

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Северо-Кавказский федеральный университет»

Колледж СКФУ в г. Ставрополе

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ
для практических занятий**

ОП. 06 Основы алгоритмизации и программирования

Специальность 09.02.11 Разработка и управление программным обеспечением

Форма обучения очная

Ставрополь

Методические указания по выполнению практических работ составлены в соответствии с ФГОС и рабочей программой учебной дисциплины ОП 06 Основы алгоритмизации и программирования для специальности 09.02.11 Разработка и управление программным обеспечением.

Разработчик:

Безпалько Е.Л.-А., преподаватель колледжа СКФУ в г. Ставрополе

Багдасарян Л.Ш., преподаватель колледжа СКФУ в г. Ставрополе

1. Пояснительная записка

Методические указания по организации и проведению практических занятий составлены в соответствии с ФГОС СПО по специальности 09.02.11 Разработка и управление программным обеспечением и рабочей программой учебной дисциплины ОП 06 Основы алгоритмизации и программирования.

Целью выполнения практических занятий является систематизация и закрепление теоретических знаний и формирование практических умений.

Особое значение для усвоения содержания дисциплины и привития практических навыков имеет правильная и четкая организация проведения и выполнения студентами практических работ под контролем преподавателя.

Перед началом выполнения каждой работы студенты должны ознакомиться с ее основными положениями, порядком выполнения работы.

По каждому практическому занятию предусматривается индивидуальный отчет перед преподавателем.

В результате освоения дисциплины обучающийся **должен знать:**

- понятие алгоритмизации, свойства алгоритмов, общие принципы построения алгоритмов, основные алгоритмические конструкции;
- эволюцию языков программирования, их классификацию, понятие системы программирования;
- основные элементы языка, структуру программы, операторы и операции, управляющие структуры, структуры данных, файлы, классы памяти;
- подпрограммы, составление библиотек подпрограмм;
- объектно-ориентированную модель программирования, основные принципы объектно-ориентированного программирования на примере алгоритмического языка: понятие классов и объектов, их свойств и методов, инкапсуляция и полиморфизма, наследования и переопределения

В результате освоения дисциплины обучающийся **должен уметь:**

- разрабатывать алгоритмы для конкретных задач;
- использовать программы для графического отображения алгоритмов;
- определять сложность работы алгоритмов;
- работать в среде программирования;
- реализовывать построенные алгоритмы в виде программ на конкретном языке программирования;
- оформлять код программы в соответствии со стандартом кодирования;
- выполнять проверку, отладку кода программы.

2. Перечень практических занятий

Наименование практического занятия	Количество часов
Раздел 1. Введение в программирование	
Практическое занятие №1. Составление блок-схем алгоритмов.	2
Раздел 2. Основные принципы алгоритмизации и программирования	
Практическое занятие №2. Знакомство с Паскаль.	1
Практическое занятие № 3. Составление программ линейной структуры.	1
Практическое занятие №4. Составление программ разветвляющейся, циклической и усложненной структуры.	1
Практическое занятие №5. Обработка одномерных и двумерных массивов.	1
Практическое занятие №6. Работа со строковыми переменными.	1
Практическое занятие №7. Работа с данными типа множество.	1
Раздел 3. Процедуры и функции	
Практическое занятие № 8. Организация и использование процедур.	1
Практическое занятие №9. Организация и использование функций.	1
Практическое занятие №10. Использование стандартных функций и процедур для работы со строками и массивами.	2
Практическое занятие №11. Работа с файлом последовательного доступа.	2
Практическое занятие №12. Работа с файлом произвольного доступа.	2
Практическое занятие №13. Использование стандартных процедур и функций для работы с файлами.	2
Практическое занятие №14. Программирование модуля.	2
Практическое занятие №15. Создание библиотеки подпрограмм.	2
Раздел 4. Основные конструкции языков программирования	
Практическое занятие №16. Использование библиотеки подпрограмм	2
Раздел 5. Объектно-ориентированное программирование	
Практическое занятие №17. Классы объектов. Компоненты и их свойства.	2
Практическое занятие №18. Событийно-управляемая модель программирования. Компонентно-ориентированный подход.	2
Практическое занятие №19. Изучение интегрированной среды разработчика.	1
Практическое занятие №20. Создание простого проекта.	1
Практическое занятие №21. Создание проекта с использованием кнопочных компонентов и компонентов для работы с текстом.	2
Практическое занятие №22. Создание проекта с использованием компонентов ввода и отображения чисел, дат и времени, и компонентов стандартных диалогов и системы меню.	2
Практическое занятие №23. Разработка оконного приложения.	2
Практическое занятие №24. Разработка оконного приложения с несколькими формами.	2
Практическое занятие №25 Изучение этапов разработки.	2
Практическое занятие №26. Объявление класса, создание экземпляров класса.	2
Практическое занятие №27. Создание наследованного класса.	2
Практическое занятие №28 Перегрузка методов.	2
Итого:	48

Раздел 1. Введение в программирование

Тема 1.1. Языки программирования

Практическое занятие №1. Составление блок-схем алгоритмов.

Цель занятия: Формирование представлений учащихся о линейных алгоритмах, навыков записи алгоритмов с помощью блок-схем и умений устанавливать соответствие между командами алгоритма, записанного словесно, и элементами блок-схемы.

Задачи: научиться разрабатывать линейные алгоритмы. Изучить интерфейс MS VISIO 2007. Изучить основные инструменты создания и оформления блок-схем.

Организационная форма занятия: практическое занятие в форме практической подготовки

Краткие теоретические сведения

Линейные алгоритмы - это такие алгоритмы, в которых действия совершаются одно за другим, в строго определенном порядке

Рассмотрим пример составления линейного вычислительного алгоритма.

Пример 1: Вычислить площадь прямоугольника по заданной длине и ширине:

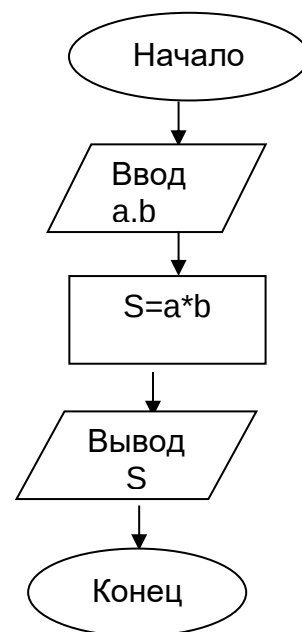
1. Ввести a, b
2. Вычислить площадь S по формуле $a*b$
3. Вывести полученный результат на экран.
4. Закончить выполнение алгоритма.

