МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания
по выполнению практических работ
по дисциплине
«АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ»
для студентов направления подготовки
43.03.01 Сервис

Ставрополь, 2026 г.

Предисловие

Если называть Python заменой BASIC, то тогда и трансформер Optimus Prime – это только замена грузовика.
Cory Dodt

Python – современный язык программирования, работающий на всех распространенных операционных системах для настольных компьютеров. Язык программирования Питон разрабатывается чуть более 20 лет. В настоящее время активно используется две версии языка – более старая версия 2 и современная версия. Версия 2 более не развивается, но до сих пор ещё используется, поскольку очень много программного обеспечения и библиотек разработано именно для версии 2. Между версиями есть существенная несовместимость, в том числе в синтаксисе команд ввода-вывода (программа на языке Python 2-й версии может не работать в 3-й версии и наоборот), но в целом они очень похожи. Мы будет использовать именно версию 3, как более современную и совершенную.

Python – современный универсальный интерпретируемый язык программирования. Его достоинства:

Кроссплатформенность и бесплатность.

Простой синтаксис и богатые возможности позволяют записывать программы очень кратко, но в то же время понятно.

По простоте освоения язык сравним с бейсиком, но куда более богат возможностями и значительно более современен.

Богатая стандартная библиотека, возможность разработки промышленных приложений (для работы с сетью, GUI, базами данных и т.д.)

Большинство школьных олимпиад по информатике поддерживают язык Python. С 2015 года в текстах задач ЕГЭ примеры приводятся также и на языке Python. Практика показывает, что задания ЕГЭ по информатике, в которых требуется написать программу, существенно проще решать с использованием языка Python, чем классических языков Бейсик, Паскаль, C/C++.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Марк Лутц. Изучаем Python. 4-е издание – очень толстый и подробный учебник.
2. Марк Саммерфилд. Программирование на Python 3 – менее толстый, не столь подробный учебник.
3. М. В. Сысоева, И. В. Сысоев Программирование для нормальных с нуля на языке Python В двух частях. Часть 1. Учебник. – Москва – Базальт СПО МАКС Пресс, 2018
4. Сузи, Р.А. Язык программирования Python Электронный ресурс : учебное пособие / Р.А. Сузи. - Язык программирования Python. - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 350 с.

Практическое занятие № 1
Запись арифметических выражений

ЦЕЛЬ: закрепление знаний по теоретическим основам алгоритмизации и программирования, приобретение навыков использования арифметических операций, функций и правил записи арифметических выражений согласно синтаксису языка Python.

Знания, умения и владения, приобретаемые обучающимся в результате освоения темы, в рамках формируемых компетенций (ПК-1: ИД-1$_{ПК-1}$, ИД-2$_{ПК-1}$**;** ПК-2: ИД-1$_{ПК-2}$, ИД-2$_{ПК-2}$): см. приложение 2: приобретение навыков разработки математических моделей (формируется часть указанных компетенций).

Теоретическая часть
1. Основные понятия алгоритмизации и программирования

1.1. Этапы решения задач на ЭВМ

Задачи, решаемые с помощью ЭВМ, можно классифицировать по различным критериям: по типу информации и информационным технологиям, по способу поиска решения (простые и переборные), по характеру целей (задачи оптимизации, управления, обучения, информационного поиска), по функциональному назначению, а также по уровню достижения цели и уровню их решения, в т. ч. по уровню автоматизации этапов их решения.

Задача становится разрешимой, если найдено правило, способ получения результата. В информатике такое правило называют алгоритмом.

Решение задачи на ЭВМ состоит из нескольких этапов, среди которых основными являются следующие:

1. **Постановка задачи:**
   - сбор информации о задаче;
   - формулировка условия задачи;
   - определение конечных целей решения задачи;
   - определение формы выдачи результатов;
   - описание данных (их типов, диапазонов величин, структуры и т. п.).

2. **Формализация (анализ и исследование задачи, модели, представление ее в виде уравнений, соотношений, ограничений и т.п.):**
   - анализ существующих аналогов задачи;
   - анализ технических и программных средств;
   - разработка математической модели;
   - разработка структур данных.

Понятие моделирования

При решении задачи обычно исследуют не реальный объект, а его модель**.**

Модель – искусственно созданный объект, обладающий всеми существенными признаками реального объекта, явления или процесса.

Моделирование – это метод познания, состоящий в создании и исследовании моделей.

Цели моделирования:

1) понять сущность изучаемого объекта;
2) научиться управлять объектом и определять наилучшие способы управления;
3) прогнозировать прямые или косвенные последствия;
4) решать прикладные задачи.

Математическая модель – это система математических соотношений (данных) – формул, уравнений, неравенств и т.д. и отношений между ними, описывающих поведение объекта с некоторой степенью точности и отражающих существенные свойства моделируемого процесса (объекта или явления).

Порядок составления математической модели:

- выделить реальный объект, на котором будет основываться математическая модель, и из множества его свойств, закономерностей, внутренних связей, отдельных характеристик явления и параметров выделяем те, которые являются существенными для решаемой задачи, и отбросить несущественные;
- определить, что считать исходными данными и результатами;
- подобрать математический объект с тем же числом подобных параметров, отражающий суть реального объекта; записать математические соотношения, связывающие результаты с исходными данными.

3. Выбор метода решения.

4. Разработка алгоритма:

- выбор метода проектирования алгоритма;
- выбор формы записи алгоритма (блок-схемы, псевдокод и др.);
- выбор тестов и метода тестирования;
- проектирование алгоритма.

5. Программирование:

- выбор языка программирования;
- уточнение способов организации данных;
- запись алгоритма на выбранном языке программирования.

6. Тестирование, отладка и исправление обнаруженных ошибок:

- синтаксическая отладка;
- отладка семантики и логической структуры;
- тестовые расчёты и анализ результатов тестирования;
- совершенствование программы.

Отладка программы

Отладка программы – это процесс поиска и устранения ошибок в программе, производимый по результатам ее прогона на компьютере.

В современных программных системах отладка осуществляется с использованием специальных программных средств, называемых отладчиками. Эти средства позволяют **исследовать внутреннее поведение программы**.

Программа-отладчик обычно обеспечивает следующие возможности:

- **пошаговое исполнение программы** с остановкой после каждой команды (оператора);
- **просмотр текущего значения любой переменной или нахождение значения любого выражения**, в том числе, с использованием стандартных функций; при необходимости можно установить новое значение переменной;
- **установку в программе "контрольных точек"**, т.е. точек, в которых программа временно прекращает свое выполнение, так что можно оценить промежуточные результаты, и др.

При отладке программ **важно помнить следующее:**

- в начале процесса отладки надо использовать **простые тестовые данные**;
- возникающие затруднения следует четко **разделять и устранять строго поочередно**;
- **не нужно считать причиной ошибок машину**, так как современные машины и трансляторы обладают чрезвычайно высокой надежностью.

Тест и тестирование программы

Тестирование – это испытание, проверка правильности работы программы в целом либо ее составных частей.

При отладке происходит локализация и устранение синтаксических ошибок и явных ошибок кодирования, в процессе же тестирования проверяется работоспособность программы, не содержащей явных ошибок. Тестирование устанавливает факт наличия ошибок, а отладка выясняет причину неправильной работы программы.

Как бы ни была тщательно отлажена программа, решающим этапом, устанавливающим ее пригодность для работы, является контроль программы по результатам ее выполнения на системе тестов.

Программу условно можно считать правильной, если её запуск для выбранной системы тестовых исходных данных во всех случаях дает правильные результаты. Но, как справедливо указывал известный теоретик программирования Э. Дейкстра, тестирование может показать лишь наличие ошибок, но не их отсутствие. Нередки случаи, когда новые входные данные вызывают "отказ" или получение неверных результатов работы программы, которая считалась полностью отлаженной.

Для реализации метода тестов должны быть изготовлены или заранее известны эталонные результаты. Вычислять эталонные результаты нужно обязательно до, а не после получения машинных результатов. В противном случае имеется опасность невольной подгонки вычисляемых значений под желаемые, полученные ранее на машине.

7. Анализ результатов решения задачи и уточнение математической модели с повторным выполнением этапов 2 - 6 (при необходимости).

8. Сопровождение программы: это работы, связанные с обслуживанием программ в процессе их эксплуатации:

- доработка программы для решения конкретных задач;
- составление документации к решённой задаче, к математической модели, к алгоритму, к программе, к набору тестов, к использованию программы.

1.2. Основы алгоритмизации

Алгоритм – это метод (способ) решения задачи, записанный по определённым правилам, в виде конечной последовательности однозначных предписаний, исполнение которых позволяет с помощью конечного числа шагов получить решение задачи, однозначно определяемое исходными данными из некоторого множества значений.

Свойства алгоритма

1. **Дискретность** (прерывность, раздельность). Алгоритм должен представлять процесс решения задачи как последовательное выполнение конечного числа простых (или ранее определенных) законченных действий шагов.
2. **Понятность** для исполнителя — т.е. исполнитель алгоритма должен знать, как его выполнять.
3. **Определенность** (точность, детерминированность). Каждое правило алгоритма должно быть четким и однозначным, содержать действия над известными данными. Каждое действие должно быть понятно исполнителю (для каждого алгоритма предполагается конкретный исполнитель).

Замечание. Часто под свойством детерминированности алгоритма понимается одновременное выполнение свойств точности и понятности.

4. **Результативность** (или конечность). Алгоритм должен приводить к решению задачи, получение определенного результата за конечное число шагов.
5. **Правильность.** Способность алгоритма обеспечить получение именно того результата, который требуется. Неправильность может объясняться неполнотой наших представлений о свойствах объекта или упущением в решении. Для доказательства правильности алгоритма задача часто делится на блоки и правильность доказывается для каждого блока, хотя такая проверка не является полной.

6. **Массовость**. Алгоритм решения задачи разрабатывается в общем виде, т.е. он должен быть применим для некоторого класса задач, различающихся лишь исходными данными. При этом исходные данные могут выбираться из некоторой области, которая называется областью применимости алгоритма.
7. **Универсальность**. Алгоритм должен быть составлен так, чтобы им мог воспользоваться любой исполнитель для решения аналогичной задачи. (Например, правила сложения и умножения чисел годятся для любых чисел, а не для каких-то конкретных.)
8. **Эффективность.** Выбор алгоритмы, который будет выполнен за минимальное время, с минимальными затратами ресурсов.

Таким образом, исполнитель действует ***формально***, т.е. отвлекается от содержания поставленной задачи, а только строго выполняет некоторые правила, инструкции и вместе с тем получать нужный результат.

Критерии качества алгоритма

1. **Связанность.** Определяется количеством промежуточных результатов. Чем выше количество промежуточных результатов, тем ниже связанность.
2. **Объем алгоритма.** Это количество операций или шагов, которые необходимо выполнить, а также сложность этих шагов.
3. **Логическая сложность.** Определяется количеством ветвлений и циклов.

Порядок выполнения алгоритма

1. Действия в алгоритме выполняются в порядке их записи.
2. Нельзя менять местами никакие два действия алгоритма.
3. Нельзя не закончив одного действия переходить к следующему.

Способы описания алгоритмов

1. ***Словесно-формульный.*** Описание алгоритма с помощью слов и формул на естественном языке.

Словесный способ не имеет широкого распространения по следующим причинам:

- такие описания строго не формализуемы;
- страдают многословностью записей;
- допускают неоднозначность толкования отдельных предписаний. «Он встретил ее на поле с цветами».

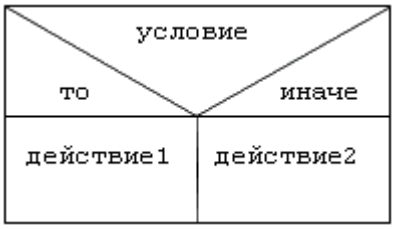
Пример. Составить алгоритм начисления зарплаты согласно следующему правилу: если стаж работы сотрудника менее 5 лет, то зарплата 130 руб., при стаже работы от 5 до 15 лет – 180 руб., при стаже свыше 15 лет зарплата повышается с каждым годом на 10 руб.

Словесно-формульное описание алгоритма решения задачи:

1. Ввести *ST*, перейти к п. 2.
2. Если *ST<5, то ZP*:.=*l30,* перейти к п. 4, иначе — перейти к п. 3.
3. Если *ST*<15, то *ZP*:=180, перейти к п.4, иначе *ZP*:=180+(*ST*-15)10, перейти к п. 4.
4. Вывести (отпечатать) значение *ZP*, перейти к п. 5.
5. Вычисления прекратить.

2. ***Табличный.*** Алгоритм представляется в форме таблицы и расчётных формул (физика, химия и т. д.).

3. ***Структурограмма:***



4. ***Синтаксическая диаграмма*** **(*формулы Бэкуса-Наура*)**

5. ***Псевдокоды.*** Полуформализованные описания алгоритмов на условном алгоритмическом языке, включающие в себя как элементы языка программирования, так и фразы естественного языка, общепринятые математические обозначения и др.

Псевдокод занимает промежуточное место между естественным и формальным языками. С одной стороны, он близок к обычному естественному языку, поэтому алгоритмы могут на нем записываться и читаться как обычный текст. С другой стороны, в псевдокоде используются некоторые формальные конструкции и математическая символика, что приближает запись алгоритма к общепринятой математической записи.

В псевдокоде не приняты строгие синтаксические правила для записи команд, присущие формальным языкам, что облегчает запись алгоритма на стадии его проектирования и дает возможность использовать более широкий набор команд. Однако в псевдокоде, так же, как и в формальных языках, есть служебные слова, смысл которых определен раз и навсегда.

Единого или формального определения псевдокода не существует, поэтому возможны различные псевдокоды, отличающиеся набором служебных слов и основных (базовых) конструкций.

Примером псевдокода является школьный алгоритмический язык.

Алгоритмический язык – это средство для записи алгоритмов в аналитическом виде, промежуточном между записью алгоритма на естественном (человеческом) языке и записью на языке ЭВМ (языке программирования).

Запись алгоритма решения задачи примера 1 на алгоритмическом языке:

```
алг зарплата(цел. ST, вещ ZP)
        арг ST
        рез ZP
нач
        если ST<5
                то ZP:=150
                иначе
                        если ST<=15
                                то ZP:=180
                                иначе ZP:=180+(ST-15)*10
                        все
        все
кон
```

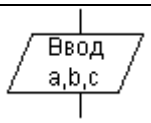
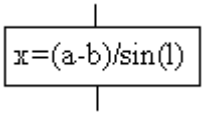
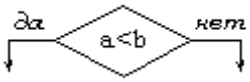
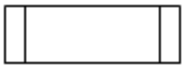
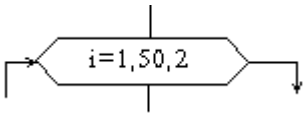
6. ***Программный***. Описание алгоритма с помощью языков программирования.
7. ***Графический.*** Алгоритм изображается в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий. Такое графическое представление называется схемой алгоритма или **блок-схемой**.

Блок-схема алгоритма представляет собой систему связанных геометрических фигур.

Правила построения блок-схем

1. В блок-схеме каждому типу действий (вводу исходных данных, вычислению значений выражений, проверке условий, управлению повторением действий, окончанию обработки и т.п.) соответствует геометрическая фигура, представленная в виде **блочного символа**. Для наглядности операции разного вида изображаются в схеме различными геометрическими фигурами.
2. Блочные символы соединяются **линиями переходов**, определяющими очередность выполнения действий. Порядок выполнения действий указывается стрелками, соединяющими блоки.
3. В схеме блоки стараются размещать сверху вниз, в порядке их выполнения.
4. Все повороты соединительных линий выполняются под углом 90 градусов.

В таблице ниже приведены наиболее часто употребляемые символы.

Название	Обозначение и пример заполнения	Выполняемая функция (пояснение)
Начало/конец (вход/выход)		Начало или конец программы, вход или выход в подпрограмму
Блоки ввода/вывода		Ввод-вывод данных
		Вывод данных на печатающее устройство
Блок вычислений		Арифметический блок определяет вычислительное действие или последовательность действий
Логический блок		Логический блок проверяет истинность или ложность условия и выбирает направления выполнения алгоритма в зависимости от условия. В блоке должны быть указаны вопрос, условие или сравнение, которые он определяет.
Предопределенный процесс		Вычисления по стандартной или пользовательской подпрограмме
Блок модификации		Выполнение действий, изменяющих пункты алгоритма, начало цикла. Внутри блока записывается параметр цикла, для которого указываются его начальное значение, граничное условие и шаг изменения значения параметра для каждого повторения.
Межстраничный соединитель		Указание связи между частями схемы, расположенной на разных страницах

Общие правила построения схемы алгоритма задачи

1. Выявить исходные данные, результаты, назначить им имена.
2. Выбрать метод (порядок) решения задачи.
3. Разбить метод решения задачи на этапы (с учетом возможностей ЭВМ).
4. Изобразить каждый этап в виде соответствующего блока схемы алгоритма и указать стрелками порядок их выполнения.
5. В полученной схеме при любом варианте вычислений:
   а) предусмотреть выдачу результатов или сообщений об их отсутствии;
   б) обеспечить возможность после выполнения любой операции, так или иначе, перейти к блоку *Останов (к выходу* схемы).

Эти правила и есть **«Основные принципы алгоритмизации».** Будем считать, что знание и применение настоящих «принципов» обязательно при составлении алгоритма любой задачи.

Типы алгоритмов

- структурированные;

- неструктурированные (т.е. с нарушением структуры – с операторами безусловного перехода);
- вспомогательные (используемые в составе других алгоритмов).

Виды алгоритмов

- линейный алгоритм;
- алгоритм ветвления;
- циклический алгоритм;
- алгоритм с подпрограммами;
- смешанные (т.е. содержащие и циклы, и ветвление, и функции).
- рекурсивный алгоритм обращается к самому себе, пока не выполнится определенное условие.

Базовые алгоритмические конструкции

Структу́рное программи́рование – методология разработки программного обеспечения, в основе которой лежит представление программы в виде иерархической структуры блоков. Предложена в 70-х годах XX века Э. Дейкстрой, разработана и дополнена Н. Виртом. Основывается на теореме о структуре.

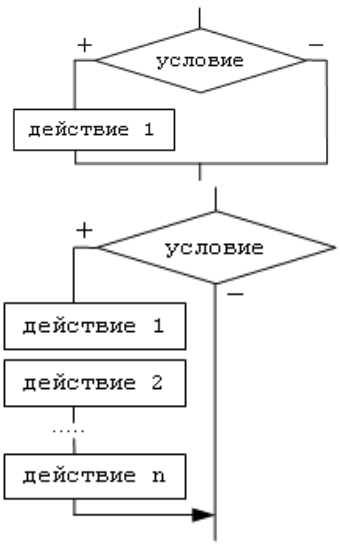
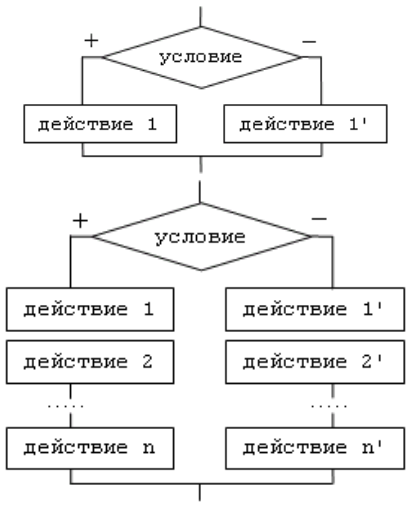
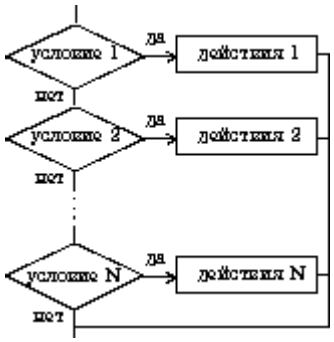
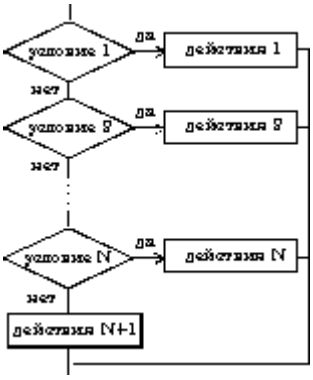
Согласно **теореме о структуре** (теорема Бёма – Якопини, 1966 г) логическая структура любого алгоритма может быть представлена комбинацией трех базовых структур: *следование, ветвление, цикл.*

Характерной особенностью базовых структур является наличие в них одного входа и одного выхода.

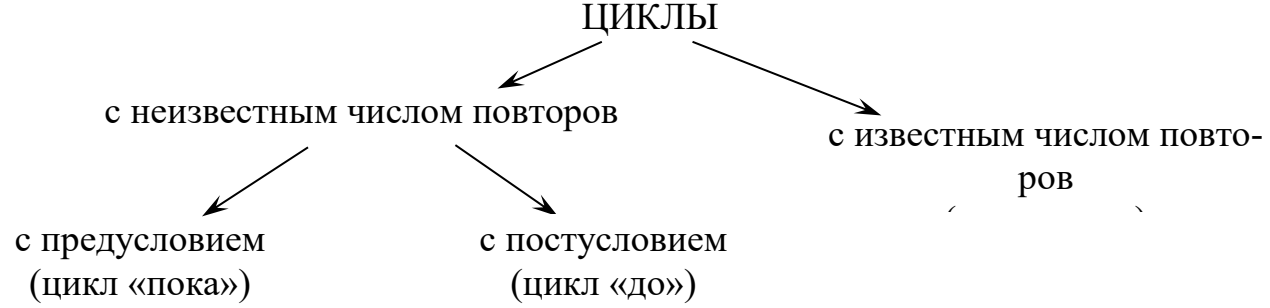
1. **Базовая структура <u>следование</u> (линейный алгоритм)**. Образуется из последовательности действий, следующих одно за другим:

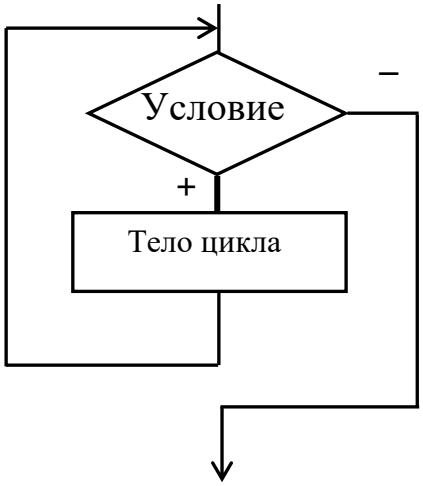
2. **Базовая структура <u>ветвление</u> (алгоритм ветвления)** обеспечивает в зависимости от результата проверки условия выбор одного из альтернативных путей работы алгоритма. Условие – вопрос, имеющий два варианта ответа: **да** или **нет.** Каждый из путей ведет к **общему выходу**, так что работа алгоритма будет продолжаться независимо от того, какой путь будет выбран. Запись ветвления выполняется в двух формах: полной и неполной.

Структура **ветвление** существует в четырех основных вариантах:

Неполная форма записи		**Полная форма записи**	
1. **если-то**		2. **если-то-иначе**	
3. **выбор**		4. **выбор-иначе**	

3. **Базовая структура <u>цикл</u>** обеспечивает **многократное выполнение** некоторой совокупности действий, которая называется **телом цикла,** над новыми данными.



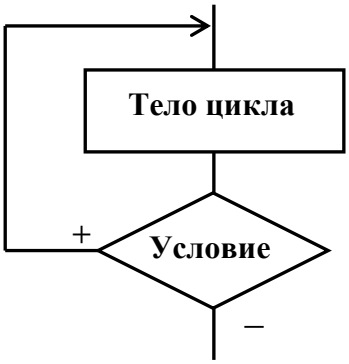


Цикл типа «пока» (Цикл с предусловием)

Выполнение цикла «пока» начинается с проверки условия, поэтому такую разновидность циклов называют циклами **с предусловием**. Переход к выполнению действия осуществляется только в том случае, если условие выполняется, иначе происходит выход из цикла. Тело цикла выполняется до тех пор, пока выполняется условие. Условие цикла необходимо подобрать так, чтобы действия, выполняемые в цикле, привели к нарушению его истинности, иначе произойдет зацикливание.

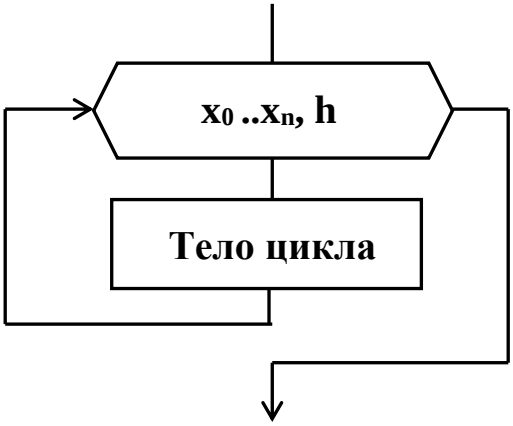
Зацикливание - бесконечное повторение выполняемых действий.

Цикл типа «до» (Цикл с постусловием)



Исполнение цикла начинается с выполнения действия. Таким образом, тело цикла будет реализовано хотя бы один раз. После этого происходит проверка условия. Поэтому цикл "до" называют циклом **с постусловием**. Если условие выполняется, то происходит возврат к выполнению действий, иначе осуществляется выход из цикла. Для предотвращения зацикливания необходимо предусмотреть действия, приводящие к ложности условия.

Цикл типа «для»



Цикл с параметром, или цикл со счетчиком, или арифметический цикл - это цикл с заранее известным числом повторов. Предписывает выполнять тело цикла для всех значений некоторой переменной (параметра цикла) в заданном диапазоне.

x_0 – начальное значение параметра; **h** – шаг; x_n – последнее значение параметра.

Для создания циклов с параметром необходимо использовать правила:

1. Параметр цикла, его начальное и конечное значения и шаг должны быть одного типа.
2. Запрещено изменять в теле цикла начальное, текущее и конечное значения для параметра.
3. Запрещено входить в цикл, минуя блок модификации.
4. После выхода из цикла значение переменной параметра неопределенно и не может использоваться в дальнейших вычислениях.
5. Из цикла можно выйти, не закончив его, тогда переменная параметр сохраняет свое последнее значение.

1.3. Теоретические основы программирования

Программирование – это раздел информатики, изучающий методы и приемы составления программ для компьютеров. Кроме того, программирование – это подготовка задачи к решению ее на компьютере.

Программа – последовательность действий, которые должен выполнить компьютер в строго указанной очерёдности для достижения конкретного результата.

Языки программирования – это совокупность средств и правил представления алгоритма в виде, приемлемом для компьютера.

Системы программирования – это набор средств ввода, редактирования, трансляции и выполнения программ на ЭВМ.

Транслятор – это устройство или комплекс программ, обеспечивающий перевод программы, написанной на символическом языке, в совокупность машинных команд, либо передающие/преобразующие данные или другую программу.

Например, транслятор воспринимает операторы одного языка и вырабатывает соответствующие операторы другого языка.

Компилятор – это транслятор, обеспечивающий перевод программы, написанной на алгоритмическом языке, в совокупность машинных команд без ее выполнения в компьютере.

Компилятор оценивает исходный текст в соответствии с синтаксической конструкцией языка и переводит на машинный язык.

Например, компилятор берет программу, написанную на языке С, и преобразует ее в программу на языке ассемблера.

Интерпретатор – это транслятор, обеспечивающий перевод каждой конструкции алгоритмического языка в машинные команды и одновременное выполнение этой конструкции в компьютере.

При исполнении программных операторов, интерпретатор должен сначала сканировать каждый оператор с целью прочтения его содержимого, а затем выполнить запрошенную операцию.

1.4. Установка Python и сред разработки

Для работы необходимо установить интерпретатор языка Питон и, для удобства написания программ, среду разработки.

Установка интерпретатора

Дистрибутивы для установки:

- Windows, 32-бита: https://www.python.org/ftp/python/3.10.5/python-3.10.5.exe
- Windows, 64-бита: https://www.python.org/ftp/python/3.10.5/python-3.10.5-amd64.exe
- Дистрибутивы для Mac OS X есть на странице https://www.python.org/downloads/release/python-3105/
- Для операционных систем GNU/Linux язык Питон скорее всего есть в дистрибутиве, для чего необходимо поставить пакет python3. Возможно, он уже установлен в вашей системе: проверить это можно командой python3.
- Для Android есть пакет QPython3.

Установка интегрированной среды разработки

Для удобства разработки кода программисты используют среды разработки (IDE).

Для обучения программированию на языке Питон рекомендуется среда разработки Wing IDE 101.

Для больших проектов рекомендуется PyCharm Community Edition.

WING IDE 101

Дистрибутив для Windows: http://wingware.com/pub/wingide-101/5.0.9/wingide-101-5.0.9-1.exe
Дистрибутивы для других операционных систем можно найти на http://wingware.com/downloads/wingide-101
Видео установки Python и Wing IDE 101 на Windows 7 (на английском): http://www.youtube.com/watch?v=OrpavIGbkSw
Обратите внимание: если вы устанавливаете среду после установки Python, то она должна сама обнаружить установленный Python. Если этого не произошло, пропишите в меню Edit-Configure python в верхнем поле ввода путь к исполняемому файлу Python (под windows это обычно что-то типа c:\python34\python.exe).

PYCHARM COMMUNITY EDITION

Версия для Windows: https://download.jetbrains.com/python/pycharm-community-2022.1.4.exe Остальные версии можно скачать здесь: http://www.jetbrains.com/pycharm/download/

1.5. Правила записи в Python арифметических выражений

Арифметические выражения

Выражение состоит из операторов и операндов. Операндами могут быть выражения или одни из его частных случаев – числа (константы) или переменные, **операторы** обозначают выполняемые над ними действия (+ сложение, - вычитание, * умножение, / деление (результат – частное, т.е. вещественное число, даже если А нацело делится на В), //– целая часть от деления (неполное частное), % остаток от деления, …).

Все основные операции языка С++ можно разбить на следующие группы:

- арифметические операции;
- логические операции;
- операции отношения;
- операции с битами информации;
- операции со строками;
- операции присваивания;
- условная операция (условное выражение или тернарная операция).

Примеры выражений Python:

```
(a + 0.12)/6
x and y or not z
(t * sin(x)-1.05e4)/((2 * k + 2) * (2 * k + 3))
```

В выражение могут входить операнды различных типов, но этого, строго говоря, не следует допускать. Если операнды имеют одинаковый тип, то результат операции будет иметь тот же тип. Если операнды различного типа, перед вычислениями выполняются преобразования типов по определенным правилам, обеспечивающим преобразование более коротких типов в более длинные для сохранения значимости и точности.

Порядок вычисления выражений определяется рангом (приоритетом) входящих в него операций (табл. 3). Принятый в С++ ранг операций наиболее близок к математическому, также как и принятый порядок их вычисления. Так, умножение и деление (мультипликативные операции) старше сложения и вычитания (аддитивные операции). Унарные операции + и – старше бинарных, т.е., знак операнда вычисляется в первую очередь. Операции типа присваивания младше прочих, что позволяет выполнить их только после того, как значение выражения вычислено полностью. Операции отношения младше арифметических операций, что позволяет использовать естественную запись логических выражений, например, `x>0 && y>0`. Здесь в первую очередь вычисляются значения отношений, которые затем являются операндами конъюнкции.

Таблица 1 – Порядок вычисления выражений

Группа	Тип действий	Операции или элементы
1	Вычисления в круглых скобках	( )
2	Вычисления значений функции	Функции
3	Унарные операции	+, -, ~ (побитовое отрицание), …
4	Операции типа умножения	*, /, %
5	Операции типа сложения	+, –
6	Операции отношения	<, <=, >, >=, ==, !=

Арифметические выражения строятся из **операндов, арифметических операций и круглых скобок.**

Круглые скобки используются для заключения в них части выражения, значения которой необходимо выполнить в первую очередь. В выражении может быть любое количество круглых скобок, причем количество открывающих круглых скобок должно быть равно количеству закрывающих. Части выражений, заключенные в круглые скобки, должны быть либо не пересекающимися, либо вложенными друг в друга.

Арифметические выражения записываются по следующим правилам:

- Запись ведётся в строчку.
- Нельзя опускать знак умножения между сомножителями.
- Для обозначения переменных используются буквы латинского алфавита.
- Операции выполняются в соответствии с *приоритетами*: сначала вычисление функций, затем умножение и деление, потом сложение и вычитание.
- Если в одном выражении записано несколько операций одинакового приоритета, унарные операции, условная операция и операции присваивания выполняются *справа налево*, остальные – *слева направо*. Например, **a = b = c** означает **a = (b = c),** а **a + b + c** означает **(a + b) + c**.
- Для изменения порядка действий используются круглые скобки.
- При использовании стандартных функций аргумент обязательно заключается в круглые скобки.

Таблица 2 – Арифметические операции с числами и встроенные функции с одним и двумя аргументами

x + y	Сложение – сумма x и y (2 + 3 = 5)
x - y	Вычитание – разность x и y (5 - 2 = 3)
x * y	Умножение – произведение x и y (2 * 3 = 6)
x / y	Деление x на y: 4 / 1.6 = 2.5, 10 / 5 = 2.0, результат дробный
x // y	Деление x на y нацело: 11 // 4 = 2, 11.8 // 4 = 2.0, результат целый, только если оба аргумента целые

x % y	Остаток от деления: 11 % 4 = 3, 11.8 % 4 = 3.8000000000000007 (присутствует ошибка, связанная с неточностью представления данных в компьютере)
x ** y	Возведение x в степень y: (2 ** 3 = 8)
abs(x)	Модуль числа x
round(x)	Округление (round(11.3) = 11)
round(x, n)	Округляет число x до n знаков после запятой: round(12.34567, 3) = 12.346
pow(x, y)	полный аналог записи x ** y
divmod(x, y)	выдаёт два числа: частное и остаток, обращаться следует так: q, r = divmod(x, y)

Модуль math

Загрузка модулей в Python осуществляется с помощью оператора import.

Самый простой способ его использования – загрузить всё содержимое модуля глобально:

```
from math import *
t = sin (pi /6)
```

Но академически правильнее, писать следующим образом:

```
import math
t = math.sin (math.pi / 6)
v = math.log (math.e)
print(t, v)
```

Таблица 3 – Наиболее употребительные функции и константы модуля math

trunc(X)	Усечение значения X до целого
sqrt(X)	Квадратный корень из X
exp(X)	Экспонента числа X
log(X), log2(X), log10(X)	Натуральный, двоичный и десятичный логарифмы X
log(X, n)	Логарифм X по основанию n
sin(X), cos(X), tan(X)	Синус, косинус и тангенс X, X указывается в радианах
asin(X), acos(X), atan(X)	Арксинус, арккосинус и арктангенс X
atan2(X, Y)	арктангенс отношения X Y с учётом квадранта
degrees(X)	Конвертирует радианы в градусы
radians(X)	Конвертирует градусы в радианы
sinh(X), cosh(X), tanh(X)	Гиперболические синус, косинус и тангенс X
asinh(X), acosh(X), atanh(X)	Обратный гиперболический синус, косинус и тангенс X
hypot(X, Y)	Гипотенуза треугольника с катетами X и Y
factorial(X)	Факториал числа X
gamma(X)	Гамма-функция X
pi	Выдаётся число _
e	Выдаётся число e

Примеры записи выражений:

Математическое выражение	Запись на Python
1. $10x+3\sqrt{\cos x}$	1. `10 * x + 3 * sqrt(cos(x))`
2. $\frac{\sin 2x-1}{x^2-1}$	2. `(sin(2*x) - 1) / (pow(x, 2) - 1)`
3. $\lvert x \rvert + 2\,ctgx$	3. `abs(x) + 2 * cos(x) / sin(x)`

Вопросы и задания

Задание I

Записать следующие выражения по правилам языка Python

1. $\sin 5x + 2\,\mathrm{tg}\,x^2$	5. $\frac{2xy+4x^2}{5y}$;
2. $2x + 3y^2 - \sin 3x$	6. $4\sqrt{\cos 3b - 2\sin b}$;
3. $\sqrt{\frac{\sin y^2}{2\cos 2y}} + 2\lvert tgy - 25 \rvert$	7. $-2\cos x^2 + 2y^7$;
4. $\sqrt{1-\frac{2}{x^2}} + \lvert x^5 - 3x \rvert$	8. $\frac{\lvert 2a + tg\,3a \rvert}{\sqrt{1+b^8}+11}$

Задание II

Выполнить согласно варианту:

1.	$a=\frac{2\cos\left(x-\frac{\pi}{6}\right)}{0.5+\sin^2 y}$,	$b=1+\frac{z^2}{3+\frac{z^2}{5}}$.
2.	$s=\left\lvert x^{\frac{y}{x}} - \sqrt[3]{\frac{y}{x}} \right\rvert + \ln x$,	$t=(y-x)\frac{y-z(y-x)}{1+(y-x)^2}$.
3.	$s=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}$,	$t=x\left(\sin x^3+\cos^2 y\right)$.
4.	$y=e^{-bt}\sin(at+b)-\sqrt{\lvert bt+a \rvert}$,	$s=b\sin\left(at^2\cos 2t\right)-1$.
5.	$w=\sqrt{x^2+b}-\frac{b^2\sin^3(x+a)}{x}$,	$y=\cos^2 x^3-\frac{x}{\sqrt{a^2+b^2}}$.

6.	$s = x^3 \cdot tg^2\left(x+b\right)^2 + \frac{\alpha}{\sqrt{x+b}}$,	$d = \frac{bx^2 - a}{e^{ax} - 1}$.
7.	$z = \frac{x^2\left(x+1\right)}{b} - \sin^2\left(x+a\right)$,	$s = \sqrt{\frac{xb}{a}} - \cos^2\left(x+b\right)^3$.
8.	$y = \sin^3\left(x^2+a\right)^2 - \sqrt{\frac{x}{b}}$,	$z = \frac{x^2}{a} + \cos\left(x+b\right)^3$.
9.	$f = \sqrt[3]{m \cdot tg\, t + \left\|c \cdot \sin t\right\|}$,	$z = m\cos\left(bt \cdot \sin t\right) + c$.
10.	$y = btg^2 x - \frac{a}{\sin\left(\frac{x}{a}\right)}$,	$d = a \cdot e^{-\sqrt{a}} \cdot \cos\left(\frac{bx}{a}\right)$.
11.	$f = \ln\left(a + x^2\right) + \sin^2\left(\frac{x}{b}\right)$,	$z = e^{-cx} \cdot \frac{x + \sqrt{x+a}}{x - \sqrt{\left\|x-b\right\|}}$.
12.	$y = \frac{a^{2x} + b^{-x} \cdot \cos\left(a+b\right)x}{x+1}$,	$k = e^{\frac{ax}{\sqrt{2}}} \cdot \cos\sqrt{\frac{bx}{2}}$.
13.	$z = \sqrt{ax \cdot \sin 2x + e^{-2x} \cdot \left(x+b\right)}$,	$w = e^3 \cdot \left\|\sin bx\right\| - \frac{x^3}{a}$.
14.	$u = \frac{a^2 x + e^{-x} \cdot \cos bx}{bx - e^{-x} \cdot \sin bx + 1}$,	$f = e^{2x} \cdot \ln\left(a+x\right) - b^{3x} \cdot \ln\left(b-x\right)$.
15.	$z = \frac{\sin x}{\sqrt{1 + m^2 \sin^2 x}} - cm \cdot \ln\left(mx\right)$,	$s = e^{-ax} \cdot \sqrt{x+1} + e^{-bx} \cdot \sqrt{x+1.5}$.
16.	$a = e^{-mx} \cdot \frac{\sqrt{m + x^2}}{\left\|m^2 - x^2\right\|}$,	$b = \ln\left(c^2 + \sqrt{c+y}\right) + e^{-cy}$.
17.	$z = x^3 \cdot \sin^2\left(x+b\right) + \frac{x}{\left(x+b^2\right)}$,	$s = \left(m^2 + y^2\right) \cdot e^{my} + \sqrt{3m + \ln\left(m+4y\right)}$.

18.	$l = \dfrac{a\cos^3 bx}{\sin a + bx\cos b - e^a}$,	$w = \dfrac{\sin\left(x^3 - a\right) + b\lvert x\rvert}{\cos\left(ax + b\right)}$.
19.	$y = \cos^2\left(x + a^2\right)^2 + \sqrt{\dfrac{x}{b^2}}$,	$z = \dfrac{a^3}{3x^2} + e^{(x+b)^3}$.
20.	$s = \left\lvert x^{\frac{x}{y}} - \sqrt[5]{\dfrac{x}{y}}\right\rvert$,	$t = (x - y)\dfrac{\sin(x + y) - tg(y - x)\cdot z}{1 + (x - y)^2}$
21.	$s = \dfrac{e^{my}}{2 + 5x} + \sqrt{\dfrac{\ln(m - 3)}{\lvert 2a\rvert}}$,	$v = \sqrt{\lvert bt - a\rvert} - e^{\sqrt{bt}}\cdot\sin(bt + a)$.
22.	$y = 1 + \dfrac{b\sin\left(at^2\cos 3t\right)}{12x}$,	$f = \sqrt[3]{m\cdot tg\, t + \lvert c\sin 3t\rvert}$
23.	$y = a\cdot tg^2(x + a) - \dfrac{b}{\sin^2\left(\dfrac{a}{x}\right)}$,	$d = b\cdot e^{-\sqrt{b}}\cdot\cos\left(\dfrac{ax}{b} + 1.4\right)$.
24.	$z = \sqrt{\dfrac{x^2}{b}} - \ln\left(a^2 + x^2\right)$,	$f = e^{-cx}\cdot\dfrac{x + \sqrt{\lvert x - b\rvert}}{x - \sqrt{x + a}}$.
25.	$s = \dfrac{\left(m^2 + y^2\right)}{5a} + \sqrt{\lvert m\rvert + tg(m - 2y)}$,	$f = \dfrac{e^{2x}}{6\cdot\ln(a + x)} - b^{3x}\cdot\sin(b - x)$.

Задание III

В интерактивном режиме вычислить значение выражений, заданных в задании I, при $a = 1$, $b = 2$, $x = 0.5$, y = 7.

Провести подробный анализ полученных результатов.

Задание IV

При работе в интерактивном режиме одной из наиболее полезных является функция help, выводящая справочную информацию об объекте, подающемся в качестве аргумента.

С помощью функции help проанализируйте функции print и input, в тетради опишите их параметры.

Контрольные вопросы

1. Что такое алгоритм?
2. Перечислите свойства алгоритмов.
3. Способы записи алгоритма.
4. Перечислите базовые алгоритмические конструкции. Какова блок-схема следования?
5. Какова блок-схема полной (неполной) формы команды ветвления?
6. Какова блок-схема цикла с предусловием?
7. Какова блок-схема цикла с постусловием?
8. Какова блок-схема цикла с параметром?
9. Что такое программа?
10. Что такое транслятор?
11. Что такое компилятор?
12. Что такое интерпретатор?
13. Что такое оператор и операнд?
14. В каких случаях при записи математических выражений на языке Python используются круглые скобки?
15. Правила использования стандартных математических функций в Python.
16. Для чего используется интерактивный режим?
17. Что необходимо для использования интегрированной среды разработки?

Список литературы, рекомендуемый к использованию по данной теме

- [1] стр. 14-16, 30-36, 79- 82, 106- 107;
- [6] стр. 5- 31.

Практическое занятие № 2
Программирование алгоритмов линейной структуры

ЦЕЛЬ: закрепление знаний о типах данных, преобразованиях типов, приобретение навыков составления и отладки программ линейной структуры на языке программирования Python

Знания, умения и владения, приобретаемые обучающимся в результате освоения темы, в рамках формируемых компетенций (ПК-1: ИД-1$_{ПК-1}$, ИД-2$_{ПК-1}$**;** ПК-2: ИД-1$_{ПК-2}$, ИД-2$_{ПК-2}$): см. приложение 2: вырабатывается навык разработки математических моделей, а также формируется способность к разработке и применению алгоритмических и программных решений (линейной структуры) в области прикладного программирования с умением объяснить теоретическую и практическую части темы (формируется часть указанных компетенций).

Теоретическая часть
2. Программирование алгоритмов линейной структуры

2.1. Общая характеристика языка программирования Python

Пайтон или **Пито́н** – язык программирования высокого уровня (Python – англоязычное название), ориентированный на повышение производительности программиста и читаемости кода. Мощность языка достигается за счет стандартной библиотеки функций, сам же синтаксис ядра включает минимум возможностей.

Структурное программирование, объектно-ориентированное программирование, функциональное, императивное и аспектно-ориентированное – вот список парадигм, которые поддерживает язык python.

Код структурируется в функции и классы, которые можно объединить в модули.

Язык python создан в конце 1980-х годов голландским программистом Гвидо ван Россумом (сотрудник института CWI). Изначально язык был ориентирован на объектную парадигму. Название языку дано в честь популярного английского комедийного телешоу 1970-х «Летающий цирк Монти Пайтона» (подразумевается дружелюбный и игровой подход в обучении).

Python – современный язык программирования, работающий на всех распространенных операционных системах для настольных компьютеров. Язык программирования Питон разрабатывается чуть более 20 лет. В настоящее время активно используется две версии языка – более старая версия 2 и современная версия 3. Версия 2 более не развивается, но до сих пор ещё используется, поскольку очень много программного обеспечения и библиотек разработано именно для версии 2. Между версиями есть существенная несовместимость, в том числе в синтаксисе команд ввода-вывода (программа на языке Python 2-й версии может не работать в 3-й версии и наоборот), но в целом они очень похожи. Будем использовать именно версию 3, как более современную и совершенную.

Python – современный универсальный интерпретируемый язык программирования. Его достоинства:

1. Кроссплатформенность и бесплатность.

2. Динамическая типизация.
3. Автоматическое управление памятью.
4. Простой синтаксис и богатые возможности позволяют записывать программы очень кратко, но в то же время понятно.
5. По простоте освоения язык сравним с бейсиком, но куда более богат возможностями и значительно более современен.
6. Богатая стандартная библиотека, возможность разработки промышленных приложений (для работы с сетью, GUI, базами данных и т.д.)

Большинство школьных олимпиад по информатике поддерживают язык Python. С 2015 года в текстах задач ЕГЭ примеры приводятся также и на языке Python. Практика показывает, что задания ЕГЭ по информатике, в которых требуется написать программу, существенно проще решать с использованием языка Python, чем классических языков Бейсик, Паскаль, C/C++.

Программирование на Питоне в настоящее время придерживается целой философии, называемой «The Zen of Python» («Дзен Питона»), автором которой является Тим Петерс.

Python постоянно развивается, появляются его новые версии, отчасти из-за этого на Python отсутствуют стандарт ANSI, ISO или другие официальные стандарты, их роль выполняет CPython.

2.2. Основные понятия языка

Основу любого языка программирования образуют три, его составляющие: алфавит, синтаксис и семантика.

Алфавит - это фиксированный для данного языка программирования набор основных символов (букв алфавита), из которых состоит любой текст на этом языке. Никакие другие символы в тексте, не допускаются. Все символы в тексте кодируются с использованием стандартной кодовой таблицы ASCII. Каждый символ кода ASCII имеет неотрицательный порядковый номер, т.е. множество всех символов является упорядоченным.

Алфавит языка Python

Изучение любого языка начинается с изучения алфавита, из букв складываются слова, из слов - предложения. То же происходит и при изучении языка программирования. Сначала мы должны уяснить, какие символы можно использовать для записи слов языка, из которых можно формировать определенные конструкции. Итак, в алфавит языка Python входят:

1. Латинские буквы от a до z и от A до Z.