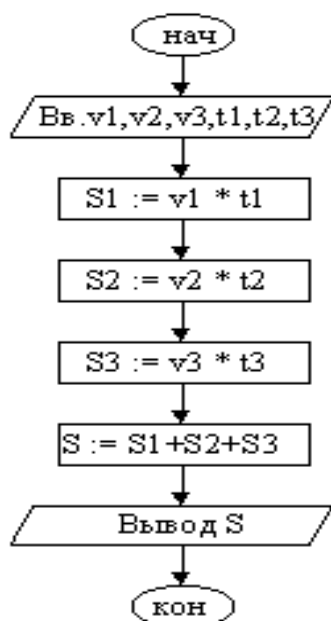
Это словесно – формульная форма записи алгоритма.

Посмотрим, как выглядит данная задача, записанная с помощью графических блоков.

Вычислительных действий в блок-схеме может быть и несколько, в зависимости от условия задачи и от оформления.

Пример 2. Пешеход шел по пересеченной местности. Его скорость движения по равнине v_1 км/ч, в гору — v_2 км/ч и под гору — v_3 км/ч. Время движения соответственно t_1, t_2 и t_3 ч. Какой полный путь прошел пешеход?





1. Ввести $v_1, v_2, v_3, t_1, t_2, t_3$.
2. $S_1 := v_1 * t_1$.
3. $S_2 := v_2 * t_2$.
4. $S_3 := v_3 * t_3$.
5. $S := S_1 + S_2 + S_3$.
6. Вывести значение S .
7. Конец.

Пример 3. Дана величина A , выражающая объем информации в байтах. Перевести A в более крупные единицы измерения информации (Килобайты – К, Мегабайты – М, Гигабайты – Г). Составьте блок-схему алгоритма решения поставленной задачи.

ИНТЕРФЕЙС ОКНА ПРИЛОЖЕНИЯ Visio

Существует несколько способов запуска этой программы. Обычно при установке программа автоматически добавляет Visio в меню Пуск, в папку Office. Откроем меню Пуск, пункт Программы, Office, и в открывшемся списке щелкнем по пункту MS Office Visio 2007. Другой способ запуска программы - двойной щелчок по ярлыку MS Visio на рабочем столе.

При открытии MS Visio по умолчанию в главном окне два дополнительных. Справа – это окно Область задач и слева – окно Библиотеки шаблонов. С помощью шаблонов создаются всевозможные документы деловой графики – рисунки, схемы и диаграммы. Синяя полоса в верхней части окна называется Строка заголовка. В ней написано имя открытого приложения, под ней – Панель меню.

СОЗДАНИЕ НОВОГО ДОКУМЕНТА С ПОМОЩЬЮ ШАБЛОНОВ

Для того, чтобы начать работу в программе MS Visio нужно либо создать новый документ, либо открыть уже созданный документ и продолжить работу в нем. Документ в MS Visio - это файл, который содержит листы с созданными пользователем изображениями. Рассмотрим сначала какие существуют способы создания нового документа. В меню Файл, выбираем подменю Новый, затем необходимо выбрать необходимый шаблон.

НАСТРОЙКА СТРАНИЦЫ Размер бумаги

Для выбора предложены размеры бумаги, которые поддерживаются текущим принтером, а также стандартные размеры. Если установить флажок Как в принтере на вкладке Размер страницы, изменения размера бумаги будут применяться также к странице документа.

В меню Файл выберите команду Предварительный просмотр, чтобы проверить соответствие размеров страницы документа размерам бумаги для печати.

Ориентация бумаги

Установите один из этих флажков, чтобы задать книжную  или альбомную  ориентацию страницы документа. Если на вкладке Размер страницы установлен флажок Как в принтере, изменения ориентации бумаги будут также применены к странице документа.

В меню Файл выберите команду Предварительный просмотр, чтобы проверить соответствие размеров страницы документа размерам бумаги для печати.

Настройка

При нажатии этой кнопки открывается диалоговое окно Настройка печати, в котором можно выбрать такие параметры печати, как поля, центрирование, принтер и источник бумаги.

Сетка

Установите этот флажок, чтобы вывести на печать линии сетки, отображаемые в окне документа.

Просмотр

В этой части окна отображается эскиз страницы, позволяющий проверить соотношение размеров текущей страницы документа и бумаги для печати.

Можно задать одинаковые размеры страницы документа и бумаги для печати. Для этого на вкладке Размер страницы установите флажок Как в принтере.

ДОБАВЛЕНИЕ НОВОЙ СТРАНИЦЫ

1. В левом нижнем углу окна документа щелкните правой кнопкой мыши вкладку страницы , а затем выберите в контекстном меню команду Вставить страницу.

2. Перейдите на вкладку Свойства страницы и введите имя для страницы или воспользуйтесь именем по умолчанию.

3. Для изменения масштаба или размера новой страницы перейдите на вкладку Масштаб документа или Размер страницы.

СОЗДАНИЕ ПРОСТОЙ БЛОК-СХЕМЫ

С помощью блок-схем можно документировать процедуры, анализировать процессы, обозначать рабочий или информационный процессы, затраты на отслеживание, эффективность и т. д.

1. В меню Файл последовательно выберите команды Создать, Бизнес или Блоксхема, а затем — команду Простая блок-схема.

2. Для каждого шага документируемого процесса перетащите в документ фигуру блок-схемы.

3. Соедините фигуры блок-схемы.

a. Перетащите фигуру из набора элементов на страницу документа и расположите ее вблизи другой фигуры.

b. Не отпуская кнопку мыши, переместите указатель на один из светло-синих треугольных маркеров. Цвет маркера изменится на темно-синий.

c. Отпустите кнопку мыши. Фигура будет размещена на странице, и к обеим фигурам будет добавлена и приклеена соединительная линия.

4. Для добавления текста в фигуру выделите ее, а затем введите текст. Закончив ввод, щелкните за пределами текстового блока.