   В Python есть различия между прописными и строчными буквами алфавита, например, `chislo`, `CHISLO`, `Chislo` - разные имена переменных. Он чувствителен к регистру.
2. Цифры от 0 до 9.
3. Специальные символы, например `+`, `-`, `*`, `/`.
4. Зарезервированные (служебные) слова: `for`, `if`, `class`, `def` и т. д.

Идентификаторы и общие правила их написания

Для того чтобы программа решения задачи обладала свойством массовости, следует употреблять не конкретные значения величин, а использовать их обозначения для возможности изменения по ходу выполнения программы их значений. Для обозначения в программе пере менных и постоянных величин используются имена - идентификаторы *(identification* - установление соответствия объекта некоторому набору символов).

Программа на Python представляет собой последовательность инструкций, которые называются операторами. Необходимо учитывать, следующее:

- в идентификатор не могут входить пробелы, специальные символы алфавита;
- идентификатор начинается только с буквы или со знака подчеркивания;
- идентификатор может состоять из букв, цифр и знака подчеркивания;
- при написании идентификаторов можно использовать как прописные, так и строчные буквы латинского алфавита;
- идентификатор не должен являться зарезервированным словом.

Синтаксис – это система правил, определяющих допустимые конструкции букв алфавита. С помощью этих конструкций, представляются отдельные компоненты алгоритма и алгоритм в целом, записанные на данном языке программирования. Таким образом, для каждой последовательности символов алфавита синтаксис позволяет ответить, на вопрос, является ли она текстом на данном языке программирования или нет.

Синтаксис устанавливает, как можно на этом языке сформировать корректный текст и писать программы. Синтаксически правильная программа интерпретируется без ошибок.

Из символов алфавита формируются *лексемы* языка:

- **идентификаторы;**
- **ключевые (зарезервированные) слова;**
- **знаки операций;**
- **константы;**
- **разделители (скобки, точка, запятая, пробельные символы).**

Лексема, или элементарная конструкция - минимальная единица языка, имеющая самостоятельный смысл. Границы лексем определяются другими лексемами, такими, как разделители или знаки операций.

Одним из важнейших синтаксических понятий любого языка программирования является понятие **идентификатора**.

Идентификатор – *имя программного объекта.*

В идентификаторе могут использоваться латинские буквы, цифры и знак подчеркивания. Прописные и строчные буквы различаются, например, `alpha`, `Alpha` и `ALPHA` – три различных имени. Первым символом идентификатора может быть буква или знак подчеркивания, но не цифра. Пробелы внутри имен не допускаются.

Для улучшения читаемости программы следует давать объектам осмысленные имена. Существует соглашение о правилах создания имен, называемое венгерской нотацией (поскольку предложил ее сотрудник компании Microsoft венгр по

национальности), по которому каждое слово, составляющее идентификатор, начинается с прописной буквы, а вначале ставится префикс, соответствующий типу величины, например, `lpfnSetFirstDialog`. Другая традиция – разделять слова, составляющие имя, знаками подчеркивания: `max_length`, `number_of_galosh` [2].

Длинна идентификатора по стандарту не ограничена, но некоторые компиляторы и компоновщики налагают на нее ограничения. Идентификатор создается на этапе объявления переменной, функции, типа, и т. п., после этого его можно использовать в последующих операторах программы. При выборе идентификатора необходимо иметь в виду следующее:

– идентификатор не должен совпадать с ключевыми словами и именами используемых стандартных объектов языка.

Примеры правильных идентификаторов: `value25`, `My_File`, `index`, `Operation`. Примеры синтаксически неправильных идентификаторов:

`1_Side` (имя не должно начинаться с цифры),

`Том 2` (не допускается пробел),

`New-Name` (не допускается дефис),

`break` (служебное слово).

Заметим, что имена, константы, служебные слова нельзя писать слитно друг с другом, они должны отделяться один от другого пробелом, переводом строки или комментарием.

Служебные (зарезервированные, ключевые) **слова** – это идентификаторы, имеющие специальное значение для транслятора. Их можно использовать только в том смысле, в котором они определены. Например, `from` - ключевое слово, обозначающее импорт нескольких функций из модуля, `return` - ключевое слово, используемое для обозначения операции: `вернуть результат`.

Знаки операций. Знак операции – это один или более символов, определяющих действие над операндами. Внутри знака операции пробелы не допускаются. Операции делятся на унарные, бинарные и тернарные по количеству участвующих в них операндов. Один и тот же знак может интерпретироваться по-разному в зависимости от контекста. Все знаки операций за исключением [ ] ,( ) и ? : представляют собой отдельные лексемы.

Семантика языка – это смысловое значение слов. В программировании – начальное смысловое значение операторов, основных конструкций языка, т.п. [1].

Пример. Первый код:

```
i = 0
S = 0
while i < 5:
      i += 1
      S += i
```

Второй код:

```
S = 0
for i in range(5):
      S += i
```

Логически эти два фрагмента кода выполняют одно и то же, результаты их работы идентичны. В то же время семантически это два разных цикла.

Таким образом,

- **синтаксис** – формальный набор правил, определяющий способ построения любых конструкций языка;
- **семантика** – множество правил, определяющих смысл синтаксических конструкций.

Комментарии – это текст, который игнорируется при компиляции, что бывает полезно программистам для пояснения тех или иных участков кода. Компилятор обрабатывает комментарии как пустое пространство. Комментарии можно использовать и для временного отключения части кода при отладке.

Комментарии в программе на Python начинаются с символа # «диез (или шарп или хеш) » (#однострочный комментарий). Внутри комментария можно использовать любые допустимые на данном компьютере символы, а не только символы из алфавита языка Python, поскольку компилятор комментарии игнорирует.

2.3. Данные и способы их организации

Данные – это информация, представленная в виде пригодном для обработки автоматическими средствами, в частности ЭВМ, при возможном участии человека.

В программах обрабатываемые данные фигурируют в качестве значений тех или иных программных объектов. Данные, которые зафиксированы в тексте и не изменяются в процессе выполнения программы, называются **константами**; остальные данные являются значениями объектов, называемых **переменными**, поскольку значения этих объектов могут изменяться в процессе выполнения программы.

Данные, с которыми работает программа, хранятся в оперативной памяти. Интерпретатору необходимо точно знать: сколько места они занимают, как именно закодированы и какие действия над ними можно выполнять. Все это задается при описании данных с помощью типа.

Тип данных однозначно определяет:

1. Формат представления данных в памяти компьютера.
2. Множество допустимых значений, принимаемое переменной или константой, принадлежащей к выбранному типу.
3. Множество допустимых действий применимых к этому типу (операции и функции).

Константы в языке Python

Итак, константа представляет значение, которое не может быть изменено. Важно заметить, что в Python нет возможности объявить константу, однако существует соглашение, что переменные, которые не должны меняться во время выполнения программы, записываются заглавными буквами: `MAX_SPEED`, `LOCAL`.

Порядок объявления и инициализации переменных

Переменная – это именованная область памяти, в которой хранятся данные определенного типа. Каждая переменная должна быть проинициализирована в тексте программы перед первым ее использованием.

Имя переменной (идентификатор) обозначает в тексте программы величину, изменяющую свое значение. Для каждой переменной в памяти компьютера выделяется некоторая область памяти, способная хранить значение. Имя позволяет осуществить доступ к области памяти, хранящей значение переменной.

Тип переменной определяет размер выделяемой памяти и способ хранения значения.

Значение переменной. Если переменная не проинициализирована начальным значением, то перед первым ее использованием, значение переменной будет не определено. Независимо от того было задано или нет начальное значение переменной (при ее описании или при вводе с клавиатуры, например) ее величина может изменяться в ходе выполнения программы.

Область видимости. Переменные в языке Python классифицируются по области видимости и делятся на две категории: локальные и глобальные.

Локальная переменная – это такая переменная, которая объявлена внутри какой-либо функции. По завершении работы функции эта переменная разрушается, а занимаемая ей память высвобождается. То есть, время жизни такой переменной ограничено тем блоком, в котором она объявлена.

Глобальная переменная – это такая переменная, которая объявлена вне любого блока. Следовательно, время ее жизни – время жизни самой программы.

Резюмируя все вышесказанное, можно сделать вывод, что прежде чем описывать какие-либо данные, необходимо выполнить следующие действия:

- выбрать имя переменной (описать идентификатор);
- определить исходя из необходимого диапазона представления чисел тип переменной;
- определить область видимости переменной;
- обнулить или проинициализировать переменную начальным значением.

2.4. Стандартные простые типы данных

Python является языком программирования с динамической неявной типизацией.

Встроенные функции, выполняющие преобразование типов:

`bool(x)` - преобразование к типу bool, использующая стандартную процедуру проверки истинности. Если x является ложным или опущен, возвращает значение False, в противном случае она возвращает True.

`bytes([строка[, кодировка [ошибки]]])` - возвращает объект типа bytes, который является неизменяемой последовательностью целых чисел в диапазоне $0 \leq X < 256$. Аргументы конструктора интерпретируются как для bytearray().

`complex([real[, imag]])` - преобразование к комплексному числу.

`float([X])` - преобразование к числу с плавающей точкой. Если аргумент не указан, возвращается 0.0.

`int([object], [основание системы счисления])` - преобразование к целому числу.

`str([object], [кодировка], [ошибки])` - строковое представление объекта.

2.5. Структура программы

Как правило, программа на Python должна состоять из следующих частей:

1) считывание данных,
2) решение задачи,
3) вывод результата.

Например, программа, считающая сумму двух чисел может иметь следующий вид:

```
a = int(input())
b = int(input())
summa = a + b
print(summa)
```

В то же время в Python программа может быть «однострочником»: print(int(input())+int(input()))

Основные принципы синтаксиса языка Python

1. Конец строки является концом инструкции (точка с запятой не требуется).

ПРИМЕР

```
a = 5
b = 3
print(a + b)
```

2. Вложенные инструкции объединяются в блоки по величине отступов. Отступ может быть любым, главное, чтобы в пределах одного вложенного блока отступ был одинаков. В тоже время для обозначения отступа рекомендуется использовать 4 пробела.

ПРИМЕР

```
if a == 5:
    print('yes')
    a += 1
```

3. Вложенные инструкции в Python записываются в соответствии с одним и тем же шаблоном, когда основная инструкция завершается двоеточием, вслед за которым располагается вложенный блок кода, обычно с отступом под строкой основной инструкции.

Правила оформления текста программы, направленные на облегчение понимания смысла и повышение наглядности, таковы:

- разделять логические части программы пустыми строками;
- разделять операнды и операции пробелами;
- в каждой строке должно быть, как правило, не более одного оператора;
- ограничивать длину строки 60-70 символами;
- отступами слева отражать вложенность операторов и блоков;

– длинные операторы располагать в нескольких строках;
– проводить алгоритмизацию так, чтобы определение одной функции занимало, как правило, не более одного экрана текста;
– стремиться использовать типовые заготовки фрагментов программ, включая и типовую структуру блока и программы в целом.

Конечно, эти правила нужно использовать совместно с *правилами именования* элементов программы и *комментирования* текста программы:
– **комментарии в тексте необходимы;**
– **имена объектов программы выбираются осмысленно.** Каждое имя подчеркивает назначение и логику объекта, например, имена библиотечных функций `sin`, `abs`, `print` и прочих говорят сами за себя. Имена объектов, введенные программистом, подчеркивают их абстрактный смысл, например, `count`, `square`, `point.x`, `point.y` и так далее.

2.6. Операторы

Оператор – это основной элемент языка программирования, представляющий собой законченную фразу и определяющий некоторый логически завершенный этап обработки данных.

Операторы условно можно подразделить на две категории: *исполняемые* – с их помощью реализуется алгоритм решаемой задачи, и *описательные*, необходимые для определения типов пользователя и объявления объектов программы, например, переменных.

Среди исполняемых операторов можно выделить:
- простые операторы (оператор-выражение, операторы передачи управления);
- составные операторы (составной оператор, блок)
- структурированные операторы (условный, выбора, цикла).

Рассмотрим **простые операторы**

Оператор «выражение»

Любое выражение, завершающееся нажатием Enter, рассматривается как оператор, выполнение которого заключается в вычислении выражения.

Примеры:

```
c = a * 1.5      # выражение с операцией присваивания
a *= b + c       # выполняется умножение с присваиванием: a = a * (b + c)
fun(i, k)        # выполняется вызов функции
```

Операция присваивания.

Общий вид:

```
идентификатор = выражение
```

Пример: `x = y + 4  # выражение с операцией присваивания`

Выполняется в два этапа:
а) вычисляется значение выражения, стоящего в правой части от знака присваивания;

б) вычисленное значение присваивается переменной, стоящей в левой части от знака присваивания, при этом предыдущее значение этой переменной теряется.

Рекомендуется строго относиться к типам данных, не смешивать типы в выражениях.

В *сложных операциях присваивания* (`+=`, `*=`, `/=` и т п.) при вычислении выражения, стоящего в правой части, используется и L-значение из левой части. Например, при сложении с присваиванием ко второму операнду прибавляется первый, и результат записывается в первый операнд, то есть выражение `a += b` является более компактной записью выражения `a = a + b.`

Еще один вид операторов, который можно отнести к операторам-выражениям, это *оператор обращения к функции* (вызов функции). Его операндам являются параметры функции.

Формат оператора:

```
имя функции ([список фактических параметров] – может отсутствовать);
```

Например:

```
# вывод данных
print ("Целое = %d, Вещественное = %f\n" % (my_int, my_float))
```

Пробелы в строке текста являются значащими, то есть, если ввести целое 5 и дробное 9.9, то строка вывода будет иметь вид:

```
Целое = 5, Вещественное = 9.900000
```

К операторам передачи управления относятся **continue**, позволяющий перейти к следующей итерации в цикле, break для выхода из цикла и **return** для возврата значения функции и прекращения её вычисления.

2.7. Организация ввода/вывода данных

Ввести данное – означает присвоить произвольное значение переменной во время выполнения программы. **Вывести данное** – означает напечатать на экране значение переменной при выполнении программы.

Ввод данных

Ввод данных можно выполнить с клавиатуры функцией `input()`:

```
m = input([str])
```

При этом на экран будет выведена строка `str`, а переменная `m` получит значение строкового типа, введённое пользователем. Строковый тип может быть преобразован, например, к типу `int` или `float`, если введённое значение – число.

Для ввода нескольких значений можно воспользоваться методом `split()`, который позволяет разбить строку на подстроки (`split` –расщеплять).

Например, для ввода значений параметра α и переменной x можно поступить так:

```
a, x = input('Введите данные (a, x): ').split()
a = float(a)
x = float(x)
```

Используемый разделитель указывается в качестве параметра метода `split()`. Если разделитель не указан, то им будет пробел. При вводе вещественного числа целая часть отделяется от десятичной дроби точкой. Если пользователь не ввел данные (просто нажал Enter) или вместо цифр и точки ввел недопустимые символы, например буквы, программа завершится аварийно на шаге приведения к типу – `float()`.

Исключительная ситуация, которая при этом возникает, может быть обработана с помощью инструкции `try`.

Вывод данных

Вывод данных на экран монитора может быть выполнен функцией `print()`. Эта функция позволяет выполнять форматированный вывод, как с использованием Си-подобного форматирования, так и с использованием форматной строки Python. Следующие строки демонстрируют, как можно форматировать вывод.

```
for x in range(1,11):
     print('%2d %3d %7.2f' % (x, x*x, x*x*x))
print("{0:.2f} {1:.2f} {2:.4f}".format(a, x, y))
```

Буква в формате числа определяет тип выводимого числа. Так, `d` – это целый тип, `f` – вещественное число. Число в формате означает то число позиций, которое будет использовано для вывода числа. Для вещественного числа указывается, после точки, количество выводимых десятичных знаков.

Во второй строке использован Си-подобный формат, в котором формат числа начинается с процента "`%`". В этом формате аргументы отделяются от форматной части строки так же символом `%` – процент. В третьей строке используется форматная строка Python, в которой в форматной строке позиции для значений аргументов выделяются фигурными скобками.

Обратите внимание на то, что сами форматные строки начинаются и завершаются одиночной или двойной кавычкой. В Python допускаются оба вида кавычек для выделения строки. Важно только, чтобы начало и конец были одинаковыми. Так же следует понимать, что в промежутках между символами форматирования могут находиться и другие символы или слова:

```
print('x=%2d x^2=%3d x^3=%7.2f' % (x, x*x, x*x*x))
print("a={0:.2f} x={1:.2f} y={2:.4f}".format(a, x, y))
```

Использование форматных строк делает вывод данных более внятным. За более подробной информацией обращайтесь к учебникам или Интернет.

Вопросы и задания

Задание I

В соответствии с вариантом составить и реализовать программы

Написать программу для расчета по двум формулам. Предварительно подготовить тестовые примеры для второй формулы с помощью калькулятора (результаты вычисления по обеим формулам должны совпадать). Часть математических функция реализована в модуле math.

1.	$z_1 = 2\sin^2(3\pi - 2\alpha)\cos^2(5\pi + 2\alpha);$	$z_2 = \frac{1}{4} - \frac{1}{4}\sin\left(\frac{5}{2}\pi - 8\alpha\right).$
2.	$z_1 = \cos\alpha + \sin\alpha + \cos 3\alpha + \sin 3\alpha;$	$z_2 = 2\sqrt{2}\cos\alpha \cdot \sin\left(\frac{\pi}{4} + 2\alpha\right).$
3.	$z_1 = \frac{\sin 2\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha + 1 - 2\sin^2 2\alpha};$	$z_2 = 2\sin\alpha.$
4.	$z_1 = \frac{\sin\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha - \cos 3\alpha + \cos 5\alpha};$	$z_2 = tg\,3\alpha.$
5.	$z_1 = 1 - \frac{1}{4}\sin^2 2\alpha + \cos 2\alpha;$	$z_2 = \cos^2\alpha + \cos^4\alpha.$
6.	$z_1 = \cos\alpha + \cos 2\alpha + \cos 6\alpha + \cos 7\alpha;$	$z_2 = 4\cos\frac{\alpha}{2} \cdot \cos\frac{5}{2}\alpha \cdot \cos 4\alpha.$
7.	$z_1 = \cos^2\left(\frac{3}{8}\pi - \frac{\alpha}{4}\right) - \cos^2\left(\frac{11}{8}\pi + \frac{\alpha}{4}\right);$	$z_2 = \frac{\sqrt{2}}{2}\sin\frac{\alpha}{2}.$
8.	$z_1 = \cos^4 x + \sin^2 y + \frac{1}{4}\sin^2 2x - 1;$	$z_2 = \sin(y + x) \cdot \sin(y - x).$
9.	$z_1 = (\cos\alpha - \cos\beta)^2 - (\sin\alpha - \sin\beta)^2;$	$z_2 = -4\sin^2\frac{\alpha - \beta}{2} \cdot \cos(\alpha + \beta).$
10.	$z_1 = \frac{\sin\left(\frac{\pi}{2} + 3\alpha\right)}{1 - \sin(3\alpha - \pi)};$	$z_2 = ctg\left(\frac{5}{4}\pi + \frac{3}{2}\alpha\right).$
11.	$z_1 = \frac{1 - 2\sin^2\alpha}{1 + \sin 2\alpha};$	$z_2 = \frac{1 - tg\alpha}{1 + tg\alpha}.$
12.	$z_1 = \frac{\sin 4\alpha}{1 + \cos 4\alpha} \cdot \frac{\cos 2\alpha}{1 + \cos 2\alpha};$	$z_2 = ctg\left(\frac{3}{2}\pi - \alpha\right).$
13.	$z_1 = \frac{\sin\alpha + \cos(2\beta - \alpha)}{\cos\alpha - \sin(2\beta - \alpha)};$	$z_2 = \frac{1 + \sin 2\beta}{\cos 2\beta}.$

14. $z_1 = \dfrac{\cos\alpha + \sin\alpha}{\cos\alpha - \sin\alpha};$	$z_2 = tg2\alpha + \sec 2\alpha.\ \left(\sec x = \dfrac{1}{\cos x}\right).$
15. $z_1 = \dfrac{\sqrt{2b + 2\sqrt{b^2 - 4}}}{\sqrt{b^2 - 4} + b + 2};$	$z_2 = \dfrac{1}{\sqrt{b+2}}.$
16. $z_1 = \dfrac{x^2 + 2x - 3 + (x+1)\sqrt{x^2 - 9}}{x^2 - 2x - 3 + (x-1)\sqrt{x^2 - 9}};$	$z_2 = \sqrt{\dfrac{x+3}{x-3}}.$
17. $z_1 = \dfrac{\sqrt{(3m+2)^2 - 24m}}{3\sqrt{m} - \dfrac{2}{\sqrt{m}}};$	$z_2 = \sqrt{m}.\quad (m > 0)$
18. $z_1 = -2 - \left(\dfrac{a+2}{\sqrt{2a}} - \dfrac{a}{\sqrt{2a}+2} + \dfrac{2}{a - \sqrt{2a}}\right) \cdot \dfrac{\sqrt{a} - \sqrt{2}}{a+2};$	$z_2 = \dfrac{1}{\sqrt{a} - \sqrt{2}}.$ (a > 0)
19. $z_1 = \left(\dfrac{1 + a + a^2}{2a + a^2} + 2 - \dfrac{1 - a + a^2}{2a - a^2}\right)^{-1} (5 - 2a^2);$	$z_2 = \dfrac{4 - a^2}{2}.$
20. $z_1 = \dfrac{(m-1)\sqrt{m} - (n-1)\sqrt{n}}{\sqrt{m^3 n} + nm + m^2 - m};$	$z_2 = \dfrac{\sqrt{m} - \sqrt{n}}{m}.$
21. $z_1 = (1 + tg\alpha)(1 + ctg\alpha) - 2;$	$z_2 = \dfrac{1}{\sin a \cos a}..$
22. $z_1 = 1 + ctg^2\alpha + \dfrac{1}{\cos^2\alpha};$	$z_2 = \dfrac{1}{\sin^2\alpha \cos^2\alpha}.$
23. $z_1 = 1 - \dfrac{1}{1 + ctg^2\alpha};$	$z_2 = \dfrac{1}{1 + tg^2\alpha}.$
24. $z_1 = \dfrac{1 - 2\cos^2\alpha}{\sin\alpha\cos\alpha} + ctg\alpha;$	$z_2 = tg\alpha.$
25. $z_1 = \sin^4\alpha - 1 + 2\sin^2\alpha\cos^2\alpha;$	$z_2 = -\cos^4\alpha.$

Задание II

Выполнить в соответствии с вариантом:

1. Составить программу вычисления объема конуса по заданному диаметру и образующей.
2. Дана сторона равностороннего треугольника. Составить программу нахождения площади этого треугольника и радиуса описанной окружности.
3. Известны радиусы двух концентрических окружностей. Составить программу нахождения площади кольца, образованного этими окружностями.
4. Составить программу нахождения суммы n членов арифметической прогрессии, для которой известен первый член, разность и число n.
5. Вычислить объем призмы, боковые грани которой - квадраты, а основанием служит равносторонний треугольник, вписанный в круг радиуса r.
6. Вычислить площадь прямоугольника, вписанного в окружность радиуса r, если отношение его сторон равно R.
7. Даны две стороны треугольника и угол между ними. Определить третью сторону и площадь.
8. Вычислить процент материала, ушедшего в отходы, если из куба с ребром a был выточен шар радиуса r ($r < a$).
9. Вычислить площадь кольца, ширина которого равна a, а отношение радиусов окружностей равно b.
10. Вычислить периметр и площадь прямоугольного треугольника, описанного около круга радиуса r, если гипотенуза треугольника равна c.
11. Вычислить диаметр трубы, пропускная способность которой позволяет заменить ею две трубы с диаметрами d_1 и d_2.
12. Вычислить высоты треугольника со сторонами a, b и c.
13. Вычислить площадь правильного n-угольника, описанного около круга радиусом r.
14. Высота конуса равна h, а радиус основания r. Вычислить объем шара, вписанного в конус.
15. Вычислить массу свинцовой трубы, длина которой равна b м (плотность свинца равна 11,4 г/см^3), толщина стенок a мм, а внутренний диаметр трубы равен d мм.
16. Дан равносторонний треугольник. Вычислить сторону, высоту и площадь этого треугольника, если радиус вписанной окружности равен r.
17. Стальной вал, имеющий b мм длины и d мм в диаметре, обтачивается на токарном станке, при этом диаметр уменьшается при обточке на S мм. Вычислить, на сколько уменьшается масса тела (плотность стали 7,4 г/см^3).
18. Вычислить объем призмы, боковые грани которой – квадраты. Основанием призмы служит равносторонний треугольник, заданный длиной стороны.
19. В конус с радиусом основания r и высотой h вписан цилиндр, радиус основания которого равен a. Вычислить объем цилиндра.

20. Даны две стороны треугольника d и c и угол между ними α. Найти радиусы окружностей: вписанной в треугольник и описанной около него.
21. Дана длина ребра куба. Найти объем куба и площадь его боковой поверхности.
22. Треугольник задан тремя сторонами. Вычислить его высоты.
23. Определить периметр правильного n-угольника, описанного около окружности радиуса R.
24. Смешано V_1 литров воды температуры T_1 и V_2 литрами воды температуры T_2. Найти объем и температуру образовавшейся смеси.
25. Известна длина окружности. Найти площадь круга, ограниченного этой окружностью.
26. Даны гипотенуза и катет прямоугольного треугольника. Найти второй катет и радиус вписанной окружности.
27. Найти площадь кольца, внутренний радиус которого равен R_1, а внешний - R_2 ($R_2 > R_1$).
28. Определить время, через которое встретятся два тела, равноускоренно движущиеся навстречу друг к другу, если известны их начальные скорости, ускорение, начальное расстояние между ними.
29. Вычислить расстояние между двумя точками с координатами (x_1, y_1) и (x_2, y_2).
30. Даны катеты прямоугольного треугольника. Найти его гипотенузу и площадь.

Задание III

Выполнить в соответствии с вариантом:

1. Радоновые ванны, применяемые для лечения, содержат $1.8 \cdot 10^6$ атомов радона в воде объемом 1.0 дм^3. На сколько молекул воды приходится один атом радона в лечебной ванне?
2. Сколько атомов ртути содержится в воздухе объемом 1.0 м^3 в помещении, зараженном ртутью, при температуре 20°С, если давление насыщенного пара ртути при этой температуре 133 мПа?
3. Какова длина ребра куба, содержащего $1.0 \cdot 10^6$ молекул идеального газа при нормальных условиях?
4. При какой температуре молекулы гелия имеют такую же среднюю скорость, как молекулы кислорода при 23°С?
5. Найдите среднюю кинетическую энергию поступательного движения молекул водорода, если при давлении 0.5 атм. их концентрация равна $1.5 \cdot 10^9$ м$^{-3}$.
6. Газ нагревается в открытом сосуде при нормальном атмосферном давлении от 27°С до 327°С. Какое приращение получает при этом число молекул в единице объема газа?

7. В сосуде объемом 1 дм^3 содержится некоторый газ при температуре 17°С. Найти приращение давления газа, если вследствие утечки газа из него выйдет 10^{21} молекул.
8. В сосуде объемом 3.0 дм^3 находится гелий массой 4.0 мг, азот массой 70 мг и $5.0 \cdot 10^{21}$ молекул водорода. Каково давление смеси, если температура ее 27°С?
9. Определить среднюю кинетическую энергию вращательного движения молекул водорода, содержащихся в 1.0 моль при 18°С.
10. При какой температуре молекулы кислорода имеют такую же среднюю скорость, как молекулы водорода при 25°С?
11. Сколько молекул содержится при нормальных условиях в 1м^3 воздуха?
12. В сосуде объемом 2.0 дм^3 находится газ под давлением 0.50 МПа. Чему равна средняя кинетическая энергия поступательного движения молекул газа?
13. Кислород массой 12 г находится при температуре 700°С, при этом 40% молекул диссоциировано на атомы. Чему равна средняя кинетическая энергия теплового движения частиц? Колебательные степени свободы молекул кислорода не возбуждаются.
14. Объём помещения 50 м^3 . температура воздуха зимой 0°С, а летом – 40°С. Какова разница в массе воздуха, заполняющего помещение зимой и летом? Принять $\mu = 29$ кг/моль.
15. Плотность воздуха при нормальных условиях 1.3 г/л. Какова плотность воздуха при температуре 100°С и давлении $4.0 \cdot 10^5$ Н/м^2?
16. Температура воздуха в баллоне объёмом 10 м^3 при давлении 700 мм рт. ст. была 15°С. После нагрева воздуха до 20°С часть воздуха была вытеснена. Найдите массу вытесненного воздуха.
17. Сколько частиц (атомов и молекул) находится в азоте массой 1.0 г, если степень диссоциации азота 7.0%.
18. Какое давление на стенки сосуда производит кислород, если средняя квадратичная скорость его молекул 400 м/с и концентрация молекул $2.7 \cdot 10^{19}$ м$^{-3}$?
19. Найдите среднюю кинетическую энергию поступательного движения молекул гелия, если при давлении 0.5 атм. их концентрация равна $1.5 \cdot 10^9$ м^3.
20. Определите температуру газа, если средняя кинетическая энергия поступательного движения его молекул равна $1.6 \cdot 10^{-19}$ Дж.
21. Каково давление газа, если в каждом кубическом сантиметре его содержится $1.0 \cdot 10^6$ молекул, а температура газа 87°С?
22. Газ нагревается в открытом сосуде при нормальном атмосферном давлении от 22°С до 320°С. Какое приращение получает при этом число молекул в единице объема газа?
23. В сосуде объемом 15.0 дм^3 находится газ под давлением 0.50 МПа. Чему равна средняя кинетическая энергия поступательного движения молекул газа?

24. При какой температуре молекулы гелия имеют такую же среднюю скорость, как молекулы водорода при 27°С?

25. Газ занимает объём 2 л при давлении $5 \cdot 10^5$ Н/м^2. Определите суммарную энергию поступательного движения молекул газа.

Задания для самостоятельной работы

1. Даны два действительных числа. Найти среднее арифметическое и среднее геометрическое этих чисел.
2. Даны четыре целых числа. Найти среднее арифметическое этих чисел и среднее геометрическое этих чисел.
3. Треугольник задан длинами своих сторон. Используя формулу Герона, найти площадь этого треугольника.
4. Идет *k*-я секунда суток. Определить сколько полных часов *H*, полных минут *M* прошло к этому моменту.
5. Даны два момента времени одних суток: H_1, H_2 - часы, M_1, M_2 - минуты. Определить интервал между этими моментами в часах *H* и минутах *M*.
6. По трем координатам вершин некоторого треугольника найти его высоту.
7. Найти объем и площадь боковой поверхности конуса радиусом *r*, высотой *h* и образующей *l*.
8. Вычислить боковую поверхность пирамиды (пирамида правильная, усеченная), если известны *p* и p_1 - полупериметры нижнего и верхнего оснований и *k* - апофема.
9. Найти объем и полную поверхность полого шара, если известны r_1 и r_2 - радиусы внешней и внутренней шаровых поверхностей.
10. Вычислить площадь поверхности и объем усеченного конуса высотой *h,* радиусами r_1 и r_2, образующей *l*.
11. В шар радиуса *r* вписан конус с углом *a* при вершине в осевом сечении конуса. Определить объем и полную поверхность конуса.
12. Вычислить объем цилиндра, вписанного в правильную шестиугольную призму, у которой каждое ребро равно *a*.
13. Найти площадь равнобокой трапеции с основаниями *A* и *B* и углом α при большем основании *A*.
14. Треугольник задан величинами своих углов и радиусом описанной окружности. Найти, стороны треугольника.
15. Найти объем правильной треугольной пирамиды, если стороны ее основания равны *a* и плоский угол при ее вершине равен α.
16. Вычислить радиус шара, вписанного в правильную четырехугольную пирамиду, если стороны основания пирамиды равны *a* и двухгранный угол при основании равен α.

17. Вычислить площадь круга и равнобедренной трапеции, описанной около него, если периметр трапеции равен *p*, а угол при нижнем основании равен *a*.
18. Вычислить объем призмы, боковые грани которой – квадраты. Основанием призмы служит равносторонний треугольник, по который известно, что радиус вписанной в него окружности равен *r*.

Контрольные вопросы

1. Что такое программа на Python?
2. Какая программа является линейной?
3. Какова структура программы на Python?
4. Что такое идентификатор?
5. По каким правилам создаются идентификаторы в Python?
6. Что такое тип величины?
7. Какие типы величин используются в языке Python?
8. Какие типы величин относятся к простым типам?
9. В каком диапазоне могут изменяться переменные типа **int** и **float**?
10. Как инициируются (задаются) переменные в Python? (Привести пример.)
11. Каково назначение отступов в программе, написанной на Python?
12. Как выполняется команда присваивания?
13. Какая инструкция предназначена для вывода на экран сообщений?
14. Какая инструкция предназначена для ввода данных?
15. Что такое программный модуль? Приведите пример программного модуля.
16. Как сохранить программу? (Устно)
17. Как осуществить интерпретацию текста программы? (Устно)
18. Как осуществить запуск программы? (Устно)
19. Как считать готовую программу с диска? (Устно)

Пример выполнения заданий

Задание I. Напишите программу для расчета по двум формулам. Отсутствующие в библиотеке языка функции выразите через имеющиеся.

$$z_1 = tg\left(x\right) - ctg\frac{7\pi}{6},$$

$$z_2 = tg\frac{7\pi}{8} - \sqrt{3}\cdot tg\frac{\pi}{8}\cdot tg\left(x\right)$$

Решение

1. Математическая модель

Для математических вычислений в Python имеются как встроенные, так и дополнительные функции и методы. Применить дополнительные математические функций можно после подключения модуля `math`.

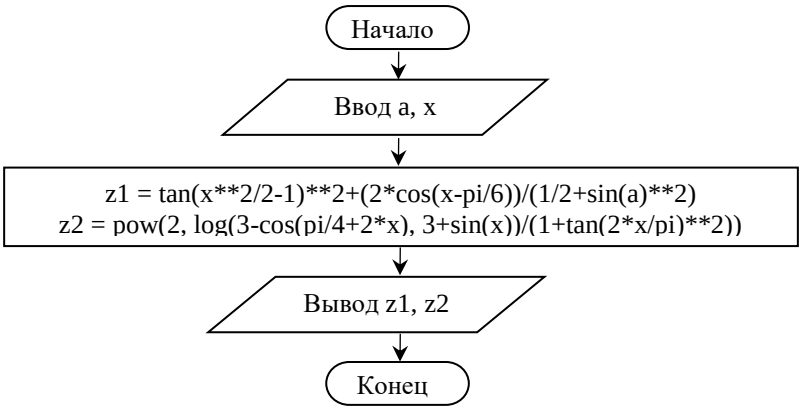
Запись выражений на языке Python примет вид:

```
z1 = tan(x**2/2-1)**2+(2*cos(x-pi/6))/(1/2+sin(a)**2)
z2 = pow(2, log(3-cos(pi/4+2*x), 3+sin(x))/(1+tan(2*x/pi)**2))
```

Аргументы: `a, x` – *вещественного типа.*

Результаты : `z1, z2` – *вещественного типа.*

2. Алгоритм (блок - схема)



3. Программа

```
from math import *
a = float(input('Введите параметр a: '))
x = float(input('Введите значение x: '))
z1 = tan(x**2/2-1)**2+(2*cos(x-pi/6))/(1/2+sin(a)**2)
print("При a = {0:.2f} и x = {1:.2f} \nZ1 = {2:.4f}".format(a, x, z1))
z2 = pow(2, log(3-cos(pi/4+2*x), 3+sin(x))/(1+tan(2*x/pi)**2))
print("Z2 = {0:.4f}".format(z2))
```

4. Результат работы программы

```
'Введите параметр a: -2
'Введите значение x: -2
При a = -2 x = -2
z1 =  1.1970
z2 =  1.1184
```

5. Результаты тестирования программы

a	x	Первое выражение	Второе выражение	
			Программа	Калькулятор
-2	-2	1.1970	1.1184	1.1184
0	-2	-0.8347	1.1184	1.1184
0	0	5.8896	1.6880	1.6880
2	0	3.7309	1.6880	1.6880
1.5	0.5	2.7712	1.7955	1.7955

Примечание: Эта таблица оформлялась в текстовом редакторе вручную.

6. Анализ результатов

При вычислении подобных выражений необходимо анализировать область допустимых значений аргументов, которые используются в выражении. Так, например, знаменатель дроби может получить нулевое значение, и программа прервётся по ошибке деления на ноль. Необходимо учитывать и допустимый диапазон аргументов используемых функций. Так, основание логарифма должно быть больше нуля и не равняться единице, а логарифмируемая функция должна быть больше нуля.

Внимательно следует относиться к выражению, в котором, например, выполняется извлечение квадратного корня или, в общем случае, возведение в степень, показатель которой является не целым числом. В этом случае может быть получен результат в виде комплексного числа или возникнет ошибка, которая приведёт к прерыванию работы программы.

В наших примерах знаменатели $½ + \sin^2\alpha$ и $1 + tg^2(2x/\pi)$ всегда неравны нулю. Кроме этого, во втором примере, основание логарифма не отрицательно и не равно единице, а логарифмируемая функция всегда больше нуля. Следовательно, x может быть любым.

Задание II. Написать программу для вычисления длины окружности и площади круга заданного радиуса.

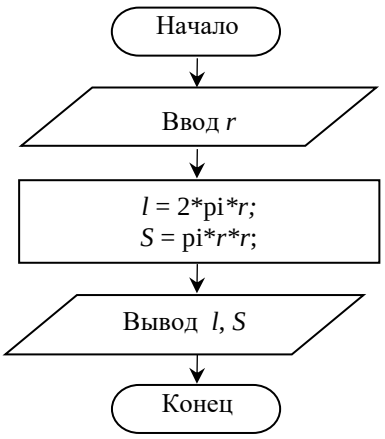
Решение

1. Математическая модель $l = 2 \cdot \pi \cdot r,\ S = \pi \cdot r^2$.

Аргументы: радиус окружности *r*, вещественного типа.

Результаты: длина окружности *l* и площадь круга *S*, вещественного типа.

2. Блок – схема



3. Программа

```
from math import pi

r = float(input("Введите радиус: r = "))
l = 2 * pi * r
S = pi * r ** 2

print("Длина окружности: l = ", l)
print("Площадь круга: S = ", S)
```

4. Результат работы программы:

```
Введите радиус: r = 5
Длина окружности: l = 31.4
Площадь круга: S = 78.5
```

Список литературы, рекомендуемый к использованию по данной теме

- [1] стр. 37- 57, 65-78, 83-89;
- [4] стр. 362-389;
- [6] стр. 32-41;
- [11] Теоретический материал: Интерактивный калькулятор (informatics.msk.ru), Теоретический материал: Запуск простейшей программы (informatics.msk.ru) - https://informatics.msk.ru/mod/book/view.php?id=3912

Практическое занятие № 3
Программирование алгоритмов разветвляющейся структуры

ЦЕЛЬ: приобретение навыков программирования вычислительных разветвляющихся процессов с использованием условного, составного операторов, оператора множественного ветвления.

Знания, умения и владения, приобретаемые обучающимся в результате освоения темы, в рамках формируемых компетенций (ПК-1: ИД-1$_{ПК-1}$, ИД-2$_{ПК-1}$**;** ПК-2: ИД-1$_{ПК-2}$, ИД-2$_{ПК-2}$): см. приложение 2: вырабатывается навык разработки математических и информационных моделей, а также формируется способность к разработке и применению алгоритмических и программных решений (разветвляющейся структуры) в области прикладного программирования с умением объяснить теоретическую и практическую часть темы (формируется часть указанных компетенций).

Теоретическая часть
3. Операторы ветвления

3.1. Простые и составные условия

В Python условие – выражение логического типа (**`bool`**), которое может принимать одно из двух значений: «истина» (**`True`**) или «ложь» (**`False`**). Класс (тип) **`bool`** наследуется от **`int`**. При приведении к логическому типу нулевое значение любого численного типа, **`None`**, а также пустая строка или не содержащий элементов итерируемый объект (`list(), tuple()`, `set()`, `dict()`, `range(0)`) вернёт **`False`**. Любое другое значение интерпретируется как **`True`**. При преобразовании к **`int`** или **`Decimal`** **`True`** возвращается значение `1`, при приведении к **`float`** – `1.0`. Для **`False`** логика та же, но значения нулевые. Преобразование **`True`** и **`False`** в строку возвращает `'True'` и `'False'` соответственно.

Используются шесть операторов, позволяющих сравнивать между собой значения выражений, числовых переменных, а также значение переменной и константу:

> (больше),	< (меньше),
== (равно),	!= (не равно),
>= (больше либо равно),	<= (меньше либо равно).

Также в Python есть дополнительные операторы для проверки принадлежности или идентичности:

`in` (принадлежит), `not in` (не принадлежит), `is` (является), `is not` (не является)

`is, is not` проверяет через `id()`

`in, not in` проверяет по значению

Условия, которые составлены с использованием одного оператора сравнения, называются **простыми условиями, например, x + y > 0**.

выражение **оператор сравнения** выражение

Из простых условий, которые являются выражениями логического типа можно строить **сложные условия**, применяя к ним, как к операндам, логические операторы: `not` (не), `and` (и), `or` (или).

Логические операции выполняются слева направо. Если значение первого операнда достаточно, чтобы определить результат операции, второй операнд не вычисляется. Для изменения порядка действий используются круглые скобки.

С**оставные условия** – это простые условия, связанные при помощи логических операторов: `x > 5 or a + b < 0.`

Ниже приведена таблица приоритета операторов, чем ниже в таблице оператор, тем более он приоритетен (значит, выполняется раньше).

Оператор	Описание
`:=`	Оператор присваивания
`lambda`	Lambda-выражение
`if – else`	Условный оператор
`or`	Логическое ИЛИ
`and`	Логическое И
`not x`	Логическое НЕ
`in, not in, is, is not, <, <=, >, >=, !=, ==`	Сравнения, включая проверку принадлежности и идентичности
`|`	Побитовое ИЛИ
`^`	Побитовое исключающее ИЛИ
`&`	Побитовое И
`<<, >>`	Сдвиги
`+, -`	Сложение и вычитание
`*, @, /, //, %`	Умножение, умножение матриц, деление, целочисленное деление, взятие остатка (% также используется для форматирования строк, в этом случае приоритет этой операции такой же)
`+x, -x, ~x`	Положительное, отрицательное, побитовое НЕ
`**`	Возведение в степень (важно отметить, что оператор возведения в степень связывает в меньшей степени, нежели арифметический или побитовый унарный оператор справа от него, то есть `2 ** -1` вернёт `0.5`)
`await x`	Выражение `await` (ожидание выполнения асинхронной функции)
`x[index], x[index:index], x(arguments...), x.attribute`	Обращение по индексу или ключу, взятие среза, вызов, взятие атрибута

Операция and

Логическую операцию **and** (и) чаще всего, если нужно, чтобы одновременно выполнялись два условия. В Python это записывается так:

объект 1 **and** объект 2

Данная операция возвращает первый объект в случае его ложности, иначе второй. Это работает не только с условиями, то есть 3 and 5 равно 5, а 0 and 5 равно 0.

На практике эта логическая операция часто применяется для создания условия принадлежности некоторой переменной указанному промежутку. Например, составим условие, которое будет истинно тогда и только тогда, когда переменная x принадлежит промежутку от 10 до 20: $10 < x < 20$. Т.е., когда $x > 10$ ***и*** $x < 20$. На языке Python это записывается так: `x > 10 and x < 20`. Также можно записать без **and**: `10 < x < 20`

Операция or

Логическую операцию **or** (или) чаще всего используют, когда хотят сформулировать условие, которое будет истинно в том случае, когда верно хотя бы одно условие из двух. В Python эта операция используется так:

объект1 **or** объект2

Данная операция возвращает первый объект в случае его истинности, иначе первый. Это работает не только с условиями, то есть 3 or 5 равно 3, а 0 and 5 равно 5.

Операция not

Логическую операцию **not** (не – логическое отрицание) используют, если нужно проверить условие (выражение) на ложность. Эта операция делает ложным истинное условие и истинным ложное. Запись операции:

not условие

В таблице 6 приводятся результаты применения логических операторов к операндам логического типа. Для наглядности вместо значения `False` используется 0, `True` – 1.

Таблица 6 – Логические операции

a	b	a and b	a **or** b	not a
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

При записи сложных условий надо учитывать приоритет операций и, в случае необходимости, использовать круглые скобки.

3.2. Составной оператор

Составной оператор – последовательность операторов, выделяемая табуляцией. Операторы, входящие в него, должны выполняться в том порядке, в котором они записаны.

Формат оператора:

```
#some code
    оператор1
    оператор2
       ...
    операторN
#some more code
```

Составной оператор (блок) используется в любом случае, когда несколько операторов следует объединить в один. Это бывает необходимо в условных операторах и операторах цикла, согласно синтаксису которых, исполнимым является только один оператор. Чаще всего, составной оператор формирует ветвь условного оператора или тело цикла в операторах цикла. Блоки, чаще всего, используются в качестве тела функции. В Python в некоторых случаях можно не делать табуляцию и писать в той же строке, например, в циклах или создании функций, но есть ограничения, например с условными конструкциями – нельзя писать `if`-ы последовательно, нельзя в той же строке писать `else`.

3.3. Условное выражение (тернарный оператор)

Эта операция тернарная, то есть имеет три операнда.
Формат операции:

```
операнд_1 if операнд_2 else операнд_3
```

Второй операнд оценивается. Если результат вычисления операнда 2 равен **True**, то результатом условной операции будет возвращение значения первого операнда, иначе – третьего операнда. Вычисляется всегда либо первый операнд, либо третий. Их тип может различаться. Условное выражение является сокращенной формой условного оператора **`if`**.

Пример: вывести на экран минимальное из двух значений.

```
a = 10
b = 5
min = b if b < a else a
print('Наименьшее число:', min)
```

Результат работы программы:

Наименьшее число: 5

Другой пример применения условной операции. Требуется, чтобы некоторая целая величина увеличивалась на 1, если ее значение меньше n, а иначе принимала значение 1:

```
i = (i + 1) if i < n else 1
```

3.4. Условный оператор if

Предназначен для выбора к исполнению одного из двух возможных операторов в зависимости от выполнения некоторого условия.

Формат оператора:

```
if  условие:
   оператор 1
else:
   оператор 2
```

Инструкция выполняется следующим образом:

1. Вычисляется значение условия (выражения), которое может иметь арифметический тип или тип указателя.
2. Если значение выражения (условия) не равно нулю (имеет значение `true`), выполняется первый оператор, следующий за условием. Если значение выражения (условия) равно `false`, то выполняются инструкции, следующие за словом `else`.

Довольно часто в случае ложности условия не нужно производить никаких действий. Допускается использование краткой формы условного оператора:

```
if  условие:
    оператор
```

Если значение логического выражения (`условия`) истинно, то выполняются инструкции, следующие за условием. Если значение выражения ложно, то выполняется инструкция, следующая непосредственно после оператора `if`.

Если после условия или служебного слова `else` требуется выполнить несколько операторов, то их оформляют в виде составного оператора.

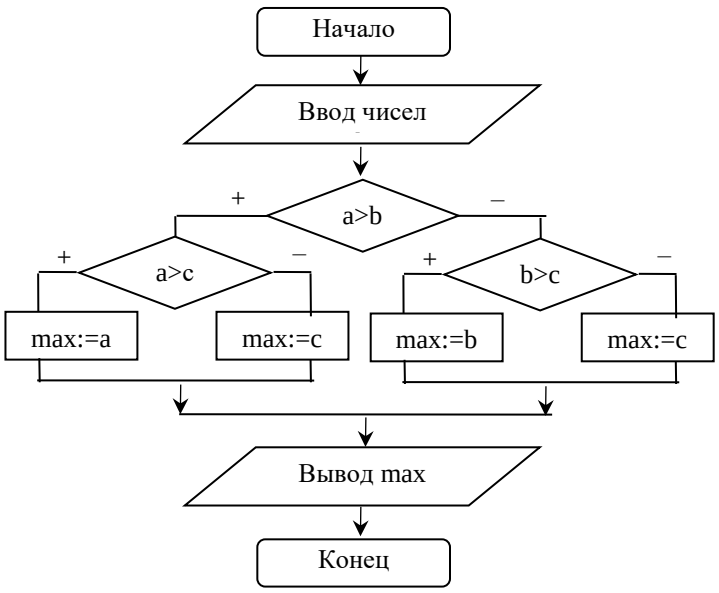
Таблица 7 – Графическая интерпретация условного оператора.

Неполная форма записи		**Полная форма записи**	
если-то	a) + условие – оператор	**если-то-иначе**	b) условие – оператор 1 оператор 2

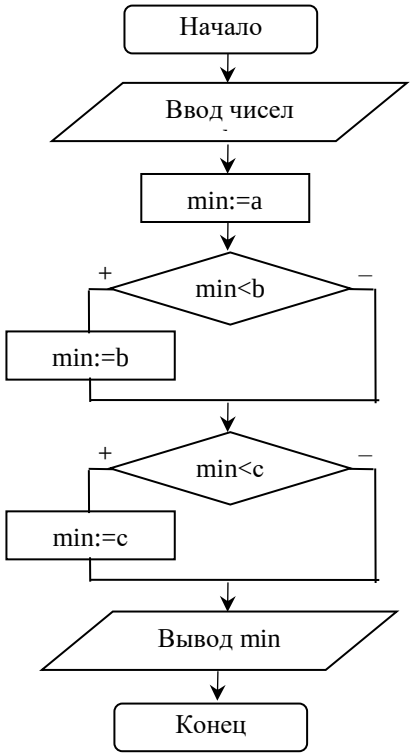
Структура команды «Ветвление» называется вложенной, если после условия или служебного слова **`else`** используются вновь условные операторы. Число вложений может быть произвольным. При этом справедливо следующее правило: служебное слово **`else`** всегда относится к ближайшему **`if`**.

Пример. Составить программу нахождения минимального из трех чисел, введенных с клавиатуры.

1 вариант решения:



2 вариант решения:



Программа на Python

```
//min_of_3_numbers;
a, b, c = (int(x) for x in input('Введите 3 числа
через пробел: ').split())
if a < b:
    if a < c:
        min = a
    else:
        min = c
else:
    if b < c:
        min = b
    else:
        min = c
print('Минимальное число =', min)
```

```
//min_of_3_numbers;
a, b, c = (int(x) for x in in-
put('Введите 3 числа через пробел:
').split())
min = a
if b < min:
    min = b
if c < min:
    min = c
print('Минимальное число =', min)
```

Результат работы программы:

Введите 3 числа через пробел:
12 5 -7
Минимальное число = -7

3.5. Каскадные условные инструкции

Если в программе нужно реализовать выбор из более, чем двух вариантов, то можно использовать инструкцию множественного ветвления, которая в языке Python реализуется при помощи каскадной условной инструкции.

Формат оператора:

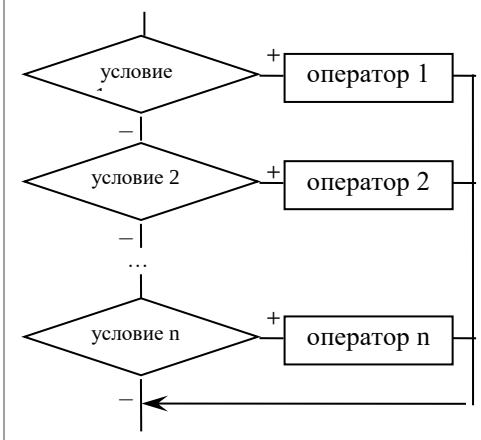
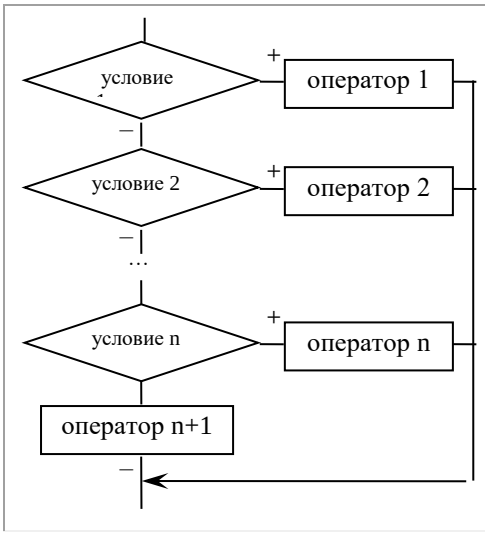
```
if условное выражение 1:
  оператор_1
elif условное выражение 2:
  оператор_2
  …
elif условное выражение n:
  оператор_n
else:
  оператор_n+1
```

Каскадный оператор выбора является обобщением условного оператора. В такой конструкции условия **`if, ..., elif`** проверяются по очереди, выполняется блок, соответствующий первому из истинных условий. Если все проверяемые условия ложны, то выполняется блок **`else`**, если он присутствует.

Затем управление передаётся оператору, следующему за каскадной условной инструкцией.

Начиная с версии 3.10 в Python реализована конструкция match, работающая аналогично switch из C++, но с некоторыми особенностями. Подробнее о данной конструкции можно узнать в документации.

Таблица 8 – Оператор выбора (блок-схема)

Неполная форма записи	**Полная форма записи**
выбор 	**выбор-иначе** 

Вопросы и задания

Задание I

В соответствии с вариантом составить и реализовать программы

Решить задачу с использованием конструкции ветвления:

1. Расстояние от дома до школы *S* км. До начала урока осталось *t* минут. Ученик идет со скоростью *v*. Определить, придёт ли он раньше звонка, после звонка или во время.
2. Даны действительные числа *x*, *y* и *z*. Найти min(*x*, *y*, *z*).
3. Вычислить значение функции $f = \begin{cases} 2x\sqrt{x}, \text{если } x > 1, \\ \dfrac{x^2}{5}, \text{если } x \le 0. \end{cases}$
4. Возвести в квадрат большее из трёх заданных чисел.
5. Определить, есть ли среди трёх заданных чисел чётные.
6. Вывести на экран три заданных числа в порядке убывания.
7. Определить, принадлежит ли точка $A(a_1, a_2)$ кольцу, определяемому окружностями: $x^2 + y^2 = 1$ и $x^2 + y^2 = 16$.
8. Вычислить значение функции $f = \begin{cases} \ln^3\left(x^2+1\right), \text{если } x \le -3, \\ arctg\dfrac{1}{2x+1}, \text{если } x > 3. \end{cases}$
9. Две окружности заданы координатами центра и радиусами. Сколько точек пересечения имеют эти окружности?
10. В два сосуда, один из которых имеет форму шара радиуса *R*1, а другой – форму куба с ребром *A*, доверху налита вода. Сравните количество воды в сосудах.
11. Вычислить значение функции: $f = \begin{cases} x\sin^2 x, \text{если } x \le 1, \\ e^x, \text{если } x > 2. \end{cases}$
12. Даны три произвольных действительных числа. Можно ли построить треугольник с такими длинами сторон?
13. Какая из точек $A(a_1, a_2)$ или $B(b_1, b_2)$ находится дальше от начала координат?
14. Две прямые заданы уравнениями: $a_1x + b_1y + c_1 = 0$ и $a_2x + b_2y + c_2 = 0$. Определить, пересекаются ли они.
15. Попадёт ли тока $A(a_1, a_2)$ в окружность заданного радиуса с центром в начале координат?

16. Вычислить значение функции $f = \begin{cases} ax^2 + bx + c, \text{ если } x < 1.2, \\ \frac{a}{x} + \sqrt{x^2 + 1}, \text{ если } x = 1.2, \\ \frac{a + bx}{\sqrt{x^2 + 1}}, \text{ если } x > 1.2. \end{cases}$

где $a = 2.8$, $b = -0.3$, $c = 4$.

17. В заборе выпилена дыра прямоугольной формы с известными размерами. Определить, пройдёт ли в эту дыру мяч заданного радиуса.

18. Две окружности заданы координатами центра и радиусами. Определить, пересекаются ли они, касаются друг друга или не имеют общих точек.

19. В григорианском календаре каждый год, номер которого делится на 4, является високосным, за исключением тех, которые делятся на 100 и не делятся на 400 нацело. Определить число дней в году по номеру года. Т.о. 1900 г. - не високосный, 2000 г. – високосный.

20. Определить, есть ли среди четырёх заданных чисел кратные пяти.

21. Вычислить значение функции $f = \begin{cases} at^2 \ln t, \text{ если } 1 \le t \le 2, \\ 1, \text{ если } t < 1, \\ e^{at} \cos bt, \text{ если } t > 2. \end{cases}$ где $a = -0.5$, $b = 2$.

22. Из трех действительных чисел a, b и c выбрать те, модули которых больше 4.

23. Даны три произвольных действительных числа. Можно ли построить треугольник с такими длинами сторон? Если да, то вывести на экран является ли заданный треугольник равносторонним, равнобедренным или разносторонним.

24. Даны действительные числа x и y. Найти $U = \max^2 (x^2y, xy^2)$.

25. Решить квадратное уравнение.

Задание II

Решить задачу с использованием конструкции выбора:

1. Дано число c. Распечатать величину этого числа в словесной форме, учитывая его знак. Предусмотреть, что $-7 \le c \le 7$.

2. Дано число m ($1 \le m \le 12$).Определить, сколько дней в месяце с номером m. Год не високосный.

3. С клавиатуры вводится натуральное число n – количество граней выпуклого многогранника. В зависимости от значения n вывести на экран фразу «Выпук-

лый многогранник имеет n граней», согласовав окончание слова "грань" с числом n. Предусмотреть, что $0 \le n \le 25$. Если выпуклого многогранника с заданным количеством граней не существует, выдать соответствующее сообщение.

4. С клавиатуры вводится цифра m (от 1 до 4). Вывести на экран названия месяцев, соответствующих времени года с номером m (считать зиму временем года № 1).

5. С клавиатуры вводится натуральное число n – количество граней выпуклого многогранника. В зависимости от значения n вывести на экран в словесной форме название правильного выпуклого многогранника или сообщить, что правильного выпуклого многогранника с таким количеством граней не существует.

   Предусмотреть, что $4 \le n \le 20$.

   Примечание. Существует пять выпуклых правильных многогранников: тетраэдр (4 грани); гексаэдр или куб (6 граней); октаэдр (8 граней); додекаэдр (12 граней); икосаэдр (20 граней)

6. Дано натуральное число $n < 25$. Вывести на экран фразу «мне n лет (год или года)» соответственно.

7. Найти значение функции: $f = \begin{cases} 2\sin^2 x, \text{ если } -1 \le x \le 1, \\ 3x^3, \text{ если } -10 < x < -1, \\ 0 \text{ в остальных случаях.} \end{cases}$

8. Дано число m ($1 \le m \le 12$).Определить, к какому времени года относится месяц с номером m.

9. Вводится целое число. Если $C > 0$, то вывести на экран название этого числа в словесной форме. В противном случае сообщить, что число отрицательное.

10. С клавиатуры вводится натуральное число n – коли$|C| < 5$чество углов многоугольника. В зависимости от значения n вывести на экран название многоугольника или сообщить, что многоугольника с таким количеством углов не существует.

    Предусмотреть, что $0 \le n \le 6$.

11. С клавиатуры вводится натуральное число n. В зависимости от значения остатка r при делении числа n на 7 вывести на экран число n в виде $n = 7k + r$, где r представить в словесной форме.

12. Дано число m. Определить полугодие, на которое приходится месяц m и количество дней в этом месяце. Предусмотреть, что $1 \le m \le 12$. Год не високосный.

13. Дано число n. Напечатать фразу "Мы успешно сдали n экзаменов", согласовав окончание слова "экзамен" с числом n. Предусмотреть, что $1 \le n \le 25$.

14. Определить время года, к которому относится месяц m и определить количество дней в этом месяце. Предусмотреть, что $1 \le m \le 12$. Год високосный.

15. Дано число c. Распечатать величину этого числа в словесной форме. Предусмотреть, что $0 \le c \le 7$.

16. Даны два числа D – день и M – месяц. Определить K – порядковый номер того дня високосного года, который имеет дату D и M.

17. Даны три числа D, M и G, определяющие день, месяц и год. Проверить образуют ли они правильную дату и вывести соответствующее сообщение. Например, 31.06.90 - неправильная дата. Год не високосный.

18. С клавиатуры вводится цифра от 0 до 9. Вывести на экран название этой цифры в словесной форме.

19. В понедельник фирма работает с 9-00 до 16-00; во вторник, среду, четверг, пятницу – с 8-00 до 19-00; в субботу – с 10-00 до 15-00; воскресенье – выходной. По заданному номеру дня недели определить часы работы.

20. Найти значение функции: $y = \begin{cases} x\sqrt{x-a^2}, \text{если } x = a+1, \\ \dfrac{a}{x} - \dfrac{1}{2}\cos 2x, \text{если } 0 \le x \le a, \\ e^x, \text{если } -a \le x < 0, \\ 1 \text{в иных случаях.} \end{cases}$ где $a = 4$.

21. С клавиатуры вводится натуральное число n. В зависимости от значения остатка r при делении числа n на 5 вывести на экран число n в виде $n = 5k + r$, где r представить в словесной форме.

22. На первом курсе студенты четырёх групп сдают зачёт по информатике. На втором курсе группа № 1 сдаёт экзамен по дисциплине «Информационные технологии в экономике», группа № 2 – экзамен по дисциплине «Информационные технологии в физике», группа № 3 – экзамен по дисциплине «Информационные технологии в географии», группа № 4 – экзамен по дисциплине «Информационные технологии в математических исследованиях». Задать с клавиатуры номер курса и номер группы и вывести на экран сообщение, какой зачёт или экзамен группа будет сдавать.

23. С клавиатуры задано целое число $N < 5$. Если оно положительное, то вывести на экран его название в словесной форме. В противном случае найти квадрат этого числа.

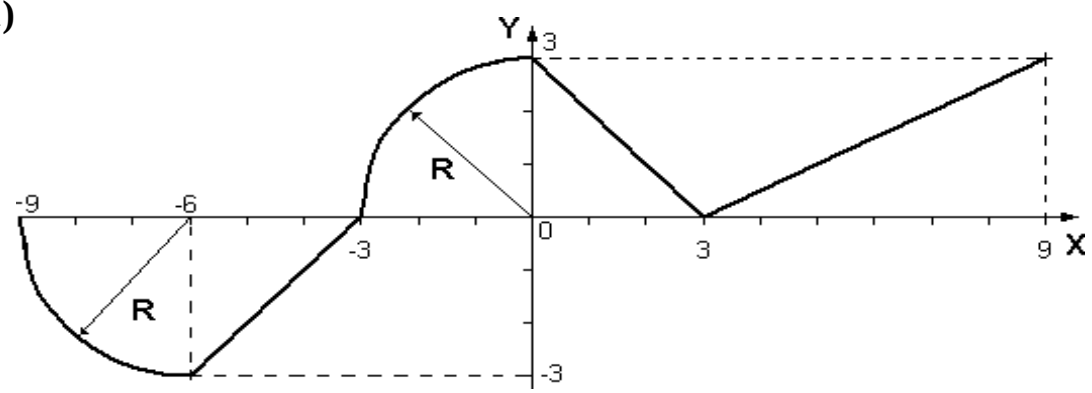
24. Студенты убирают яблоки. Нормы следующие: студент 1 курса должен собрать в день не менее 50 кг яблок, студент второго курса – не менее 60 кг в день, студент 3 курса – не менее 70 кг в день. Зная курс, на котором учится студент, определить, выполнил ли он дневную норму.

25. Дано число m ($1 \le m \le 12$). Определить, сколько дней в месяце с номером m. Учесть високосный или не високосный год.

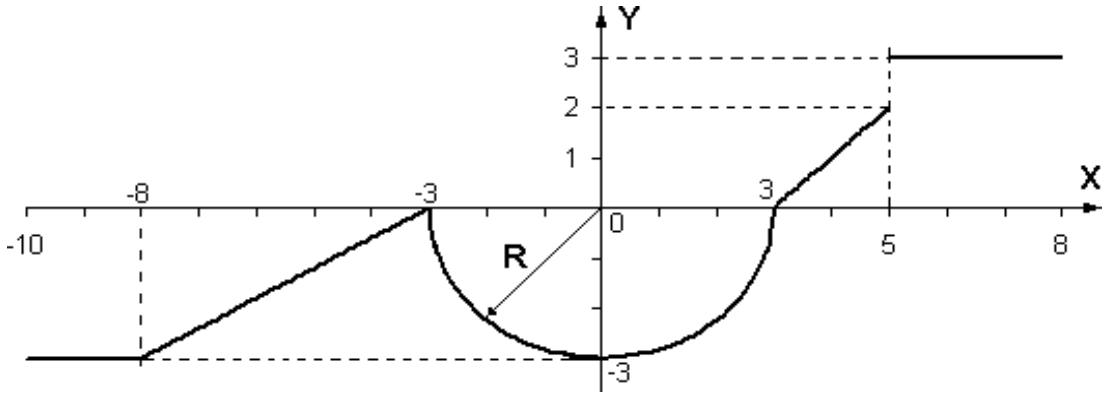
Задание III

Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика. Параметры, необходимые для решения задания следует получить из графика и определить в программе. График функции выбрать согласно варианту.

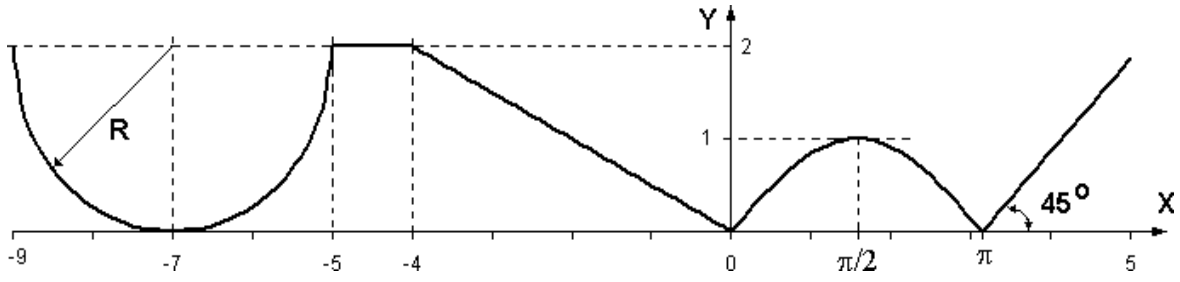
1)



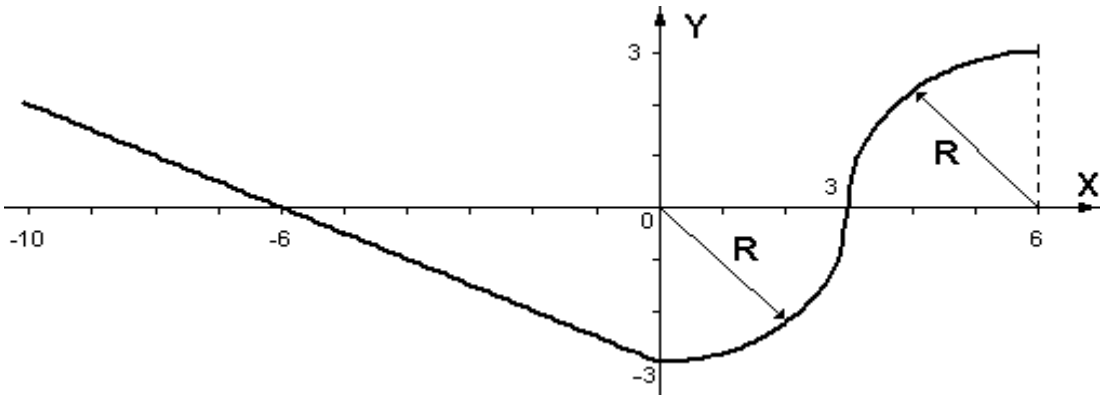
2)



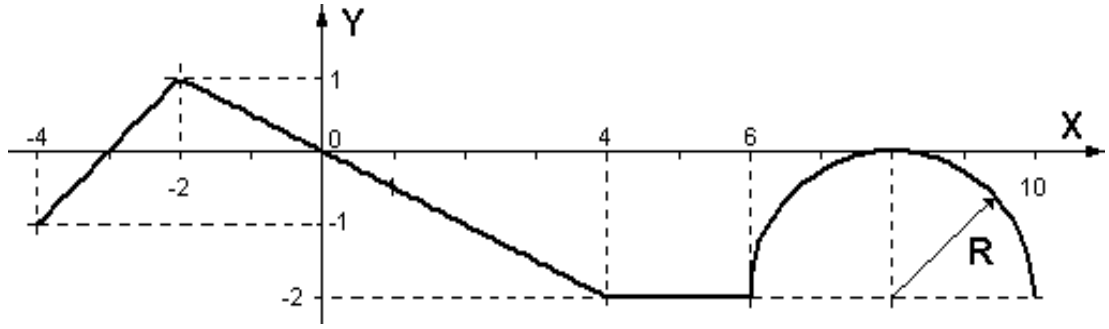
3)



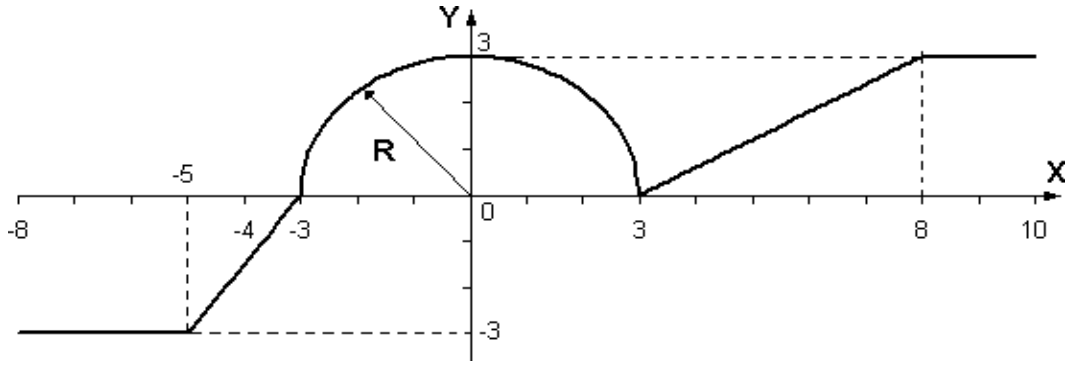
4)


Y
X
3
-3
0
-10
-6
3
6
R
R

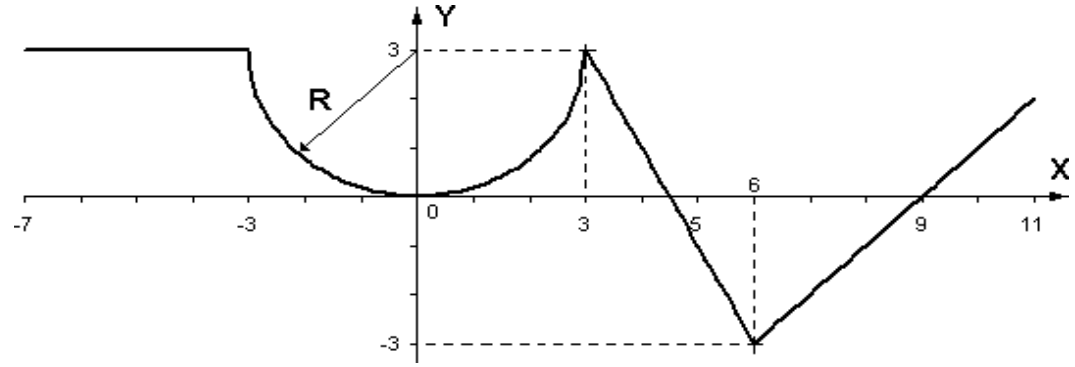
5)


Y
X
1
-1
-2
0
-4
-2
4
6
10
R

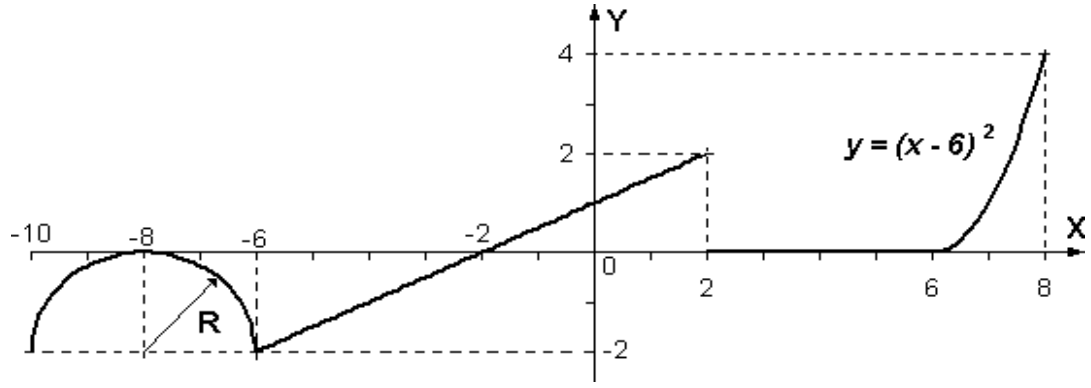
6)


Y
X
3
-3
0
-8
-5
-4
-3
3
8
10
R

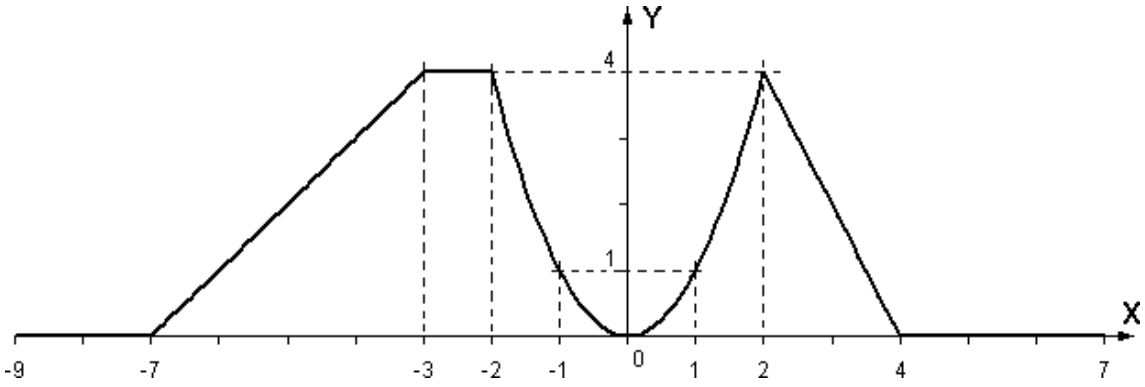
7)


Y
X
3
-3
0
-7
-3
3
5
6
9
11
R

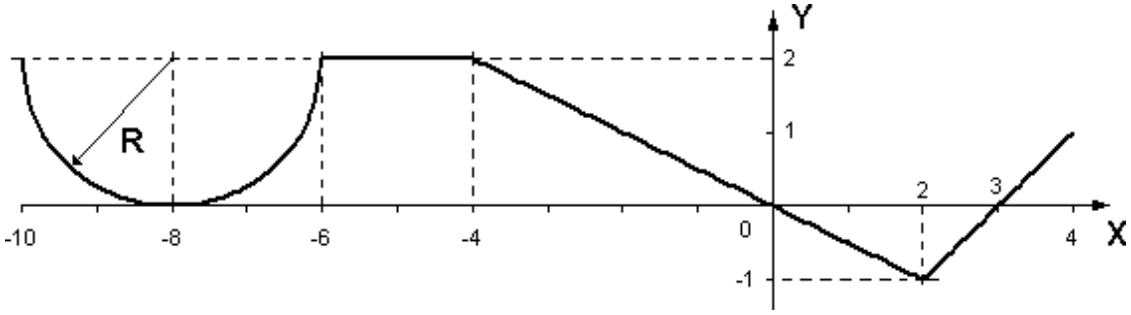
8)


Y
X
4
2
-2
0
-10
-8
-6
-2
2
6
8
R
y = (x - 6) 2

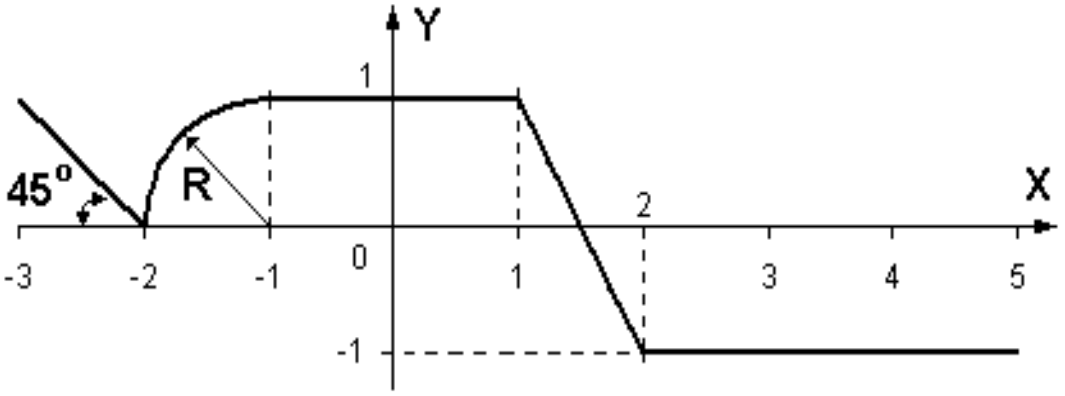
9)


Y
X
4
1
-9
-7
-3
-2
-1
0
1
2
4
7

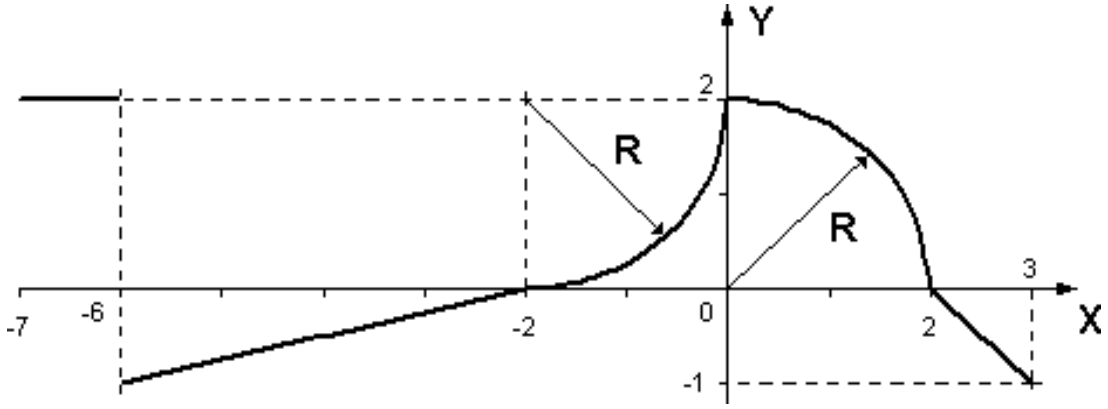
10)


Y
X
R
2
1
-1
-10
-8
-6
-4
0
2
3
4

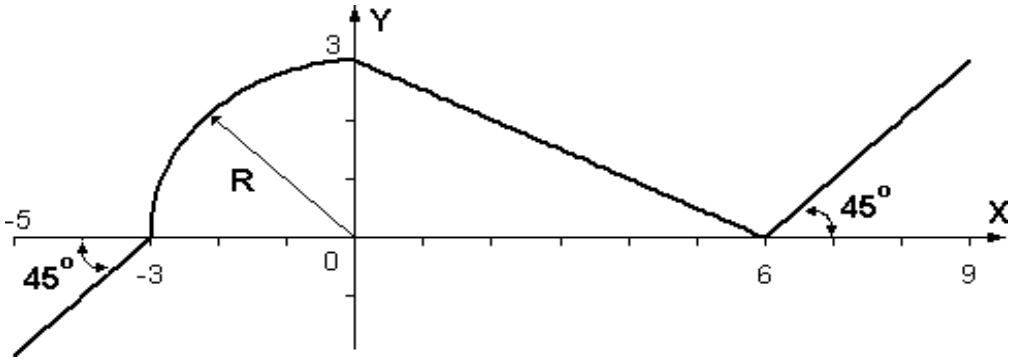
11)


Y
X
45°
R
1
-1
0
-3
-2
-1
1
2
3
4
5

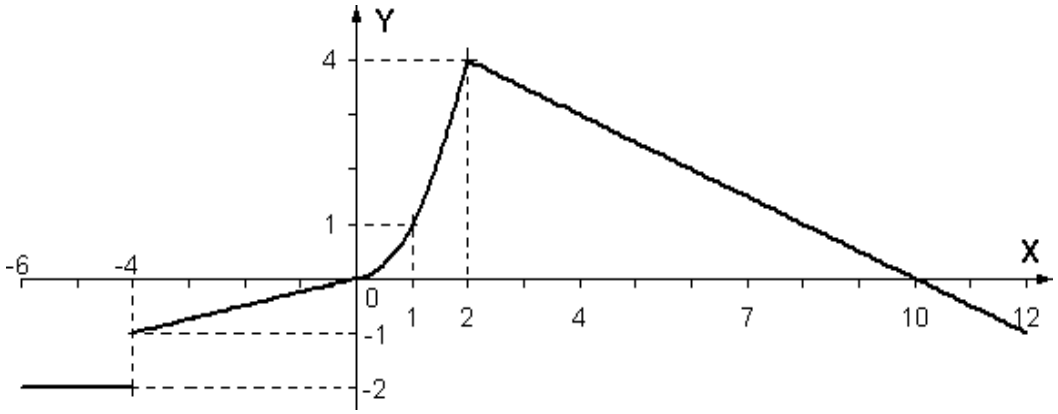
12)


Y
X
R
R
2
-1
0
-7
-6
-2
2
3

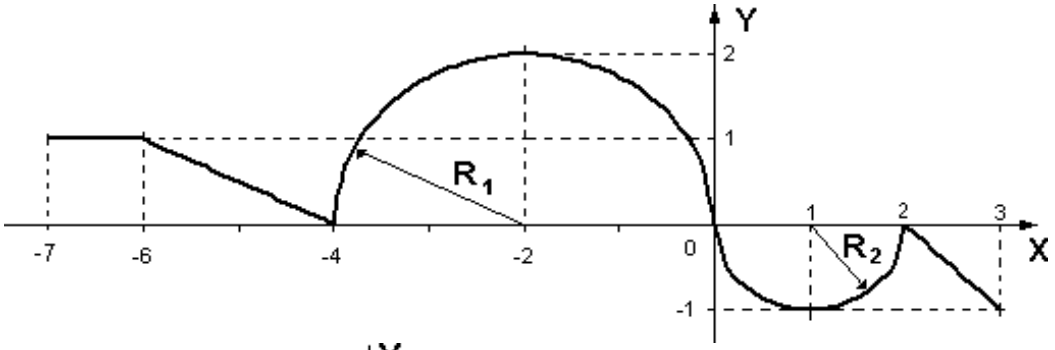
13)


Y
3
R
-5
45°
-3
0
6
9
45°
X

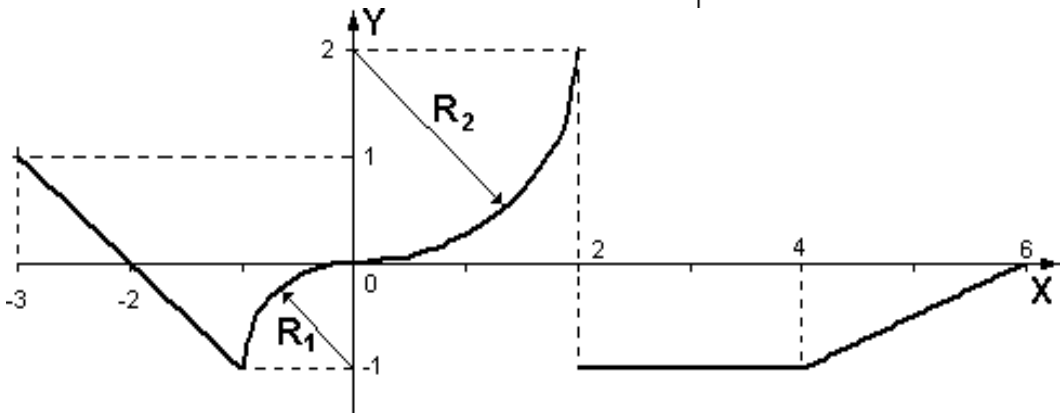
14)


Y
4
1
-6
-4
0
1
2
4
7
10
12
X
-1
-2

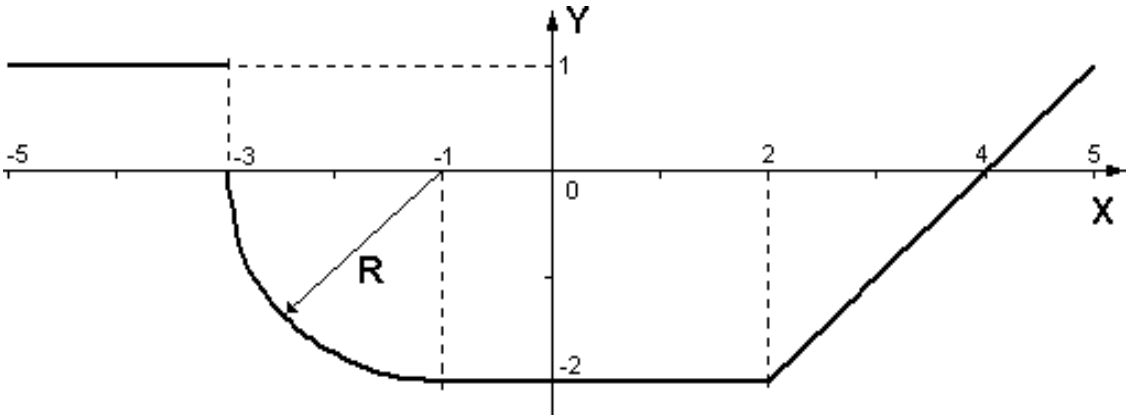
15)


Y
2
1
R1
-7
-6
-4
-2
0
1
2
3
R2
X
-1

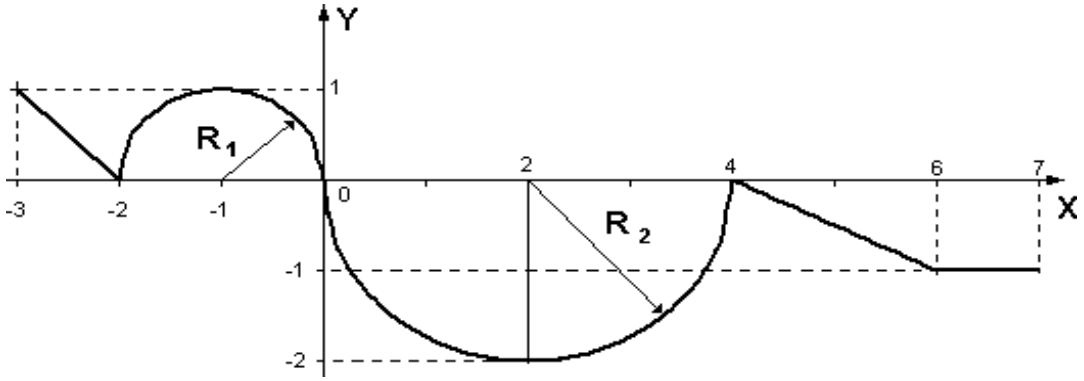
16)


Y
2
R2
1
2
4
6
-3
-2
0
X
R1
-1

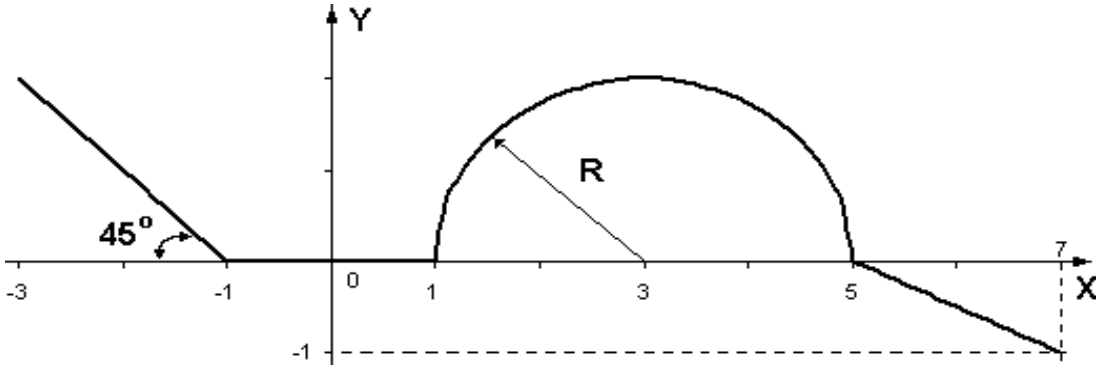
17)


Y
1
-5
-3
-1
0
2
4
5
X
R
-2

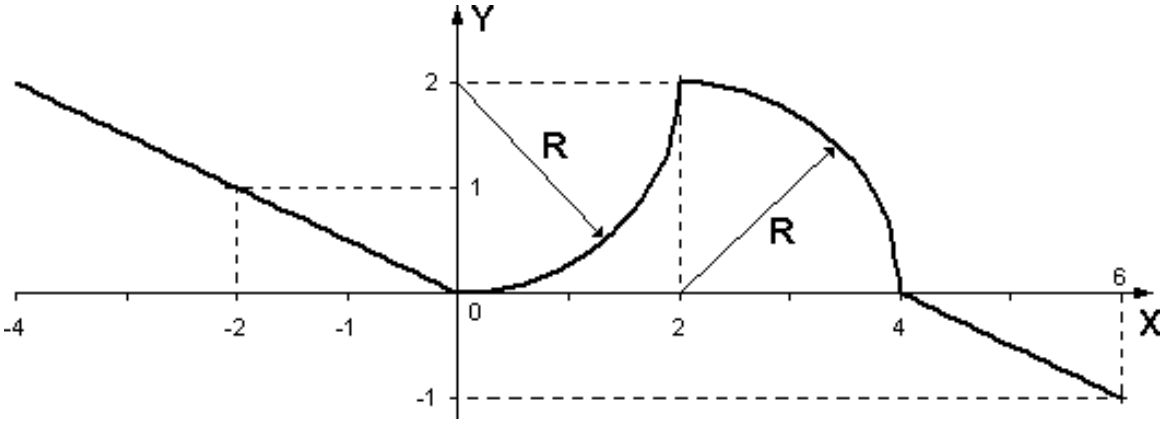
18)


Y
1
R 1
-3
-2
-1
0
2
4
6
7
X
R 2
-1
-2

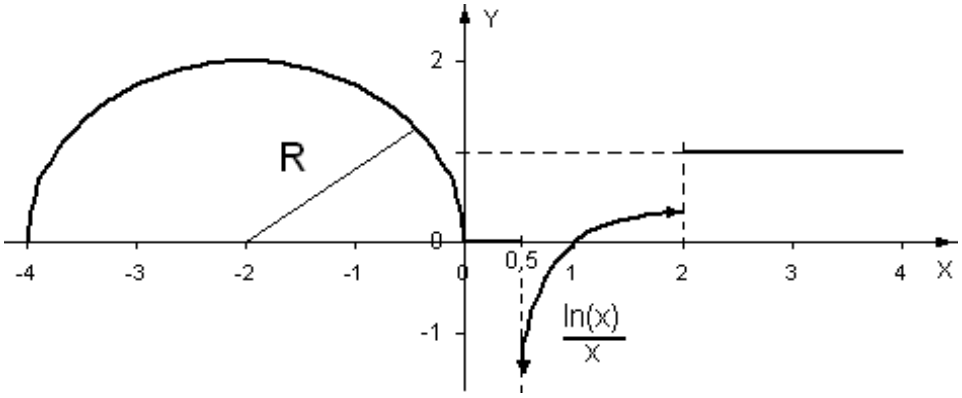
19)


Y
R
45°
-3
-1
0
1
3
5
7
X
-1

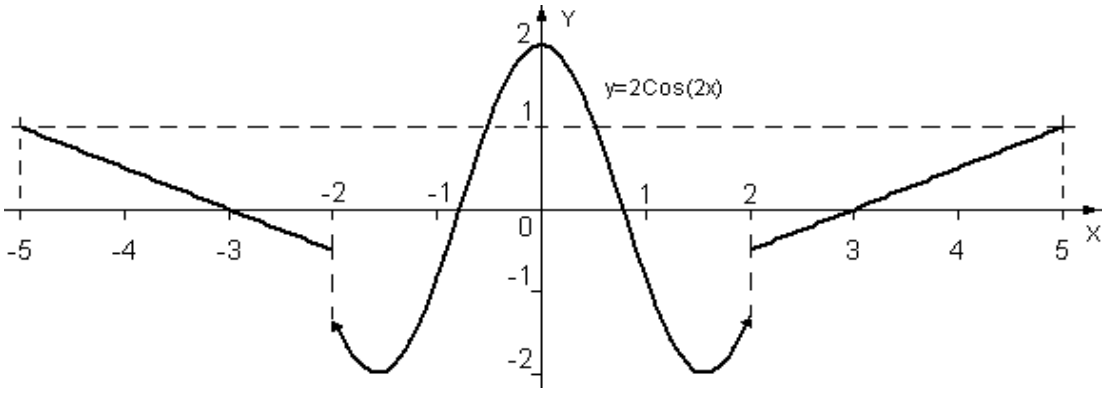
20)


Y
2
R
1
R
6
-4
-2
-1
0
2
4
X
-1

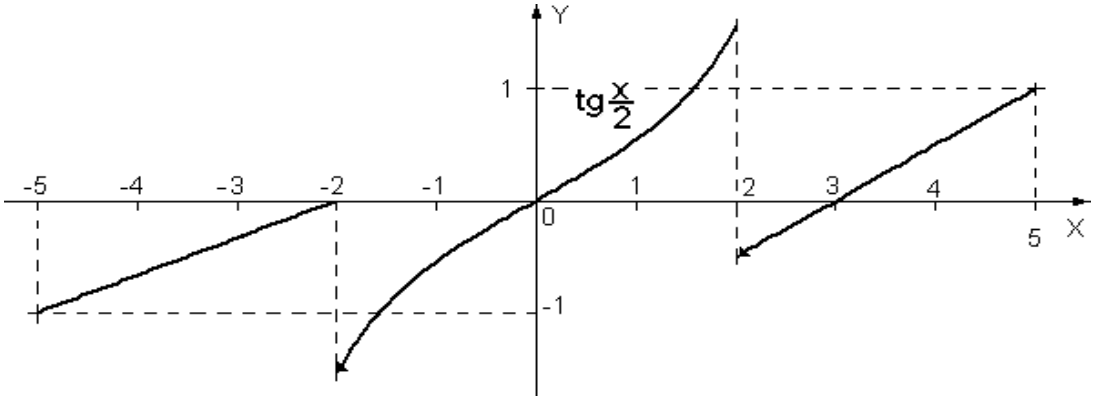
21)


Y
2
R
0
-4
-3
-2
-1
0
0,5
1
2
3
4
X
-1
ln(x)
x

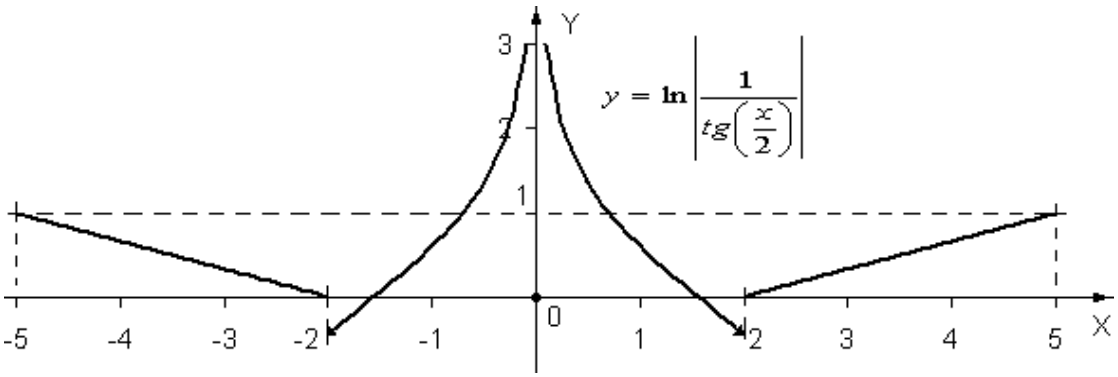
22)


Y
2
y=2Cos(2x)
1
-2
-1
1
2
0
-5
-4
-3
3
4
5
X
-1
-2

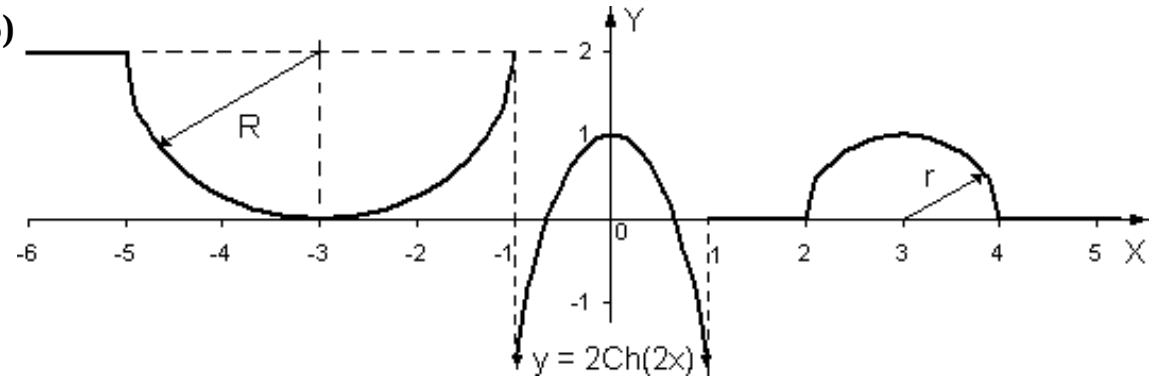
23)


Y
1
tg x/2
-5
-4
-3
-2
-1
1
2
3
4
0
5
X
-1

24)

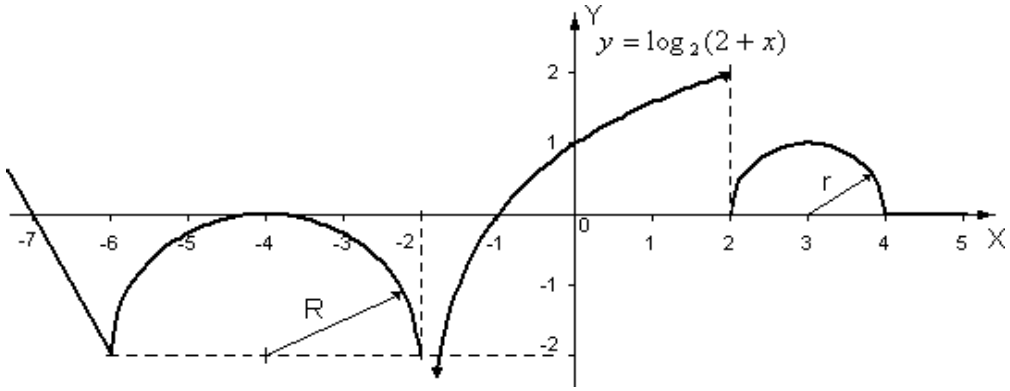

Y
3
y = ln |1 / tg(x/2)|
2
1
0
-5
-4
-3
-2
-1
1
2
3
4
5
X

25)

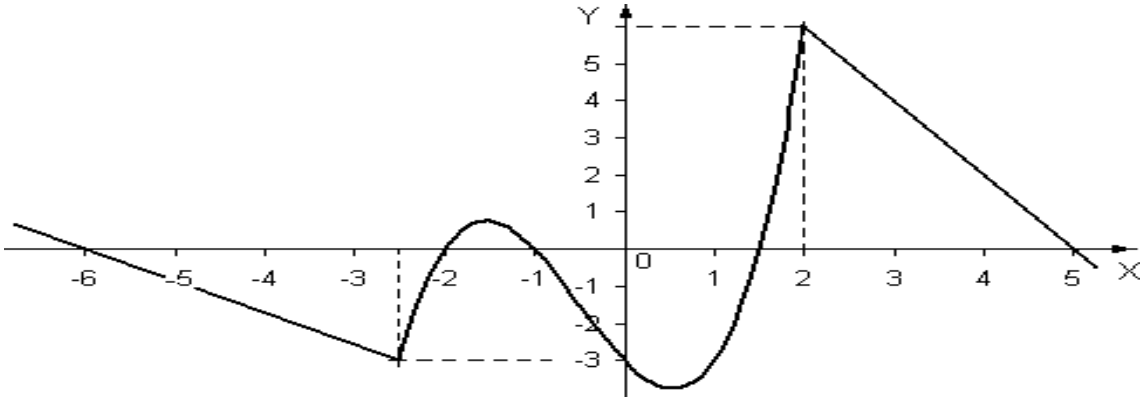


Гиперболический косинус может быть описан формулой: $Ch(x) = \frac{1}{2}\left(e^{x} + e^{-x}\right)$.