5. Для отображения последовательности шагов процесса фигуры в блок-схеме можно пронумеровать.

a. В блок-схеме выделите фигуры, которым нужно присвоить номера.

В меню Сервис последовательно выберите команды Надстройки, Дополнительные решения Visio, а затем — команду Нумерация фигур.

c. На вкладке Общие в группе Операция установите переключатель в положение Автономерация.

d. В группе Применить к установите переключатель в положение Выбранные фигуры, а затем нажмите кнопку ОК Совет. Для автоматической нумерации новых фигур блок-схемы по мере их перетаскивания на страницу, в диалоговом окне Нумерация фигур установите флажок Продолжать нумерацию фигур при перетаскивании на страницу

Задания для практического занятия:

1. Рассмотреть примеры 1-3.

2. Разработать словесно -формульный алгоритм по индивидуальному заданию.
3. Разработать блок-схему алгоритма решения задачи средствами MS Visio.
4. Оформить отчет в MS Word.

Индивидуальные задания

Вариант 1.

1. Дана сторона квадрата. Найти его периметр.
2. Дан радиус окружности. Найти длину окружности и площадь круга.
3. Известны количество жителей в государстве и площадь его территории. Определить плотность населения в этом государстве.
4. Даны катеты прямоугольного треугольника. Найти его периметр.
5. Треугольник задан координатами своих вершин. Найти периметр и площадь треугольника.
6. Дано двузначное число. Найти:
 - а) число десятков в нем;
 - б) число единиц в нем;
 - в) сумму его цифр;
 - г) произведение его цифр.

Вариант 2

1. Дан радиус окружности. Найти ее диаметр.
2. Даны два целых числа. Найти их среднее арифметическое.
3. Даны катеты прямоугольного треугольника. Найти его гипотенузу.
4. Даны два числа. Найти их сумму, разность, произведение, а также частное от деления первого числа на второе.
5. Даны длины сторон прямоугольного параллелепипеда. Найти его объем и площадь боковой поверхности.
6. Дано трехзначное число. Найти:
 - а) число единиц в нем;
 - б) число десятков в нем;
 - в) сумму его цифр;
 - г) произведение его цифр.

Вариант 3

1. Дана длина ребра куба. Найти объем куба и площадь его боковой поверхности.
2. Даны два целых числа. Найти их среднее геометрическое.
3. Найти площадь кольца по заданным внешнему и внутреннему радиусам.
4. Даны два числа. Найти среднее арифметическое и среднее геометрическое их модулей.
5. Даны длины сторон прямоугольника. Найти его периметр и длину диагонали.
6. Дано двузначное число. Получить число, образованное при перестановке цифр заданного числа.

Методика анализа результатов, полученных в ходе практического занятия

1. В результате выполнения заданий у Вас должны быть алгоритмы описанные словесно-формульным способом и в виде блок-схемы, разработанной в MS Visio по индивидуальным заданиям.

.

Порядок выполнения отчета по практическому занятию

1. Разработать программу по индивидуальному заданию.
2. Предоставить преподавателю отчет в MS Word.

Составление словесных алгоритмов.

Краткие теоретические сведения

Команда ветвления - это составная команда алгоритма, в которой в зависимости от условия P выполняется или одно $S1$, или другое $S2$ действие. Из команд следования и команд ветвления составляются разветвляющиеся алгоритмы (алгоритмы ветвления). Примером разветвляющегося алгоритма будет нахождение большего из двух чисел, введенных с клавиатуры.

Команда ветвления может быть полной и неполной формы.

Неполная форма команды ветвления используется тогда, когда необходимо выполнять действие S только в случае соблюдения условия P . Если условие P не соблюдается, то команда ветвления завершает свою работу без выполнения действия. Примером команды ветвления неполной формы будет уменьшение в два раза только четного числа.

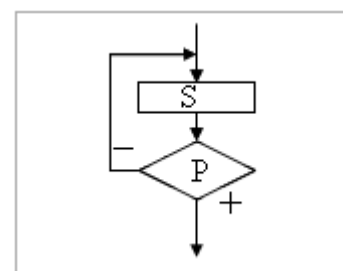
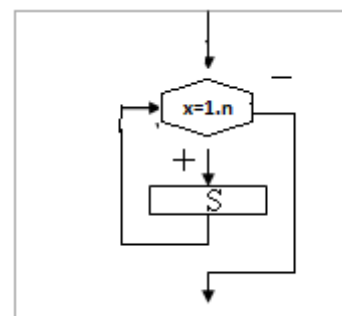
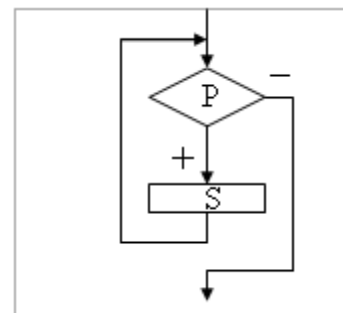
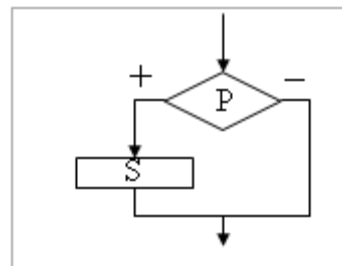
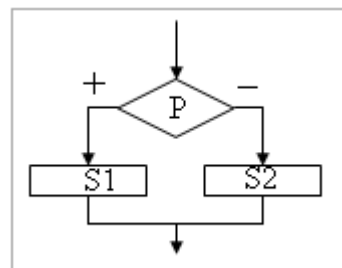
Команда повторения - это составная команда алгоритма, в которой в зависимости от условия P возможно многократное выполнение действия S . Из команд следования и команд повторения составляются циклические алгоритмы (алгоритмы повторения). На рисунке представлена команда повторения с предусловием. Называется она так потому, что вначале проверяется условие, а уже затем выполняется действие. Причем действие выполняется, пока условие соблюдается. Пример циклического алгоритма может быть следующий. Пока с клавиатуры вводятся положительные числа, алгоритм выполняет нахождение их суммы.

Команда повторения с предусловием не является единственно возможной. Разновидностью команды повторения с предусловием является **команда повторения с параметром**. Она используется тогда, когда известно количество повторений действия. В блок-схеме команды повторения с параметром условие записывается не в ромбе, а в шестиугольнике. Примером циклического алгоритма с параметром будет нахождение суммы первых 20 натуральных чисел.