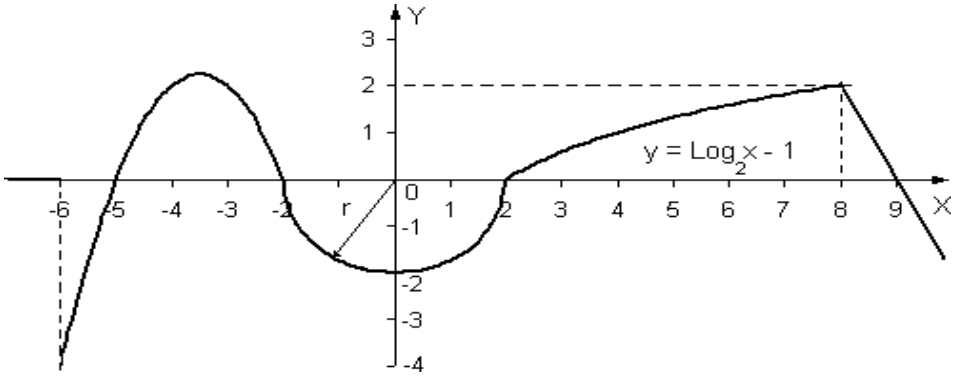
26)



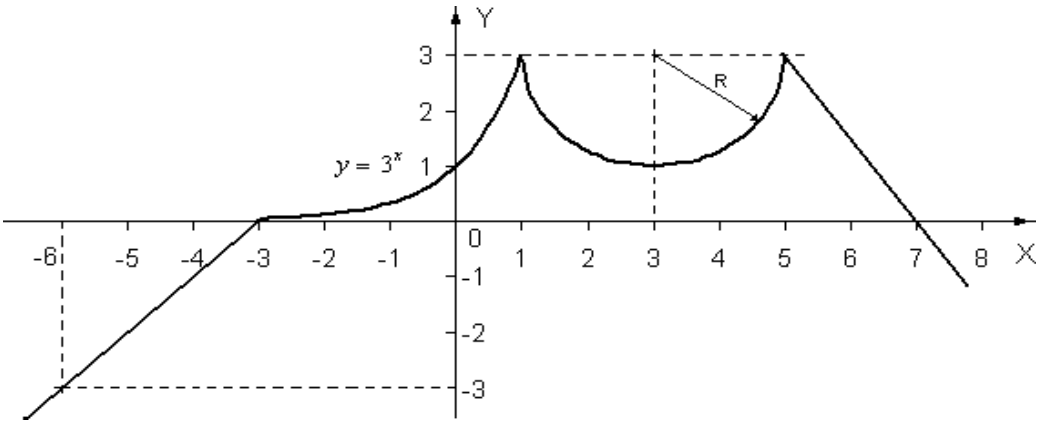
27)



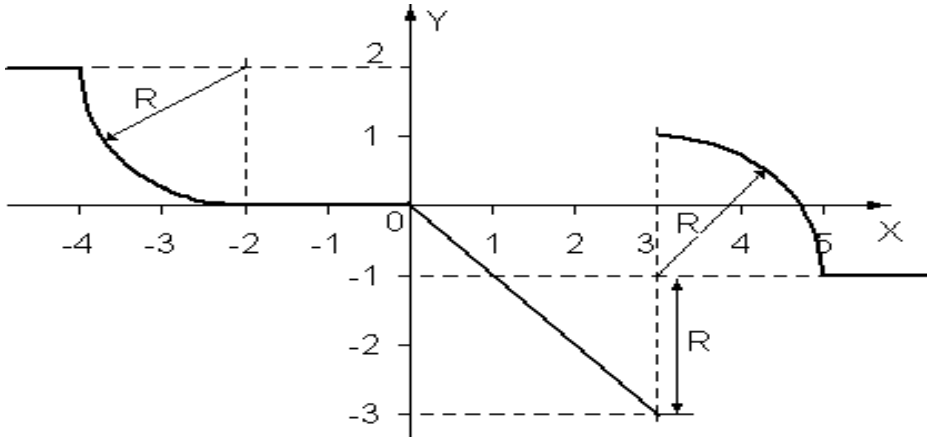
28)



29)



30)



Дополнительное требование к заданию №30: Программа должна быть написана так, что бы решения получались при различных значениях `R`, вводимых с клавиатуры. Центр положения левой четверти окружности изменяется в соответствии с введённым радиусом, а правой – остаётся постоянным при `X = 3`, `Y = -1`.

Задания для самостоятельной работы

1. Определить *D* и *M*-дату *K*-го по счету дня високосного года. Месяц вывести в словесной форме. *D* - день, *M* - месяц. Предусмотреть, что $1 \le K \le 366$.
2. Компания по снабжению электроэнергией взимает плату с клиентов по тарифу:
   - 7 р. за 1 кВт/ч за первые 250 кВт/ч;
   - 17 р. за кВт/ч, если потребление свыше 250, но не превышает 300 кВт/ч;
   - 20 р. за кВт/ч, если потребление свыше 300 кВт/ч.

   Потребитель израсходовал *n* кВт/ч. Подсчитать плату.
3. Школьники сдают нормы по прыжкам в длину. Если длина прыжка больше 2,50 м, то оценка – «5», если от 2 м до 2,5 м - оценка «4»; от 1,5 м до 2 м -

оценка «3»; если меньше 1,5 м – «2». Выставить школьнику оценку, если известна длина его прыжка. Примечание: 1 м = 100 см.

4. При покупке товара на сумму от 200 до 500 руб предоставляется скидка 3%, при покупке товара на сумму от 500 до 800 – скидка 5%, при покупке товара на сумму от 800 до 1000 руб – скидка 7%, свыше 1000 руб – скидка 10%. Покупатель приобрел 8 рулонов обоев по цене $X1$ и две банки краски по цене $X2$. Сколько он заплатил?

5. Студенты убирают урожай помидоров. При сборе до 50 кг в день работа оплачивается из расчёта 30 руб. за 1 кг; при сборе от 50 до 75 кг в день – 50 руб. за 1 кг; при сборе от 75 до 90 кг в день – 65 руб. за 1 кг; при сборе свыше 90 кг в день – 70 руб. за 1 кг плюс 120 руб. премия. Студент собрал X кг за день. Определить его заработок.

6. Ученики начальной школы сдают технику чтения. В 1 классе скорость чтения до 20 слов в мин считается низкой, от 21 до 40 слов в мин – средней, от 41 до 50 – выше среднего, более 51 слова в мин – высокой. Во втором классе скорость чтения до 30 слов в мин считается низкой, от 31 до 60 слов в мин – средней, от 61 до 70 – выше среднего, более 71 слова в мин – высокой. Задать с клавиатуры номер класса и количество прочитанных учеником в минуту слов. Оценить скорость чтения.

7. Студенты убирают яблоки. Студентам 1 курса при сборе до 50 кг в день работа оплачивается из расчёта 30 коп. за 1 кг; при сборе от 50 до 75 кг в день – 50 коп. за 1 кг; при сборе от 75 до 90 кг в день – 65 коп. за 1 кг; при сборе свыше 90 кг в день – 70 коп. за 1 кг плюс 20 руб премия. Студентам 2 курса при сборе до 60 кг в день работа оплачивается из расчёта 35 коп. за 1 кг; при сборе от 61 до 80 кг в день – 55 коп. за 1 кг; при сборе от 81 до 95 кг в день – 65 коп. за 1 кг; при сборе свыше 95 кг в день – 70 коп. за 1 кг плюс 30 руб премия.

   Студент собрал X кг яблок за день. Определить его заработок.

8. С клавиатуры задаётся целое число $A > -5$. Для положительного однозначного числа определить, является ли оно чётным; для положительного двузначного определить, кратно ли оно 3; для положительного трёхзначного числа определить, кратно ли оно 5; для числа, большего 999, определить, кратно ли оно 10. Если число $A < 0$, то вывести на экран его название в словесной форме.

9. Школьники сдают нормы по прыжкам в длину. Для учеников 5 класса нормы следующие: если длина прыжка больше 2,50 м, то оценка - 5, если от 2 м до 2,5 - оценка 4; от 1,5 м до 2 м - оценка 3; если меньше 1,5 м - 2. Нормы для учеников 6 класса: если длина прыжка больше 3,0 м, то оценка - 5, если от 2,5 м до 3,0 - оценка 4; от 2,0 м до 2,5 м - оценка 3; если меньше 2,0 м – оценка 2. Ученики других классов прыжки не сдают.

Выставить школьнику оценку, если известны номер его класса и длина прыжка.

10. Школьники сдают нормы по прыжкам в длину. Для учеников 2 класса минимальная длина прыжка должна быть не менее 1,2 м; для учеников 3 класса – не менее 1,5 м; для 4 класса – не менее 1,7 м; для 5 класса не менее 2,0 м.

    Зная номер класса и длину прыжка ученика, определить, сдан ли норматив.

11. С клавиатуры задаётся натуральное число $N < 100$. Если оно однозначное, то вывести на экран его название в словесной форме. Если число двузначное, то сообщить об этом и определить остаток от деления его на 2, 3, 5.

12. Компания по снабжению электроэнергией взымает плату с клиентов по тарифу:

    - ✓ 7 р. за 1 кВт/ч за первые 250 кВт/ч;
    - ✓ 17 р. за кВт/ч, если потребление свыше 250, но не превышает 300 кВт/ч;
    - ✓ 20 р. за кВт/ч, если потребление свыше 300 кВт/ч.

    Существуют три категории потребителей. Тариф 1 категории составляет 75% общей стоимости, тариф 2 категории – половину общей стоимости. Остальные оплачивают полностью.

    Потребитель израсходовал n кВт/ч. Подсчитать плату, зная его категорию.

13. С клавиатуры задано натуральное число $N < 10000$. Сообщить, является ли оно однозначным, двузначным, трёхзначным или четырёхзначным. В каждом случае определить, чётное оно или нечётное.

14. Ученики начальной школы сдают технику чтения. Нормы следующие: в 1 классе ребёнок должен читать не менее 20 слов в минуту; во втором классе – не менее 50 слов в мин; в 3 классе – не менее 60 слов в мин, в 4 классе – не менее 70 слов в минуту. Задавая с клавиатуры номер класса и количество прочитанных учеником в минуту слов, определить, сдана ли техника чтения.

15. Даны три целых числа определяющие дату: год, месяц, день. Считая, что год не високосный, определить дату следующего дня.

Контрольные вопросы

1. Что такое составной оператор?
2. Какова полная (неполная) форма команды ветвления (блок-схема)?
3. Каков алгоритм выполнения команды ветвления?
4. Каков алгоритм выполнения команды множественного ветвления (каскадной условной инструкции)? Блок-схема.
5. Какие операторы сравнения используются в Python?
6. Что называется простым условием? Приведите примеры.
7. Что такое составное условие? Приведите примеры.

8. Какие логические операторы допускаются при составлении сложных условий?
9. Каков результат применения оператора логическое "И" (and)?
10. Каков результат применения оператора логическое "ИЛИ" (or)?
11. Каков результат применения оператора логическое отрицание (!)?
12. Каков общий вид (формат) инструкции «Ветвление»?
13. Каков алгоритм выполнения тернарной операции? Приведите пример.
14. Каков общий вид (формат) каскадной условной инструкции?
15. Может ли оператор ветвления содержать внутри себя другие ветвления?

Пример выполнения заданий

Задание I. Две окружности заданы радиусами. Определить какая из данных окружностей будет иметь большую площадь. Вывести значение большей площади на экран.

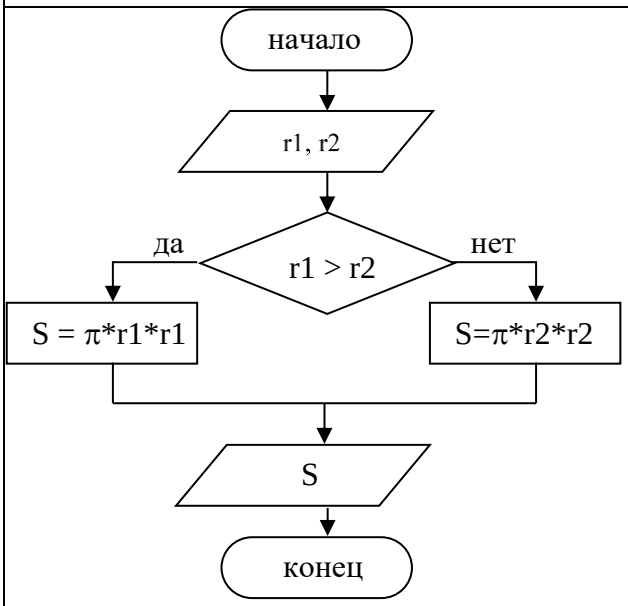
Решение

1. Математическая модель

Наибольшая площадь будет у окружности с наибольшим радиусом.

Аргументы: r1, r2 вещественного типа – радиусы окружностей.
Результаты: S вещественного типа – значение наибольшей площади.

2. Алгоритм



3. Программа

```
pi = 3.14159265359
r1, r2 = map (float, input(" Введите
радиусы r1 и r2: ").split())
if r1>r2:
      S = pi*pow(r1, 2)
else:
      S = pi*pow(r2, 2.0)
print(" S = ", S)
```

4. Результат работы программы

```
Введите радиусы r1 и r2: 6  3
S = 113.04
```

Задание II. Дан номер месяца *m*. Вывести на экран название этого месяца.

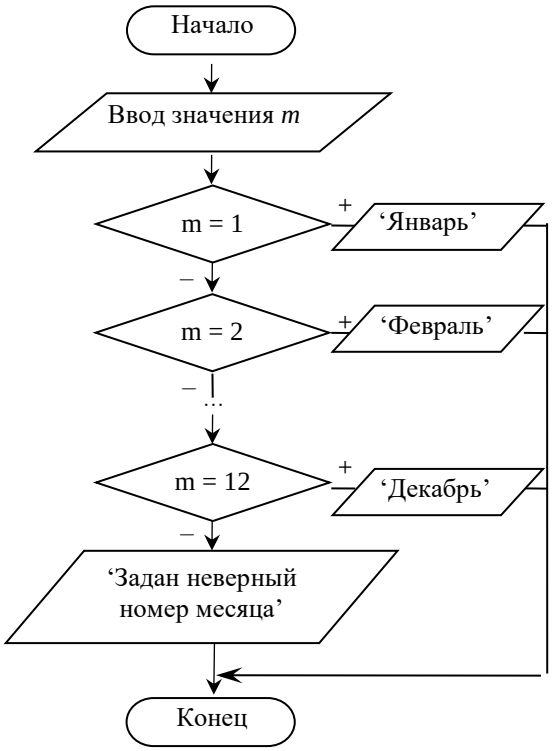
Решение

1. Математическая модель

Для решения задачи необходимо использовать инструкцию выбора.

Аргументы: ***m*** целого типа - номер месяца.
Результаты: текстовое сообщение.

2. Алгоритм	3. Программа Python

	`a = int(input("Введите номер месяца - "))`
	
	`if a == 1:`
	`  print("Январь")`
	`elif a == 2:`
	`  print("Февраль")`
	`elif a == 3:`
	`  print("Март")`
	`elif a == 4:`
	`  print("Апрель")`
	`elif a == 5:`
	`  print("Май")`
	`elif a == 6:`
	`  print("Июнь")`
	`elif a == 7:`
	`  print("Июль")`
	`elif a == 8:`
	`  print("Август")`
	`elif a == 9:`
	`  print("Сентябрь")`
	`elif a == 10:`
	`  print("Октябрь")`
	`elif a == 11:`
	`  print("Ноябрь")`
	`elif a == 12:`
	`  print("Декабрь")`
	`else:`
	`  print("Ошибка ввода")`

4. Результат работы программы

```
Введите номер месяца - 5
Май
```

Список литературы, рекомендуемый к использованию по данной теме

- [1] стр. 58-64, 109-110;
- [4] стр. 394-409;
- [6] стр. 42-48;
- [11] Теоретический материал: Синтаксис условной инструкции (informatics.msk.ru) (https://informatics.msk.ru/mod/book/view.php?id=4033).

Практическое занятие № 4
Программирование алгоритмов циклической структуры

ЦЕЛЬ: приобретение навыков программирования вычислительных повторяющихся процессов с использованием циклических операторов.

Знания, умения и владения, приобретаемые обучающимся в результате освоения темы, в рамках формируемых компетенций (ПК-1: ИД-1$_{ПК-1}$, ИД-2$_{ПК-1}$**;** ПК-2: ИД-1$_{ПК-2}$, ИД-2$_{ПК-2}$): см. приложение 2: вырабатывается навык разработки математических и информационных моделей, а также формируется способность к разработке и применению алгоритмических и программных решений (циклической структуры) в области прикладного программирования с умением объяснить теоретическую и практическую часть темы (формируется часть указанных компетенций).

Теоретическая часть
4. Циклы

Цикл - вид конструкции, используемый для вычислений, повторяющихся многократно.

Блок, ради которого и организуется цикл, называется **телом цикла**. Остальные операторы служат для управления процессом повторения вычислений: это начальные установки, проверка условия продолжения цикла и модификация параметра цикла (рис. 4.1). Один проход цикла называется **итерацией.**

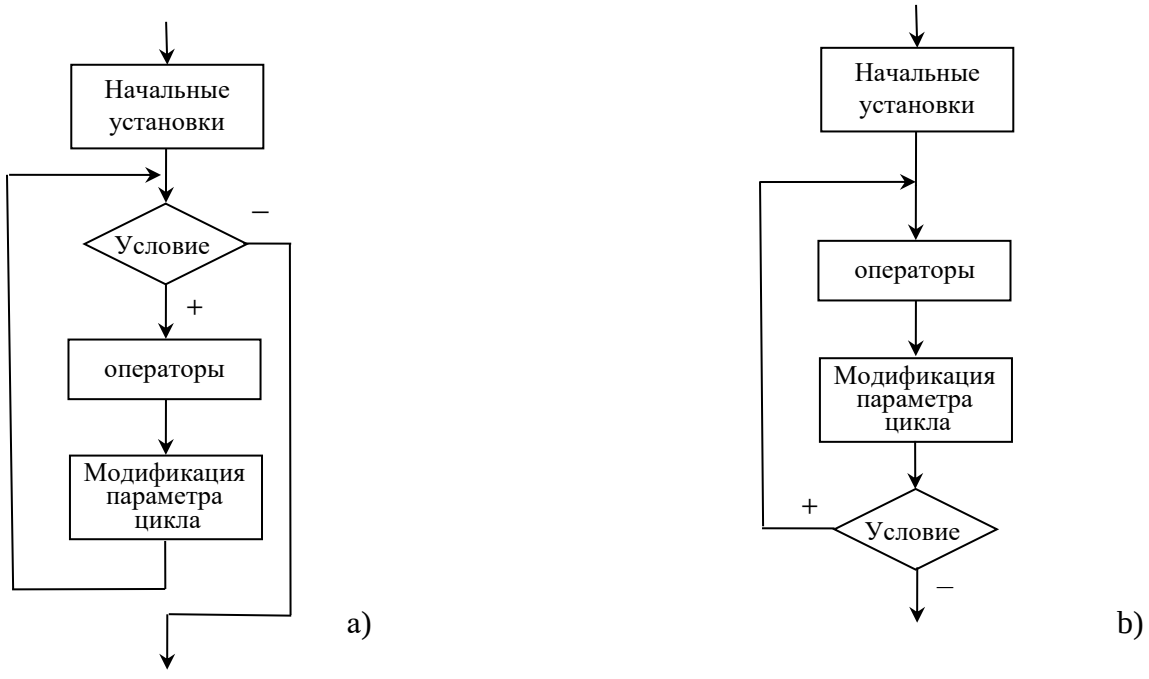


Рисунок 4.1 – Структурные схемы выполнения цикла

Начальные установки служат для того, чтобы до входа в цикл задать значения переменных, которые в нем используются.

Проверка условия продолжения цикла выполняется на каждой итерации либо до тела цикла (тогда говорят о цикле с предусловием, рис. 4.1а), либо после тела цикла (цикл с постусловием, рис 4.1б). Разница между ними состоит в том, что тело цикла с постусловием всегда выполняется хотя бы один раз, после чего проверяется

надо ли его выполнять еще раз. Проверка необходимости выполнения цикла с предусловием делается до тела цикла, поэтому возможно, что он не выполнится ни разу.

Параметром цикла называется переменная, которая используется при проверке условия цикла и принудительно изменяется на каждой итерации, причем, как правило на одну и туже величину. Если параметр цикла целочисленный, он называется счетчиком цикла.

Для реализации циклических вычислений предусмотрены три оператора: цикл с предусловием, цикл с постусловием, цикл с параметром. Первые два оператора используются в том случае, когда число повторений цикла неизвестно. Если же число повторений цикла может быть определено перед его началом, то используется оператор цикла с параметром.

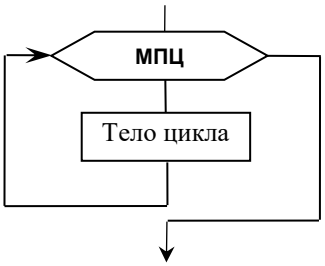
4.1. Оператор цикла с параметром (for)

Цикл обхода (цикл с параметром или со счетчиком, арифметический цикл, цикл «для») – этот цикл используется либо для повторения какой-либо последовательности действий заданное число раз, либо для изменения значения переменной в цикле от некоторого начального значения до некоторого конечного.

Формат оператора:

```
for i in range(a, b, h):
     Оператор1 #Тело цикла
[else:
     Оператор2]
```

Графическая интерпретация оператора цикла **for**



МПЦ – модификация параметра цикла
range(a, b, h) – функция, генерирующая множество значений от a до b-1 с шагом h.
По умолчанию a = 0, h = 1.

Цикл с параметром реализован как цикл с предусловием.

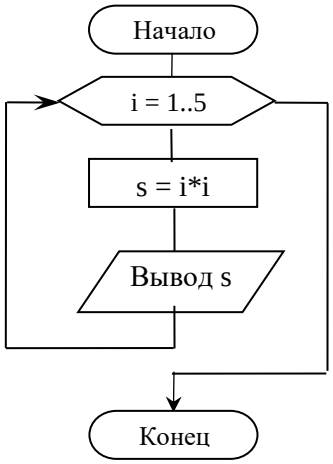
Модификации выполняются после каждой итерации цикла и служат обычно для изменения параметра цикла. Простой или составной *оператор* представляет собой тело цикла.

Значение переменной-счетчика можно использовать в теле цикла; изменение его значения тоже допускается, но считается плохим стилем программирования.

Помимо range можно использовать другие итерируемые объекты, например, кортежи и списки. Использование списка [1, 3, 5] будет почти полностью аналогично использованию range(1, 6, 2).

Для всех операторов цикла выход из цикла осуществляется как вследствие естественного окончания оператора цикла, так и с помощью операторов перехода (**continue** – переход к следующей итерации) и выхода (**break** – выход из цикла, оператор возврата из функции **return**).

Пример: Вывести на экран таблицу квадратов первых пяти чисел.




```
#numbers_and_them_squares
print("\nЧисла и их квадраты:\n ")
for i in range(1, 6):
      s = i*i
      print(i,"^ 2 = ",s)
```

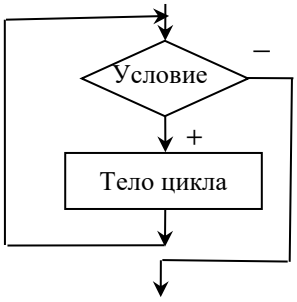
Числа и их квадраты:
1 ^ 2 = 1
2 ^ 2 = 4
3 ^ 2 = 9
4 ^ 2 = 16
5 ^ 2 = 25

4.2. Оператор цикла с предусловием (while)

Формат оператора:

```
while выражение:
  Оператор1 #Тело цикла
[else:
  Оператор2]
```

Графическая интерпретация оператора:



Оператор **while** позволяет многократно выполнять одни и те же действия в зависимости от значения выражения (условия). Тип выражения должен быть арифметическим или приводимым к нему. Выражение вычисляется перед каждой итерацией цикла.

Работа оператора

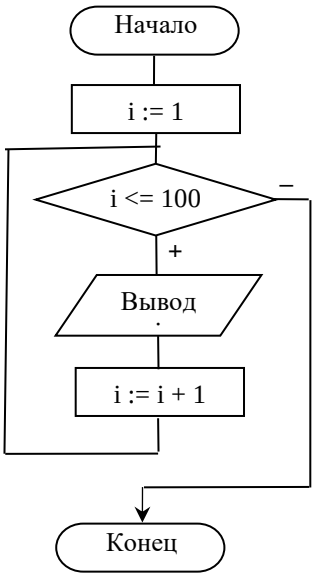
При входе в цикл вычисляется значение выражения (условия). Если при входе в цикл значение условия равно **false**, то вход в цикл не осуществляется и управление передается оператору, следующему непосредственно за оператором цикла. Если условие истинно (не равно false), то происходит вход в цикл и однократное выполнение тела цикла, представленного простым или составным оператором. Как только достигнут конец цикла, управление снова передается на его заголовок, где снова вычисляется значение выражения. Если значение выражения все еще равно **true,** то тело цикла выполняется еще один раз и т. д. до тех пор, пока значение условия не станет равно **false.** Если оно постоянно будет равно **true**, то цикл будет бесконечным, т.е. произойдет зацикливание.

Замечания:

– для того, чтобы тело цикла выполнилось хотя бы один раз, необходимо, чтобы перед выполнением тела цикла выражение (условие) было истинно;

– для того, чтобы цикл завершился, необходимо, чтобы в теле цикла изменялись значения переменных, входящих в условие.

Пример: Вывести на экран числа от 1 до 100.




```
# Schet_ot_1_do_100;
print("\nЧисла от 1 до 100:\n ")
i=1 #начальное значение параметра цикла
while (i<=100): # перебираем первые 100 чисел
    print(" ",i,end="") # вывод на экран значения
i
    i+=1 # Увеличение параметра цикла на 1
```

Результат работы программы:

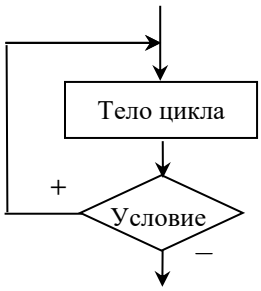
```
Числа от 1 до 100:

  1  2  3  4  5  6  7  8  9  10  11  12  13  14
15  16  17  18  19  20  21  22  23 …
```

4.3. Оператор цикла с постусловием

Формат оператора:

В Python цикл с постусловием отсутствует.

Графическая интерпретация оператора



Работа цикла

Сначала выполняется простой или составной оператор, составляющий тело цикла, а затем вычисляется выражение (условие). Если оно истинно (не равно **`false`**), тело цикла выполняется еще раз. Цикл завершается, когда выражение станет равным **`false`** или в теле цикла будет выполнен какой-либо оператор передачи управления. Тип выражения должен быть арифметическим или приводимым к нему.

Замечания:

– последовательность инструкций, являющихся телом цикла, всегда будет выполнена хотя бы один раз;

– для того, чтобы цикл завершился, необходимо, чтобы в теле цикла изменялись значения переменных**,** входящих в условие.

4.4. Вложенные циклы

Тело цикла может содержать любой оператор, в том числе и оператор цикла. Структура цикла, содержащая вложенный цикл, называется *кратным циклом*. Число вложений может быть произвольным. Если цикл содержит один вложенный цикл, то он называется *двойным циклом* (рис. 4.2).

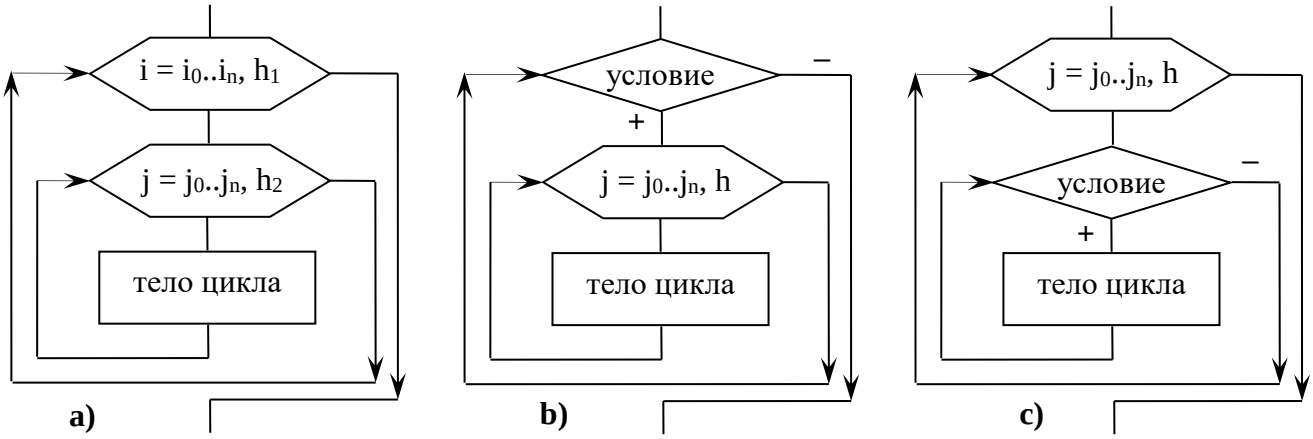


Рисунок 4.2 – Блок-схемы некоторых из возможных вариантов реализации двойного цикла

Цикл, который содержит вложенный цикл, называется *внешним*. В двойном цикле вложенный цикл называется *внутренним.* На рисунке 4.2.a предложена блок-схема алгоритма, в котором для реализации и внешнего и внутреннего циклов использован оператор цикла с параметром (**`for`**). Такой вид двойного цикла используется при работе с двумерными массивами (таблицами). На рисунке 4.2.b для реализации внешнего цикла использован оператор цикла с предусловием (**`while`**), а внутреннего цикла – оператор цикла с параметром (**`for`**). На рисунке 4.2.c внешний цикл представлен оператором цикла с параметром (**`for`**), а внутренний – оператором цикла с предусловием (**`while`**).

Переменная внутреннего цикла всегда меняется быстрее, чем внешнего. Это означает, что для каждого значения внешней переменной цикла меняются все значения внутренней переменной.

Внешний и внутренний циклы могут использовать любой вид операторов цикла Python (**`while, for`**) (рис. 4.2).

Пример. Дано число вида $\overline{ab}$, где $a = 1..2,\ b = 0..9$. Вывести на экран все числа $\overline{ab}$.

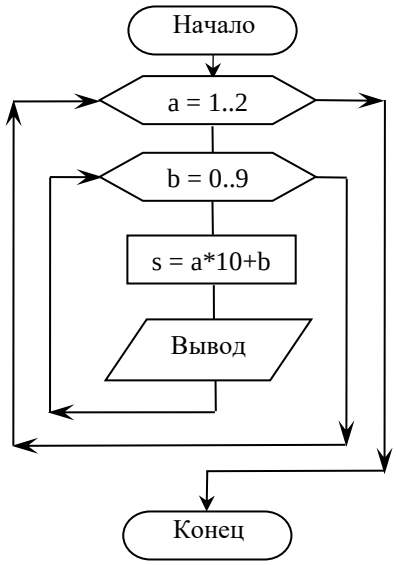
1. **Математическая модель**

Выпишем все эти числа в ряд:

10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29

Сначала увеличиваются единицы от 0 до 9, при этом значение десятков не меняется. Только когда перебор единиц завершён, десятки увеличиваются на 1. После этого опять происходит изменение единиц от 0 до 9. В переменной a – будут десятки, в b – единицы.

2. Блок-схема



3. Программа

```
for a in range(1, 3):
   for b in range(0, 10):
       print(a * 10 + b, end = ‘ ‘)
```

4.

Результат работы программы

10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29

Вопросы и задания

Задание I

В соответствии с вариантом составить и реализовать программы
Используя оператор цикла с параметром, решить предложенную задачу:

1. Вычислить сумму и произведение первых n натуральных чисел. Предусмотреть ввод *n* с клавиатуры.

2. Вычислить $S = \sum_{n=1}^{k} (-1)^{n} n^2,\ k \geq 10.$

3. Вычислить сумму первых n членов ряда с общим членом $a_k = \frac{k+1}{\sqrt{k^2+2}}, k = \overline{1,n}$

.

4. Определить средний рост девочек, и мальчиков одного класса. В классе учится n учеников. ($n \geq 15$).

5. Вводя в цикле по 4 оценки каждого студента группы, подсчитать число студентов, не имеющих оценок 2 и 3. В группе учится *n* студентов. Предусмотреть ввод *n* с клавиатуры.

6. Вводя в цикле по 4 оценки, полученные студентами в сессию, определить средний балл группы по всем экзаменам. В группе учится *n* студентов. Предусмотреть ввод *n* с клавиатуры.

7. Задано n троек, чисел a, b, c. Вводя их по очереди с клавиатуры в порядке возрастания и интерпретируя их как длины сторон треугольника, определить, сколько троек может быть использовано для построения треугольника.

8. В ЭВМ по очереди поступают результаты соревнований по плаванию на дистанции 200 м, в которых участвует n спортсменов ($n > 10$). Выдать на экран дисплея лучший результат после ввода результата очередного спортсмена.

9. В ЭВМ вводятся по очереди координаты n точек ($n \geq 10$). Определить, сколько из них попадет в круг радиуса r с центром в точке (a, b).

10. Ученикам первого класса назначается стакан молока (200 мл), если их вес составляет меньше 30 кг. Определить, сколько литров молока потребуется, ежедневно для одного класса, состоящего из n учеников. После взвешивания вес каждого ученика вводится в ЭВМ.

11. В ЭВМ вводятся номер по списку и рост учеников 10 класса. Вывести на экран номера по списку тех учеников, рост которых больше 170 см.

12. В соревнованиях по бегу на дистанции 100 м принимают участие n спортсменов. Вводя очереди номера по списку и результаты участников в ЭВМ, определить, сколько из них выполнили норму, и напечатать их номера (норма равна 13,2 сек.)

13. В ЭВМ вводятся по очереди координаты n точек. Определить, сколько из них попадет в кольцо с внутренним радиусом r_1, внешним r_2 и центром в начале координат.

14. В ЭВМ по очереди вводятся координаты n точек. Определить, сколько из них принадлежит фигуре, ограниченной осью абсцисс и аркой синусоиды, построенной для аргумента со значениями от 0 до π.

15. Задана окружность радиуса r с центром в начале координат. Вводя последовательно в ЭВМ координаты n точек, являющихся центрами других окружностей того же радиуса, определить, сколько из этих окружностей пересекает данную.

16. Найти все трехзначные натуральные числа, сумма цифр которых равна их произведению.

17. Найти сумму целых положительных чисел, больших 20, меньших 100 и кратных 3.

18. Вычислить сумму $S = 1 + \frac{1}{2^3} + \frac{1}{3^3} + ... + \frac{1}{n^3}$, где $n \geq 10$.

19. Вводя в цикле по 4 оценки, полученные студентами в сессию, определить число неуспевающих студентов группы. В группе учится n студентов. Предусмотреть ввод n с клавиатуры.

20. Цилиндр объема 1 куб.ед. имеет высоту *h*. Определить радиус основания для значений *h* от 1 до 6 с шагом 0.5 ед.

21. Вывести на печать последовательность с общим членом $a_k = \sin\left(k^2 + b\right)$, где k = 0, 1, ..., *K*. Предусмотреть ввод *K* и *b* с клавиатуры. Вычислить количество элементов, удовлетворяющих условию: $0 \le a_k \le 0.5$.

22. В ЭВМ вводятся поочередно *n* точек. Определить, сколько из них принадлежит фигуре, ограниченной осью ординат и правой полуокружностью $x^2 + y^2 = r^2$.

23. В ЭВМ по очереди вводятся координаты *n* точек. Определить, сколько из них принадлежит фигуре, ограниченной линией $|x| + |y| = 1$.

24. Составить программу вычисления значений функции $p(x) = \frac{1}{16}\left(231x^6 - 315x^4 + 105x^2 - 5\right)$ при изменении *x* 0 до 1 с шагом 0.1.

25. Вычислить значение выражения $\frac{(x-2)(x-4)(x-6)...(x-64)}{(x-1)(x-3)(x-5)...(x-63)}$ для произвольного действительного числа *x* из области определения данного выражения.

Задание II

Используя оператор цикла с предусловием, для данного значения x найти сумму ряда S с точностью до члена ряда, по абсолютной величине меньшего eps = 0.0001. Сравнить эту сумму со значением контрольной функции y.

1. $S = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + \frac{x}{1!} + \frac{x^2}{2!} + ...; x = 1.5; y = e^x.$

2. $S = \sum_{n=0}^{\infty} \frac{x^n \ln^n 3}{n!} = 1 + \frac{x \ln 3}{1!} + \frac{x^2 \ln^2 3}{2!} + ...; x = 1; y = 3^x.$

3. $S = \sum_{n=0}^{\infty} x^n = 1 + x + x^2 + x^3 + ...; x = 0.7; y = \frac{1}{1-x}.$

4. $S = \sum_{n=0}^{\infty} (-1)^n \frac{x^n}{2 \cdot 2^n} = \frac{1}{2} - \frac{x}{4} + \frac{x^2}{8} - ...; x = 1.2; y = \frac{1}{2+x}.$

5. $S = \sum_{n=1}^{\infty} \left(\frac{1}{2^n} + \frac{1}{3^n}\right) x^{n-1} = \frac{5}{6} + \frac{13}{36}x + \frac{35}{216}x^2 + ...; x = -0.8; y = \frac{5-2x}{6-5x+x^2}.$

6. $S = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - ...; x = 0.3; y = \cos x.$

7. $S = \sum_{n=1}^{\infty} (-1)^{n-1} \frac{x^n}{n} = x - \frac{x^2}{2} + \frac{x^3}{3} - ...; x = 0.4; y = \ln(1+x).$

8. $S=\sum_{n=1}^{\infty}(-1)^{n+1}\frac{\sin(nx)}{n}=\sin x-\frac{\sin 2x}{2}+\frac{\sin 3x}{3}+\ldots;x=-\frac{\pi}{2};y=\frac{x}{2}.$

9. $S=\sum_{n=0}^{\infty}\frac{x^{2n+1}}{(2n+1)!}=x+\frac{x^3}{3!}+\frac{x^5}{5!}+\ldots;x=2;y=\frac{e^x-e^{-x}}{2}.$

10. $S=\sum_{n=0}^{\infty}(-1)^n\frac{x^{2n+1}}{2n+1}=x-\frac{x^3}{3}+\frac{x^5}{5}-\ldots;x=0.6;y=arctg\,x.$

11. $S=\sum_{n=0}^{\infty}(-1)^n\frac{x^{2n+1}}{(2n+1)!}=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\ldots;x=1.4;y=\sin x.$

12. $S=\sum_{n=0}^{\infty}\frac{x^{2n}}{(2n)!}=1+\frac{x^2}{2!}+\frac{x^4}{4!}+\ldots;x=0.5;y=\frac{e^x+e^{-x}}{2}.$

13. $S=\sum_{n=1}^{\infty}\frac{\sin(2n-1)x}{2n-1}=\sin x+\frac{\sin 3x}{3}+\ldots;x=\frac{\pi}{2};y=\frac{\pi}{4}.$

14. $S=\sum_{n=1}^{\infty}(-1)^{n+1}\frac{\cos nx}{n}=\cos x-\frac{\cos 2x}{2}+\ldots;x=\frac{\pi}{8};y=\ln\left(2\cos\frac{x}{2}\right).$

15. $S=\sum_{n=1}^{\infty}\frac{\cos nx}{n}=\cos x+\frac{\cos 2x}{2}+\ldots;x=\pi;y=-\ln\left(2\sin\frac{x}{2}\right).$

16. $S=2\sum_{n=1}^{\infty}\frac{x^{2n-1}}{2n-1}=2\left(x+\frac{x^3}{3}+\frac{x^5}{5}+\ldots\right);x=-0.4;y=\ln\frac{1+x}{1-x}.$

17. $S=\sum_{n=1}^{\infty}\frac{1}{(2n-1)x^{2n-1}}=\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+\ldots;x=3;y=\frac{1}{2}\ln\frac{x+1}{x-1}.$

18. $S=\sum_{n=1}^{\infty}\frac{1}{(2n-1)}\left(\frac{x-1}{x+1}\right)^{2n-1}=\frac{x-1}{x+1}+\frac{1}{3}\left(\frac{x-1}{x+1}\right)^3+\ldots;x=0.6;y=\frac{1}{2}\ln x.$

19. $S=\sum_{n=0}^{\infty}(-1)^n\frac{x^{2n}}{n!}=1-\frac{x^2}{1!}+\frac{x^4}{2!}-\frac{x^6}{3!}\ldots;x=-0.7;y=e^{-x^2}.$

20. $S=\sum_{n=0}^{\infty}(n+1)x^n=1+2x+3x^2+4x^3+\ldots;x=-0.7;y=\frac{1}{(1-x)^2}.$

21. $S=\sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1}=x+\frac{x^3}{3}+\frac{x^5}{5}+\ldots;x=0.35;y=\ln\sqrt{\frac{1+x}{1-x}}.$

22. $S=\sum_{n=1}^{\infty}\frac{(-1)^n}{n}(x+1)^{2n}=-(x+1)^2+\frac{1}{2}(x+1)^4+\ldots;x=-1.2;y=\ln\frac{1}{2+2x+x^2}.$

23. $S = 1 - \sum_{n=1}^{\infty} (-1)^{n+1} \frac{1 \cdot 3 \cdot 5 \cdot \ldots \cdot (2n-3)}{2 \cdot 4 \cdot 6 \cdot \ldots \cdot 2n} x^n = 1 + \frac{1}{2}x - \frac{1}{2 \cdot 4}x^2 + \frac{1 \cdot 3}{2 \cdot 4 \cdot 6}x^3 - \ldots;$

$x = -0.8; \quad y = \sqrt{1+x}.$

24. $S = 1 + \sum_{n=1}^{\infty} \frac{1 \cdot 3 \cdot 5 \cdot \ldots \cdot (2n-1)}{2 \cdot 4 \cdot 6 \cdot \ldots \cdot 2n} x^{2n} = 1 + \frac{1}{2}x^2 + \frac{1 \cdot 3}{2 \cdot 4}x^4 + \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6}x^6 + ..;$

$x = 0.64; \quad y = \frac{1}{\sqrt{1-x^2}}.$

25. $S = \sum_{n=1}^{\infty} \frac{\sin nx}{n} = \sin x + \frac{\sin 2x}{2} + \ldots; x = \frac{\pi}{3}; y = \frac{\pi - x}{2}.$

Задание III

1. Какое количество теплоты потребуется для испарения 100 г воды при 100, 110, …, 200°С?
2. Определите внутреннюю энергию 1 г гелия при температуре 10, 15, …, 50°С.
3. В сосуде объёмом 1 см^3 находится 10^{16} молекул газа. Определить давление газа в сосуде при температуре 0, 1,…,10°С.
4. Какова средняя квадратичная скорость молекул кислорода при 10, 12, …, 20°С?
5. Сколько теплоты выделится при конденсации 100, 200, …, 900 г водяного пара?
6. В котле ёмкостью 6.0 м^3 содержится 25 кг перегретого пара. Каково давление пара, если его температура равна 200, 220, …, 300°С?
7. Какова средняя квадратичная скорость молекул водорода при температуре 10, 15, …, 50°С?
8. Определите внутреннюю энергию 1 г гелия при температуре 20, 22, …, 40°С.
9. В сосуде объёмом 480 см^3 при давлении $2.5 \cdot 10^4$ Н/м^2 находится газ. Подсчитать количество молекул газа для значений температуры 10, 11, …, 20°С.
10. Сколько теплоты потребуется для испарения 200 г воды при температуре 100, 110, …, 200°С?
11. Какова средняя квадратичная скорость молекул углекислого газа при температуре 0, 5, …, 30°С ?
12. Определить среднюю квадратичную скорость молекул водорода при 0, 5,…, 50°С.
13. В сосуде объёмом 450 см^3 при давлении $2.5 \cdot 10^4$ Н/м^2 находится газ. Подсчитать количество молекул газа для значений температуры 10, 12, …, 20°С.