В команде повторения с постусловием вначале выполняется действие S и лишь затем, проверяется условие P . Причем действие повторяется до тех пор, пока условие не соблюдается. Примером команды повторения с постусловием будет уменьшение положительного числа до тех пор, пока оно неотрицательное. Как только число становится отрицательным, команда повторения заканчивает свою работу.

С помощью соединения только этих элементарных конструкций (последовательно или вложением) можно "собрать" алгоритм любой степени сложности.

Пример 1. Определить среднее арифметическое двух чисел, если a положительное и частное (a/b) в противном случае.



Запись алгоритма в виде блок-схемы

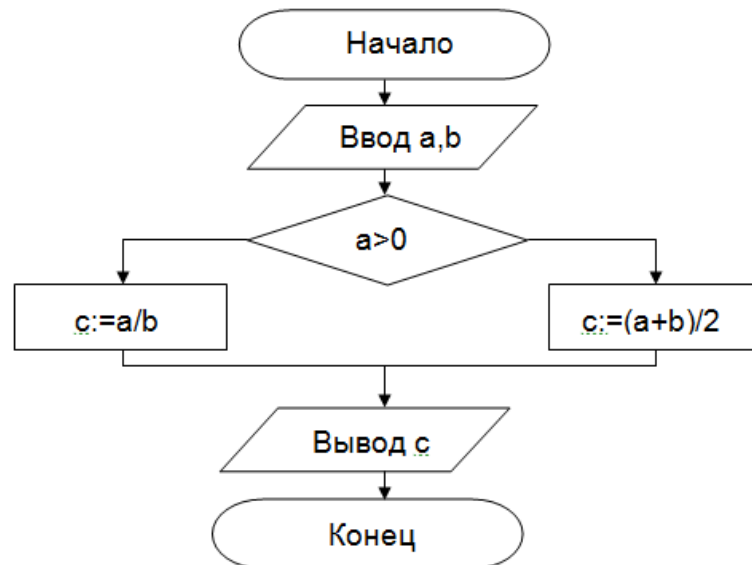
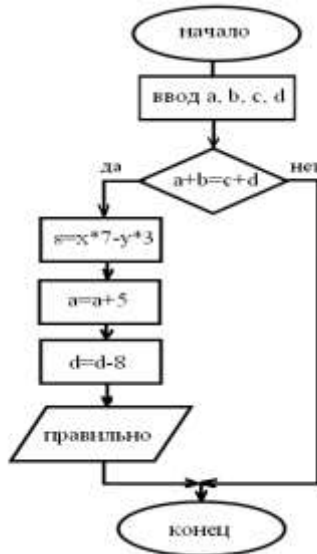


Рисунок . Блок-схема алгоритма с ветвлением

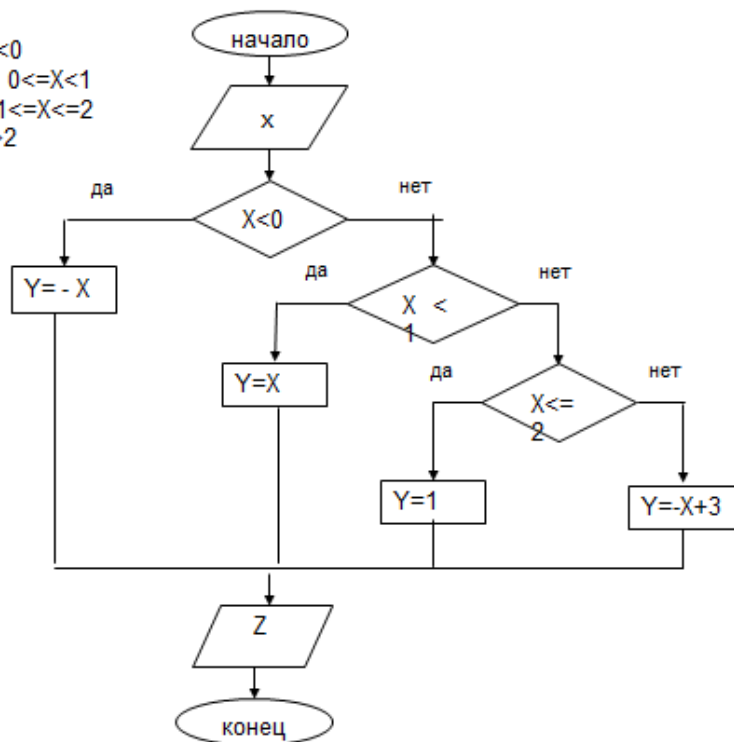
Пример 2. Составить блок-схему решения задачи:

Если $a+b=c+d$ то сделать следующее: напечатать сообщение "Правильно", вычислить $s=x*7-y*3$, сложить a с пятью, вычесть из d восемь.



Пример 3. По заданной функции нарисовать блок-схему.

$$Y = \begin{cases} -X, & \text{если } X < 0 \\ X, & \text{если } 0 \leq X < 1 \\ 1, & \text{если } 1 \leq X \leq 2 \\ -X + 3, & \text{если } X > 2 \end{cases}$$



Пример 4. Составить алгоритм нахождения суммы целых чисел в диапазоне от 1 до 10. Запись алгоритма в виде блок-схемы:

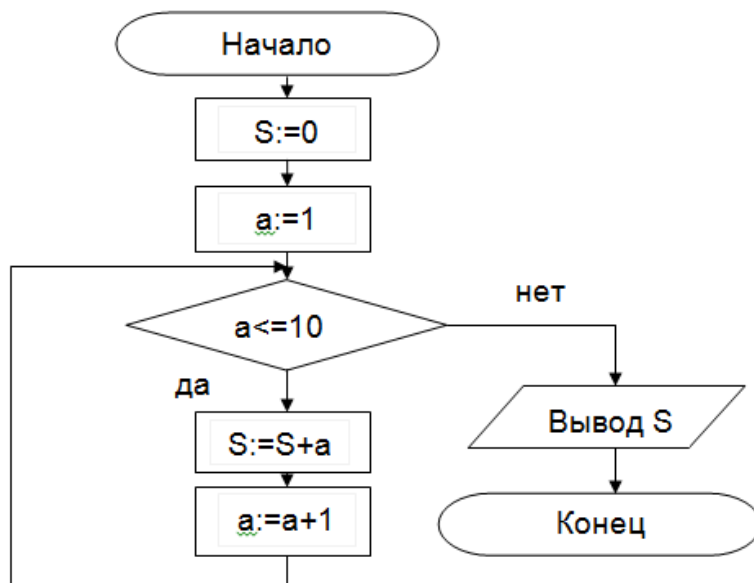


Рисунок - Циклический алгоритм с предусловием

В алгоритме с постусловием сначала выполняется тело цикла, а затем проверяется условие окончания цикла. Решение задачи нахождения суммы первых десяти целых чисел в данном случае будет выглядеть следующим образом:
Запись алгоритма в виде блок-схемы:

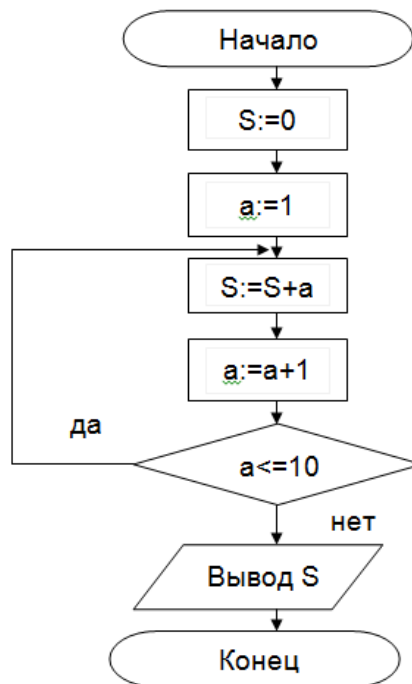


Рисунок 3 - Циклический алгоритм с постусловием

Вопросы для закрепления теоретического материала к практическому занятию:

1. Что такое алгоритм?
2. Свойства алгоритма.
3. Способы записи алгоритма.
4. Основные элементы блок-схемы.
5. Виды алгоритмов.
6. Отличительные особенности алгоритмов с предусловием и постусловием.

Задания для практического занятия:

1. Рассмотреть примеры.
2. Составить алгоритм решения задачи в виде блок-схемы по индивидуальному заданию
3. Оформить отчет

Индивидуальные задания

Вариант 1.

1. Задано число x . Вычислить функцию s .
 $s = -1$, если $x < 0$
 $s = 0$, если $x = 0$
 $s = 1$, если $x > 0$
1. Составить алгоритм нахождения суммы S и произведения p четных чисел от 1 до n .

Вариант 2.

1. Составить блок-схему решения задачи. Заданы два числа k и m .
 Переменной n присвоить значение 1, если $k < m$, 0, если $k = m$, и -1, если $k > m$.
2. Составить алгоритм нахождения суммы S и произведения p нечетных чисел от 1 до n .

Вариант 3.

1. Составить блок-схему вычисления функции:

$$y = \begin{cases} X, & \text{если } X < 2 \\ 2, & \text{если } 2 \leq X \leq 3 \\ -X + 5, & \text{если } X > 3 \end{cases}$$

2. Составить алгоритм нахождения суммы S и произведения p чисел, кратных

3 от 1 до n .

Методика анализа результатов, полученных в ходе практического занятия

2. В результате выполнения заданий у Вас должны быть разработаны алгоритмы средствами MS Visio.

Порядок выполнения отчета по практическому занятию

1. Разработать блок-схемы по индивидуальному заданию.
2. Предоставить преподавателю отчет.

Содержание отчета

1. Цель работы и задание.
2. Условие задачи.
3. Алгоритм, написанный с помощью блок-схемы.

Пример отчета

Составление словесных алгоритмов

Учебная цель: сформировать умение разрабатывать разветвляющиеся и циклические алгоритмы

Индивидуальное задание

ЗАДАЧА. Разработать алгоритм вычисления наибольшего числа из двух чисел x и y .

Этап 1. Математическое описание решения задачи.

Из курса математики известно, если $x > y$, то наибольшее число x , если $x < y$, то наибольшее число y , если $x = y$, то число x равно числу y .

Этап 2. Определение входных и выходных данных.

Входными данными являются значения чисел x и y . Выходным данными являются:

- наибольшее число
- любое из чисел, если числа равны

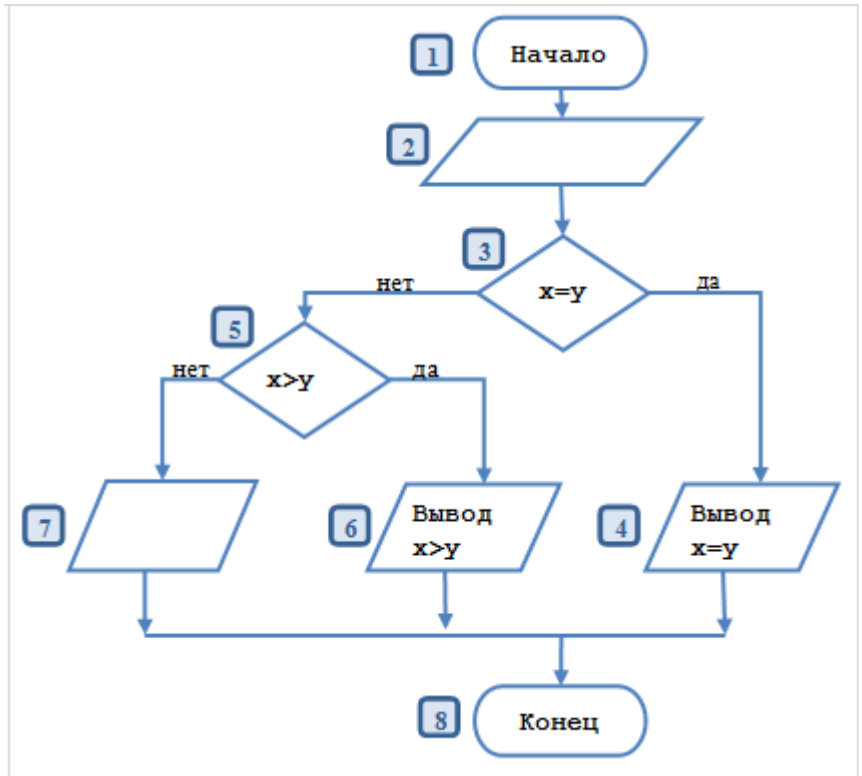
Для решения задачи нам необходимо знать значения x и y .