14. Определите внутреннюю энергию 1 г водорода при температуре 20, 21, …, 30°C.
15. Определить среднюю кинетическую энергию вращательного движения молекул водорода, содержащихся в 1.0 моль при 10, 11, …, 20°C.
16. Плотность воздуха при нормальных условиях 1.3 г/л. Какова плотность воздуха при давлении $4.0 \cdot 10^5$ Н/м^2 и температуре 50, 60, …, 100°C?
17. Определите внутреннюю энергию 1 г углекислого газа при температуре 10, 15, …, 50°C.
18. В колбе содержится газ при нормальных условиях. Сколько молекул газа содержится в колбе, если её ёмкость равна 0.1, 0.2, …, 0.5 л?
19. Какова средняя квадратичная скорость молекул водорода при 0, 2, …, 20°C ?
20. Какова средняя квадратичная скорость молекул кислорода при температуре 0, 2, …, 10°C?
21. В сосуде объёмом 400 см^3 при давлении $2.5 \cdot 10^4$ Н/м^2 находится газ. Подсчитать количество молекул газа для значений температуры 0, 1, …, 10°C.
22. Сколько теплоты потребуется для испарения 150 г воды при температуре 100, 105, …, 150°C?
23. Определите внутреннюю энергию 1 г гелия при температуре 20, 21, …, 30°C.
24. Какова средняя квадратичная скорость молекул углекислого газа при температуре 0, 5, …, 30°C ?
25. Определите внутреннюю энергию 1 г кислорода при температуре 10, 15, …, 50°C.

Контрольные вопросы

1. Какие базовые алгоритмические конструкции можно реализовать средствами языка Python?
2. Что называется циклом?
3. Что такое тело цикла?
4. Что такое параметр цикла?
5. Что такое итерация?
6. Что такое зацикливание?
7. Какие типы циклов существуют?
8. Какие инструкции используются для реализации циклов на Python?
9. Какая инструкция используется для реализации цикла с параметром?
10. Можно ли использовать в теле цикла переменную, являющуюся параметром цикла?
11. Какая инструкция используется для реализации цикла с предусловием?
12. От чего зависит количество повторений тела цикла с предусловием?

13. Цикл обхода (цикл с параметром или со счетчиком, арифметический цикл, цикл «для») – это цикл с заранее известным числом повторений? Так ли это?
14. Какие из конструкций повторения могут привести к зацикливанию?
15. Как работает цикл со счетчиком?
16. Функция **`range(a, b, h)`:** особенности работы, параметры?

Пример выполнения заданий

Задание I. Составить программу табулирования функции

$$y = e^x \frac{2.42x^2 - 6.47x + 1.03}{x - 5.8}$$ на отрезке [-4, 4] с шагом 0.2.

Решение

1. Математическая модель

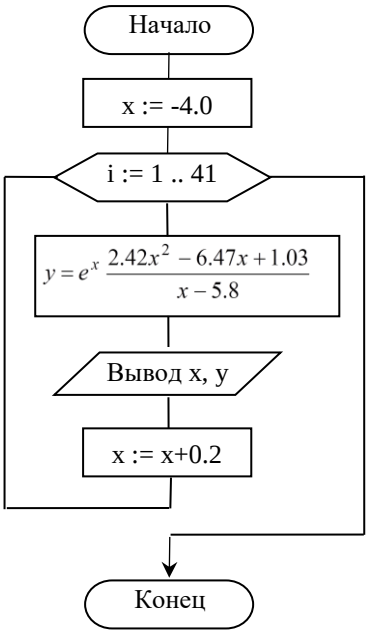
Прежде всего, определим число повторений тела цикла: $S = \frac{4-(-4)}{0.2} + 1 = 41.$

Аргументы: переменная x вещественного типа.

Результаты: значение заданной функции y (вещественного типа).

Промежуточные величины: счетчик i целого типа.

2. Алгоритм



3. Программа

```
import math as m


x = 4.0

for i in range(1, 42):
     y = m.exp(x) * (2.42 * m.pow(x, 2) -
6.47 * x +1.03) / (x - 5.8)
     print('x = ', x, 'y = ', y)
     x += 0.2
```

4. Результат работы программы

```
…
x = -3.6   y = -0.161865
x = -3.4   y = -0.185016
…
x = -2.2   y = -0.373639
…
```

Задание II. Для x = 0.47 найти сумму ряда

$$y = x + \left(\frac{1}{2} \cdot \frac{x^3}{3} + \frac{1 \cdot 3}{2 \cdot 4} \cdot \frac{x^5}{5} + \ldots + \frac{1 \cdot 3 \cdot 5 \cdot \ldots \cdot (2n-1)}{2 \cdot 4 \cdot 6 \cdot \ldots \cdot 2n} \cdot \frac{x^{2n+1}}{2n+1} + \ldots \right)$$

с точностью до члена ряда, меньшего (по абсолютной величине) eps = 0.0001, используя оператор цикла с предусловием. Сравнить полученную сумму со значением функции для контроля $y = \arcsin x$.

Решение

1. Математическая модель. Обозначим через p1 последовательные произведения нечетных чисел. Через p2 – последовательные произведения четных чисел.

Аргументы: `x` – переменная вещественного типа, `-1 < x < 1`.

Результаты: `summa` – сумма членов ряда вещественного типа;

`y` – значение контрольной функции (вещественного типа).

Промежуточные величины:

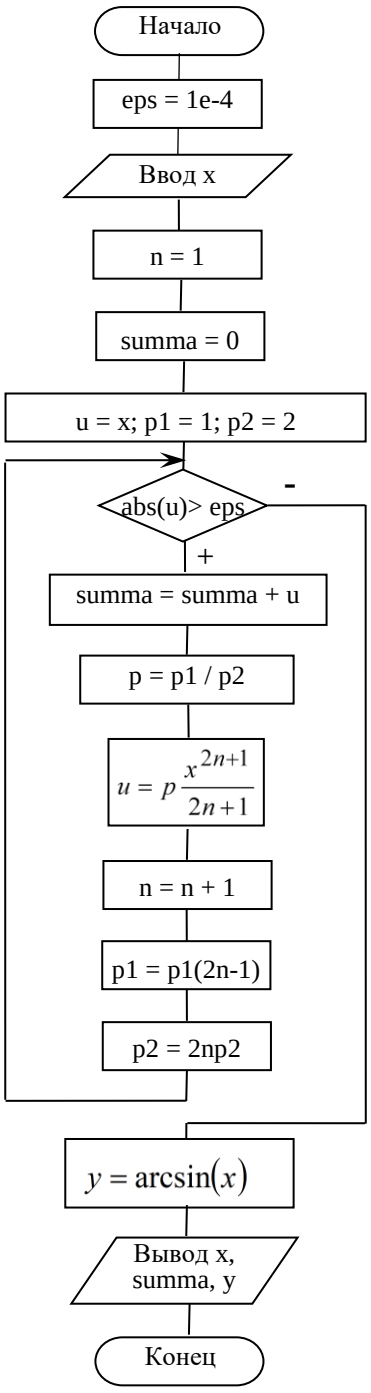
`eps` – константа, с которой сравнивается абсолютная величина текущего члена ряда: члены ряда представляют собой убывающую последовательность, поэтому согласно условию вычисление суммы членов ряда будет продолжаться до тех пор пока абсолютное значение текущего члена ряда не окажется меньше `eps`;

`p1` – последовательные произведения нечетных чисел, вещественного типа;

`p2` – последовательные произведения четных чисел, вещественного типа;

`p` – частное, получаемое при делении `p1` на `p2`, вещественного типа;

`n` – номер члена ряда, целого типа; `u` – текущий член ряда, вычисляемый по формуле $u = p\frac{x^{2n+1}}{2n+1}$, вещ. типа;

2. Алгоритм	3. Программа
	


```
// вычисление суммы членов ряда

import math
eps = 0.0001
# описание переменных, ввод x, начальные
# установки

x = float(input("Введите значение x: "))
n = 1
summa = 0
u = x
p1 = 1
p2 = 2
#организация вычисления суммы членов ряда
while(abs(u) > eps):
      summa += u
      p = p1/p2
      u = p*pow(x, 2*n+1)/(2*n+1)
      n+=1
      p1 = p1*(2*n-1)
      p2 = p2*2*n
# вычисление значения контрольной функции
y = math.asin(x)
#вывод результатов
print(" Сумма данного ряда при x =",x,", с
точностью 0.0001, S = {0:.4}".format(summa))
print(" Значение контрольной функции, y =
{0:.4}".format(y))
```

4. Результат работы программы

```
Введите значение x: 0.47
Сумма данного ряда при x = 0.4700 с точностью
до 0.0001, S = 0.4893
Значение контрольной функции, y = 0.4893
```

Вывод: С точностью до 0.0001 значение суммы ряда и значение контрольной функции совпадают, следовательно, вычисление суммы членов ряда запрограммировано верно.

Список литературы, рекомендуемый к использованию по данной теме

- [1] стр. 65-79
- [4] стр.410-435
- [6] стр. 67-93
- [11] Теоретический материал: Цикл for (informatics.msk.ru) - https://informatics.msk.ru/mod/book/view.php?id=4159, https://informatics.msk.ru/mod/book/view.php?id=4383

Практическое занятие №5
Использование функций для решения прикладных задач

ЦЕЛЬ: закрепление теоретических знаний и приобретение навыков программирования функций и рекурсивных функций.

Знания, умения и владения, приобретаемые обучающимся в результате освоения темы, в рамках формируемых компетенций (ПК-1: ИД-1$_{ПК-1}$, ИД-2$_{ПК-1}$**;** ПК-2: ИД-1$_{ПК-2}$, ИД-2$_{ПК-2}$): см. приложение 2: вырабатывается навык разработки математических и информационных моделей, а также формируется способность к разработке и применению алгоритмических и программных решений (с использованием функций и рекурсивных алгоритмических конструкций) в области прикладного программирования с умением объяснить теоретическую и практическую часть темы (формируется часть указанных компетенций).

Теоретическая часть
5. Подпрограммы

5.1. Понятие подпрограммы

При решении задач часто встречаются случаи, когда для разных наборов исходных данных приходится выполнять одни и те же команды. В некоторых случаях для организации повторяющихся вычислений можно воспользоваться командой повторения. Иногда имеет смысл выделить повторяющиеся последовательности команд в самостоятельный блок, вынеся его за пределы основной программы и в нужном месте организовать вызов этого блока, указав соответствующий набор параметров. Такой блок, реализующий некоторый вполне законченный этап обработки информации и допускающий обращение к нему из различных частей основной программы, называется **подпрограммой.** Подпрограмма оформляется в виде замкнутого участка программы, имеющего чётко обозначенный вход и выход. Применение подпрограмм экономит оперативную память, т.к. для хранения переменных, использующихся только в подпрограмме, память отводится только на время выполнения подпрограммы.

Большинство языков программирования содержат следующие разновидности подпрограмм:
- встроенные подпрограммы;
- библиотечные подпрограммы;
- подпрограммы, создаваемые программистом.

В Python подпрограммы реализуются в виде функций. Различают библиотечные (стандартные) функции и функции, создаваемые программистом.

Библиотечные (стандартные) функции

Python предоставляет стандартные функции, имеющие определённые имена и выполняющие стандартные действия. Эти функции, определённы в языке и объединены в модули. Чтобы использовать библиотечную функцию, необходимо подключить к программе соответствующий модуль. Чтобы вызвать такую функцию, достаточно в теле основной или пользовательской функции указать её имя и список фактических параметров.

Функции, создаваемые программистом

Python позволяет программисту определить свою собственную функцию и в дальнейшем использовать ее точно так же, как и библиотечные функции. Собственную функцию можно как объявить и использовать в одном файле, так и поместить в файл (модуль) и импортировать её в другие программы (модули).

5.2. Формальные и фактические параметры

Подпрограмма (в случае Python – функция) составляется формально. Используемые в ней переменные, называются **формальными**, так как они не существуют в том же качестве, как обычные переменные и константы.

Формальные параметры – это идентификаторы переменных, через которые передается информация из основной программы в подпрограмму и обратно. Их имена используются в теле подпрограммы, но на самом деле она будет обрабатывать фактические параметры, переданные при вызове, которые могут иметь совершенно другие имена. При вызове подпрограммы из основной программы формальные параметры заменяются **фактическими**, между ними устанавливается взаимно однозначное соответствие, удовлетворяющее одному условию: соответствующие друг другу параметры должны совпадать по порядку следования и по типу. Блок подпрограммы выполняется для фактического набора параметров.

Фактические параметры – это информация о том, какие значения нужно передать аргументам подпрограммы и каким именам программы передать значения результатов подпрограммы.

Описание подпрограммы задает формальную схему обработки информации, в то время как оператор вызова подпрограммы осуществляет настройку этой формальной схемы на конкретную обработку фактических данных.

5.3. Локальные и глобальные переменные

Переменная – это именованная область памяти, в которой хранятся данные определенного типа. У переменной есть имя и значение. Имя служит для обращения к области памяти, в которой хранится значение. Во время выполнения программы значение переменной можно изменять. Перед использованием любая переменная должна быть описана.

Область действия идентификатора – это часть программы, в которой его можно использовать для доступа связанной с ним области памяти. В зависимости от области действия переменная может быть **локальной** или **глобальной**.

Если переменная определена внутри подпрограммы, она называется **локальной**, область ее действия – от точки описания до конца подпрограммы, включая все вложенные подпрограммы. Если переменная определена вне любого блока, она называется **глобальной** и областью ее действия считается файл, в котором она определена, от точки описания до его конца.

Python позволяет связать идентификатор внутри некоторого блока с глобальным идентификатором, это можно сделать, указав в начале блока на необходимость использования глобального идентификатора. Это делается с помощью оператора `global`:

```
global _идентификатор_
```

Предпочтительно не использовать `global`, так как это ухудшает понимание кода.

5.4. Функции

Под **функцией** понимается совокупность операторов, совместно выполняющих некоторое действие, имеющее определенную цель, оформленная в соответствии с требованиями языка программирования.

С понятием функции в языке Python связано два следующих компонента:
- описание функции;
- вызов функции.

Описание функции состоит из двух частей: заголовка и тела. Описание функции имеет следующую форму записи:

```
def имя_функции ([список_параметров]):    #заголовок функции
    #объявления и операторы
    #или, другими словами, тело функции
    #тело функции выделяется табуляцией
```

В языках программирования с **явной типизацией** тип возвращаемого значения задаётся при описании функции, но т.к. Python обладает **строгой неявной динамической типизацией**, задавать тип возвращаемого значения не нужно.

Тело функции представляет собой последовательность объявлений и операторов, описывающих определенный алгоритм. Важным оператором тела функции является оператор возврата в точку вызова: **`return [выражение].`** Оператор **`return`** имеет двойное назначение. Он обеспечивает немедленный возврат в вызывающую функцию и может использоваться для передачи вычисленного значения функции. В теле функции может быть несколько операторов **`return`,** но может не быть и ни одного. В последнем случае возврат в вызывающую программу происходит после выполнения последнего оператора тела функции, при том такая функция считается функцией, не возвращающей значение.

Вызов функции может быть оформлен в виде оператора, если у функции *отсутствует* возвращаемое значение, или в виде выражения, если *существует* возвращаемое значение.

В первом случае оператор имеет следующий формат:

```
имя_функции([список_аргументов])
```

Например: `print(“Hello World!”)`

Во втором случае выражение записывается следующим образом:

```
Переменная = имя_функции (список_аргументов);
```

Например: `z = max(x, y)`

Значение вычисленного выражения является возвращаемым значением функции. Возвращаемое значение передается в место вызова функции и является результатом ее работы.

Число и типы фактических параметров (аргументов) должны совпадать с числом и типом формальных параметров (параметров) функции. При вызове функции фактические параметры (аргументы) подставляются вместо формальных параметров (параметров).

Ниже приводится пример функции с возвращаемым значением.

Пример 1. Составить программу, содержащую обращение к функции вычисления максимума из двух чисел.

Возможное решение данной задачи имеет вид:

```
//описание функции max находится в данном файле ниже тела главной #
описание пользовательской функции
def max(a, b):  #Заголовок функции
    if a > b:       #Тело функции
        return a    #Тело функции
    else:           #Тело функции
        return b    #Тело функции

x = int(input('Введите x: '))
y = int(input('Введите y: '))
z = max(x, y)       #Вызов функции
print('z =', z)
```

Описание функции находится в одном файле с главной программой.

Вызов функции является выражением в правой части оператора присваивания `z = max(x,y)`, при выполнении которого **значения аргументов x и y подставляются вместо параметров a и b соответственно** (передача параметров в функцию по значению). После выполнения тела функции **возвращаемое значение передается в место вызова функции и присваивается переменной z.**

В рассмотренной программе функция имеет возвращаемое значение. Существуют задачи, которые не требуют передачи возвращаемого значения. Такой пример приведен ниже.

Пример 2. Составить программу, которая получает последовательно имена и возраст трёх друзей Полли, а потом выводит на экран эти данные в формате: `_имя_друга_ is _возраст друга_ years old.`

```
def print_age(name, years):                    #Заголовок процедуры
  print(name + ' is ' + str(years) + ' years old.') #Тело процедуры

friend = input('Name: ')
age = int(input(' age: '))
print_age(friend, age)                         #Вызов процедуры
friend = input('Name: ')
age = int(input(' age: '))
print_age(friend, age)                          #Вызов процедуры
friend = input('Name: ')
```

```
age = int(input(' age: '))
print_age(friend, age)                          #Вызов процедуры
```

У функции `print_age` отсутствует возвращаемое значение, поэтому обращение к функции осуществляется оператором вызова функции `print_age(friend, age)`, то есть нету этапа присваивания, функция просто выполняется.

5.5. Аргументы вида *args и **kwargs

Помимо стандартных аргументов, можно использовать аргумент вида `*args`, позволяющий использовать дальнейшие передаваемые объекты в виде кортежа. Это позволяет реализовывать, например, функцию max, которая принимает неограниченное количество чисел:

```
def max(*nums):
    max_num = nums[0]
    for num in nums:
        if num > max_num:
            max_num = num
    return max_num
```

Результат вызова функции `max(4, 6, 3)`:

```
6
```

Так же существует возможность использовать аргумент вида `**kwargs`, отличающийся от `*args` тем, что вместо кортежа используется словарь. Пример, демонстрирующий работу функции с подобным аргументом:

```
def f(**kwargs):
    for key in kwargs.keys():
        print(key, kwargs[key])
```

Результат работы для `f(a=4, b=3, c=6)`:

```
a 4
b 3
c 6
```

Можно использовать в функции оба представленных аргумента, при этом `*args` должно стоять раньше `**kwargs`

Подробнее использование упомянутых выше типов данных описано в следующих практических.

5.6. Итерация и рекурсия. Основные понятия

Итеративный процесс можно определить как процесс вычислений, основанный на повторении последовательности операций, при котором на каждом шаге повторения используется результат предыдущего шага. Примерами итеративных алгоритмов могут служить алгоритмы вычисления факториала натурального числа,

последовательности чисел Фибоначчи, вычисление суммы членов последовательности с заданным числом слагаемых и т.д. Итеративные процессы программируются обычно с помощью циклических конструкций.

Пример. Вычисление факториала числа *n*.

```
def factor(n):
   f = 1
   for i in range(2, n+1):
          f *= i
   return f
```

В то же время повторение последовательности действий можно осуществить и другим методом, основанным на принципе возможности явного включения в конструкции языка программирования конструкции того же вида. В частности, в языке программирования Python всякая функция может вызывать сама себя. Такой вызов называется **рекурсией.** Функция, вызывающая саму себя, называется **рекурсивной**.

Вообще алгоритм решения задачи называется *рекурсивным*, если он выражается в терминах решения более простой версии той же задачи и, кроме того, в нем предусмотрено решение самого простого варианта задачи.

При рекурсивном вызове функций в Python используется стековая организация хранения локальных параметров.

Стек является простейшей динамической структурой организации данных. Добавление элементов в стек и выборка из него выполняются из одного конца, называемого вершиной стека. Говорят, что стек реализует принцип обслуживания LIFO (last in – first out, последним пришел – первым ушел (обслужен)).

Стеки широко применяются в системном программном обеспечении, компиляторах, в различных рекурсивных алгоритмах.

Различия между итерацией и рекурсией:

1) итерации необходим цикл и вспомогательные величины, в то время как рекурсия обходится без вспомогательных величин и обычно проще для понимания, короче и нагляднее итерации;

2) итерация требует меньше места в оперативной памяти и меньших затрат машинного времени, чем рекурсия, которой необходимы затраты на управление стеком;

3) при реализации рекурсии велика вероятность переполнения стека.

Поэтому при выборе способа описания функции следует вначале решить, чему отдать предпочтение – эффективности программы или ее компактности. Однако **следует избегать рекурсий там, где есть очевидное итерационное решение.** Вместе с тем существует много хороших примеров применения рекурсии (построение графических образов разнообразных узоров, работа с двоичными деревьями).

При рекурсивном описании функции последняя всегда должна содержать, по крайней мере, одну альтернативную нерекурсивную ветвь алгоритма с условием окончания вычислений, заканчивающуюся оператором возврата результата.

Различают две формы рекурсивных подпрограмм:

- **прямая рекурсия**;
- **косвенная рекурсия**.

В **первом случае** функция содержит вызов этой же функции.

Косвенная рекурсия возникает в том случае, когда две или более функций вызывают друг друга. В случае двух функций: некоторая функция обращается к другой функции, а та в свою очередь вызывает первоначальную функцию. Для предотвращения зацикливания одна или обе функции должны проверять условие завершения.

Пример: Составить рекурсивную функцию, вычисляющую факториал числа n по формуле (учесть, что 0! = 1):

$$n! = \begin{cases} 1, n = 1; \\ n \cdot (n-1)!, n > 1. \end{cases}$$

```
def fact(num):
      if num == 0:
            return 1
      return num * fact (num - 1)

n = int(input('Введите число '))
print(fact (n))
```

То же самое можно записать короче:

```
def fact(n):
      return n*fact(n-1) if (n > 1) else 1
```

Пример: Распечатать последовательность, состоящую из *n* букв A и *n* букв B.

```
def abn(n):
      print("A")
      if n > 1:
            abn(n-1)
      print("B")
```

5.7. Итераторы, итерируемые объекты и генераторы в языке Python

Итератором в Python называются объекты, для которых реализованы методы `__next__`, который возвращает следующий элемент итератора или вызывает исключение, если элементов не осталось и метод `__iter__`, возвращающий итератор. Для вызова данных методов используются функции `next` и `iter`.

Итераторы используются в циклах for, так как он преобразует подающийся объект в итератор. Данный объект называется **итерируемым объектом**, то есть с

помощью функции `iter` из него можно получить итератор. Например, Объект, возвращаемый функцией `range`, является итерируемым. Представим на его примере работу итераторов:

```
≫a = iter(range(1, 4))
≫a = iter(a)
≫next(a)
1
≫next(a)
2
≫next(a)
3
≫next(a)
Traceback (most recent call last):
  File "<pyshell#24>", line 1, in <module>
    next(a)
StopIteration
```

Как видно, когда элементы закончились, произошла ошибка при использовании функции next. Этого можно избежать, задав второму параметру какое-либо значение, возвращаемое вместо ошибки:

```
≫next(a, 0)
0
```

Генераторами в Python называются функции, в которых используется ключевое слово `yield`, и генераторные выражения вида **`(объект for переменная in итерируемый объект)`**. Все генераторы являются итераторами, то есть для них работают функции `next` и `iter`.

Главным преимуществом генераторов является тот факт, что они не хранят в памяти все возвращаемые объекты и не тратят время на их нахождение пока это не потребуется. Допустим, нам нужно в программе последовательно находить числа Фибоначчи, тогда лучше всего будет использовать представленную функцию:

```
≫def fib():
    a, b = 0, 1
    while True:
        a, b = b, a+b
        yield a
```

При вызове данная функция вернёт генератор, позволяющий находить следующее число Фибоначчи при использовании функции `next`:

```
≫a = fib()
≫for i in range(8):
    print(next(a))
1
1
2
3
5
8
13
21
```

Вопросы и задания

Задание I

Изучить порядок объявления и описания функций, выполнить общее для всех задание:

реализовать (отладить и запустить) программы из блока теории (**Пример 1** и **Пример 2**), оформить в тетради, записав условие, блок-схему и результат работы.

Задание II

В соответствии с вариантом составить и реализовать программу, использующую пользовательскую функцию:

1. Три прямоугольных треугольника заданы катетами. Определить, площадь какого из них больше.

2. Три отрезка заданы координатами вершин. Определить, есть ли среди них отрезки равной длины.

3. В порт в среднем приходит три корабля в день. Какова вероятность того, что в порт придёт 2 корабля? 4 корабля? Вычисление вероятности производится по формуле: $P = 3^k e^{-3}/ k!$.

4. Найти среднее арифметическое действительных корней трёх квадратных уравнений.

5. Даны три различных бруска, имеющих форму кубов с заданным произвольным ребром. Из каждого выпилена правильная четырехугольная пирамида (основание – грань куба). Вычислить объем материала, ушедшего в отходы.

6. Даны действительные числа *a* и *b*. Вычислите,

$g(1.2, a)+g(b, a)-g(2a- 1, ab)$, где $g(x,y)=\dfrac{\left(x^2+y^2\right)}{x^2+2xy+3y^2+4}$.

7. Заданы длины рёбер произвольной треугольной пирамиды. Вычислить площадь полной поверхности этой пирамиды.

8. Вычислить объем «снеговика», состоящего из трех шаров с произвольными радиусами.

9. Каждое из трех колец задано внутренним и внешним радиусами. Чему равна площадь наибольшего кольца?

10. Вычислить значение выражения a! + b! + c! + d!.

11. Вычислить площадь «елки», состоящей из четырех треугольников с известными сторонами.

12. Найти минимальное значение среди корней трех квадратных уравнений (каждое уравнений задано тройкой коэффициентов).

13. Даны действительные числа a и b. Вычислить

$$\max(u + v^2, 3.14),$$

где $u = \max(a, b)$, $v = \max(ab, a + b)$, а max – подпрограмма для вычисления максимального значения для двух данных чисел.

14. Даны радиусы трех шаров. Вычислить среднее арифметическое объемов этих шаров.

15. Дана арифметическая прогрессия a_n,, a_1 – первый ее элемент, d – разность. Найти второй, третий, четвертый и седьмой элементы и их сумму.

16. Даны действительные числа s и t. Вычислить $f(t, -2s, 1.17) + f(2.2, t, s - t)$,

$$f(x, y, z) = \frac{2x - y - \sin z}{5 + |z|}.$$

17. Найти наибольшее из пяти заданных чисел, используя подпрограмму нахождения наибольшего из двух чисел.

18. Вычислить значение функции:

$$y = \varphi(3) + \varphi(7.6) \cdot (\varphi(11.2) - \varphi(5)), \text{ если } \varphi(x) = \sqrt{2x^2 - 4x + e^x}$$

19. Вычислить значение функций

$$f(x) = 2\sin x - 4.2\cos x + e^x \text{ и } g(x) = \frac{2x^2}{10\sqrt{5x}}.$$

для значений $x =$ 0,1; 0,35; 0,53.

20. Вычислить значение функции $x = 2f(15) – 4f(-1)g(11) + g(32)$, если

$$f(x) = 10x + e^{2x};\ g(x) = \sqrt{2x + x^3}\ .$$

21. Из куба со стороной a выпилили куб со сторой b ($a > b$). Какой процент материала ушёл в отходы?

22. Каждая из трёх окружностей задана координатами центра и координатами точки на окружности. Какова наибольшая длина окружности?

23. Найти наименьшее из четырёх заданных чисел, используя подпрограмму нахождения наименьшего из двух чисел.

24. Даны действительные числа a и b. Вычислить

$$\min(2u - v^2, 4.5),$$

где $u = \min(a, b)$, $v = \min(a/b, a - b)$, а min – подпрограмма для вычисления минимального значения для двух данных чисел.

25. Вычислить большие вещественные корни квадратных уравнений $ax^2 + bx + c = 0$ и $tx^2 + px + q = 0$, если они существуют. В противном случае вывести на экран монитора соответствующее сообщение.

Задание III

В соответствии с вариантом, используя прямую рекурсию, составить и реализовать программу. Проверить правильность работы программы альтернативными методами/средствами. Скриншот результата проверки вставить в отчет.

1. Описать рекурсивную функцию NOD(A, B) целого типа, находящую наибольший общий делитель (НОД) двух целых положительных чисел A и B, используя алгоритм Евклида: НОД(A, B) = НОД(B, A mod B), если B ≠ 0; НОД(A, 0) = A. С помощью этой функции найти НОД(A, B), НОД(A, C), НОД(A, D), если даны числа A, B, C, D.

2. Для n = 12 найти числа Фибоначчи. Числа Фибоначчи: F(0) = 1, F(1) = 1 F(n) = F(n - 2) + F(n - l)

3. Даны целые числа m и n, где $0 \le m \le n$, вычислить, используя рекурсию, число сочетаний C(n, m) по формуле: $C_n^0 = C_n^n = 1$, $C_n^m = C_{n-1}^m + C_{n-1}^{m-1}$ при $0 \le m \le n$. Воспользовавшись формулой $C_n^m = \dfrac{n!}{m!(n-m)!}$ можно проверить правильность результата.

4. Опишите рекурсивную функцию, которая по заданным вещественному x и целому n вычисляет величину x^n согласно формуле:

$$x^n = \begin{cases} 1, если\ n = 0, \\ \dfrac{1}{x^{|n|}}, n < 0, \\ x \cdot \left(x^{n-1}\right), n > 0. \end{cases}$$

5. Задана последовательность положительных чисел, признаком конца которых служит отрицательное число. Используя рекурсию, подсчитать количество чисел и их сумму.

6. Напишите рекурсивную функцию $K(n)$, которая возвращает количество цифр в заданном натуральном числе n, используя формулу:

$$K(n) = \begin{cases} 1, если\ n < 10, \\ K(n\ div 10) + 1, n \ge 10. \end{cases}$$

7. Определим функцию $S(n)$, вычисляющую сумму цифр заданного натурального числа:

$$S(n) = \begin{cases} n, \ если\ n < 10, \\ S(n\ div\ 10) + n \bmod 10, \ \ если\ n \ge 10. \end{cases}$$

Описать рекурсивную функцию DigitSum(K) целого типа, которая находит сумму цифр целого числа K, не используя оператор цикла. С помощью этой функции найти суммы цифр для пяти данных целых чисел.

8. Дан вектор X из n вещественных чисел. Найти минимальный элемент вектора, используя вспомогательную рекурсивную функцию, находящую минимум среди элементов вектора X, начиная с n -го.

9. Напишите рекурсивную функцию для нахождения биномиальных коэффициентов (для заданного $M \ge i \ge j > 0$ вычислите все C_i^j):

$$C_n^m = \begin{cases} 1, \text{ если } n = m \text{ или } m = 0; \\ C_{n-1}^{m-1} + C_{n-1}^m. \end{cases}$$

10. Напишите программу вычисления функции Аккермана для всех неотрицательных целых аргументов m и n:

$$A(m,n) = \begin{cases} A(0,n) = n+1, \\ A(m,0) = A(m-1,1), m > 0, \\ A(m,n) = A(m-1, A(m,n-1)), m,n > 0. \end{cases}$$

11. Для заданных границ интегрирования a и b вычислите значение определенного интеграла следующего вида:

$$\int e^{ax} \cos^n x dx = \begin{cases} \dfrac{e^{ax} \cos^{n-1} x (a \cos x + n \sin x)}{a^2 + n^2} + \dfrac{n(n-1)}{a^2 + n^2} \int e^{ax} \cos^{n-2} x dx, \; n \ge 2, \\ \dfrac{-e^{ax} (\sin x + a \cos x)}{a^2 + n^2}, n = 1, \\ \dfrac{e^{an}}{a}, n = 0. \end{cases}$$

12. Для заданных границ интегрирования a и b вычислите значение определенного интеграла следующего вида:

$$\int \cos^n x dx = \begin{cases} \dfrac{\cos^{n-1} x \sin x}{n} + \dfrac{n-1}{n} \int \cos^{n-2} x dx, \; n > 2, \\ \dfrac{x}{2} + \dfrac{1}{4} \sin 2x, n = 2, \\ \sin x, n = 1. \end{cases}$$

13. Для заданных границ интегрирования a и b вычислите значение определенного интеграла следующего вида:

$$\int \sin^n x dx = \begin{cases} -\dfrac{\sin^{n-1} x \cos x}{n} + \dfrac{n-1}{n} \int \sin^{n-2} x dx, \; n > 2, \\ \dfrac{x}{2} - \dfrac{1}{4} \sin 2x, n = 2, \\ -\cos x, n = 1. \end{cases}$$

14. Для заданных границ интегрирования a и b вычислите значение определенного интеграла следующего вида:

$$\int \frac{dx}{\sin^n x} = \begin{cases} -\frac{1}{n-1} \cdot \frac{\cos x}{\sin^{n-1} x} + \frac{n-2}{n-1} \int \frac{dx}{\sin^{n-2} x}, \; n \geq 2, \\ \ln tg \frac{x}{2}, n = 1, \\ x, n = 0. \end{cases}$$

15. Для заданных границ интегрирования a и b вычислите значение определенного интеграла следующего вида:

$$\int \frac{dx}{\cos^n x} = \begin{cases} \frac{1}{n-1} \cdot \frac{\sin x}{\cos^{n-1} x} + \frac{n-2}{n-1} \int \frac{dx}{\cos^{n-2} x}, \; n \geq 2, \\ \ln tg \left(\frac{\pi}{4} + \frac{x}{2} \right), n = 1, \\ x, n = 0. \end{cases}$$

16. Для заданных границ интегрирования a и b вычислите значение определенного интеграла следующего вида:

$$\int x^n e^{ax} dx = \begin{cases} \frac{x^n e^{ax}}{a} - \frac{n}{a} \int x^{n-1} e^{ax} dx, \; n > 1, \\ \frac{e^{ax}}{a^2} (ax - 1), n = 1. \end{cases}$$

17. Для заданных границ интегрирования a и b вычислите значение определенного интеграла следующего вида:

$$\int x^n a^{mx} dx = \begin{cases} \frac{x^n a^{mx}}{n \ln a} - \frac{n}{m \ln a} \int x^{n-1} a^{mx} dx, \; n > 1, \\ \frac{x a^{mx}}{m \ln a} - \frac{a^{mx}}{m (\ln a)^2}, n = 1. \end{cases}$$

18. Для заданных границ интегрирования a и b вычислите значение определенного интеграла следующего вида:

$$\int \ln^n x dx = \begin{cases} x \ln^n x - n \int \ln^{n-1} x dx, \; n > 1, \\ x \ln x - x, n = 1. \end{cases}$$

19. Для заданных границ интегрирования a и b вычислите значение определенного интеграла следующего вида:

$$\int x^m \ln^n x dx = \begin{cases} \dfrac{x^{m+1}}{m+1}\ln^n x - \dfrac{n}{m+1}\int x^m \ln^{n-1} x dx,\ n>1, \\ x^{m+1}\left[\dfrac{\ln x}{m+1} - \dfrac{1}{(m+1)^2}\right], n=1. \end{cases}$$

20. Для заданных границ интегрирования a и b вычислите значение определенного интеграла следующего вида:

$$\int \frac{dx}{\left(a^2+x^2\right)^n} = \begin{cases} \dfrac{1}{2(n-1)a^2}\left[\dfrac{x}{\left(a^2+x^2\right)^{n-1}} + (2n-3)\int \dfrac{dx}{\left(a^2+x^2\right)^{n-1}}\right], n>1, \\ \dfrac{1}{a}arctg\dfrac{x}{a}, n=1. \end{cases}$$

21. Для заданных границ интегрирования a и b вычислите значение определенного интеграла следующего вида:

$$\int tg^n x dx = \begin{cases} \dfrac{tg^{n-1}x}{n-1} - \int tg^{n-2} x dx,\ n \geq 2, \\ -\ln|\cos x|, n=1, \\ x, n=0. \end{cases}$$

22. Для заданных границ интегрирования a и b вычислите значение определенного интеграла следующего вида:

$$\int ctg^n x dx = \begin{cases} -\dfrac{ctg^{n-1}x}{n-1} - \int ctg^{n-2} x dx,\ n \geq 2, \\ \ln|\sin x|, n=1, \\ x, n=0. \end{cases}$$

23. Для заданных границ интегрирования a и b вычислите значение определенного интеграла следующего вида:

$$\int x^m \sin ax dx = \begin{cases} -\dfrac{x^m}{a}\cos ax + \dfrac{m}{a}\int x^{m-1}\cos ax dx,\ m \geq 1, \\ -\dfrac{\cos ax}{a}, m=0. \end{cases}$$

24. Для заданных границ интегрирования a и b вычислите значение определенного интеграла следующего вида:

$$\int x^m \cos ax dx = \begin{cases} -\dfrac{x^m}{a}\sin ax + \dfrac{m}{a}\int x^{m-1}\sin ax dx,\ m \geq 1, \\ -\dfrac{\sin ax}{a}, m=0. \end{cases}$$

25. Напишите рекурсивную функцию, перемножающую два целых числа, одно из которых неотрицательно, без использования операции умножения.

Контрольные вопросы

1. Что такое подпрограмма?
2. Как подпрограмму можно реализовать в Python?
3. Какие параметры называются формальными/фактическими?
4. Параметры по умолчанию.
5. Аргументы вида *args и **kwargs.
6. Что такое переменная?
7. Что такое локальная переменная? Что такое глобальная переменная?
8. Что такое область действия идентификатора?
9. Какова область действия локальных идентификаторов? Какова область действия глобальных идентификаторов?
10. Каково назначение ключевого слова global?
11. Что такое функция?
12. Перечислите составные части описания функции.
13. Как описывается функция пользователя в программе?
14. Что такое возвращаемое значение функции?
15. Как осуществляется вызов функций?
16. Какой процесс называется итеративным?
17. Каким образом реализуются итеративные процессы?
18. Какой алгоритм называется рекурсивным?
19. Какая функция называется прямо рекурсивной? Косвенно рекурсивной?
20. Что такое стек? В какой последовательности происходит заполнение стека и выбор элементов из стека?
21. Что должно обязательно присутствовать в теле рекурсивно описанной функции?
22. Перечислите различия между итерацией и рекурсией.
23. Что произойдет, если рекурсивный алгоритм будет вызывать сам себя «бесконечное» число раз?
24. Как предотвратить бесконечное выполнение рекурсивного алгоритма?
25. Верно ли, что решение задачи, реализуемое рекурсивным алгоритмом, можно выразить, используя итерацию?
26. Итераторы, итерируемые объекты и генераторы в языке Python

Пример выполнения задания II

Задание II

Задание. Задавая с клавиатуры координаты вершин треугольника, определить, является ли он равносторонним.

Решение

1. Математическая модель

Треугольник является равносторонним, если длины его сторон равны. Находим длины сторон треугольника, используя подпрограмму нахождения расстояния между двумя точками.

Аргументы: a1, b1, c1, a2, b2, c2 целого типа – координаты вершин треугольника.

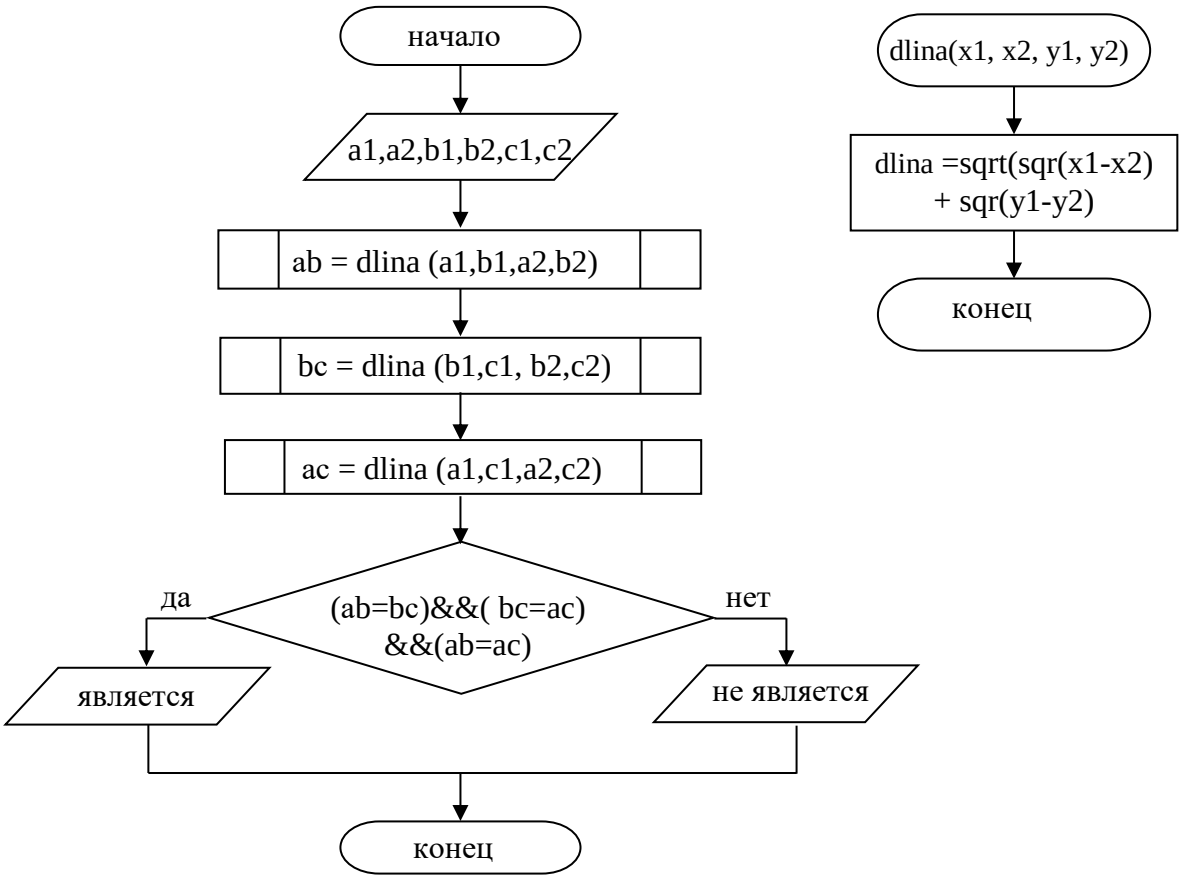
Результаты: текстовое сообщение.

Промежуточные величины: ab, bc, ac вещественного типа – длины сторон треугольника.

Расстояние между двумя точками A(x1,y1) и B(x2,y2) находим по формуле:

$$AB = sqrt(sqr(x2 - x1) + sqr(y2 - y1).$$

2. Алгоритм – основная программа: – подпрограмма:



3. Программа

```
import math
def dlina(x1, y1, x2, y2):
    return math.sqrt((x1-x2)**2 + (y1-y2)**2)
```

```
    a1, a2, b1, b2, c1, c2 = map(int, input("Задайте координаты точек A,
B, C: ").split())
    ab = dlina(a1, a2, b1, b2)
    bc = dlina(b1, b2, c1, c2)
    ac = dlina(a1, a2, c1, c2)

    if ab == bc == ac:
        print("Треугольник является равносторонним")
    else:
        print("Треугольник не является равносторонним")
```

4. Результат работы программы

```
Задайте координаты точек A, B, C: 0 0 4 0 2 2
Треугольник не является равносторонним
```

Задание III

Задание. Для заданных границ интегрирования a и b вычислите значение определенного интеграла следующего вида:

$$\int x^n e^{cx} dx = \begin{cases} \dfrac{x^n e^{cx}}{c} - \dfrac{n}{c}\int x^{n-1} e^{cx} dx,\ n > 1, \\ \dfrac{e^{cx}}{c^2}(cx-1), n = 1; \end{cases}$$

Решение

1. Математическая модель

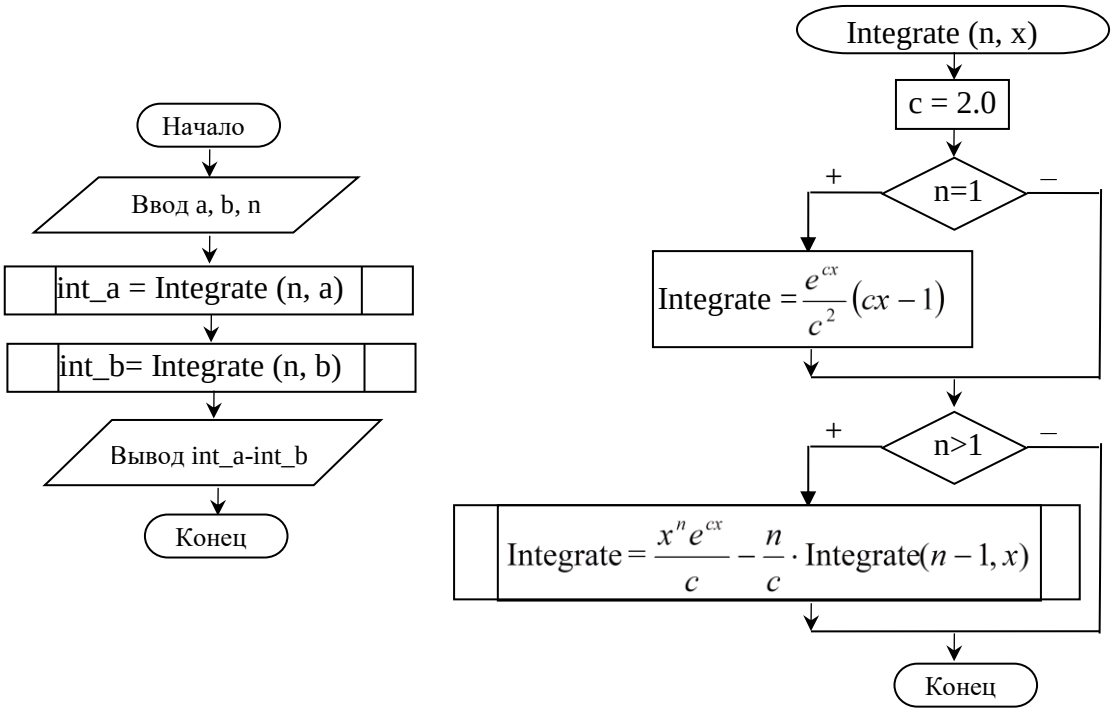
Аргументы: границы интегрирования a, b целого типа;

степень n целого типа.