Этап 3. Разработка алгоритма решения задачи.

Словесное описание алгоритма

1. Начало алгоритма.
2. Ввод значений x и y .
3. Сравниваем x и y .
Если $x = y$, то переход к шагу 4, иначе к шагу 5.
4. Вывод информации: числа x и y равны. Переход к шагу 8.
5. Сравниваем x и y .
Если $x > y$, то переход к шагу 6, иначе к шагу 7.
6. Вывод информации: число x больше y . Переход к шагу 8.
7. Вывод информации: число y больше x . Переход к шагу 8.
8. Конец алгоритма.

Запись алгоритма на языке блок-схем



ЗАДАЧА. Разработать алгоритм вычисления суммы натуральных чисел от 1 до 100.

Этап 1. Математическое описание решения задачи.

Обозначим сумму натуральных чисел через S . Тогда формула вычисления суммы натуральных чисел от 1 до 100 может быть записана так:

$$S = 1 + 2 + 3 + \dots + 97 + 98 + 99 + 100 = \sum_{i=1}^n X_i$$

где X_i – натуральное число X с номером i , который изменяется от 1 до n , $n=100$ – количество натуральных чисел.

Этап 2. Определение входных и выходных данных.

Входными данными являются натуральные числа: 1, 2, 3, 4, 5, ..., 98, 99, 100.

Выходные данные – значение суммы членов последовательности натуральных чисел.

Параметр цикла – величина, определяющая количество повторений цикла. В нашем случае i – номер натурального числа.

Подготовка цикла заключается в задании начального и конечного значений параметра цикла.

- начальное значение параметра цикла равно 1,
- конечное значение параметра цикла равно n ,
- шаг цикла равен 1.

Для корректного суммирования необходимо предварительно задать начальное значение суммы, равное 0.

Тело цикла. В теле цикла будет выполняться накопление значения суммы чисел, а также вычисляться следующее значение параметра цикла по формулам:

$$S = S + i; \quad I = I + 1;$$

Условие продолжения цикла: цикл должен повторяться до тех пор, пока не будет добавлен последний член последовательности натуральных чисел, т.е. пока параметр цикла будет меньше или равен конечному значению параметра цикла.

Этап 3. Разработка алгоритма решения задачи.

Введем обозначения: S – сумма последовательности, i – значение натурального числа.

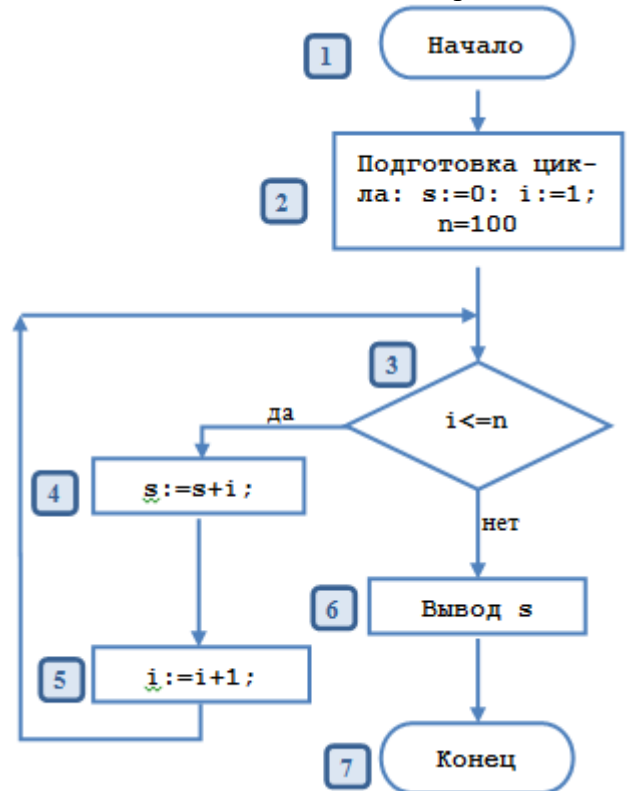
Начальное значение цикла $i=1$, конечное значение цикла $i=100$, шаг цикла 1.

Словесное описание алгоритма

1. Начало алгоритма.
2. Подготовка цикла: $S:=0$; $i:=1$; $n=100$;
3. Проверка условия. Если $i \leq n$, то перейти к шагу 4, иначе к шагу 6.
4. Накопление суммы: $S:=S+i$;
5. Вычисление следующего значения параметра цикла: $i:=i+1$;
6. Вывод информации: сумма натуральных чисел – S .
7. Конец алгоритма.

Запись алгоритма на языке блок-схем

В схеме алгоритма решения задачи цифрами указаны номера элементов алгоритма. Номера элементов соответствуют номерам шагов словесного описания алгоритма.



Раздел 2. Основные принципы алгоритмизации и программирования

Тема 2.1. Операторы языка программирования

Практическое занятие №2. Знакомство с Паскаль.

Учебная цель: сформировать умение разрабатывать простейшие программы в среде программирования ABC Pascal

Организационная форма занятия: практическое занятие в форме практической подготовки

Ход практического занятия:

1. Повторить теоретический материал по теме практической работы.
2. Ответить на вопросы для закрепления теоретического материала.
3. Разработать простейшие программы.
4. Оформить отчет.

Краткие теоретические сведения

Язык Паскаль назван в честь французского математика и философа Блеза Паскаля (1623-1662) и разработан в 1968-1971 гг. Никлаусом Виртом.

Система программирования Pascal ABC достаточно простая и дружелюбная среда программирования. Во время работы используются три основных **окна**: окно редактора кода, окно ввода и окно вывода.

Окно редактора кода

Большую часть рабочей области, её верхнюю часть занимает окно редактора кода. В него вводится исходный текст программы.

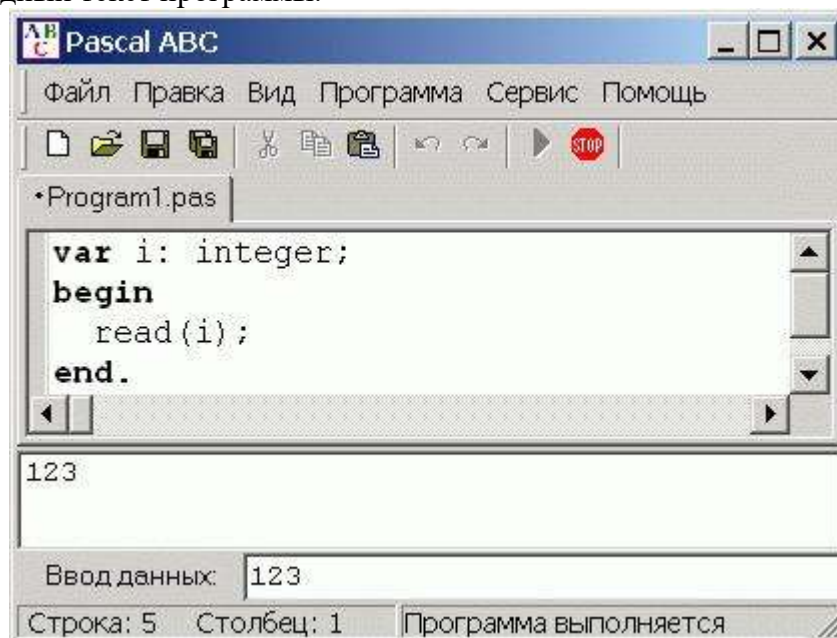


Рисунок 1 – Окно системы программирования Pascal ABC

Горячие клавиши, которые можно использовать при работе с текстом программы:

F2, Ctrl-S- сохранить файл.

F3, Ctrl-O- загрузить файл.

F12- сохранить файл под новым именем.

Ctrl-Shift-S- сохранить все открытые файлы.

Ctrl-Tab, Ctrl-Shift-Tab - перейти к следующему/предыдущему окну редактора.

Ctrl-Shift-I- увеличить отступ выделенного блока.

Ctrl-Shift-U - уменьшить отступ выделенного блока.

Окно вывода

Под окном редактора расположено окно вывода. Оно предназначено для вывода данных процедурами `write` и `writeln`, а также для вывода сообщений об ошибках и предупреждений во время работы программы.

Окно вывода может быть скрыто. Клавиша F5 показывают/скрывают окно вывода. Для скрытия окна вывода используется также клавиша Esc.

Окно вывода обязательно открывается при любом выводе в него.

Для очистки окна вывода следует нажать комбинацию клавиш Ctrl-Del или кнопку .

Окно ввода

Окно ввода открывается при выполнении процедур `read` и `readln` ходе работы программы: Ввод данных в в окно ввода сопровождается эхо-выводом в окно вывода (см. рис). После нажатия клавиши Enter данные из окна ввода попадают в соответствующие переменные, окно ввода закрывается, и программа продолжает работать дальше.

Запуск и остановка программы

Для запуска программы в текущем окне редактора следует нажать клавишу F9 или кнопку  панели инструментов.

Программа вначале компилируется во внутреннее представление, после чего, если не найдены ошибки, программа начинает выполняться. При выполнении программы кнопка запуска программы становится неактивной, кнопка останова программы, наоборот, активной и в строке статуса отображается информация "Программа выполняется".

Выполнение программы можно в любой момент прервать нажатием комбинации клавиш Ctrl-F2 или кнопки . При этом в окне вывода появится сообщение: • Программа прервана пользователем.

Любая программа на языке Паскаль имеет начало (`begin`) и конец (`end`). Это обозначается блоком:

Begin

Код программы

End.

Код программы представляет собой набор последовательных шагов, которые приводят к выполнению поставленной задачи, и может достигать несколько тысяч строк. В нашем случае – это вывод на экран простого сообщения.

Программа будет иметь вид (рис. 2.):

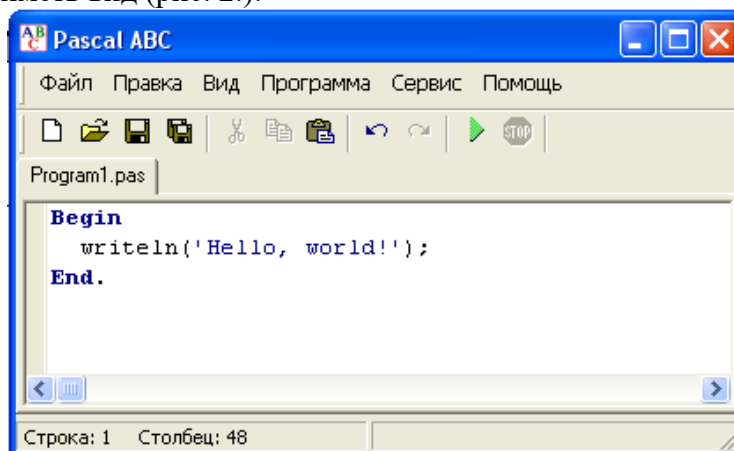


Рисунок 2 - Программа для вывода на экран сообщения "Hello, world!".

Для вывода на экран информации можно использовать команды `write` или `writeln`. Во втором

случае при выводе сообщения на экран курсор переходит на другую строчку (ln - от слова line-линия). Внутри круглых скобок необходимо указать текстовое сообщение в одинарных кавычках. В конце команды ставится **точка с запятой**. Как и в любом предложении в конце программы должна стоять **точка**.

Вопросы для закрепления теоретического материала к практическому занятию:

1. Назначение языка программирования Pascal.
2. Что такое [среда программирования](#)?
3. Состав среды программирования.
4. Алфавит языка программирования Pascal.
5. Что такое идентификатор? Ограничения при создании идентификатора.
6. Из каких разделов может состоять программа на языке Pascal?
7. Какие из разделов программы на языке Pascal необходимы?
8. Для чего служит раздел описания переменных, раздел операторов?
9. Как записываются операторы вывода на экран в Pascal?
- 10.
11. Что такое константа. Примеры констант в Pascal
12. Поясните: имя и значение переменной.
13. Операции языка Pascal. Приоритет операций.

Задания для практического занятия:

1. Выполнить задания 1-5.
2. Разработать программу по индивидуальному заданию

Инструкция по выполнению практической работы

Задание 1. Запуск программы Pascal ABC и сохранение файла.