Результаты: разность int_b - int_a вещественного типа.

Промежуточные величины: значение интеграла для нижней границы интегрирования int_a; значение интеграла для верхней границы интегрирования int_b (вещественного типа).

2. Алгоритм



3. Программа

```
// вычисление значения определенного интеграла
from math import exp


C = 2.0


def integrate(n, x):
    if n == 1:
        return exp(C * x) / (C * C) * (C * x-1)
    return pow(x, n) * exp(C * x) / C - (n / C) * integrate(n-1, x)


a, b = sorted(map(float, input("Введите a и b: ").split()))
n = int(input("Введите степень n: "))
int_a = integrate(n, a)
int_b = integrate(n, b)

print("Значение определённого интеграла при a =", a, "и b =", b, "равно",
integral_value := int_b-int_a)
```

4. Результат работы программы

```
Введите a и b: -3 3
Введите степень n: 5
Значение определённого интеграла при a = -3.0 и b = 3.0 равно
25568.135436929646
```

5. Проверка правильности работы программы

Для проверки дополним предыдущий код фрагментом, приведённым ниже. Воспользуемся модулем sympy для работы с символьной математикой, а именно функцией sympy.integrate, позволяющей находить неопределённые и определённые интегралы:

```
import sympy as sp
x = sp.Symbol("x")

f = (x**n)*sp.exp(C*x)
print("Проверяющее значение -", verifying_value := sp.integrate(f, (x, a, b)))
print("Разница в полученных значениях  -", abs(integral_value - verifying_value))
```

Результат проверки для заданных выше значений:

```
Проверяющее значение - 25568.1354369296
Разница в полученных значениях  - 0
```

Список литературы, рекомендуемый к использованию по данной теме

- [4] стр. 491-502
- [6] стр. 93-108
- [11] Теоретический материал: Функции (informatics.msk.ru) - https://informatics.msk.ru/mod/book/view.php?id=4384

Практическое занятие №6
Использование числовых массивов

ЦЕЛЬ: освоение способов инициализации списка, приобретение навыков организации ввода-вывода и обработки списка.

Знания, умения и владения, приобретаемые обучающимся в результате освоения темы, в рамках формируемых компетенций (ПК-1: ИД-1$_{ПК-1}$, ИД-2$_{ПК-1}$**;** ПК-2: ИД-1$_{ПК-2}$, ИД-2$_{ПК-2}$): см. приложение 2: вырабатывается навык разработки математических, информационных и имитационных моделей, а также формируется способность к разработке и применению алгоритмических и программных решений в области прикладного программирования при использовании структурированных типов данных с умением объяснить теоретическую и практическую часть темы (формируется часть указанных компетенций).

Теоретическая часть
6. Структурированные упорядоченные типы данных

6.1. Понятие структурированного типа данных

Синонимами названия «базовый тип» являются названия «стандартные», «предопределенные». Реализация этих типов заложена в стандарте языка, и программист не вправе изменить предопределенное. Производным типом (определяемым программистом) является такой тип, который должен быть описан пользователем перед употреблением. К производным типам можно отнести массивы, перечисления, функции, структуры, ссылки, объединения и классы. Полезно помнить, что тип данного определяет его размещение в памяти и набор операций.

Структурированный тип представляет собой набор элементов – данных, объединенных программистом в единый блок в соответствии с особенностями решаемой задачи. Данные, входящие в блок, могут быть однотипными или разнотипными, весь блок данных имеет общее имя. Каждый элемент отличается от других либо порядковым номером (индексом) – в массиве, либо своим именем и, возможно, типом – в структуре, списке.

Любой из структурированных типов данных характеризуется множественностью образующих этот тип элементов. Переменная или константа структурированного типа всегда имеет несколько компонент. Каждая из этих компонент, в свою очередь, может принадлежать структурированному типу, что позволяет говорить о возможной вложенности типов.

В C++ можно выделить следующие структурированные типы: массивы, структуры, объединения, классы, файлы.

В Python к составным типам относят: списки, строки, множества, словари и т.д.

6.2. Понятие типа массив

При использовании простых переменных каждой области памяти для хранения данных соответствует своё имя. Если с группой величин одинакового типа требуется выполнять однообразные действия, им дают одно имя, а различают по по-

рядковому номеру. Это позволяет компактно записывать множество операций с помощью циклов. Конечная именованная последовательность однотипных величин называется **массивом**.

Итак, традиционно под словом «массив» в мире программирования понимается конечная именованная последовательность однотипных величин. Но в языке Python 3, хоть и имеется возможность использования массивов (встроенный модуль `array`, а также `array` из модуля `numpy`), это не является частой практикой. Чаще принято использовать списки (`lists`), которые выполняют почти те же самые функции. Список – конечная последовательность величин. Как видите, пропали слова «именованная» и «однотипных». Это говорит о том, что списки могут существовать без имени и содержать величины любых типов вместе. Таким образом, списки имеют определённые преимущества перед массивами, что в очередной раз отсылает нас к гибкости и мощности инструментов, предоставляемых языком программирования Python 3.

6.3. Одномерные массивы. Списки Python

Одномерный массив – массив, с одним параметром (измерением), характеризующим количество элементов одномерного массива. **Размерность** `n` = 1.

На рисунке 7.1 показана структура целочисленного одномерного массива `a`. Размер этого массива – 10 элементов (ячеек).

-5	2	0	-6	12	56	8	23	4	-3
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
a[-10]	a[-9]	a[-8]	a[-7]	a[-6]	a[-5]	a[-4]	a[-3]	a[-2]	a[-1]

Рисунок 7.1 – Одномерный массив

Максимальный индекс одномерного массива `a` равен 9, но размер массива 10 ячеек, так как нумерация ячеек массива всегда начинается с 0. Индекс ячейки – это целое неотрицательное число, по которому можно обращаться к каждой ячейке массива и выполнять какие-либо действия над ней (ячейкой).

```
int a[10];//пример объявления массива C++, изображенного на рисунке 7.1,
                // где int – целочисленный тип данных;
                //     a – имя одномерного массива;
                //     10 – размер одномерного массива a.
//инициализация:
a = {-5, 2, 0, -6, 12, 56, 8, 23, 4, -3};
```

В Python (список аналогичный массиву C++):

```
a = [-5, 2, 0, -6, 12, 56, 8, 23, 4, -3]
```

Список (list) в Python - упорядоченная изменяемая коллекция (контейнер) объектов произвольных типов (почти как массив, но типы могут отличаться).
Основные свойства списка:

- список хранит несколько элементов под одним именем (как и множество);
- элементы списка могут повторяться (в отличие от множества);

- элементы списка упорядочены и проиндексированы, доступна операция среза (как в строке);
- элементы списка можно изменять (в отличие от строки);
- элементами списка могут быть значения любого типа: целые и действительные числа, строки и даже другие списки.

6.4. Основные действия над списками

1. Инициализация списка: присвоение каждому элементу начального значения:

а) инициализация:

```
a = [0, 5, -7, 100, 15]    # a[0] = 0, a[1] = 5, a[2] = -7,
                           # a[3] = 100, a[4] = 15
```

Инициализация списка выполняется в квадратных скобках после знака равно, каждый элемент списка отделяется от предыдущего запятой.

Чтобы программно определить число элементов в таком списке, используется Функция `len`:

```
lenMas = len(a) # число элементов
```

Итак, для списка можно выполнить начальную инициализацию значений его элементов, для этого нужно задать список инициализирующих значений. Например, в году всегда 12 месяцев, значение числа дней в каждом месяце известно, значит, такая структура может быть задана списком:

```
month = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
```

Эквивалентом инициализации является простое **присваивание** вида:

```
month[0] = 31         # Январь
...
month[11] = 31        # Декабрь
```

```
Помимо этого, в Python реализована возможность обращения к элементу
списка с конца:
month[-12] = 31       # Январь
...
month[-1] = 31        # Декабрь
```

```
Также можно задать изначально пустой список:
a = []
```

б) ввод элементов списка с клавиатуры:

```
N = 10
a = [0] * N
print(" Введите массив: ")
for i in  range(N):
```

```
        print("\t",i, " элемент: ")
        a[i] = int(input())
```

в) формирование списка с помощью генератора псевдослучайных чисел:

```
from random import randint      # подключение модуля и взятие функции
                                #генерации псевдослучайных чисел
N = 10          # определяем количество элементов списка
a = [0]* N # инициализируем список нулями
for i in range(n):
    a[i] = randint(-10,10) # a[i] ∈ [-10, 10]
```

г) вычисление элементов списка по формуле:

```
for i in range(n):
     a[i] = 6 * i - 2;
```

2. Инициализация массива: использование генератора списков:

В Python реализована конструкция генератор списков, имеющая вид

```
[объект for переменная in итерируемый объект]
```

и позволяющая упростить написание и чтение кода. Ниже представлены примеры из предыдущего пункта, реализованные с помощью данной конструкции:

а) ввод элементов списка с клавиатуры (аналог 1б):

```
N = 10
a = [int(input) for i in range(N)]
```

б) формирование списка с помощью генератора псевдослучайных чисел (аналог 1в):

```
from random import randint # подключение модуля и взятие
                           # функции генерации псевдослучайных чисел
N = 10
a = [randint(-10, 10) for i in range(N)]
```

г) вычисление элементов списка по формуле (аналог 1г):

```
a = [6 * i – 2 for i in range(n)]
```

Если при создании списка нужно учитывать какие-либо условия, то можно использовать следующие конструкции:

```
[объект for переменная in итерируемый_объект if условие]
```

```
[объект1 if условие else объект2 for переменная in итерируемый_объект]
```

3. Вывод списка на экран:

```
a = [1, 2, 3, 4, 5]
print(" Массив A: ")
print(a)              # Вывод списка
[1, 2, 3, 4, 5]
print(*a)             # Вывод элементов списка
1 2 3 4 5
```

Обратите внимание на использование `*a` в функции `print`. Если мы передаём как аргумент выражение виде `*итерируемый_объект`, то данный объект распаковывается, передавая каждый его элемент как отдельный параметр.

4. Обработка списка

Для работы со списками реализовано множество методов. Основные из них представлены ниже, более подробную информацию можно узнать, используя функцию help.

`list.append(x)` – добавляет элемент в конец списка.

`list.pop(i=-1)` – удаляет i-ый элемент списка, возвращая его значение.

`list.extend(L)` – добавляет в список все элементы объекта L.

`list.insert(i, x)` – добавляет x в список под индексом x.

При работе со списками можно использовать ключевые слова и различные операторы:

`object in list` – возвращает True, если объект есть в списке, иначе False

`list1 + list2` – возвращает список, элементами которого являются сначала элементы list1, потом list2

`list * N` – возвращает результат сложения N списков.

Можно обращаться сразу к нескольким элементам списка, используя конструкцию срез `list[start:end:step]`. Возвращает элементы с индексами, принадлежащим полуотрезку [start, end) с шагом step. При присваивании значений срезу меняются значения в самом списке.

Если не задавать start и end, то будут задействованы все элементы до границ списка, если не задавать значение step, то оно будет принято за 1. Для задания start, end и step можно использовать отрицательные значения.

Пример работы со срезами для списка `a = [1, 2, 3, 4, 5, 6]`:

```
a[2:4]          # [3, 4]
a[:5]           # [1, 2, 3, 4, 5]
#[3:]           # [4, 5, 6]
a[::2]          # [1, 3, 5]
a[4:1:-1]       # [5, 4, 3]
a[::-1]         # [6, 5, 4, 3, 2, 1]
```

6.5. Кортежи

Помимо списков в Python реализован такой тип данных, как кортеж. Его главным отличием от списка является его неизменяемость, то есть нельзя изменить значение какого-либо элемента или добавить/убрать какой-либо элемент. Для инициализации кортежей используются символы (и), но в некоторых случаях их можно не использовать:

```
a = (1, 2, 3, 4, 5)
a = 1, 2, 3, 4, 5     # Данная запись аналогична записи выше
```

Для инициализации кортежей нет конструкции, аналогичной генератору списков. При использовании символов (и) в генераторе списков вместо [и] будет создан объект типа generator.

6.6. Строки в Python

Строки в Python работают практически идентично кортежам. Основными отличиями являются реализованные для них методы, вид при выводе и тот факт, что элементами строки могут быть только символы. Они являются неизменяемыми, операторы работают с ними аналогично. Для большей информации о реализованных специально для строк методов можно обратиться к документации или воспользоваться функцией `help`.

Для инициализации строки можно использовать функцию str или набор символов, заключённый в одинарные или двойные кавычки.

Стоит также отметить, что в Python не существует отдельного типа данных для символов. Для работы с одиночными символами необходимо использовать строки длины 1.

6.7. Многомерные массивы и особенности их реализации в Python

Многомерным массивом является массив, элементы которого являются массивами. Аналогом многомерных массивов в Python являются списки списков. Например:

```
a = [
     [1, 2, 3],
     [4, 5, 6],
     [7, 8, 9]
]
```

Записью `a[i]` мы обратимся к списку с индексом `i` из `a`, соответственно, запись `a[i][j]` аналогична записи `(a[i])[j]`, то есть мы обращаемся к элементу с индексом `j` из списка `a[i]`. Применим данные ниже операции:

```
a[1] = [10, 11, 12]
a[2][0] = 123
```

В итоге a будет иметь вид:

```
[
[1, 2, 3],
[10, 11, 12],
[123, 8, 9]
]
```

Элементом списка может быть любой объект Python, в том числе и функции, при этом размер их ограничен только памятью вашего устройства. Все представленные ниже записи корректны:

```
a = [1, "23", [23, 54], [412, [34, 34]]]
b = [min, max, sum, sorted]
print(b[0](1, 2))    # Выведет 1
```

Учитывайте, что list.copy не стоит применять на списках, содержащих другие изменяемые элементы, так как в этом случае их изменение в копии приведёт к изменению в оригинале и наоборот. Для копирования в этом случае следует применять функцию deepcopy из модуля copy. Аналогично с иными типами данных, которые могут хранить какие-либо объекты.

Вопросы и задания

Задание I

Выполнить общее для всех задание, изучив порядок описания, ввода-вывода и обработки массивов:

1. Набрать и отладить программу нахождения суммы элементов массива, стоящих в нечетных позициях. Оформить в тетради, записав условие, код, блок-схему и результат работы, объяснить назначение операторов 0-4.

```
from random import randint                          # Оператор 0
n = int(input("Введите количество элементов массива: "))
L = [randint(-10,10) for i in range(n)]             # Оператор 1
print(L)                                            # Оператор 2
S = 0                                               # Оператор 3
for i in range(0, n, 2):
    S += L[i]                                       # Оператор 4
print('S = ',S)
```

2. Набрать и отладить программу нахождения суммы элементов массива, кратных 3. Оформить в тетради, записав условие, код, блок-схему и результат работы, объяснить назначение операторов 0-3.

```
from random import randint
n = int(input("Введите количество элементов массива: "))
L = [0]*n                   # Оператор 0
for i in range(n):
    L[i] = randint(-10,10) # Оператор 1
print(L)
S = 0
for i in range(0, n):       # Оператор 2
   if L[i]%3==0:
        S += L[i]                # Оператор 3
print('S = ',S)
```

Задание II

В соответствии с вариантом составить и реализовать программу:

1. Даны два массива разных размеров. Определить, какие элементы первого массива и сколько раз встречаются во втором массиве.
2. Массив содержит $2n$ чисел. Из суммы первых n его элементов вычесть сумму последних n элементов.
3. Транспонировать массив, т.е. по a_1, a_2,..., a_n сформировать a_n, a_{n-1}, ..., a_1. В полученном массиве найти индекс минимального элемента.
4. Из заданного целочисленного массива удалить все повторяющиеся элементы, оставив только их первые вхождения, т.е. из заданного массива получить новый массив, состоящий из различных целых чисел.

5. Заменить отрицательные числа в массиве их квадратами, оставив остальные без изменения.

6. В заданном массиве найти среднее арифметическое положительных чисел, среднее арифметическое отрицательных чисел и число нулей.

7. В массиве из $2n$ чисел найти сумму квадратов элементов с четными индексами и сумму кубов элементов с нечетными индексами.

8. Из чисел $a_1, a_2,..., a_n$ выбрать те, которые больше по модулю заданного числа c, и образовать из них новый массив, сохранив порядок следования элементов.

9. Из массива целых чисел составить три других, в первый из которых записать числа, кратные 5, во второй - числа, кратные 7, а в третий - остальные числа.

10. Задан массив из 100 целых случайных чисел, принадлежащих промежутку [0, 100]. Найти сумму тех элементов массива, которые больше 15, но меньше 45, а также вычислить количество этих элементов.

11. В линейном массиве заменить все элементы на число m (m – индекс максимального элемента).

12. Дан массив, состоящий как из положительных, так и отрицательных чисел. Нужно сначала записать положительные числа, а затем отрицательные в том же порядке, как они были расположены в исходном массиве. Если есть нули, записать их в последнюю очередь.

13. Найти сумму элементов данного массива. Разделить каждый элемент исходного массива на полученное значение.

14. Вычислить сумму и разность массивов одного размера.

15. Найти среднее арифметическое значение элементов заданного массива. Преобразовать исходный массив, вычитая из каждого элемента среднее значение.

16. Даны два массива одинакового размера. Рассматривая их как арифметические векторы, найти длины этих векторов и их скалярное произведение.

17. Заданы два массива разных размеров. Объединить их в один массив, включив второй массив между k-ым и $(k + 1)$-ым элементами первого (k вводится с клавиатуры).

18. Вычесть из положительных элементов данного массива элемент с номером k_1 а к отрицательным элементам прибавить элемент с номером k_2. Нулевые элементы заменить 1. Номера k_1 и k_2 вводятся с клавиатуры.

19. К четным элементам целочисленного массива прибавить данное число a, а из элементов с четными номерами вычесть данное число b.

20. Дан первый член геометрической прогрессии и ее знаменатель. Сформировать одномерный массив, элементами которого служат первые *n* членов этой прогрессии.
21. Сформировать массив из первых 30 членов последовательности Фибоначчи.
22. Вставить одно и то же число, введенное с клавиатуры, перед каждым отрицательным элементом заданного целочисленного массива.
23. Дан массив четного размера. Поменять местами его половины следующим образом: первый элемент - с последним, второй - с предпоследним элементом и т.д.
24. Даны два целочисленных массива одинакового размера. Получить третий ей соответствующих элементов данных массивов.
25. Задан массив из *n* целых случайных чисел, принадлежащих промежутку [-25, 25]. Найти произведение тех элементов массива, которые больше 1, но меньше 15, а также вычислить количество четных элементов массива.

Задание III

В соответствии с вариантом составить и реализовать программу.

1. Дана матрица размера n*n, найдите сумму её с её транспонированной матрицей.
2. Реализуйте функцию нахождения максимального элемента матрицы, полученной в результате умножения вводимой матрицы A на вводимую матрицу B
3. С помощью элементарные преобразований получите из матрицы размера n*n соответствующую ей верхнюю треугольную матрицу.
4. С помощью элементарные преобразований получите из матрицы размера n*n соответствующую ей нижнюю треугольную матрицу.
5. С помощью элементарных преобразований получить из матрицы размера n*m соответствующую ей ступенчатую матрицу.
6. Реализуйте возведение матрицы размера m*m в степень n.
7. Реализуйте деление матрицы на сумму максимальных чисел из каждой её строки
8. Пока возможно для матрицы размера n*n, меняйте i строку со строкой i+1, если элемент матрицы с индексом i, i больше элемента с индексом i+1, i+1, после этого повторите аналогично, но меняйте местами столбцы, а элемент с индексом i, i должен быть меньше элемента с индексом i+1, i+1.
9. В матрице размера n*n для всех возможных i разделите строку i и столбец i на элемент с индексом i, i.
10. На основе матрицы размера n*m составьте соответствующую ей бинарную матрицу, которая указывает на элементы, являющиеся одновременно максимальными в своих строке и столбце.

11. Реализуйте игру «Крестики-нолики» для одного игрока, в которой действия второго игрока должны быть реализованы вашей программой, также можно реализовать любую другую игру на основе «Крестиков-ноликов», но в этом случае она должна быть сложнее.
12. Для двоичной матрицы размера n*n, если этой возможно, создайте матрицу размера n*n, в которой элемент с индексом i, j больше соседних по вертикали и горизонтальных элементов, если элемент двоичной матрицы с индексом i, j равен 1. Если это невозможно, то выведите информацию об этом.
13. Реализуйте функцию получения из матрицы размера n*m аналогичной матрицы, но перевёрнутой на 90 градусов, размер полученной матрицы будет m*n.
14. В двоичной матрице размера m*n найдите самую длинную вертикальную или горизонтальную последовательность из единиц.
15. Дана матрица размера m*n, элементами которой являются “#” и “.”. Вместо “.” может поставлено любое число от 0 до n, где n-количество соседних “#” по вертикали и горизонтали возле данной “.”. Найдите количество матриц, которое можно составить в итоге.

Контрольные вопросы

1. Дайте определение производного типа данных, структурированного типа.
2. Дайте определение массива.
3. Имя, размер и размерность массива.
4. Каким может быть тип элементов списка?
5. Правила инициализации списка.
6. Как осуществляется доступ к элементам списка?
7. Как осуществляется ввод списка с клавиатуры?
8. Какие способы ввода/создания списка вы знаете?
9. Вывод списка.
10. Какие основные свойства списка вам известны?
11. Какие способы добавления элемента в список вам известны?
12. Какие способы удаления элементов из списка вам известны?
13. Что вам известно про срезы списков?
14. Как осуществляется перебор элементов списка?
15. Какие методы списков вам известны?
16. Что представляет из себя матрица при программной реализации с использованием списков?

Пример выполнения задания II

Задание. Массив D содержит 24 значения атмосферного давления за каждый час в течение суток. Определить, какое значение атмосферного давления было наибольшим и в какое время оно было зафиксировано.

Решение

1. Математическая модель

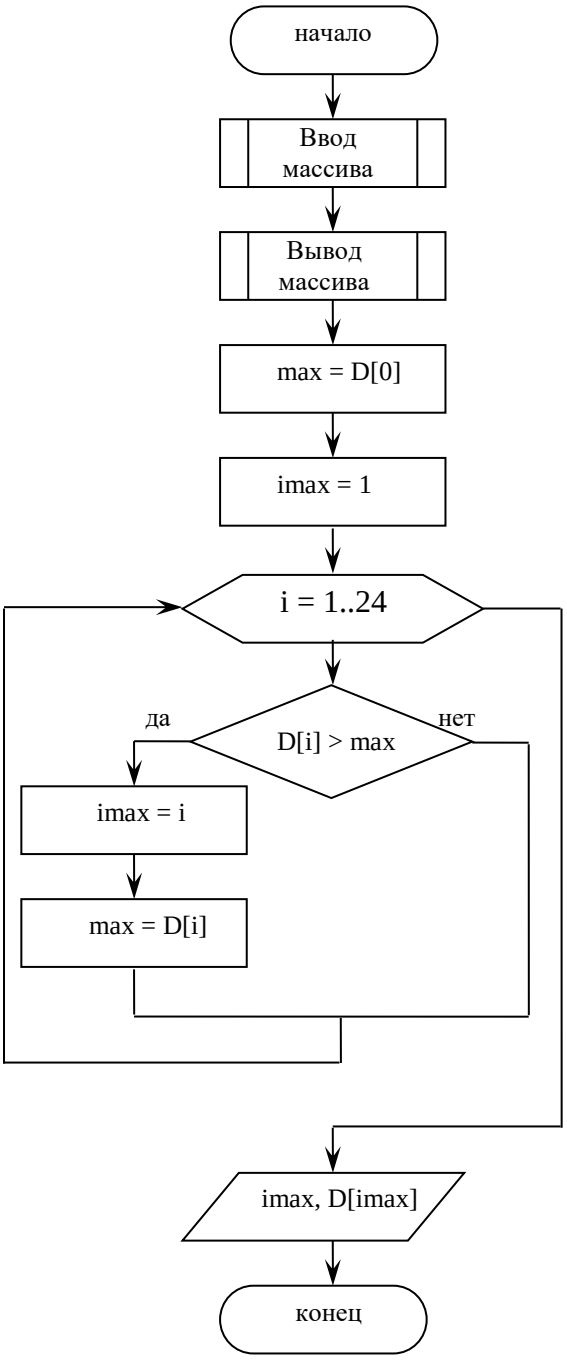
Значением атмосферного давления являются элементы массива D[24], значением времени – индексы элементов. Решение задачи сводится к поиску максимального элемента в массиве и определению его индекса.

Аргументы: D[24] – массив целого типа.

Результаты: imax целого типа – индекс максимального элемента;
D[imax] целого типа – значение максимального элемента.

Промежуточная величина: max целого типа – максимальное значение элемента массива.

2. Алгоритм



3. Программа

```
def InputM(mass):
   for i in range(24):
      mass[i] = int(input())


def PrintM(mass):
   for i in  range(24):
      print(mass[i],end = ' ')
   print()

D = [0]*24
print(" Введите массив: ")
InputM(D)
print(" Массив значений давле-
ния:")
PrintM(D)
max = D[0]
imax = 1
for i in  range(24):
   if (D[i]>max):
         imax = i+1
         max = D[i]
print(" Максимальное значение
давле-ния :",D[imax-
1],"в",imax,"ч.")
```

4. Результат работы программы:

```
Массив значений давления:
  748  743  756  748  741  760  748
743  757  748  756  756  751  760
754
  756  752  758  758  757  750  747
759  750
Максимальное значение давления 760
в 6 ч.
```

Список литературы, рекомендуемый к использованию по данной теме

- [1] стр. 123 - 127
- [4] стр 219-279, 302-309
- [6] стр. 48 – 52, 112 - 132
- [11] https://informatics.msk.ru/course/view.php?id=156#section-9, https://informatics.msk.ru/course/view.php?id=156#section-10, https://informatics.msk.ru/course/view.php?id=156#section-13, https://informatics.msk.ru/course/view.php?id=156#section-14.

Практическое занятие №7
Множества и словари

ЦЕЛЬ: освоение способов работы с множествами и словарями.

Знания, умения и владения, приобретаемые обучающимся в результате освоения темы, в рамках формируемых компетенций (ПК-1: ИД-1$_{ПК-1}$, ИД-2$_{ПК-1}$**;** ПК-2: ИД-1$_{ПК-2}$, ИД-2$_{ПК-2}$): см. приложение 2: вырабатывается навык разработки математических, информационных и имитационных моделей, а также формируется способность к разработке и применению алгоритмических и программных решений в области прикладного программирования с умением объяснить теоретическую и практическую часть темы (формируется часть указанных компетенций).

Теоретическая часть
7. Множества и словари

7.1 Множества

Множества в Python представляют собой неупорядоченный итерируемый тип данных, хранящий неповторяющиеся и неизменяемые объекты. Множества не предоставляют возможности обращаться к какому-либо их элементу по индексу, однако проверка на наличие элемента выполняется в среднем за константное время, реализованы пересечение, объединение и разница множеств за линейное время. Для инициализации множеств используются { и }, а также функция set, можно использовать генератор множеств, похожий на генератор списков:

```
a  = set()                          # Пустое множество
a = set((1, 2, 3))                  # Множество цифр 1, 2, 3.
a = {1, 2, 3}                       # Аналогично записи выше.
a = {i**2 for i in range(1, 5)}     # {1, 4, 9, 16}
```

Для инициализации списка нельзя использовать запись вида `a = {}`, так как в этом случае буден создан словарь.

В множества можно добавлять элементы и удалять из них элементы:

`set.add(x)` – добавляет x в множество

`set.discard(x) / set.remove(x)` – удаляет x из множества, set.remove вызывает ошибку, если элемент отсутствует в множестве.

`set.pop()` – удаляет случайный элемент списка, возвращает его значение.

Для множеств реализовано использование различных операторов.

`set1 | set2` – объединение множеств
`set1 & set2` – пересечение множеств
`set1 - set2` – разность множеств
`set1 ^ set2` - исключающее или множеств
`set1 <= set2` – проверка, является ли set1 подмножеством set2.

У множеств есть методы, работающие аналогично представленным выше операторам, некоторые могут принимать сразу несколько множеств (аргумент имеет вид *sets). Описанные ниже методы не изменяют исходный массив:

```
set1.issubset(set2) / set1.issuperset(set2) – set1 <= set2 / set1 >= set2
set.union(*sets) – set | sets
set.intersection(*sets) – set & sets
set.difference(*sets) – set - sets
set1.symmetric_difference(set2) – set ^ sets
```

Если необходимо изменить исходное множество, то нужно к представленным выше методам добавить в конце названия функции "_update". Исключением является объединение, оно будет иметь вид set.update(sets).

Существует также неизменяемый аналог множества - frozenset.

7.2 Словари

Словарь в Python – тип данных, хранящий неупорядоченные пары из ключа и значения. Ключи в словаре не повторяются и могут быть представлены только неизменяемыми типами данных. Доступ к значениям словаря осуществляется по ключу аналогично доступу по индексу у списка. Для инициализации списка можно использовать символы { и }, функцию dict и генератор словарей:

```
a = dict()                          # Пустой словарь, аналогично a = {}
a = {'a': 1, 'b': 2, 'c': 3}
a = dict(a=1, b=2, c=3)                  # аналогично 2-ой строке
a = dict((('a', 1), ('b', 2), ('c', 3)))      # аналогично 2-ой строке
a = {i: i**2 for i in range(1, 4)}           # {1: 1, 2: 4, 3: 9)
```

Добавить в словарь новую пару или изменить существующую можно используя конструкцию вида dict[key] = value.

Для словарей реализованы методы, возвращающие объекты, хранящие ключи, значения и пары элементов:

dict.keys() – возвращает объект, хранящий ключи и работающий подобно множествам.

dict.values() – возвращает объект, хранящий все значения и работающий подобно кортежам без возможности обращения по индексу.

dict.items() – возвращает объект, хранящий все пары элементов и работающий подобно множествам.

Также для словарей реализованы методы, подобные реализуемым для множеств:

dict.pop(key) – удаляет из словаря элемент по ключу, возвращает значение.

dict.popitem() – удаляет случайный элемент словаря и возвращает пару (ключ, значение)

dict.update(*dicts) – добавляет в данный словарь элементы словарей, подаваемых в качестве аргумента, если ключи совпадают, то в приоритете значения словарей, расположенных правее.

7.3 Объект defaultdict и анонимные функции

Модуль collections содержит в себе класс `defaultdict`, который позволяет подать в качестве аргумента функцию, которая при обращении без аргументов возвращает какой-либо объект. В дальнейшем, если мы будем обращаться к ключу, у которого ранее не было задано значение, ему будет сперва задан объект, возвращаемый функцией. Пример работы `defaultdict` (`int()` возвращает 0):

```
a = defaultdict(int)
a[2] += 3
print(a[2])# 3
```

В качество аргумента может быть удобно подавать анонимные функции. Они имеют вид **`lambda`** ***параметры: возвращаемое значение***, и используется, когда в качестве аргумента необходимо подать функцию, которая нигде больше не используется. Пример использования `lambda` совместно с `defaultdict`:

```
a = defaultdict(lambda: 3)
a[2] += 3
print(a[2])# 6
```

Можно задать переменной анонимную функцию и использовать как стандартную функцию в дальнейшем, **но это считается плохой практикой и не рекомендовано к использованию при написании кода**:

```
f = lambda x, y: (x*x + y*y)**0.5
print(f(3, 4))         # 5.0
```

Вопросы и задания

Задание I

Освоить теоретический материал, выполнить общее для всех задание:

1. Отладьте представленную ниже программу, в которой реализован алгоритм Дейкстры. В результате работы программа должна выводить длину минимального пути от 1 до 7 вершины представленного графа:

```
from math import inf
from collections import defaultdict

graph = {
    1: {2: 4, 3: 3},
    2: {1: 4, 4: 6, 6: 2},
    3: {1: 3, 5: 2, 6: 8},
    4: {2: 6, 6: 1, 7: 3},
    5: {3: 2, 6: 4, 7: 2},
    6: {2: 2, 3: 8, 4: 1, 5: 4},
    7: {4: 3, 5: 2}
}

path_len = defaultdict(lambda: inf)
```

```
visited = set()
node = 1
value = 0
final_node = 7

path_len[node] = value
while node != final_node:
    for next_node in (graph[node].keys() | visited):
        new_value = value + graph[node][next_node]
        if new_value < path_len[next_node]:
            path_len[next_node] = new_value
    visited.add(node)
    path_len.pop(node)
    node, value = max(path_len.items(), key=lambda x: x[1])

print(path_len[final_node])
```

2. Объясните, что выполняют представленные ниже строки и почему именно они использовались в программе?

```
path_len = defaultdict(lambda: inf)

path_len.pop(node)
```

3. В одной из используемых в коде функций был использован аргумент `key=lambda x: x[1]`. Параметр `key` также используется и в некоторых других функциях, например, sorted. Изучите принципы его работы и опишите своими словами.

Задание II

В соответствии с вариантом составить и реализовать программу, используя множества. Для решения б) и в) недопустимо использовать циклы и логические операторы (функции len, sum, max, min и т. д. необходимо использовать для решения в)). Если логическое выражение в словесной форме может быть воспринято двояко, то любое соответствующее ему решение будет верным:

Вариант 1

Даны 3 множества целых чисел, первое состоит из чисел кратных 3, второе из чисел кратных 9, третье из отрицательных чисел. Программа должна выполнять данные ниже задачи:

а) Ввод чисел с сохранением их в соответствующих множествах и вывод данных множеств.

б) Нахождение множества положительных чисел, не кратных 9, но кратных 3. Данное множество не должно храниться в какой-либо переменной.

в) Нахождение наименьшего числа, хранимого в минимум двух из этих множеств. Если такого числа нет, то вывести 0.

Вариант 2

Даны 3 множества целых чисел, первое состоит из чётных чисел, второе из чисел, не кратных 3, третье из положительных чисел. Программа должна выполнять данные ниже задачи:

а) Ввод чисел с сохранением их в соответствующих множествах и вывод данных множеств.

б) Нахождение множества положительных нечётных чисел не кратных 3. Данное множество не должно храниться в какой-либо переменной.

в) Нахождение наименьшего такого числа, что оно принадлежит только 1 из представленных множеств. Если такого нет, то выведите 0.

Вариант 3

Даны 3 множества целых чисел, первое состоит из чисел, кратных 3, второе из чисел, кратных 5, третье из чисел, кратных 10. Программа должна выполнять данные ниже задачи:

а) Ввод чисел с сохранением их в соответствующих множествах и вывод данных множеств.

б) Нахождение множества чисел, кратных 15, но не кратных 2. Данное множество не должно храниться в какой-либо переменной.

в) Нахождение наибольшего числа, кратного 3 и не кратного 5 или кратного 10.

Вариант 4

Даны 3 множества целых чисел, первое состоит из чисел, которые делятся на 3 с остатком 2, второе из чисел, которые делятся на 3 с остатком 2, третье из чисел, кратных 2. Программа должна выполнять данные ниже задачи:

а) Ввод чисел с сохранением их в соответствующих множествах и вывод данных множеств.

б) Нахождение множества нечётных чисел, которые делятся на 6 остатком 5. Данное множество не должно храниться в какой-либо переменной.

в) Нахождение наибольшего числа, которое делится на 6 или с остатком 2, или с остатком 4, или без остатка. Если такого числа нет, то выведите 0.

Вариант 5

Даны 3 множества целых чисел, первое состоит из чисел, которые больше 25, второе из чётных чисел, третье чисел, модуль которых является двузначным числом. Программа должна выполнять данные ниже задачи:

а) Ввод чисел с сохранением их в соответствующих множествах и вывод данных множеств.

б) Нахождение множества нечётных чисел с двузначным модулем меньше 26. Данное множество не должно храниться в какой-либо переменной.

в) Нахождение наименьшего по модулю чётного числа, которые не является двузначным или больше 25. Если такого числа нет, то выведите 0.

Вариант 6

Даны 3 множества вещественных чисел, первое состоит из чисел, которые округляются до целых в большую сторону, второе из чисел, которые округляются до десятков в меньшую сторону, третье из чисел, целая часть которых чётная. Программа должна выполнять данные ниже задачи:

а) Ввод чисел с сохранением их в соответствующих множествах и вывод данных множеств.

б) Нахождение множества чисел с чётной целой частью, которые округляются до целых в ту же сторону, что и до десятых. Данное множество не должно храниться в какой-либо переменной.

в) Нахождение суммы чисел с нечётной целой частью, которые или только округляются до десятков в меньшую сторону, или только округляются до целых в большую сторону.

Вариант 7

Даны 3 множества вещественных чисел, первое состоит из чисел, разряды единиц и десятых которых совпадают, второе из чисел, целая часть которых равно 0, третье из чисел, квадрат которых больше 5. Программа должна выполнять данные ниже задачи:

а) Ввод чисел с сохранением их в соответствующих множествах и вывод данных множеств.

б) Нахождение множества чисел, десятые которых не равны 0, разряды десятых и целых совпадают, квадрат меньше 5. Данное множество не должно храниться в какой-либо переменной.

в) Нахождение суммы чисел, квадрат которых больше 5 или разряд целых не совпадает с разрядом десятых и равен 0.

Вариант 8

Даны 3 множества вещественных чисел, первое состоит из чисел, квадрат которых меньше квадратного корня, второе из чисел больше 3, третье из чисел меньше 0.5. Программа должна выполнять данные ниже задачи:

а) Ввод чисел с сохранением их в соответствующих множествах и вывод данных множеств.

б) Нахождение множества чисел, меньше 0.5 или больше 3, но не входящих в промежуток [-1, 1]. Данное множество не должно храниться в какой-либо переменной.

в) Нахождение среднего арифметического чисел, которые или только больше 3, или только меньше 0.5, или только в промежутке [-1, 1].

Вариант 9

Даны 3 множества строк, первое состоит из строк, которые начинаются на «a», второе состоит из строк, оканчивающихся на «z», в третьем строки длиной больше 5. Программа должна выполнять данные ниже задачи:

а) Ввод строк с сохранением их в соответствующих множествах и вывод данных множеств.

б) Нахождение множества строк короче 6 символов, которые начинаются на «a» и заканчиваются на «z». Данное множество не должно храниться в какой-либо переменной.

в) Вывод такой строки, что её символы стоят в обратном порядке строки любого из множеств, начинается на «z» или заканчивается на «a» и длиннее 5 символов.

Вариант 10

Даны 3 множества строк, первое состоит из строк, которые начинаются на «сл», второе состоит из строк, оканчивающихся на «он», в третьем строки длиной меньше 7. Программа должна выполнять данные ниже задачи:

а) Ввод строк с сохранением их в соответствующих множествах и вывод данных множеств.

б) Нахождение множества строк больше 6 символов, которые начинаются на «сл» и не заканчиваются на «он». Данное множество не должно храниться в какой-либо переменной.

в) Нахождение самой длинной строки, которая начинается на «сл», заканчивается на «он» и короче 7 символов.

Вариант 11

Даны 3 множества строк, первое состоит из строк, которые содержат подстроку «ла», второе состоит из строк, первый и последний символы которых совпадают, в третьем строки чётной длины. Программа должна выполнять данные ниже задачи:

а) Ввод строк с сохранением их в соответствующих множествах и вывод данных множеств.

б) Нахождение множества строк чётной длины, содержащих подстроку «ла», но не содержащих одинаковые первый и последний символы. Данное множество не должно храниться в какой-либо переменной.

в) Нахождение самой короткой строки, которая начинается и заканчивается на одинаковые символы или чётной, но не содержит подстроку «ла».

Вариант 12

Даны 3 множества строк, первое состоит из строк, которые не содержат одинаковых символов, второе состоит из строк, содержащих 1 символ « », в третьем строки длины меньше 8. Программа должна выполнять данные ниже задачи:

а) Ввод строк с сохранением их в соответствующих множествах и вывод данных множеств.

б) Нахождение множества строк, не содержащих одинаковые символы и содержащих 1 символ « », но длиннее 7. Данное множество не должно храниться в какой-либо переменной.

в) Вывод каждого символа (не более 1 раза) строки, длина которой меньше 8, которая содержит повторяющиеся символы и содержит не 1 символ « ».

Вариант 13

Даны 3 множества строк, первое состоит из палиндромов, второе состоит из строк длиной 4, третье состоит из строк длиной 6. Программа должна выполнять данные ниже задачи:

а) Ввод чисел с сохранением их в соответствующих множествах и вывод данных множеств.

б) Нахождение множества строк, являющихся палиндромами, но не длины 4 или 6. Данное множество не должно храниться в какой-либо переменной.

в) Нахождение количества строк, длина которых 4 или 6, но не являющихся палиндромами.

Вариант 14

Даны 3 множества строк, первое состоит из строк только из заглавных английских букв, второе состоит из строк, начинающихся на «ABC», третье состоит из строк длиной 6. Программа должна выполнять данные ниже задачи:

а) Ввод чисел с сохранением их в соответствующих множествах и вывод данных множеств.

б) Нахождение множества строк, состоящих не только из заглавных английских букв, начинающихся на «ABC» и длиной 6. Данное множество не должно храниться в какой-либо переменной.

в) Нахождение самой длинной строки, состоящей только из заглавных английских букв, не начинающейся на «ABC» и не длины 6.

Вариант 15

Даны 3 множества строк, первое состоит из строк только из строчных английских букв, второе состоит из строк, заканчивающихся на «xyz», третье состоит из строк длиной 4. Программа должна выполнять данные ниже задачи:

а) Ввод чисел с сохранением их в соответствующих множествах и вывод данных множеств.

б) Нахождение множества строк, состоящих не только из строчных английских букв, заканчивающихся на «xyz» и длиной 4. Данное множество не должно храниться в какой-либо переменной.

в) Нахождение самой короткой строки, состоящей только из строчных английских букв, не заканчивающихся на «xyz» и не длины 4.

Задание III

В соответствии с вариантом составить и реализовать программу. Предусмотреть обработку данных с использованием МЕНЮ, содержащим пункты: ввод данных, просмотр всех данных, вывод данных по ключу, удаление данных по ключу, поиск 1, поиск 2 (если указан в задании), выход. Если ключ совпадает с какими-либо хранимыми данными, то допускается не хранить его дважды, но вывод этих данных всё ещё должен осуществляться.

Вариант 1

1. Описать словарь student, где в качестве ключей используются ФИО, а в качестве

значений словари со следующими данными: ФИО, номер группы, успеваемость (массив из пяти элементов).
2. Написать программу, выполняющую следующие действия:
   - ввод данных с сохранением в словаре;
   - поиск записей, в которых средний бал студента больше 4,0;
   - поиск записей, в которых совпадают номера групп студентов.

Вариант 2

1. Описать словарь aeroflot, где в качестве ключей используются номера рейсов, а в качестве значений словари со следующими данными: название пункта назначения рейса, номер рейса, вместимость, тип самолета.
2. Написать программу, выполняющую следующие действия:
   - ввод данных с сохранением в словаре;
   - поиск записей, в которых пункт назначения совпадает с введенным с клавиатуры;
   - поиск записей, в которых вместимость самолета не менее введенной с клавиатуры.

Вариант 3

1. Описать словарь train, где в качестве ключей используются номера поездов, а в качестве значений словари со следующими данными: номер поезда, название пункта назначения рейса, время отправления, время в пути.
2. Написать программу, выполняющую следующие действия:
   - ввод данных с сохранением в словаре;
   - поиск поездов, отправляющихся после времени, введенного с клавиатуры в течении ближайших двух часов;
   - поиск поездов, отправляющихся в пункт назначения, введенный с клавиатуры.

Вариант 4

1. Описать словарь tourist, где в качестве ключей используются номера маршрутов, а в качестве значений словари со следующими данными: номер маршрута, наименование начального пункта маршрута, наименование конечного пункта маршрута, время в пути.
2. Написать программу, выполняющую следующие действия:
   - ввод данных с сохранением в словаре;
   - поиск маршрутов, начинающихся в пункте, наименование которого введено с клавиатуры;
   - поиск маршрутов, общее время прохождения которых не превышает времени, введенного с клавиатуры.

Вариант 5

1. Описать словарь note, где в качестве ключей используются номера телефона, а в качестве значений словари со следующими данными: фамилия, имя, номер телефона.
2. Написать программу, выполняющую следующие действия:
   - ввод данных с сохранением в словаре;
   - поиск всех записей, у которых начало номера соответствует введённому

набору символов;
- поиск всех номеров, которые принадлежат людям с введённой фамилией.

Вариант 6

1. Описать словарь znak, где в качестве ключей используются ФИО, а в качестве значений словари со следующими данными: ФИО, дата рождения, знак зодиака.
2. Написать программу, выполняющую следующие действия:
   - ввод данных с сохранением в словаре;
   - поиск информации о людях, родившихся сегодня;
   - поиск информации о людях, родившихся под веденным знаком зодиака.

Вариант 7

1. Описать словарь price, где в качестве ключей используются наименования товаров, а в качестве значений словари со следующими данными: наименование товара, название магазина, в который товар отгружен, стоимость товара.
2. Написать программу, выполняющую следующие действия:
   - ввод данных с сохранением в словаре;
   - поиск информации о магазинах, в которые отгружен товар, название которого введено с клавиатуры;
   - поиск информации о товарах, стоимость которых не превышает стоимость, введенную с клавиатуры.

Вариант 8

1. Описать словарь order, где в качестве ключей используются ИНН, а в качестве значений словари со следующими данными: фамилия и инициалы налогоплательщика, ИНН налогоплательщика, сумма налогообложения.
2. Написать программу, выполняющую следующие действия:
   - ввод данных с сохранением в словаре;
   - поиск информации о налогоплательщиках с введённой фамилией;
   - поиск информации о налогоплательщиках, чей суммарный налог превышает сумму, введенную с клавиатуры.

Вариант 9

1. Описать словарь car, где в качестве ключей используются регистрационные номера, а в качестве значений словари со следующими данными: марка автомобиля, государственный регистрационный номер, фамилия владельца, инициалы владельца, пробег поквартально (массив из четырех элементов).
2. Написать программу, выполняющую следующие действия:
   - ввод данных с сохранением в словаре;
   - поиск информации о владельцах автомобилей, марка которого введена с клавиатуры;
   - поиск информации об автомобилях, у которых суммарный пробег за 2 и 3 кварталы не превышает значение, введенное с клавиатуры.

Вариант 10

1. Описать словарь student, где в качестве ключей используются серии и номера паспортов, а в качестве значений словари со следующими данными: фамилия,

имя, отчество, серия и номер паспорта, дата выдачи.
2. Написать программу, выполняющую следующие действия:
   - ввод данных с сохранением в словаре;
   - поиск записей, в которых совпадает имя студента;
   - поиск записей, в которых совпадает месяц выдачи паспорта.

Вариант 11

1. Описать словарь aeroflot, где в качестве ключей используются номера рейсов, а в качестве значений словари со следующими данными: номер рейса, время отправления, время начала регистрации, тип самолета.
2. Написать программу, выполняющую следующие действия:
   - ввод данных с сохранением в словаре;
   - поиск рейсов, регистрация на которые уже началась по отношению ко времени, введенному с клавиатуры, но еще не закончилась (временем окончания регистрации считается время отправления минус 20 минут).

Вариант 12

1. Описать словарь worker, где в качестве ключей используются фамилии и инициалы, а в качестве значений словари со следующими данными: фамилия и инициалы работника, дата рождения, год начала трудовой деятельности, год поступления на работу.
2. Написать программу, выполняющую следующие действия:
   - ввод данных с сохранением в словаре;
   - поиск сотрудников, чей трудовой стаж не менее введенного с клавиатуры;
   - поиск сотрудников, родившихся в течение месяца, введенного с клавиатуры.

Вариант 13

1. Описать словарь train, где в качестве ключей используются номера поездов, а в качестве значений словари со следующими данными: номер поезда, название пункта назначения рейса, количество спальных, купейных и плацкартных мест (список из трех элементов), количество свободных спальных, купейных и плацкартных мест (список из трех элементов).
2. Написать программу, выполняющую следующие действия:
   - ввод данных с сохранением в словаре;
   - поиск поездов, отправляющихся в пункт назначения, введенный с клавиатуры и имеющих количество свободных плацкартных мест, заданных пользователем.
   - Поиск поезда с наибольшим отношением свободных мест к общему количеству для введённого типа места.

Вариант 14

1. Описать словарь note, где в качестве ключей используются даты, а в качестве значений словари со следующими данными: фамилия, имя, дата встречи, место встречи.
2. Написать программу, выполняющую следующие действия:
   - ввод данных с сохранением в словаре;

- поиск информации о встречах, с человеком с фамилией и именем, введенными с клавиатуры;
- поиск информации о встречах, назначенных в месте, введенном с клавиатуры.

Вариант 15

1. Описать словарь price, где в качестве ключей используются кортежи вида (наименование магазина, наименование товара), а в качестве значений словари со следующими данными: наименование магазина, наименование товара, количество товара на складе, стоимость товара.
2. Написать программу, выполняющую следующие действия:
   - ввод данных с сохранением в словаре;
   - поиск информации о магазинах, в которых есть товар, название которого ведено с клавиатуры, со стоимостью, не превышающей значение, введенное с клавиатуры.

Контрольные вопросы

1. В каких случаях можно использовать множества?
2. Какие действия над множествами можно совершать?
3. Как можно пройтись по всем элементам множества?
4. Может ли множество несколько одинаковых элементов?
5. Может ли множество содержать элементы разных типов? Все ли типы данных может содержать множество?
6. В каких случаях можно использовать словари?
7. Что является ключом словаря, а что значением? Какая разница между ними?
8. Как можно пройтись по всем элементам словаря?
9. Какие действия над словарями можно совершать?
10. Может ли функция быть аргументом другой функции? Если да, то приведите примеры.
11. В чём отличительная особенность defaultdict и когда его лучше использовать вместо стандартного словаря?
12. В каких случаях используются анонимные функции?
13. Почему в некоторых ситуациях лучше использовать анонимные функции, а не стандартные?

Пример выполнения задания I

Задание

1. Описать словарь university, где в качестве ключей используются названия институтов, а в качестве значений словарь, ключами которого являются коды групп, а в качестве значений словари со следующими данными: название института, код группы, множество ФИО студентов группы, словарь с ключами в виде названий предметов, а значениями в виде среднего арифметического итоговых оценок студентов.
2. Написать программу, выполняющую следующие действия:
   - ввод данных с сохранением в словаре;

- Поиск группы с наибольшим средним арифметическим хранимых данных об оценках для каждого института.
- Поиск предмета с наивысшей средней арифметической оценок среди всех групп.

1. Решение

Для хранения данных воспользуемся типом данных defaultdict с аргументом dict, так как это позволит нам в одной строке задавать ключи с названием института и кодом группы из-за отсутствия необходимости задавать каждому новому ключу, обозначающему институт, словарь в качестве значения.

Для ввода данных опишем функцию input_grades, представляющую интерфейс ввода и возвращающую кортеж из названия института, кода группы и словаря с информацией о группе. Для большей читаемости кода опишем в ней подфункции input_ grades и input_students, представляющие интерфейс ввода и возвращающие словарь оценок и список студентов.

Для вывода опишем функции output_group, output_institute и output_university. Первые две выводят информацию о поданных в качестве аргументов словарей с информацией о группе и институте. Третья представляет интерфейс для выбора выводимых данных, позволяя вывести информацию о всех институт, одном выбранном и о выбранной из института группы.

Для предотвращения ошибок и опечаток при вводе воспользуемся функций get_close_matches модуля difflib. Данная функция принимает в качестве аргументов строку и объект, хранящий строки, а возвращает список строк, в котором элементы упорядочены по убыванию схожести с поданной строкой (Для большей информации о данной функции можно воспользоваться функцией help).

Первый поиск задания будет выполняться функцией search1. Она будет представлять собой генератор, который посредством перебора для каждого института находит подходящую группу и возвращает информацию о ней.

Второй поиск задания будет выполняться функцией search2. Создадим объект типа defaultdict с аргументом lambda: [0, 0]. После этого переберём все предметы для каждой группы и соответствующему значению по ключу, являющимся названием дисциплины, будем добавлять к 0-му элементу оценку, к 1-му единицу. Для поиска нужного предмета воспользуемся функцией max, в качестве аргументов подадим названия предметов и анонимную функцию, находящую по названию предмета среднюю оценку по нему.

2. Программа

```
import os
from collections import defaultdict
from difflib import get_close_matches


def input_new_group():
    def input_students():
        os.system("cls")
        print("Вводите ФИО студентов построчно. Для окончания введите пустую
строку")
        students = list()
```

```
        while student := input().title():
            students.append(student)
        return students

    def input_grades():
        os.system("cls")
        print("Вводите предмет и среднюю оценку по нему построчно. Для окончания
ввода введите пустую строку")
        grades = dict()
        while subject_with_grade := input().split():
            subject, grade = " ".join(subject_with_grade[:-1]), int(sub-
ject_with_grade[-1])
            grades[subject.capitalize()] = float(grade)
        return grades

    os.system("cls")
    institute = input("Введите институт: ").capitalize()
    grout_code = input("Введите код группы: ").upper()
    group = dict(
        students=input_students(),
        grades=input_grades()
    )
    return institute, grout_code, group


def output_group(group, group_code):
    print(f"Студенты группы с кодом {group_code}:")
    for student in group["students"]:
        print("\t" + student)
    print(f"Средние оценки студентов по предметам:")
    for subject, grade in group["grades"].items():
        print(f"\t{subject}: {grade}")


def output_institute(institute, institute_name):
    print(f"Информация о группах {institute_name}", end="\n"*2)

    for group_code, group in institute.items():
        output_group(group, group_code)
        print()


def output_university(university):
    os.system("cls")
    institute_name = input("Введите название института или нажмите Enter: ").ti-
tle()
    if institute_name and institute_name not in university:
        matches = get_close_matches(institute_name, university.keys())
        if len(matches) == 0:
            print("\nДанный институт не найден")
            input("\nНажмите Enter Для продолжения")
            return None
        else:
```

```
            institute_name = matches[0]

    if institute_name:
        institute = university[institute_name]
        group_code = input("Введите код группы или нажмите Enter: ")
        print()
        if group_code and group_code not in institute:
            matches = get_close_matches(group_code, institute.keys())
            if len(matches) == 0:
                print("\nДанная группа не найдена")
                input("\nНажмите Enter Для продолжения")
                return None
            else:
                group_code = matches[0]
        if group_code:
            group = institute[group_code]
            output_group(group, group_code)
        else:
            output_institute(institute, institute_name)

    else:
        print("Информация обо всех институтах:", end="\n")
        for institute_name, institute in university.items():
            output_institute(institute, institute_name)
            print(end="\n\n")

    input("Нажмите Enter Для продолжения")

def delete_from(university):
    os.system("cls")
    institute_name = input("Введите название института или нажмите Enter для
удаления всех данных: ").title()
    if institute_name and institute_name not in university:
        matches = get_close_matches(institute_name, university.keys(), cut-
off=0.8)
        if len(matches) == 0:
            print("\nДанный институт не найден")
            input("\nНажмите Enter Для продолжения")
            return None
        else:
            institute_name = matches[0]

    if institute_name:
        institute = university[institute_name]
        group_code = input("Введите код группы или нажмите Enter для удаления
института: ")
        print()
        if group_code and group_code not in institute:
            matches = get_close_matches(group_code, institute.keys(), cut-
off=0.8)
            if len(matches) == 0:
                print("\nДанная группа не найдена")
                input("\nНажмите Enter Для продолжения")
```

```
                return None
            else:
                group_code = matches[0]
        if group_code:
            institute.pop(group_code)
        else:
            university.pop(institute_name)

    else:
        university.clear()
    print("Удаление завершено")
    input("Нажмите Enter Для продолжения")

def get_choice():
    os.system("cls")

    print(
        "1 - Добавить новую группу",
        "2 - Удалить данные",
        "3 - Просмотреть информацию о группе",
        "4 - Просмотреть информацию о лучших группах",
        "5 - Найти предмет с наилучшей оценкой среди всех групп",
        "6 - Прекратить выполнение программы",
        sep="\n", end="\n"*2
    )

    try:
        choice = int(input("Введите номер команды - "))
    except ValueError:
        choice = 0
    return choice


def search1(university):
    for institute in university.values():
        best_grade, best_group = 0, (None, None)
        for group_code, group in institute.items():
            grades = group["grades"].values()
            if mean_grade := sum(grades)/len(grades) > best_grade:
                best_grade = mean_grade
                best_group = group_code, group
        yield best_group


def search2(university):
    subjects = defaultdict(lambda: [0, 0])
    for institute in university.values():
        for group in institute.values():
            for subject, grade in group["grades"].items():
                subjects[subject][0] += grade
                subjects[subject][1] += 1
    return max(subjects.keys(), key=lambda x: subjects[x][0]/subjects[x][1])
```

```
university = defaultdict(dict)
finish = False
while not finish:
    choice = get_choice()
    if choice == 1:
        institute_name, group_code, group = input_new_group()
        if matches := get_close_matches(institute_name, university.keys(), cut-
off=0.85):
            institute_name = matches[0]
            if matches := get_close_matches(group_code, university[insti-
tute_name].keys(), cutoff=0.95):
                group_code = matches[0]
        university[institute_name][group_code] = group
    elif choice == 2:
        delete_from(university)
    elif choice == 3:
        output_university(university)
    elif choice == 4:
        os.system("cls")
        print("Лучшие группы каждого института:", end="\n"*2)
        for group_code, group in search1(university):
            output_group(group, group_code)
        input("\nНажмите Enter Для продолжения")
    elif choice == 5:
        os.system("cls")
        print("Предметом с наивысшей оценкой является", search2(university))
        input("\nНажмите Enter Для продолжения")
    elif choice == 6:
        finish = True
    else:
        os.system("cls")
        input("Введена неверная команда. Нажмите Enter для продолжения")
```

Список литературы, рекомендуемый к использованию по данной теме

- [1] стр. 128 - 135
- [4] стр 195-203, 279-300, 585-591
- [6] стр. 53 – 59
- [11] https://informatics.msk.ru/course/view.php?id=156#section-16

Практическое занятие №8
Файлы

ЦЕЛЬ: изучение функции open и базовых способов работы с файлами, переменных sys.stdin, sys.stdout, sys.stderr

Знания, умения и владения, приобретаемые обучающимся в результате освоения темы, в рамках формируемых компетенций (ПК-1: ИД-1$_{ПК-1}$, ИД-2$_{ПК-1}$**;** ПК-2: ИД-1$_{ПК-2}$, ИД-2$_{ПК-2}$): см. приложение 2: вырабатывается навык разработки математических, информационных и имитационных моделей, а также формируется способность к разработке и применению алгоритмических и программных решений (с использованием файлов) в области прикладного программирования с умением объяснить теоретическую и практическую часть темы (формируется часть указанных компетенций).

Теоретическая часть
8. Файлы

8.1 Знакомство с файлами

Работа с файлами в Python организована через потоки, но, в отличие от C++, этого не видно столь отчётливо – всё спрятано за классами, за наследованием. Что нужно программисту, чтобы открыть файл, так это функция **open.** Первый параметр – путь к файлу. Это может выглядеть так:

```
file = open(“input.txt”)
```

В данном случае мы открываем файл `input.txt`, находящемуся в той же директории, что и наша программа. Указывать расширение обязательно!

Произвольный файл открывается так:

```
file = open(“C:/Users/User/Desktop/input.txt”)
```

Нам приходится указывать полное имя файла. Про расширение не забываем. Также следует обратить внимание на то, что косая черта в полном имени файла наклонена вправо, а не влево, как указывается в Windows.

Второй параметр – режим, в котором мы открываем файл. Можно указать этот параметр как без имени, так и с именем **`mode`**, поскольку он именованный.

Режим	Расшифровка
`r`	Чтение
`w`	Запись (имевшееся в файле содержимое стирается)
`x`	Создание нового файла и открытие на запись
`a`	Запись (начиная с конца имеющегося содержимого)
`b`	Двоичный режим
`t`	Текстовый режим

Режимы можно комбинировать, при этом “r”, “w”, “x” или “a” идёт раньше, чем “b” или “t”. Значение данного параметра по умолчанию – “rt”, то есть файл открывается на чтение в текстовом режиме. Предположим, что мы хотим сделать совсем другое – открыть файл не на чтение, а на запись, и писать не текст, а байты:

```
file = open(“output.txt”, “wb”)
```

Забежим вперёд – файл нужно закрыть. Открытый файл – потенциальный источник проблем и просто паразит, который тратит ресурсы компьютера. Первый способ это сделать – функция **close.** Выглядеть это может так:

```
file = open(“input.txt”, mode=“rb”)
# таинственная магия применяется к содержимому файла
# two hours later…

file.close()
```

Когда программист напечатает значительную часть текста программы, он забудет либо то, что файл нужно закрыть, либо имя переменной. Есть более удобный способ закрыть файл, и начинается он с того, что файл открывается иначе. Для этого используем менеджер контекста **with**:

```
with open(“input.txt”) as file:
      # код
```

После ключевого слова **as** мы пишем всего лишь имя переменной, псевдоним, по которому будем обращаться к объекту, созданному функцией open(). Менеджер контекста автоматически закроет файл, когда завершится соответствующий блок кода.

Как говорилось ранее, открытый файл – потенциальный источник ошибок или, иначе, исключений (exceptions). Было бы неплохо эти исключения “поймать”, не дать им остановить работу программы. Это можно сделать так:

```
try:
      with open(“input.txt”) as file:
            # код
except:
      # код в случае исключения
```

В блок try нужно поместить минимум кода, притом там должен быть именно такой код, который с наибольшей вероятностью приведёт к исключению. Плохой идеей является всю задачу решить внутри try и, в частности, внутри with! Вместо этого в with мы лишь прочитаем файл, т. е. загрузим его в оперативную память, а потом закроем и будем работать с копией данных в оперативной памяти. С записью в файл то же самое: мы сначала создадим текст или последовательность байт, которую хотим записать, и лишь затем откроем файл на запись.

8.2 Чтение из файла

Первый способ прочитать данные из файла – функция **read.**

```
with open("test.txt", 'rt') as file:
    text = file.read()
print(text)
```

Читаем из файла весь текст функцией read(), сохраняем полученную строку в переменной text, закрываем файл, и выводим, что прочитали. А прочитали мы крокозябры:

```
РєРѕС€РєР°
СЃРѕР±Р°РєР°
РІР°СЃСЏ РїРµС‚СЏ РІР°РЅСЏ
>>>
```

У функции open() есть именованный параметр **encoding,** в котором указывается кодировка. Исправим предыдущий код:

```
with open("test.txt", 'rt', encoding="utf-8") as file:
    text = file.read()
print(text)
```

Теперь программа работает правильно:

```
кошка
собака
вася петя ваня
>>>
```

Функция read() возвращает одну строку, включающую символы переноса строки \n. Допустим, нам нужно содержимое файла получить не в виде одной большой строки, а в виде списка строк. Первое, что приходит на ум – функция split().

```
with open("test.txt", 'rt', encoding="utf-8") as file:
    lines = file.read().split("\n")
print(lines)
```

Код почти не изменился, а уже намного удобнее стало. Функция split() ещё и убрала символы переноса строки:

```
['кошка', 'собака', 'вася петя ваня']
>>>
```

Можно было сразу итерировать file, но тогда код получился бы длиннее:

```
with open("test.txt", 'rt', encoding="utf-8") as file:
    lines = list()
```

```
    for line in file:
        lines.append(line)
print(lines)
```

Символы переноса строки никуда не исчезли. Нужны они вам или нет – зависит от задачи.

```
['кошка\n', 'собака\n', 'вася петя ваня']
>>>
```

Второй способ читать данные из файла – функция **readline.** Она считывает одну строчку файла, т. е. весь текст до первого переноса строки. При следующем вызове с этим же файлом она прочитает вторую строчку, потом – третью, и так далее. С этой функцией может получиться такой код:

```
with open("test.txt", 'rt', encoding="utf-8") as file:
    lines = list()
    line = file.readline()
    while line:
        lines.append(line)
        line = file.readline()
print(lines)
```

Символы переноса строки по-прежнему здесь.

```
['кошка\n', 'собака\n', 'вася петя ваня']
>>>
```

Но всё это время мы изобретали велосипед. Если файл нужно прочитать по строкам, самым простым способом будет третий – с помощью функции **readlines.** Это не та же самая функция, в конце появилась буква s! Коротко и элегантно:

```
with open("test.txt", 'rt', encoding="utf-8") as file:
    lines = file.readlines()
print(lines)
```

Символы переноса строки остаются:

```
['кошка\n', 'собака\n', 'вася петя ваня']
```

При работе с файлами учитывайте, что построчное считывание может тратить меньше памяти, чем считывание всего файла, и позволяет работать с файлами огромных размеров.

8.3 Запись в файл

Как же записывать в файл? Тут тоже не всё так однозначно. Первый способ – функция **write**.

```
with open("test.txt", 'wt', encoding="utf-8") as file:
```

```
    file.write("кошка\nсобака\nвася петя ваня")
```

Перенос строки нужно указывать явно при всех способах записи в файл. Второй способ – функция **writelines.** Она принимает список строк, и записывает их в файл подряд. Если в строке не указан явно символ переноса строки, то она станет частью предыдущей.

```
with open("test.txt", 'wt', encoding="utf-8") as file:
    file.writelines(["кошка\n", "собака\n", "вася петя ваня"])
```

Наконец, можно использовать именованный параметр **file** функции **print.** Убедитесь, что файл ещё не закрыт!

```
with open("test.txt", 'wt', encoding="utf-8") as test:
    print("кошка\nсобака\nвася петя ваня", end="", file=test)
```

Не забудьте про `end=""`, иначе в конце файла появится пустая строка.

8.4 stdin, stdout, stderr

Встроенный модуль sys содержит в себе различные переменные и функции, которые так или иначе связаны с работой интерпретатора. В данный момент наиболее интересными для нас являются **sys input.stdin, sys.stdout, sys.stderr**, которые позволяют работать с вводом, выводом и выводом ошибок так же, как с файлами.

Например, мы может использовать **sys.stdin.readline** вместо **input** и **sys.stdout.write** вместо **print**.

Так же это позволяет нам заменять стандартные ввод, вывод и вывод ошибок, с которыми работают input, print и т.д. на другие файлы, например:

sys.stdin = open("input.txt", "r")

Обратите внимание, что это будет работать только если мы импортируем именно **sys**, а не его содержимое.

Вопросы и задания

Задание I

1. Создать в Блокноте следующий текстовый файл:

   ```
   У меня спросили: сколько будет x Опер y ?
   А я не знаю! А n Опер k ? Тоже!
   Помогите!
   ```

   Например:

   ```
   У меня спросили: сколько будет 7 * 2 ?
   А я не знаю! А 9 / 4 ? Тоже!
   Помогите!
   ```

2. Вам известна структура файла. Вывести содержимое файла на экран, а в выходной файл записать результаты:

```
x Опер y = Рез1
n Опер k = Рез2
```

Например:

```
7 * 2 = 14
9 / 4 = 2.25
```

3. Исходные данные берутся из таблицы согласно варианту:

Вар	x	Опе	y	n	Опе	k	Вар	x	Опе	y	n	Опе	k
1	15	+	4	7	*	8	***9***	23	+	37	13	*	5
2	18	-	19	18	/	4	***10***	7	-	42	37	/	6
3	9	*	6	56	-	37	***11***	34	*	3	14	-	53
4	23	/	5	31	+	29	***12***	21	/	5	11	+	77
5	7	+	23	14	/	4	***13***	12	+	25	20	/	6
6	34	-	67	11	*	3	***14***	15	-	72	54	*	2
7	21	*	2	20	+	11	***15***	18	*	4	7	+	55
8	12	/	5	54	-	32	***16***	9	/	18	18	-	81

Задание II

Оценивание данного задания зависит от эффективности использования памяти. Для создания входных файлов и проверки решения используйте данный исполняемый файл.

1. Даны файлы input0.txt и input1.txt. Создайте файл output.txt, в который построчно запишите наибольшее число из каждой тройки чисел сначала из input0.txt, а потом из input1.txt.

2. Даны файлы input0.txt и input1.txt. Создайте файл output.txt, в который построчно запишите наименьшее число из каждой пятёрки чисел сначала из input0.txt, а потом из input1.txt.

3. Даны файлы input0.txt и input1.txt. Создайте файл output.txt, в который построчно запишите сумму чисел каждой строки сначала из input0.txt, а потом из input1.txt.

4. Даны файлы input0.txt и input1.txt. Создайте файл output.txt, в который построчно запишите отсортированные строки чисел сначала из input0.txt, а потом из input1.txt.

5. Даны файлы input0.txt и input1.txt. Создайте файл output.txt, в который построчно запишите среднее арифметическое каждой строки чисел сначала из input0.txt, а потом из input1.txt.

6. Даны файлы input0.txt, input1.txt, …, input499.txt. Создайте файл output.txt, в который запишите построчно максимальное число в каждом из файлов.

7. Даны файлы input100.txt, input101.txt, …, input399.txt. Создайте файл output.txt, в который запишите построчно минимальное число в каждом из файлов.

8. Даны файлы input0.txt, input1.txt, …, input499.txt. Создайте файл output.txt, в который запишите построчно сумму чисел каждого из файлов.

9. Даны файлы input0.txt, input1.txt, …, input499.txt. Создайте файл output.txt, в который запишите построчно модуль разницы количества чётных и нечётных чисел каждого из файлов.

10. Даны файлы input0.txt, input1.txt, …, input499.txt. Создайте файл output.txt, в который запишите наибольшее по модулю число каждого из файлов.

11. Даны файлы input0.txt и input1.txt с одинаковым количеством строк. Создайте файл output.txt, в каждой его строке запишите максимальное значение из чисел соответствующих строк input0.txt и input1.txt.

12. Даны файлы input0.txt и input1.txt с одинаковым количеством строк. Создайте файл output.txt, в каждой его строке запишите минимальное значение из чисел соответствующих строк input0.txt и input1.txt.

13. Даны файлы input0.txt и input1.txt с одинаковым количеством строк. Создайте файл output.txt, в каждой его строке запишите отсортированные в порядке возрастания числа из соответствующих строк input0.txt и input1.txt.

14. Даны файлы input0.txt и input1.txt с одинаковым количеством строк. Создайте файл output.txt, в каждой его строке запишите отсортированные в порядке убывания числа из соответствующих строк input0.txt и input1.txt.

15. Даны файлы input0.txt и input1.txt с одинаковым количеством строк. Создайте файл output.txt, в каждой его строке запишите НОД чисел из соответствующих строк input0.txt и input1.txt.

Задание III

Добавьте в вашу программу для 3 задания 7 практического занятия возможность сохранять данные в файл и загружать данные из файла с помощью соответствующих пунктов меню.

Контрольные вопросы

1. Как начать работать с файлами в Python?
2. Опишите основные параметры функции open
3. Какие аргументы используются для параметра mode функции open и что они означают?
4. Опишите конструкцию with … as … :, расскажите, чем она лучше открытия файла с помощью стандартного присваивания.
5. Как можно считать данные из файла?
6. В каких случаях некоторые способы считывания данных из файла лучше остальных? Приведите пример.
7. Как можно записать данные в файл?
8. В чём отличие открытия файла на запись и дозапись?
9. Для чего используется модуль sys?
10. Что представляют из себя переменные sys.stdin, sys.stdout. sys.stderr
11. Как можно использовать функцию print для записи данных в файл?

Пример выполнения задания III

Задание

Дополнить программу, добавив возможность сохранения файлов в программу и загрузку данных из файлов.

1. Решение

Будем хранить информацию о количестве поездов в первой строке файла, а дальше для каждого файла построчно данную информацию: номер поезда, название пункта отправления рейса, количество мест, количество свободных мест. Загрузку будем осуществлять аналогично стандартному вводу, но взятием данных из файлов.

2. Программа

```
import os


DEPARTURE = "Пункт отправления"
PLACES = "Количество мест"
FREE_PLACES = "Количество свободных мест"

PATH = r"D:\Учёба\Алгоритмизация и программирование\Задания\trains.txt"


def safe_trains(trains, path=PATH):
    def safe_train(num, train_info, file):
        file.write(
            f"{num}\n"
            f"{train_info[DEPARTURE]}\n"
            f"{' '.join(map(str, train_info[PLACES]))}\n"
            f"{' '.join(map(str, train_info[PLACES]))}\n"
        )

    with open(path, "w") as file:
        file.write(f"{len(trains)}\n")
        for num, train_info in trains.items():
            safe_train(num, train_info, file)


def load_trains(path=PATH):
    def load_train(file):
        num = file.readline().strip()
        train_info = {
            DEPARTURE: file.readline().strip().title(),
            PLACES: list(map(int, file.readline().split())),
            FREE_PLACES: list(map(int, file.readline().split()))
        }

        return num, train_info

    with open(path, "r") as file:
        trains = dict()
```

```
            n = int(file.readline())
            for _ in range(n):
                num, train_info = load_train(file)
                trains[num] = train_info

        return trains


    def get_train_info():
        num = input("Введите номер поезда - ").strip()
        train_info = {
            DEPARTURE:     input("Введите     пункт     отправления     -
").strip().title(),
            PLACES: list(map(int, input("Введите количество спальных мест,
в плацкарте и в купе: ").split())),
            FREE_PLACES: list(map(int, input("Введите количество свободных
спальных мест, в плацкарте и в купе: ").split()))
        }

        return num, train_info


    def print_train_info(num, train_info):
        print(f"Информация о поезде с номером {num}")
        print(f"Пункт отправления: {train_info[DEPARTURE]}")
        print(f"Количество  мест  спальных,  в  плацкарте  и  в  купе:",
*train_info[PLACES])
        print(f"Количество свободных мест спальных, в плацкарте и в купе:",
*train_info[FREE_PLACES])


    def find_with_departure(trains, departure):
        print(f"Поезда с пунктом отправления {departure}:\n")
        for train in filter(lambda x: trains[x][DEPARTURE] == departure,
trains.keys()):
            print_train_info(train, trains[train])
            print()


    def find_best_train(trains):
        print("Поезд с наилучшим соотношением свободных мест ко всем:\n")
        print_train_info(train   :=   max(trains.keys(),   key=lambda   x:
sum(trains[x][FREE_PLACES]) / sum(trains[x][PLACES])),
                         trains[train])


    def get_choice():
        os.system("cls")

        print(
            "1 - Добавить новый поезд",
            "2 - Удалить данные",
            "3 - Просмотреть информацию о поезде",
```

```
            "4 - Посмотреть информацию о всех поездах",
            "5 - Посмотреть информацию о поездах из пункта назначения",
            "6 - Найти поезд с наилучшим отношением свободных мест ко всем
местам",
            "7 - Сохранить информацию о поездах",
            "8 - Загрузить информацию о поездах",
            "9 - Прекратить выполнение программы",
            sep="\n", end="\n" * 2
        )

        try:
            choice = int(input("Введите номер команды - "))
        except ValueError:
            choice = 0
        return choice


    choice = 0
    try:
        trains = load_trains(PATH)
    except ValueError:
        trains = dict()

    while choice != 9:
        choice = get_choice()
        os.system("cls")
        if choice == 1:
            num, train_info = get_train_info()
            trains[num] = train_info
        elif choice == 2:
            num = input("Введите номер поезда для удаления - ").strip()
            try:
                trains.pop(num)
            except KeyError:
                print("Данный поезд не существует")
            else:
                print("Поезд успешно удалён")
        elif choice == 3:
            num = input("Введите номер поезда - ").strip()
            try:
                train_info = trains[num]
            except KeyError:
                print("Данный поезд не существует")
            else:
                print_train_info(num, train_info)
        elif choice == 4:
            for num, train_info in trains.items():
                print_train_info(num, train_info)
                print()
        elif choice == 5:
            departure    =    input("Введите    пункт    отправления    -
").strip().title()
            find_with_departure(trains, departure)
```

```
elif choice == 6:
    find_best_train(trains)
elif choice == 7:
    safe_trains(trains)
elif choice == 8:
    trains = load_trains()
elif choice == 9:
    pass
else:
    print("Введён неправильный номер команды")
input("\nНажмите Enter для продолжения")
```

Список литературы, рекомендуемый к использованию по данной теме

- [4] стр. 309-322;
- https://informatics.msk.ru/course/view.php?id=156#section-15.

Практическое занятие № 9
Задачи сортировки и поиска

ЦЕЛЬ: Закрепление теоретических знаний и приобретение практических навыков по составлению алгоритмов и программ различных методов сортировки.

Знания, умения и владения, приобретаемые обучающимся в результате освоения темы, в рамках формируемых компетенций (ПК-1: ИД-1$_{ПК-1}$, ИД-2$_{ПК-1}$**;** ПК-2: ИД-1$_{ПК-2}$, ИД-2$_{ПК-2}$): см. приложение 2: вырабатывается навык разработки математических, информационных и имитационных моделей, а также формируется способность к разработке и применению алгоритмических и программных решений в области прикладного программирования с умением объяснить теоретическую и практическую часть темы (формируется часть указанных компетенций).

Теоретическая часть
9. Алгоритмы решения задач внутренней сортировки и алгоритмы поиска информации

9.1. Сложность алгоритмов

Теория сложности обеспечивает методологию анализа **вычислительной сложности** различных криптографических методов и алгоритмов.

Алгоритм – это формально описанная вычислительная процедура, получающая исходные данные, называемые также входом алгоритма или его аргументом, и выдающая результат вычислений на выход.

Чаще всего, говоря о сложности, имеют в виду *временную сложность алгоритма*: *T*. Временем работы алгоритма называется число элементарных шагов, которые он выполняет. Под элементарными шагами понимают такие базисные операции, как сложение, умножение, замены, безусловные передачи управления и вызовы подпрограмм. *T* обычно представляется в виде функции от *n*, где *n* — это размер входных данных. *Функция времени вычислений* – $T_A(n)$.

Обычно вычислительная сложность алгоритма выражается с помощью нотации "*O* большого", т.е. описывается порядком величины вычислительной сложности. Это просто член разложения функции сложности, быстрее всего растущий с ростом *n*, все члены низшего порядка игнорируются. Например, если временная сложность данного алгоритма равна $4n^2+7n+2$, то вычислительная сложность порядка n^2, записываемая как $O(n^2)$ [4].

Эта нотация позволяет увидеть, как объем входных данных влияет на требования к времени и объему памяти. Например, если $T = O(n)$, то удвоение входных данных удвоит и время выполнения алгоритма. Если $T = O(2^n)$, то добавление одного бита к входным данным удвоит время выполнения алгоритма.

Обычно алгоритмы классифицируются в соответствии с их временной или пространственной сложностью. Алгоритм называют **постоянным,** если его сложность не зависит от *n:* $O(1)$. Алгоритм является **линейным,** если его временная

сложность $O(n)$. Алгоритмы могут быть **квадратичными, кубическими** и т.д. Все эти алгоритмы – **полиномиальные,** их сложность $O(n^m)$, где m – константа. Алгоритмы с полиномиальной временной сложностью называются алгоритмами **с полиномиальным временем.**

Алгоритмы, сложность которых равна $O(t^{f(n)})$, где t – константа, большая, чем 1, а $f(n)$ – некоторая полиномиальная функция от n, называются **экспоненциальными.** Подмножество экспоненциальных алгоритмов, сложность которых равна $O(c^{f(n)})$, где где c – константа, а $f(n)$ возрастает быстрее, чем постоянная, но медленнее, чем линейная функция, называется **суперполиномиальным.** Алгоритмы, сложность которых равна $O(n!)$, называются алгоритмами с факториальной сложностью. Таким образом, справедливо следующее соотношение:

$$O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n) < O(n!).$$

В настоящее время алгоритм считается практически полезным только в том случае, если его временная функция растет полиномиально относительно размеров входных данных, причем ограниченной степени. Практичными являются алгоритмы сложности $O(n)$, $O(n^2)$. Не практичны экспоненциальные и факториальные алгоритмы. Их сложность превосходит любую полиномиальную оценку.

9.2. Постановка задачи поиска. Последовательный и бинарный поиск

В практической деятельности приходится решать такие прикладные задачи, в которых необходимо локально или автономно произвести поиск нужных объектов. Рассмотрим задачу поиска элемента в массиве.

Пусть задан массив, состоящий из одного или нескольких элементов любого типа, отличного от файлового. Предположим, что в некотором массиве хранится множество из n ($n \times m$) элементов и необходимо определить положение некоторого элемента в этом массиве. Каждый элемент массива имеет индекс (индексы), причем индексы различны и однозначно идентифицируют элементы массива.

Задача поиска в этом случае состоит в отыскании индекса (индексов) элемента по заданному свойству элемента.

Существуют два возможных результата поиска:

- поиск оказался удачным, т.е. позволил определить положение элемента в массиве;
- поиск оказался неудачным, т.е. достигнут конец массива, но элемент с заданным свойством отсутствует.

Последовательный (линейный) поиск

Простейшим методом поиска является последовательный просмотр таблицы, одномерного массива или строки до нахождения элемента с требуемыми свойствами или установления факта, что такого элемента нет. Для массива, данные в котором не упорядочены сортировкой, единственный путь поиска заданного элемента состоит в сравнении каждого элемента массива с заданным. При совпадении некоторого элемента массива с заданным, его позиция в массиве фиксируется. Этот

алгоритм называется **последовательным,** или **линейным,** поиском. Ядром такого алгоритма является цикл по индексу массива. Так, если в последовательности a нужно найти x, о котором никакой дополнительной информации нет, то соответствующий цикл можно записать следующим образом:

```
for i, item in enumerate(a):
    if item == x:
        index = i
```

Этот способ решения обладает двумя существенными недостатками:

- если значение x встречается в массиве несколько раз, то будет найдено последнее из них;
- после того, как нужное значение уже найдено, массив просматривается до конца, то есть всегда выполняется n сравнений.

Для устранения этих недостатков необходимо прервать просмотр массива сразу после обнаружения заданного числа. Так как в этом случае число повторений не известно, необходимо использовать цикл с предусловием:

```
while (i < n) and (a[i] != x):
    i += 1
```

Бинарный поиск

Наиболее эффективным и распространенным методом непоследовательного поиска в одномерных *упорядоченных* массивах является **бинарный** поиск, который называют также методом *половинного деления,* методом *дихотомии,* методом *логарифмического* поиска или методом *деления пополам.* Основная идея бинарного поиска состоит в том, что сначала искомый элемент x сравнивается со средним элементом массива a. Результат сравнения либо приводит к решению задачи, либо позволяет определить, в какой части массива – левой или правой – продолжать поиск. После каждой такой итерации область поиска сокращается вдвое и не более чем через [log2n] сравнений мы закончим поиск, то есть либо попадем на элемент x, либо покажем, что x не принадлежит массиву a[n]. Например, при n = 1000 бинарный поиск в 100 раз быстрее последовательного, а при n = 1000000 – в 50000 раз. Разумеется, если массив неупорядоченный, то применить к нему бинарный поиск нельзя. Таким образом, факт упорядоченности элементов в массиве играет ключевую роль при бинарном поиске.

9.3. Постановка задачи сортировки данных

От порядка расположения данных в памяти ЭВМ, во многом зависит скорость и простота выполнения алгоритмов, предназначенных для обработки этих данных. Поэтому в программировании часто возникает задача перегруппировки данных в невозрастающем или неубывающем, порядке. Такая задача называется упорядочением или сортировкой элементов данной структуры, в простейшем случае – элементов одномерного массива.

Задача сортировки связана со многими важными приложениями, кроме того, служат хорошей иллюстрацией анализа сложности алгоритмов и тем самым позволяют разумно выбирать лучшие среди, казалось бы, равноценных методов. Каждый из алгоритмов сортировки имеет свои достоинства и недостатки, и выбирать алгоритмы нужно, исходя из конкретной постановки задачи. Выбор алгоритма сортировки существенно зависит от структуры обрабатываемых данных, поэтому все методы сортировки подразделяют на два класса: внутренний (сортировка массивов) и внешний (сортировка последовательных файлов). Различие: при внутренней сортировке все данные хранятся в оперативной памяти ЭВМ, а при внешней - во внешней памяти. При внутренней сортировке имеются более гибкие возможности для построения алгоритмов, так как все данные выстроены в виде массива и как бы лежат перед пользователем, он видит каждый элемент и имеет к нему прямой доступ. При внешней сортировке доступ к данным ограничен, поскольку они из-за своего размера в оперативной памяти не помещаются.

Уточним математическую постановку задачи сортировки данных [7].

Пусть надо упорядочить n элементов $m_1, m_2, \ldots, m_n$, которые назовем записями. Каждой записи m_j поставим в соответствие свой ключ k_j, который и будет управлять процессом сортировки. Помимо ключа запись может содержать и дополнительную информацию, которая не влияет на сортировку**,** но всегда остается в этой записи. Задача заключается *в нахождении такой перестановки* $m_{p_1}, m_{p_2}, \ldots, m_{p_n}$, *записей* $m_1, m_2, \ldots, m_n$, *после которой ключи будут располагаться в неубывающем* (*невозрастающем) порядке:*

$$k_{p_1} \le k_{p_2} \le \ldots \le k_{p_n} \; (k_{p_1} \ge k_{p_2} \ge \ldots \ge k_{p_n})$$

Сортировка называется **устойчивой,** если она удовлетворяет дополнительному условию: записи с одинаковыми ключами остаются в прежнем порядке, т.е. $p_i < p_j$, если $k_{p_i} = k_{p_j}$ и $i < j$.

Так как каждый ключ идентифицирует соответствующую запись, то эти записи можно и не определять при сортировке, рассматривая лишь их ключи (в простейшем случае, когда запись состоит лишь из сортируемых элементов**,** понятия записи и ключ совпадают).

При внутренней сортировке выбранный метод должен экономно использовать время работы процессора и память. Хорошие алгоритмы затрачивают на сортировку n записей время порядка $n \log n$, а мерой эффективности может служить число необходимых сравнений ключей и число перестановок записей. Эти числа являются функциями от n – числа сортируемых записей.

При сортировке массивов будем предполагать, что перестановки, переводящие элементы массива в нужный порядок, должны выполняться *на том же месте,* т.е. без использования промежуточного массива. Методы, в которых элементы из массива A передаются в результирующий массив B, представляют значительно меньший интерес.

9.4. Прямые и быстрые методы внутренней сортировки

Все методы внутренней сортировки делятся на прямые и быстрые (или улучшенные) [7].

Для прямых методов сортировки требуется порядка n^2 сравнений ключей, они более просты и удобны для объяснения характерных черт основных принципов большинства сортировок. Программы этих методов легко понимать и они короче программ быстрых алгоритмов.

Улучшенные методы сортировки требуют небольшого числа операций, но эти операции обычно сами более сложны, и поэтому для достаточно малых n прямые методы оказываются быстрее, хотя при больших значениях n их использовать не следует.

Методы внутренней сортировки можно разбить в соответствии с определяющими их принципами на три класса:

1. **Сортировка вставками**: элементы просматриваются по одному, и каждый новый элемент вставляется на подходящую позицию среди ране упорядоченных элементов.
2. **Сортировка с помощью выбора**: сначала выделяется наименьший (или наибольший) элемент и каким-либо образом отделяется от остальных, затем выбирается наименьший (наибольший) из оставшихся элементов и т.д.
3. **Сортировка с помощью обмена**: последовательно просматриваются пары соседних элементов; если элементы в паре образуют инверсию (т.е. неупорядочены), то они меняются местами.

Все прямые методы сортировки фактически передвигают каждый элемент массива на каждом элементарном шаге на одну позицию. Поэтому требуют $O(n^2)$ таких шагов. В основу любых улучшенных методов положен принцип перемещения элементов на каждом шаге сортировки на большие расстояния. Такой подход позволяет существенно уменьшить время сортировки, но усложнить алгоритмы выполнения.

Улучшенный метод обмена – Шейкерная сортировка, метод Хоара (самый быстрый из известных на сегодняшний день методов внутренней сортировки),
усовершенствование сортировки прямого включения – метод Шелла,
прямого выбора – сортировка с помощью бинарного дерева.

9.5. Сортировка вставками

Рассмотрим процесс сортировки методом вставки (прямого включения) массива a[1], a[2], ..., a[n]. Вначале в исходном массиве выделяется уже отсортированная, например по неубыванию, часть $a[1] \le a[2] \le ... \le a[i-1]$ и не отсортированная a[i], a[i+l],..., a[n]. Затем с шагом 1, начиная с i = 2 чередуя сравнения и перемещения элемента a[i], в отсортированной части массива определяется такой индекс (ключ) j ($1 \le j \le i-1$), чтобы выполнялись неравенства $a[j-l] \le a[i] \le a[j]$. Если такой индекс найден, то элемент a[i] вставляется на соответствующее место. Для этого выполняется сдвиг на одну позицию вправо элементов отсортированной части: a[j], …, a[i-1]. После этого осуществляется переход к следующему значению i. В противном случае просто происходит переход к следующему значению i. С

каждым шагом отсортированная часть массива увеличивается на один элемент. Очевидно, что для выполнения полной сортировки потребуется n -1 шаг, где n - число элементов исходного массива.

Покажем процесс сортировки вставками на примере целочисленного массива a[1..12], состоящего из чисел 22, -14, 13, 28, 6, 41, -5, 18, 0, 35, 6,-15 (значком * обозначены вставленные элементы, выделены элементы отсортированной части)

Исходный массив:	**22**	-14	13	28	6	41	-5	18	0	35	6	-15
1-й шаг	**-14***	**22**	13	28	6	41	-5	18	0	35	6	-15
2-й шаг:	**-14**	**13***	**22**	28	6	41	-5	18	0	35	6	-15
3-й шаг:	**-14**	**13**	**22**	**28***	6	41	-5	18	0	35	6	-15
4-й шаг:	**-14**	**6***	**13**	**22**	**28**	41	-5	18	0	35	6	-15
5-й шаг:	**-14**	**6**	**13**	**22**	**28**	**41***	-5	18	0	35	6	-15
6-й шаг:	**-14**	**-5***	**6**	**13**	**22**	**28**	**41**	18	0	35	6	-15
7-й шаг:	**-14**	**-5**	**6**	**13**	**18***	**22**	**28**	**41**	0	35	6	-15
8-й шаг:	**-14**	**-5**	**0***	**6**	**13**	**18**	**22**	**28**	**41**	35	6	-15
9-й шаг:	**-14**	**-5**	**0**	**6**	**13**	**18**	**22**	**28**	**35***	**41**	6	-15
10-й шаг:	**-14**	**-5**	**0**	**6**	**6***	**13**	**18**	**22**	**28**	**35**	**41**	-15
11-й шаг:	**-15***	**-14**	**-5**	**0**	**6**	**6**	**13**	**18**	**22**	**28**	**35**	**41**

Как видно из примера, в процессе сортировки на первом шаге первый элемент исходного массива считается отсортированным, а следующий за ним элемент сравнивается с ним. Если этот элемент больше, он остается на своем месте, в противном случае первый элемент сдвигается на вторую позицию, а второй помещается на первую (как в нашем примере). Далее все выбираемые элементы из неотсортированной части массива последовательно сравниваются с элементами отсортированной части, начиная с первого, до тех пор, пока не встретится больший или равный ему. Этот элемент и все последующие элементы отсортированной части перемещаются на одну позицию вправо (или вниз, если массив расположен в столбик), освобождая место для нового элемента. Этот процесс закончится при выполнении одного из двух условий:

- найден первый слева элемент a[j] ≥ a[i], что говорит о необходимости вставки a[i] между a[j] и a[j-1], в частности, если j=1, то a[i] помещается в первую позицию, что влечет за собой необходимость увеличения диапазона индекса массива на единицу;
- достигнут правый конец отсортированной части массива (правый барьер), следовательно, элемент a[i] нужно оставить на прежнем месте.

Заметим, что сравнение и перемещение элемента a[i] в отсортированной части массива можно проводить и справа налево, начиная с (i-1)-го элемента.

9.6. Сортировка с помощью прямого выбора

Сортировка массива размером n в неубывающем (невозрастающем) порядке методом прямого выбора выполняется по следующему алгоритму.

1. Выбирается элемент данного массива с минимальным (максимальным) значением (или ключом).
2. Выбранный элемент и первый элемент меняются местами.

3. Этот процесс повторяется с оставшимися n - 1 элементами, затем с n - 2 элементами и т.д. до тех пор, пока не останется один, самый большой/меньший элемент.

Всего потребуется n - 1 раз выполнить указанную последовательность действий. В процессе сортировки будет увеличиваться отсортированная часть массива, а неотсортированная соответственно уменьшаться.

В отличие от сортировки вставками, когда а каждом шаге рассматривается только один очередной элемент исходного массива и **все** элементы отсортированной части, среди которых ищется место для вставки, при прямом выборе для поиска одного элемента с наименьшим (наибольшим) ключом просматриваются все элементы исходного массива и найденный элемент помещается как очередной в отсортированную часть массива.

Напомним, что во всех рассматриваемых методах сортировки мы не используем промежуточных массивов.

Схему сортировки методом прямого выбора проиллюстрируем на примере массива, рассмотренного в предыдущем пункте. Элементы, которые меняются местами на очередном шаге, выделены полужирны шрифтом, а минимальные элементы неотсортированной части, кроме того, отмечены звездочкой. Отсортированная часть подчеркнута.

Исходный массив:	**22**	-14	13	28	6	41	-5	18	0	35	6	**-15***
1-й шаг	-15	**-14***	13	28	6	41	-5	18	0	35	6	22
2-й шаг:	-15	-14	**13**	28	6	41	**-5***	18	0	35	6	22
3-й шаг:	-15	-14	-5	**28**	6	41	13	18	**0***	35	6	22
4-й шаг:	-15	-14	-5	0	**6***	41	13	18	28	35	6	22
5-й шаг:	-15	-14	-5	0	6	**41**	13	18	28	35	**6***	22
6-й шаг:	-15	-14	-5	0	6	6	**13***	18	28	35	41	22
7-й шаг:	-15	-14	-5	0	6	6	13	**18***	28	35	41	22
8-й шаг:	-15	-14	-5	0	6	6	13	18	**28**	35	41	**22***
9-й шаг:	-15	-14	-5	0	6	6	13	18	22	**35**	41	**28***
10-й шаг:	-15	-14	-5	0	6	6	13	18	22	28	**41**	**35**
11-й шаг:	-15	-14	-5	0	6	6	13	18	22	28	35	41

9.7. Сортировка с помощью прямого обмена

Алгоритм данного метода сортировки заключается в последовательных просмотрах массива от начала к концу и обмене местами *соседних* элементов, если они образуют *инверсию.* Опишем алгоритм сортировки по **неубыванию** подробнее.

Пусть задан массив a размером n. Начинаем просмотр массива с первой пары элементов a[1] и a[2]. Если первый элемент этой пары больше второго, то меняем их местами, иначе оставляем их без изменения и сравниваем вторую пару элементов a[2] и a[3]. Если a|2] > a[3], то меняем их местами и т.д. На первом шаге будут просмотрены все пары элементов массива: a[i] и a[i+1] для i от 1 до n -1. В результате такого просмотра и необходимых обменов *максимальный* элемент переместится в конец массива и будет являться отсортированной частью. На втором шаге аналогичная процедура проводится с первого до (n - 1)-го элемента. Тем самым

второй по величине элемент массива переместится на предпоследнее место. Отсортированная часть будет содержать два элемента. Эти действия продолжаются до тех пор, пока количество элементов в неотсортированной части массива не уменьшится до двух. На последнем шаге выполняется упорядочение оставшихся двух элементов. После (n - 1) шагов массив окажется отсортированным по **неубыванию.** Для иллюстрации отсортируем массив по неубыванию методом простого обмена. Отсортированная часть выделена полужирным шрифтом.

Исходный массив:	22	-14	13	28	6	41	-5	18	0	35	6	-15
1-й шаг	-14	13	22	6	28	-5	18	0	35	6	-15	**41**
2-й шаг:	-14	13	6	22	-5	18	0	28	6	-15	**35**	**41**
3-й шаг:	-14	6	13	-5	18	0	22	6	-15	**28**	**35**	**41**
4-й шаг:	-14	6	-5	13	0	18	6	-15	**22**	**28**	**35**	**41**
5-й шаг:	-14	-5	6	0	13	6	-15	**18**	**22**	**28**	**35**	**41**
6-й шаг:	-14	-5	0	6	6	-15	**13**	**18**	**22**	**28**	**35**	**41**
7-й шаг:	-14	-5	0	6	-15	**6**	**13**	**18**	**22**	**28**	**35**	**41**
8-й шаг:	-14	-5	0	-15	**6**	**6**	**13**	**18**	**22**	**28**	**35**	**41**
9-й шаг:	-14	-5	-15	**0**	**6**	**6**	**13**	**18**	**22**	**28**	**35**	**41**
10-й шаг:	-14	-15	**-5**	**0**	**6**	**6**	**13**	**18**	**22**	**28**	**35**	**41**
11-й шаг:	-15	**-14**	**-5**	**0**	**6**	**6**	**13**	**18**	**22**	**28**	**35**	**41**

Сортировку методом прямого обмена называют еще методом "пузырька". Это название происходит от образной интерпретации, при которой элементы данного массива расположены вертикально и в **процессе** выполнения сортировки на каждом шаге более легкие элементы, как пузырьки в ванне с водой, поднимаются до уровня, соответствующего их весу.

9.8. Функции и методы объектов Python, решающие задачи поиска и сортировки

Так как Python обладает довольно большим количеством встроенных модулей и функций, многие алгоритмы в нём уже реализованы и, чаще всего, будут работать эффективнее написанных самостоятельно.

Для объектов типа `str`, `list`, `tuple` реализован метод `index`, позволяющий находить индекс первого слева вхождение элемента в данный объект. При этом для строк возможен как поиск отдельного символа, так и подстроки. Помимо искомого значения эти методы также могут принимать аргументы, обозначающие в каком промежутке мы производим поиск. В случае отсутствия элемента произойдёт `ValueError`.

Функция `sorted` принимает итерируемый объект и возвращает список из его элементов, отсортированных в порядке возрастания. Параметр `key` принимает как аргумент функцию, которая определяет «стоимость» каждого элемента, по умолчанию `None`. Параметр `reversed` принимает как аргумент булево значение, если `True`, то сортировка будет по убыванию, по умолчанию `False`. Для объектов list существует метод sort, который сортирует сам список, а не создаёт новый. Данные сортировки работают за O(n*log(n)) и являются устойчивыми.

Двоичный поиск реализован в модуле bisect, подробнее с ним вы можете ознакомиться используя встроенную документацию.

Вопросы и задания

Задание I

Реализуйте поиск индекса элемента в списке и в матрице (*i*, *j*) двумя способами: используя встроенные функции и методы, представленные в **9.8.**, а также не используя их.

Задание II

Реализуйте сортировку матрицы по следующим условиям соответственно варианту, как используя функции и методы, представленные в **9.8.** (можно использовать numpy), так и не используя их:

1. Сортировка всех элементов слева-направо сверху-вниз в порядке возрастания.

2. Сортировка всех строк по наибольшему значению в ней в порядке возрастания, сортировка в каждой строке в порядке убывания.

3. Сортировка всех строк по наименьшему значению в ней в порядке возрастания, сортировка в каждой строке в порядке возрастания.

4. Сортировка столбцов по сумме элементов в них в порядке возрастания.

5. Сортировка строк в порядке возрастания по их элементам, которые являются членами главной диагонали. Последующая аналогичная сортировка по столбцам.

6. Сортировка всех строк по их норме в порядке возрастания (норма задана стандартным скалярным произведением)

7. Пусть fib-значением некоторой строки называется число, получаемое сложением произведений *i*-го числа строки на *i*-е число Фибоначчи для всех чисел строки (0 и 1 элементы чисел Фибоначчи равны 1 и 1). Сортировка строк матрицы по их fib-значению в порядке возрастания.

Задание III

Реализуйте игру по угадыванию числа, загаданного пользователем на промежутке [-1024; 1024]. Программа должна узнавать у пользователя, больше или меньше загаданное им число, чем данное программой, или было ли число угадано. Программа должна справляться с задачей меньше, чем за 15 запросов.

Контрольные вопросы

1. Понятие сложности алгоритмов.
2. На какие классы делятся алгоритмы в соответствии с их временной или пространственной сложностью?
3. Постановка задачи сортировки данных.
4. Прямые и быстрые методы внутренней сортировки.
5. Алгоритм сортировки массива методом вставки.
6. Алгоритм сортировки массива методом прямого выбора.
7. Алгоритм сортировки массива методом прямого обмена.

8. Понятие инверсии.
9. Постановка задачи поиска элемента в массиве.
10. Алгоритм последовательного (линейного) поиска.
11. Алгоритм бинарного поиска.
12. Реализованные методы для нахождения индекса элемента
13. Функция sorted и её параметры
14. Модуль bisect и его функции.

Список литературы, рекомендуемый к использованию по данной теме

- [4] стр. 275-2764
- [5] стр. 58-128;
- https://informatics.msk.ru/course/view.php?id=156#section-12.

Литература

Перечень основной литературы:

1. Шелудько, В. М. Основы программирования на языке высокого уровня Python Электронный ресурс : Учебное пособие / В. М. Шелудько. – Ростов-на-Дону, Таганрог : Издательство Южного федерального университета, 2017. – 146 . с.
2. Шелудько, В. М.; Язык программирования высокого уровня Python: функции, структуры данных, дополнительные модули : учебное пособие / В.М. Шелудько ; Министерство науки и высшего образования РФ ; Федеральное государственное автономное образовательное учреждение высшего образования «Южный федеральный университет»
3. Северенс, Ч. Введение в программирование на Python / Ч. Северенс. – 2-е изд., испр. – Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 231 с. : схем., ил. - http://biblioclub.ru/
4. Марк Лутц “Изучаем Python”, том 1 издание 5: Пер. с англ. – СПб.: ООО “Диалектика”, 2019. – 832 с. :
5. Дж. Вандер Плас «Python для сложных задач. Наука о данных и машинное обучение» . – СПб.: Питер, 2018. – 576 с

Перечень дополнительной литературы:

6. М. В. Сысоева, И. В. Сысоев Программирование для нормальных с нуля на языке Python
7. Шкаберина, Г. Ш.; Программирование. Основы языка Python Электронный ресурс / Шкаберина Г. Ш., Резова Н. Л. : учебное пособие. - Красноярск : СибГУ им. академика М. Ф. Решетнёва, 2018. – 92 с.
8. Северенс, Ч. Введение в программирование на Python / Ч. Северенс. – 2-е изд., испр. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 231 с.
9. Хахаев, И. А Практикум по алгоритмизации и программированию на Python : курс / И.А. Хахаев. - 2-е изд., исправ. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. – 179 с
10. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. – М.: МЦНМО, 2000.
11. Д. П. Кириенко. Программирование на языке Python (школа 179 г. Москвы) - Курс: Д. П. Кириенко. Программирование на языке Python (школа 179 г. Москвы) (informatics.msk.ru)

Приложение 1

Порядок выполнения работ в ходе практических занятий

1. Ознакомиться с теоретическим материалом, соответствующим теме практического занятия, и подготовить ответы на предложенные контрольные вопросы.
2. Из списка задач выбрать соответствующие своему варианту. Для каждой задачи разработать математическую модель, алгоритм и программу решения.
3. Реализовать программу на ЭВМ.
4. Для получения оценки по итогам практического занятия знать ответы на контрольные вопросы и предоставить письменный отчёт, содержащий:

   а) тему практического занятия;

   б) номер варианта;

   в) цель;

   г) решение каждой задачи, включающее в себя:

   - условие задачи;
   - математическую модель;
   - алгоритм;
   - листинг программы с отметкой преподавателя о её выполнении;
   - исходный набор данных и соответствующий ему результат;
   - анализ результатов работы программы.

Приложение 2

Перечень планируемых результатов обучения по дисциплине, соотнесённых с планируемыми результатами освоения образовательной программы

Таблица П1 – Перечень планируемых результатов

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине, характеризующие этапы формирования компетенций, индикаторов
ОПК-8 Способен понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности	**ИД-1опк-8** Способен понимать принципы работы современных информационных технологий	Демонстрирует знание принципы работы современных информационных технологий и умение использовать современныех информационные технологии для решения прикладных задач
	ИД-2опк-8 Способен использовать современные информационные технологии для решения задач профессиональной деятельности	Умеет разрабатывать программное и информационное обеспечение компьютерных систем, вычислительных комплексов, баз данных

Таблица П2 – **Описание показателей и критериев оценивания**

Компетенция (ии), индикатор (ы)	Уровни сформированности компетенции(ий)			
	Минимальный уровень не достигнут (неудовлетворительно) 2 балла	**Минимальный уровень (удовлетворительно) 3 балла**	**Средний уровень (хорошо) 4 балла**	**Высокий уровень (отлично) 5 баллов**
Компетенция: **ОПК-8 Способен понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности**				
Результаты обучения по дисциплине: *Индикатор:* **ИД-1опк-8** Способен понимать принципы работы современных информационных технологий	Не демонстрирует способности понимания принципов работы современных информационных технологий Существенные пробелы во владении знаниями разделов дисциплины. Знания фрагментарные, неструктурированные, декларативные. Слабое понимание предмета дисциплины. Примерное представление о существующих методах приёмах решения прикладных задач с использованием языка программирования Питон. Начальный уровень самостоятельности мышления, умение частично	Владение плохо структурированными декларативными знаниями, частичное распознавание отдельных блоков знания и соотнесение их между собой. Понимание ключевых аспектов предмета в рамках рабочей программы дисциплины. Умение применять ограниченный спектр стандартных методов и аналитических приёмов, допуская существенные ошибки для решения задач с использованием	Полное, но не детальное владение знаниями математических и естественных наук - строго в рамках рабочей программы. Общее понимание материала в соответствии с рабочей программой дисциплины. Умение применять полный спектр методов и аналитических приёмов, допуская несущественные ошибки при решении задач с использованием языка программирования Питон. Умение проводить сложный анализ и демонстрировать уверенную доказательную аргументацию, формулировать исследовательский вопрос, находить решение, допуская отдельные неточности, оценивать критически альтернативные подходы	Расширенный по сравнению с рабочей программой диапазон владения знаниями всех разделов дисциплины включая информацию из дополнительных источников. Глубокое понимание предмета дисциплины. Умение эффективно применять современные методы и аналитические приёмы, направленные на решение задач с использованием языка программирования Питон. Отличные аналитические способности и всесторонняя убедительная доказательная аргументация.

	или с ошибками воспроизводить структуру имеющегося знания.	языка программирования Питон.	к решению задач с использованием языка программирования Питон.	
Результаты обучения по дисциплине: *Индикатор:* **ИД-2$_{ОПК-8}$** Способен использовать современные информационные технологии для решения задач профессиональной деятельности	Не способен использовать современные информационные технологии для решения задач профессиональной деятельности	Удовлетворительно применяет информационные технологии и среды программирования для решения профессиональных задач с использованием языка программирования Питон	Хорошо применяет информационные технологии и среды программирования для решения профессиональных задач с использованием языка программирования Питон	Высокий уровень самостоятельности мышления. Умение формулировать актуальный исследовательский вопрос, находить оптимальное решение и критически оценивать существующие альтернативные подходы к решению математических задач с использованием языка программирования Питон.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания
по организации и проведению самостоятельной работы
по дисциплине
«АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ»
для студентов направления подготовки
02.03.01 Математика и компьютерные науки

Ставрополь, 2026 г.

Общие положения

Самостоятельная работа - планируемая учебная, учебно-исследовательская, научно-исследовательская работа студентов, выполняемая во внеаудиторное (аудиторное) время по заданию и при методическом руководстве преподавателя, но без его непосредственного участия (при частичном непосредственном участии преподавателя, оставляющем ведущую роль за работой студентов).

Самостоятельная работа студентов (СРС) в ВУЗе является важным видом учебной и научной деятельности студента. Самостоятельная работа студентов играет значительную роль в рейтинговой технологии обучения.

К основным видам самостоятельной работы студентов относятся:

– формирование и усвоение содержания конспекта лекций на базе рекомендованной лектором учебной литературы, включая информационные образовательные ресурсы (электронные учебники, электронные библиотеки и др.);

– написание докладов;

– подготовка к семинарам, практическим и лабораторным работам, их оформление;

– составление аннотированного списка статей из соответствующих журналов по отраслям знаний (педагогических, психологических, методических и др.);

– выполнение учебно-исследовательских работ, проектная деятельность;

– подготовка практических разработок и рекомендаций по решению проблемной ситуации;

– выполнение домашних заданий в виде решения отдельных задач, проведения типовых расчетов, расчетно-компьютерных и индивидуальных работ по отдельным разделам содержания дисциплин и т.д.;

– компьютерный текущий самоконтроль и контроль успеваемости на базе электронных обучающих и аттестующих тестов;

– выполнение курсовых работ (проектов) в рамках дисциплин;

– выполнение выпускной квалификационной работы и др.

Методика организации самостоятельной работы студентов зависит от структуры, характера и особенностей изучаемой дисциплины, объема часов на ее изучение, вида заданий для самостоятельной работы студентов, индивидуальных качеств студентов и условий учебной деятельности.

Процесс организации самостоятельной работы студентов включает в себя следующие этапы:

- подготовительный (определение целей, составление программы, подготовка методического обеспечения, подготовка оборудования);
- основной (реализация программы, использование приемов поиска информации, усвоения, переработки, применения, передачи знаний, фиксирование результатов, самоорганизация процесса работы);
- заключительный (оценка значимости и анализ результатов, их систематизация, оценка эффективности программы и приемов работы, выводы о направлениях оптимизации труда).

Самостоятельная работа по дисциплине **«Алгоритмизация и программирование»** направлена на формирование следующих **компетенций**:

ПК-1 Способен демонстрировать базовые знания математических и естественных наук, основ программирования и информационных технологий

ПК-2. Способен преподавать математику и информатику в образовательных организациях общего образования и среднего профессионального образования на основе полученного фундаментального образования и научного мировоззрения

1. Цель и задачи самостоятельной работы

Ведущая цель организации и осуществления СРС совпадает с целью обучения студента – формирование соответствующего набора компетенций будущего бакалавра по направлению подготовки «Математика и компьютерные науки».
При организации СРС важным и необходимым условием становятся формирование умения самостоятельной работы для приобретения знаний, навыков и возможности организации учебной и научной деятельности. Целью самостоятельной работы студентов является овладение фундаментальными знаниями, профессиональными умениями и навыками деятельности по профилю, опытом творческой, исследовательской деятельности.
Самостоятельная работа студентов способствует развитию самостоятельности, ответственности и организованности, творческого подхода к решению проблем учебного и профессионального уровня.
Задачами СРС являются:

- систематизация и закрепление полученных теоретических знаний и практических умений студентов;
- углубление и расширение теоретических знаний;
- формирование умений использовать нормативную, правовую, справочную документацию и специальную литературу;
- развитие познавательных способностей и активности студентов: творческой инициативы, самостоятельности, ответственности и организованности;
- формирование самостоятельности мышления, способностей к саморазвитию, самосовершенствованию и самореализации;
- развитие исследовательских умений;
- использование материала, собранного и полученного в ходе самостоятельных занятий на семинарах, на практических и лабораторных занятиях, при написании курсовых и выпускной квалификационной работ, для эффективной подготовки к итоговым зачетам и экзаменам.

2. Порядок выполнения самостоятельной работы студентом

2.1. Методические рекомендации по работе с учебной литературой

При работе с книгой необходимо подобрать литературу, научиться правильно ее читать, вести записи. Для подбора литературы в библиотеке используются алфавитный и систематический каталоги.
Важно помнить, что рациональные навыки работы с книгой - это всегда большая экономия времени и сил.
Правильный подбор учебников рекомендуется преподавателем, читающим лекционный курс. Необходимая литература может быть также указана в методических разработках по данному курсу.
Изучая материал по учебнику, следует переходить к следующему вопросу только после правильного уяснения предыдущего, описывая на бумаге все выкладки и вычисления (в том числе те, которые в учебнике опущены или на лекции даны для самостоятельного вывода).
При изучении любой дисциплины большую и важную роль играет самостоятельная индивидуальная работа.
Особое внимание следует обратить на определение основных понятий курса. Студент должен подробно разбирать примеры, которые поясняют такие определения, и уметь строить аналогичные примеры самостоятельно. Нужно добиваться точного представления о том, что изучаешь.
Полезно составлять опорные конспекты. При изучении материала по учебнику полезно в тетради (на специально отведенных полях) дополнять конспект лекций. Там же следует отмечать вопросы, выделенные студентом для консультации с преподавателем.

Выводы, полученные в результате изучения, рекомендуется в конспекте выделять, чтобы они при перечитывании записей лучше запоминались.
Опыт показывает, что многим студентам помогает составление листа опорных сигналов, содержащего важнейшие и наиболее часто употребляемые формулы и понятия. Такой лист помогает запомнить формулы, основные положения лекции, а также может служить постоянным справочником для студента.
Чтение научного текста является частью познавательной деятельности. Ее цель – извлечение из текста необходимой информации. От того на сколько осознанна читающим собственная внутренняя установка при обращении к печатному слову (найти нужные сведения, усвоить информацию полностью или частично, критически проанализировать материал и т.п.) во многом зависит эффективность осуществляемого действия.
Выделяют ***четыре основные установки в чтении научного текста***:
информационно-поисковый (задача – найти, выделить искомую информацию)
усваивающая (усилия читателя направлены на то, чтобы как можно полнее осознать и запомнить как сами сведения излагаемые автором, так и всю логику его рассуждений)
аналитико-критическая (читатель стремится критически осмыслить материал, проанализировав его, определив свое отношение к нему)
творческая (создает у читателя готовность в том или ином виде – как отправной пункт для своих рассуждений, как образ для действия по аналогии и т.п. – использовать суждения автора, ход его мыслей, результат наблюдения, разработанную методику, дополнить их, подвергнуть новой проверке).
Основные виды систематизированной записи прочитанного:
Аннотирование – предельно краткое связное описание просмотренной или прочитанной книги (статьи), ее содержания, источников, характера и назначения;
Планирование – краткая логическая организация текста, раскрывающая содержание и структуру изучаемого материала;
Тезирование – лаконичное воспроизведение основных утверждений автора без привлечения фактического материала;
Цитирование – дословное выписывание из текста выдержек, извлечений, наиболее существенно отражающих ту или иную мысль автора;
Конспектирование – краткое и последовательное изложение содержания прочитанного.
Конспект – сложный способ изложения содержания книги или статьи в логической последовательности. Конспект аккумулирует в себе предыдущие виды записи, позволяет всесторонне охватить содержание книги, статьи. Поэтому умение составлять план, тезисы, делать выписки и другие записи определяет и технологию составления конспекта.
Методические рекомендации по составлению конспекта:

1. Внимательно прочитайте текст. Уточните в справочной литературе непонятные слова. При записи не забудьте вынести справочные данные на поля конспекта;
2. Выделите главное, составьте план;
3. Кратко сформулируйте основные положения текста, отметьте аргументацию автора;
4. Законспектируйте материал, четко следуя пунктам плана. При конспектировании старайтесь выразить мысль своими словами. Записи следует вести четко, ясно.
5. Грамотно записывайте цитаты. Цитируя, учитывайте лаконичность, значимость мысли.

В тексте конспекта желательно приводить не только тезисные положения, но и их доказательства. При оформлении конспекта необходимо стремиться к емкости каждого предложения. Мысли автора книги следует излагать кратко, заботясь о стиле и выразительности написанного. Число дополнительных элементов конспекта должно быть логически обоснованным, записи должны распределяться в определенной последовательности, отвечающей логической структуре произведения. Для уточнения и дополнения необходимо оставлять поля.
Овладение навыками конспектирования требует от студента целеустремленности, повседневной самостоятельной работы.

2.2. Методические рекомендации по подготовке к практическим и лабораторным занятиям

Для того чтобы практические и лабораторные занятия приносили максимальную пользу, необходимо помнить, что упражнение и решение задач проводятся по вычитанному на лекциях материалу и связаны, как правило, с детальным разбором отдельных вопросов лекционного курса. Следует подчеркнуть, что только после усвоения лекционного материала с определенной точки зрения (а именно с той, с которой он излагается на лекциях) он будет закрепляться на практических занятиях как в результате обсуждения и анализа лекционного материала, так и с помощью решения проблемных ситуаций, задач. При этих условиях студент не только хорошо усвоит материал, но и научится применять его на практике, а также получит дополнительный стимул (и это очень важно) для активной проработки лекции.

При самостоятельном решении задач нужно обосновывать каждый этап решения, исходя из теоретических положений курса. Если студент видит несколько путей решения проблемы (задачи), то нужно сравнить их и выбрать самый рациональный. Полезно до начала вычислений составить краткий план решения проблемы (задачи). Решение проблемных задач или примеров следует излагать подробно, вычисления располагать в строгом порядке, отделяя вспомогательные вычисления от основных. Решения при необходимости нужно сопровождать комментариями, схемами, чертежами и рисунками.

Следует помнить, что решение каждой учебной задачи должно доводиться до окончательного логического ответа, которого требует условие, и по возможности с выводом. Полученный ответ следует проверить способами, вытекающими из существа данной задачи. Полезно также (если возможно) решать несколькими способами и сравнить полученные результаты. Решение задач данного типа нужно продолжать до приобретения твердых навыков в их решении.

2.3. Методические рекомендации по самопроверке знаний

После изучения определенной темы по записям в конспекте и учебнику, а также решения достаточного количества соответствующих задач на практических занятиях и самостоятельно студенту рекомендуется, провести самопроверку усвоенных знаний, ответив на контрольные вопросы по изученной теме.

В случае необходимости нужно еще раз внимательно разобраться в материале.

Иногда недостаточность усвоения того или иного вопроса выясняется только при изучении дальнейшего материала. В этом случае надо вернуться назад и повторить плохо усвоенный материал. Важный критерий усвоения теоретического материала - умение решать задачи или пройти тестирование по пройденному материалу. Однако следует помнить, что правильное решение задачи может получиться в результате применения механически заученных формул без понимания сущности теоретических положений.

2.4. Методические рекомендации по написанию научных текстов (докладов, рефератов, эссе, научных статей и т.д.)

Перед тем, как приступить к написанию научного текста, важно разобраться, какова истинная цель вашего научного текста - это поможет вам разумно распределить свои силы и время. Во-первых, сначала нужно определиться с идеей научного текста, а для этого необходимо научиться либо относиться к разным явлениям и фактам несколько критически (своя идея – как иная точка зрения), либо научиться увлекаться какими-то известными идеями, которые нуждаются в доработке (идея – как оптимистическая позиция и направленность на дальнейшее совершенствование уже известного). Во-вторых, научиться организовывать свое время, ведь, как известно, свободное (от всяких глупостей) время – важнейшее условие настоящего творчества, для него наконец-то появляется время. Иногда именно на организацию такого времени уходит немалая часть сил и талантов.

Писать следует ясно и понятно, стараясь основные положения формулировать четко и недвусмысленно (чтобы и самому понятно было), а также стремясь структурировать свой текст. Каждый раз надо представлять, что ваш текст будет кто-то читать и ему захочется сориентироваться в нем, быстро находить ответы на интересующие вопросы (заодно представьте себя на месте такого человека). Понятно, что работа, написанная «сплошным текстом» (без заголовков, без выделения крупным шрифтом наиболее важным мест и т, п.), у культурного читателя должна вызывать брезгливость и даже жалость к автору (исключения составляют некоторые древние тексты, когда и жанр был иной и к текстам относились иначе, да и самих текстов было гораздо меньше – не то, что в эпоху «информационного взрыва» и соответствующего «информационного мусора»).

Объем текста и различные оформительские требования во многом зависят от принятых в конкретном учебном заведении порядков.

Реферат (доклад) - это самостоятельное исследование студентом определенной проблемы, комплекса взаимосвязанных вопросов.

Реферат не должна составляться из фрагментов статей, монографий, пособий. Кроме простого изложения фактов и цитат, в реферате должно проявляться авторское видение проблемы и ее решения.

Рассмотрим основные этапы подготовки реферата студентом.

Выполнение реферата начинается с выбора темы.

Затем студент приходит на первую консультацию к руководителю, которая предусматривает:

- обсуждение цели и задач работы, основных моментов избранной темы;
- консультирование по вопросам подбора литературы;
- составление предварительного плана.

Следующим этапом является работа с литературой. Необходимая литература подбирается студентом самостоятельно.

После подбора литературы целесообразно сделать рабочий вариант плана работы. В нем нужно выделить основные вопросы темы и параграфы, раскрывающие их содержание.

Составленный список литературы и предварительный вариант плана уточняются, согласуются на очередной консультации с руководителем.

Затем начинается следующий этап работы - изучение литературы. Только внимательно читая и конспектируя литературу, можно разобраться в основных вопросах темы и подготовиться к самостоятельному (авторскому) изложению содержания реферата. Конспектируя первоисточники, необходимо отразить основную идею автора и его позицию по исследуемому вопросу, выявить проблемы и наметить задачи для дальнейшего изучения данных проблем.

Систематизация и анализ изученной литературы по проблеме исследования позволяют студенту написать работу.

Рабочий вариант текста реферата предоставляется руководителю на проверку. На основе рабочего варианта текста руководитель вместе со студентом обсуждает возможности доработки текста, его оформление. После доработки реферат сдается на кафедру для его оценивания руководителем.

Требования к написанию реферата

Написание 1 реферата является обязательным условием выполнения плана СРС по любой дисциплине профессионального цикла.

Тема реферата может быть выбрана студентом из предложенных в рабочей программе или фонде оценочных средств дисциплины, либо определена самостоятельно, исходя из интересов студента (в рамках изучаемой дисциплины). Выбранную тему необходимо согласоваться с преподавателем.

Реферат должен быть написан научным языком.

Объем реферата должен составлять 20-25 стр.

Структура реферата:

- Введение (не более 3-4 страниц). Во введении необходимо обосновать выбор темы, ее актуальность, очертить область исследования, объект исследования, основные цели и задачи исследования.

- Основная часть состоит из 2-3 разделов. В них раскрывается суть исследуемой проблемы, проводится обзор мировой литературы и источников Интернет по предмету исследования, в котором дается характеристика степени разработанности проблемы и авторская аналитическая оценка основных теоретических подходов к ее решению. Изложение материала не должно ограничиваться лишь описательным подходом к раскрытию выбранной темы. Оно также должно содержать собственное видение рассматриваемой проблемы и изложение собственной точки зрения на возможные пути ее решения.
- Заключение (1-2 страницы). В заключении кратко излагаются достигнутые при изучении проблемы цели, перспективы развития исследуемого вопроса
- Список использованной литературы (не меньше 10 источников), в алфавитном порядке, оформленный в соответствии с принятыми правилами. В список использованной литературы рекомендуется включать работы отечественных и зарубежных авторов, в том числе статьи, опубликованные в научных журналах в течение последних 3-х лет и ссылки на ресурсы сети Интернет.
- Приложение (при необходимости).

Требования к оформлению:

- текст с одной стороны листа;
- шрифт Times New Roman;
- кегль шрифта 14;
- межстрочное расстояние 1,5;
- поля: сверху 2,5 см, снизу – 2,5 см, слева - 3 см, справа 1,5 см;
- реферат должен быть представлен в сброшюрованном виде.

Порядок защиты реферата:

Защита реферата проводится на практических занятиях, после окончания работы студента над ним и исправления всех недочетов, выявленных преподавателем в ходе консультаций. На защиту реферата отводится 5-7 минут времени, в ходе которого студент должен показать свободное владение материалом по заявленной теме. При защите реферата приветствуется использование мультимедиа-презентации.

Оценка реферата

Реферат оценивается по следующим критериям:

• соблюдение требований к его оформлению;

• необходимость и достаточность для раскрытия темы приведенной в тексте реферата информации;

• умение студента свободно излагать основные идеи, отраженные в реферате;

• способность студента понять суть задаваемых преподавателем и сокурсниками вопросов и сформулировать точные ответы на них.

Критерии оценки:

Оценка «отлично» выставляется студенту, если в докладе студент исчерпывающе, последовательно, четко и логически стройно излагает материал; свободно справляется с задачами, вопросами и другими видами применения знаний; использует для написания доклада современные научные материалы; анализирует полученную информацию; проявляет самостоятельность при написании доклада.

Оценка «хорошо» выставляется студенту, если качество выполнения доклада достаточно высокое. Студент твердо знает материал, грамотно и по существу излагает его, не допуская существенных неточностей в ответе на вопросы по теме доклада.

Оценка «удовлетворительно» выставляется студенту, если материал доклада излагается частично, но пробелы не носят существенного характера, студент допускает неточности и ошибки при защите доклада, дает недостаточно правильные формулировки, наблюдаются нарушения логической последовательности в изложении материала.

Оценка «неудовлетворительно» выставляется студенту, если он не подготовил доклад или допустил существенные ошибки. Студент неуверенно излагает материал доклада, не отвечает на вопросы преподавателя.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания
по организации и проведению самостоятельной работы
по дисциплине

«АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ»

для студентов направления подготовки
43.03.01 Сервис

Ставрополь, 2026 г.

Общие положения

Самостоятельная работа - планируемая учебная, учебно-исследовательская, научно-исследовательская работа студентов, выполняемая во внеаудиторное (аудиторное) время по заданию и при методическом руководстве преподавателя, но без его непосредственного участия (при частичном непосредственном участии преподавателя, оставляющем ведущую роль за работой студентов).

Самостоятельная работа студентов (СРС) в ВУЗе является важным видом учебной и научной деятельности студента. Самостоятельная работа студентов играет значительную роль в рейтинговой технологии обучения.

К основным видам самостоятельной работы студентов относятся:

− формирование и усвоение содержания конспекта лекций на базе рекомендованной лектором учебной литературы, включая информационные образовательные ресурсы (электронные учебники, электронные библиотеки и др.);

− написание докладов;

− подготовка к семинарам, практическим и лабораторным работам, их оформление;

− составление аннотированного списка статей из соответствующих журналов по отраслям знаний (педагогических, психологических, методических и др.);

− выполнение учебно-исследовательских работ, проектная деятельность;

− подготовка практических разработок и рекомендаций по решению проблемной ситуации;

− выполнение домашних заданий в виде решения отдельных задач, проведения типовых расчетов, расчетно-компьютерных и индивидуальных работ по отдельным разделам содержания дисциплин и т.д.;

− компьютерный текущий самоконтроль и контроль успеваемости на базе электронных обучающих и аттестующих тестов;

− выполнение курсовых работ (проектов) в рамках дисциплин;

− выполнение выпускной квалификационной работы и др.

Методика организации самостоятельной работы студентов зависит от структуры, характера и особенностей изучаемой дисциплины, объема часов на ее изучение, вида заданий для самостоятельной работы студентов, индивидуальных качеств студентов и условий учебной деятельности.

Процесс организации самостоятельной работы студентов включает в себя следующие этапы:

- подготовительный (определение целей, составление программы, подготовка методического обеспечения, подготовка оборудования);
- основной (реализация программы, использование приемов поиска информации, усвоения, переработки, применения, передачи знаний, фиксирование результатов, самоорганизация процесса работы);
- заключительный (оценка значимости и анализ результатов, их систематизация, оценка эффективности программы и приемов работы, выводы о направлениях оптимизации труда).

Самостоятельная работа по дисциплине **«АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ»** направлена на формирование следующих **компетенций**:

ОПК-8 Способен понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности

1. Цель и задачи самостоятельной работы

Ведущая цель организации и осуществления СРС совпадает с целью обучения студента – формирование соответствующего набора компетенций будущего бакалавра по направлению подготовки «Математика и компьютерные науки».

При организации СРС важным и необходимым условием становятся формирование умения самостоятельной работы для приобретения знаний, навыков и возможности организации учебной и научной деятельности. Целью самостоятельной работы студентов является овладение фундаментальными знаниями, профессиональными умениями и навыками деятельности по профилю, опытом творческой, исследовательской деятельности. Самостоятельная работа студентов способствует развитию самостоятельности, ответственности и организованности, творческого подхода к решению проблем учебного и профессионального уровня.

Задачами СРС являются:

- систематизация и закрепление полученных теоретических знаний и практических умений студентов;
- углубление и расширение теоретических знаний;
- формирование умений использовать нормативную, правовую, справочную документацию и специальную литературу;
- развитие познавательных способностей и активности студентов: творческой инициативы, самостоятельности, ответственности и организованности;
- формирование самостоятельности мышления, способностей к саморазвитию, самосовершенствованию и самореализации;
- развитие исследовательских умений;
- использование материала, собранного и полученного в ходе самостоятельных занятий на семинарах, на практических и лабораторных занятиях, при написании курсовых и выпускной квалификационной работ, для эффективной подготовки к итоговым зачетам и экзаменам.

2. Порядок выполнения самостоятельной работы студентом

2.1. Методические рекомендации по работе с учебной литературой

При работе с книгой необходимо подобрать литературу, научиться правильно ее читать, вести записи. Для подбора литературы в библиотеке используются алфавитный и систематический каталоги.

Важно помнить, что рациональные навыки работы с книгой - это всегда большая экономия времени и сил.

Правильный подбор учебников рекомендуется преподавателем, читающим лекционный курс. Необходимая литература может быть также указана в методических разработках по данному курсу.

Изучая материал по учебнику, следует переходить к следующему вопросу только после правильного уяснения предыдущего, описывая на бумаге все выкладки и вычисления (в том числе те, которые в учебнике опущены или на лекции даны для самостоятельного вывода).

При изучении любой дисциплины большую и важную роль играет самостоятельная индивидуальная работа.

Особое внимание следует обратить на определение основных понятий курса. Студент должен подробно разбирать примеры, которые поясняют такие определения, и уметь строить аналогичные примеры самостоятельно. Нужно добиваться точного представления о том, что изучаешь.

Полезно составлять опорные конспекты. При изучении материала по учебнику полезно в тетради (на специально отведенных полях) дополнять конспект лекций. Там же следует отмечать вопросы, выделенные студентом для консультации с преподавателем.

Выводы, полученные в результате изучения, рекомендуется в конспекте выделять, чтобы они при перечитывании записей лучше запоминались.

Опыт показывает, что многим студентам помогает составление листа опорных сигналов, содержащего важнейшие и наиболее часто употребляемые формулы и понятия. Такой лист помогает

запомнить формулы, основные положения лекции, а также может служить постоянным справочником для студента.
Чтение научного текста является частью познавательной деятельности. Ее цель – извлечение из текста необходимой информации. От того на сколько осознанна читающим собственная внутренняя установка при обращении к печатному слову (найти нужные сведения, усвоить информацию полностью или частично, критически проанализировать материал и т.п.) во многом зависит эффективность осуществляемого действия.
Выделяют ***четыре основные установки в чтении научного текста***:
информационно-поисковый (задача – найти, выделить искомую информацию)
усваивающая (усилия читателя направлены на то, чтобы как можно полнее осознать и запомнить как сами сведения излагаемые автором, так и всю логику его рассуждений)
аналитико-критическая (читатель стремится критически осмыслить материал, проанализировав его, определив свое отношение к нему)
творческая (создает у читателя готовность в том или ином виде – как отправной пункт для своих рассуждений, как образ для действия по аналогии и т.п. – использовать суждения автора, ход его мыслей, результат наблюдения, разработанную методику, дополнить их, подвергнуть новой проверке).
Основные виды систематизированной записи прочитанного:
Аннотирование – предельно краткое связное описание просмотренной или прочитанной книги (статьи), ее содержания, источников, характера и назначения;
Планирование – краткая логическая организация текста, раскрывающая содержание и структуру изучаемого материала;
Тезирование – лаконичное воспроизведение основных утверждений автора без привлечения фактического материала;
Цитирование – дословное выписывание из текста выдержек, извлечений, наиболее существенно отражающих ту или иную мысль автора;
Конспектирование – краткое и последовательное изложение содержания прочитанного.
Конспект – сложный способ изложения содержания книги или статьи в логической последовательности. Конспект аккумулирует в себе предыдущие виды записи, позволяет всесторонне охватить содержание книги, статьи. Поэтому умение составлять план, тезисы, делать выписки и другие записи определяет и технологию составления конспекта.
Методические рекомендации по составлению конспекта:

1. Внимательно прочитайте текст. Уточните в справочной литературе непонятные слова. При записи не забудьте вынести справочные данные на поля конспекта;
2. Выделите главное, составьте план;
3. Кратко сформулируйте основные положения текста, отметьте аргументацию автора;
4. Законспектируйте материал, четко следуя пунктам плана. При конспектировании старайтесь выразить мысль своими словами. Записи следует вести четко, ясно.
5. Грамотно записывайте цитаты. Цитируя, учитывайте лаконичность, значимость мысли.

В тексте конспекта желательно приводить не только тезисные положения, но и их доказательства. При оформлении конспекта необходимо стремиться к емкости каждого предложения. Мысли автора книги следует излагать кратко, заботясь о стиле и выразительности написанного. Число дополнительных элементов конспекта должно быть логически обоснованным, записи должны распределяться в определенной последовательности, отвечающей логической структуре произведения. Для уточнения и дополнения необходимо оставлять поля.
Овладение навыками конспектирования требует от студента целеустремленности, повседневной самостоятельной работы.

2.2. Методические рекомендации по подготовке к практическим и лабораторным занятиям
Для того чтобы практические и лабораторные занятия приносили максимальную пользу, необходимо помнить, что упражнение и решение задач проводятся по вычитанному на лекциях материалу и связаны, как правило, с детальным разбором отдельных вопросов лекционного курса. Следует подчеркнуть, что только после усвоения лекционного материала с определенной точки

зрения (а именно с той, с которой он излагается на лекциях) он будет закрепляться на практических занятиях как в результате обсуждения и анализа лекционного материала, так и с помощью решения проблемных ситуаций, задач. При этих условиях студент не только хорошо усвоит материал, но и научится применять его на практике, а также получит дополнительный стимул (и это очень важно) для активной проработки лекции.
При самостоятельном решении задач нужно обосновывать каждый этап решения, исходя из теоретических положений курса. Если студент видит несколько путей решения проблемы (задачи), то нужно сравнить их и выбрать самый рациональный. Полезно до начала вычислений составить краткий план решения проблемы (задачи). Решение проблемных задач или примеров следует излагать подробно, вычисления располагать в строгом порядке, отделяя вспомогательные вычисления от основных. Решения при необходимости нужно сопровождать комментариями, схемами, чертежами и рисунками.
Следует помнить, что решение каждой учебной задачи должно доводиться до окончательного логического ответа, которого требует условие, и по возможности с выводом. Полученный ответ следует проверить способами, вытекающими из существа данной задачи. Полезно также (если возможно) решать несколькими способами и сравнить полученные результаты. Решение задач данного типа нужно продолжать до приобретения твердых навыков в их решении.

2.3. Методические рекомендации по самопроверке знаний
После изучения определенной темы по записям в конспекте и учебнику, а также решения достаточного количества соответствующих задач на практических занятиях и самостоятельно студенту рекомендуется, провести самопроверку усвоенных знаний, ответив на контрольные вопросы по изученной теме.
В случае необходимости нужно еще раз внимательно разобраться в материале.
Иногда недостаточность усвоения того или иного вопроса выясняется только при изучении дальнейшего материала. В этом случае надо вернуться назад и повторить плохо усвоенный материал. Важный критерий усвоения теоретического материала - умение решать задачи или пройти тестирование по пройденному материалу. Однако следует помнить, что правильное решение задачи может получиться в результате применения механически заученных формул без понимания сущности теоретических положений.

2.4. Методические рекомендации по написанию научных текстов (докладов, рефератов, эссе, научных статей и т.д.)
Перед тем, как приступить к написанию научного текста, важно разобраться, какова истинная цель вашего научного текста - это поможет вам разумно распределить свои силы и время.
Во-первых, сначала нужно определиться с идеей научного текста, а для этого необходимо научиться либо относиться к разным явлениям и фактам несколько критически (своя идея – как иная точка зрения), либо научиться увлекаться какими-то известными идеями, которые нуждаются в доработке (идея – как оптимистическая позиция и направленность на дальнейшее совершенствование уже известного). Во-вторых, научиться организовывать свое время, ведь, как известно, свободное (от всяких глупостей) время – важнейшее условие настоящего творчества, для него наконец-то появляется время. Иногда именно на организацию такого времени уходит немалая часть сил и талантов.
Писать следует ясно и понятно, стараясь основные положения формулировать четко и недвусмысленно (чтобы и самому понятно было), а также стремясь структурировать свой текст. Каждый раз надо представлять, что ваш текст будет кто-то читать и ему захочется сориентироваться в нем, быстро находить ответы на интересующие вопросы (заодно представьте себя на месте такого человека). Понятно, что работа, написанная «сплошным текстом» (без заголовков, без выделения крупным шрифтом наиболее важным мест и т, п.), у культурного читателя должна вызывать брезгливость и даже жалость к автору (исключения составляют некоторые древние тексты, когда и жанр был иной и к текстам относились иначе, да и самих текстов было гораздо меньше – не то, что в эпоху «информационного взрыва» и соответствующего «информационного мусора»).

Объем текста и различные оформительские требования во многом зависят от принятых в конкретном учебном заведении порядков.
Реферат (доклад) - это самостоятельное исследование студентом определенной проблемы, комплекса взаимосвязанных вопросов.
Реферат не должна составляться из фрагментов статей, монографий, пособий. Кроме простого изложения фактов и цитат, в реферате должно проявляться авторское видение проблемы и ее решения.
Рассмотрим основные этапы подготовки реферата студентом.
Выполнение реферата начинается с выбора темы.
Затем студент приходит на первую консультацию к руководителю, которая предусматривает:
- обсуждение цели и задач работы, основных моментов избранной темы;
- консультирование по вопросам подбора литературы;
- составление предварительного плана.
Следующим этапом является работа с литературой. Необходимая литература подбирается студентом самостоятельно.
После подбора литературы целесообразно сделать рабочий вариант плана работы. В нем нужно выделить основные вопросы темы и параграфы, раскрывающие их содержание.
Составленный список литературы и предварительный вариант плана уточняются, согласуются на очередной консультации с руководителем.
Затем начинается следующий этап работы - изучение литературы. Только внимательно читая и конспектируя литературу, можно разобраться в основных вопросах темы и подготовиться к самостоятельному (авторскому) изложению содержания реферата. Конспектируя первоисточники, необходимо отразить основную идею автора и его позицию по исследуемому вопросу, выявить проблемы и наметить задачи для дальнейшего изучения данных проблем.
Систематизация и анализ изученной литературы по проблеме исследования позволяют студенту написать работу.
Рабочий вариант текста реферата предоставляется руководителю на проверку. На основе рабочего варианта текста руководитель вместе со студентом обсуждает возможности доработки текста, его оформление. После доработки реферат сдается на кафедру для его оценивания руководителем.
Требования к написанию реферата
Написание 1 реферата является обязательным условием выполнения плана СРС по любой дисциплине профессионального цикла.
Тема реферата может быть выбрана студентом из предложенных в рабочей программе или фонде оценочных средств дисциплины, либо определена самостоятельно, исходя из интересов студента (в рамках изучаемой дисциплины). Выбранную тему необходимо согласоваться с преподавателем.
Реферат должен быть написан научным языком.
Объем реферата должен составлять 20-25 стр.
Структура реферата:

- Введение (не более 3-4 страниц). Во введении необходимо обосновать выбор темы, ее актуальность, очертить область исследования, объект исследования, основные цели и задачи исследования.
- Основная часть состоит из 2-3 разделов. В них раскрывается суть исследуемой проблемы, проводится обзор мировой литературы и источников Интернет по предмету исследования, в котором дается характеристика степени разработанности проблемы и авторская аналитическая оценка основных теоретических подходов к ее решению. Изложение материала не должно ограничиваться лишь описательным подходом к раскрытию выбранной темы. Оно также должно содержать собственное видение рассматриваемой проблемы и изложение собственной точки зрения на возможные пути ее решения.
- Заключение (1-2 страницы). В заключении кратко излагаются достигнутые при изучении проблемы цели, перспективы развития исследуемого вопроса

- Список использованной литературы (не меньше 10 источников), в алфавитном порядке, оформленный в соответствии с принятыми правилами. В список использованной литературы рекомендуется включать работы отечественных и зарубежных авторов, в том числе статьи, опубликованные в научных журналах в течение последних 3-х лет и ссылки на ресурсы сети Интернет.
- Приложение (при необходимости).

Требования к оформлению:

- текст с одной стороны листа;
- шрифт Times New Roman;
- кегль шрифта 14;
- межстрочное расстояние 1,5;
- поля: сверху 2,5 см, снизу – 2,5 см, слева - 3 см, справа 1,5 см;
- реферат должен быть представлен в сброшюрованном виде.

Порядок защиты реферата:

Защита реферата проводится на практических занятиях, после окончания работы студента над ним и исправления всех недочетов, выявленных преподавателем в ходе консультаций. На защиту реферата отводится 5-7 минут времени, в ходе которого студент должен показать свободное владение материалом по заявленной теме. При защите реферата приветствуется использование мультимедиа-презентации.

Оценка реферата

Реферат оценивается по следующим критериям:

• соблюдение требований к его оформлению;

• необходимость и достаточность для раскрытия темы приведенной в тексте реферата информации;

• умение студента свободно излагать основные идеи, отраженные в реферате;

• способность студента понять суть задаваемых преподавателем и сокурсниками вопросов и сформулировать точные ответы на них.

Критерии оценки:

Оценка «отлично» выставляется студенту, если в докладе студент исчерпывающе, последовательно, четко и логически стройно излагает материал; свободно справляется с задачами, вопросами и другими видами применения знаний; использует для написания доклада современные научные материалы; анализирует полученную информацию; проявляет самостоятельность при написании доклада.

Оценка «хорошо» выставляется студенту, если качество выполнения доклада достаточно высокое. Студент твердо знает материал, грамотно и по существу излагает его, не допуская существенных неточностей в ответе на вопросы по теме доклада.

Оценка «удовлетворительно» выставляется студенту, если материал доклада излагается частично, но пробелы не носят существенного характера, студент допускает неточности и ошибки при защите доклада, дает недостаточно правильные формулировки, наблюдаются нарушения логической последовательности в изложении материала.

Оценка «неудовлетворительно» выставляется студенту, если он не подготовил доклад или допустил существенные ошибки. Студент неуверенно излагает материал доклада, не отвечает на вопросы преподавателя.