1. Запустите программу Pascal ABC: Пуск- Все программы – Pascal ABC - Pascal ABC (рис.1.)

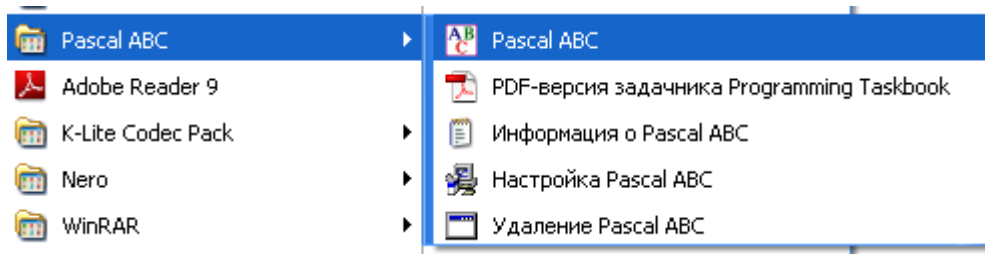


Рисунок 3 - Запуск программы Pascal ABC

2. Откроется окно (рис.4.), в которое необходимо вводить программный код. По умолчанию название программы Program1.pas.

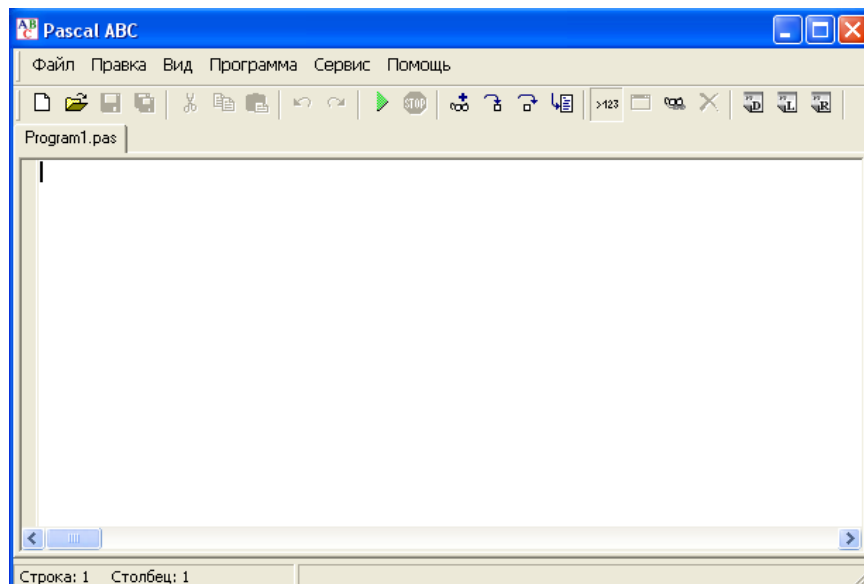


Рисунок 4 - Окно для ввода программного кода.

3. Сохраните программу под именем First.pas (первая программа). Для этого откройте Файл – Сохранить как... и в появившемся окне укажите имя файла. Например, можно сохранить так D:\ОАИП\группа\Петров\First.pas.

Задание 2. Написать первую простейшую программу “Hello, world!” (Здравствуй, мир!). Программа должна вывести на экран сообщение “Hello, world!”.

Введите код программы, как показано на рис.2.

2. Запустите программу на выполнение. Для этого нажмите в меню Программа – Выполнить или F9 или зеленую стрелочку.

3. Если ошибок не обнаружено, то появится результат в нижней части экрана (рис. 5.):

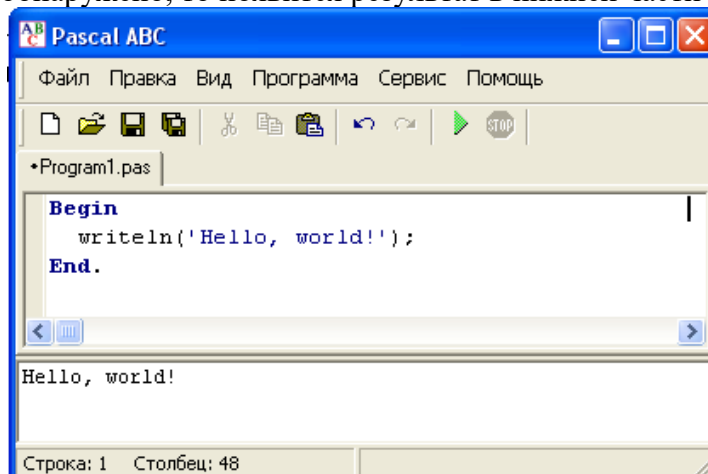


Рисунок 5 - Вывод сообщения “Hello, world!” на экран.

Несколько сообщений можно выводить через запятую, например:

writeln('Я в колледже',' изучаю',' Паскаль.');

Тот же результат будет, если написать:

writeln('Я в колледже изучаю Паскаль.');

Для перехода на другую строчку можно просто написать writeln;

Задание 3. Напишите программу, которая выводит строчку: ****!!!!****.

Задание 4. Напишите программу, которая рисует картинку:

```

  ~
(o o)
// V \\
/( _ )\

```

^^ ^^

Задание 5. Напишите программу, которая рисует картинку:

```
(  
 \  
 )  
##----->  
 )  
 /  
 (  

```

Индивидуальное задание.

1. Напишите программу, которая выводит любую картинку.

Методика анализа результатов, полученных в ходе практической работы

1. В результате выполнения заданий у Вас должны быть работающие программы, позволяющие выводить данные на экран.
2. В результате выполнения индивидуальных заданий у Вас должно быть работающая программа.

Порядок выполнения отчета по практической работе

1. Разработать программу по индивидуальному заданию.
2. Предоставить преподавателю работающую версию приложения.

Практическое занятие № 3. Составление программ линейной структуры.

Учебная цель: сформировать умение записи на языке Паскаль алгебраических выражений, составления и реализации на ЭВМ простейших линейных программы.

Организационная форма занятия: практическое занятие в форме практической подготовки

Ход практического занятия:

1. Повторить теоретический материал по теме практического занятия.
2. Ответить на вопросы для закрепления теоретического материала.
3. Разработать блок-схемы.
4. Разработать программу в среде ABC Pascal.
5. Подготовить тестовые задания и произвести отладку программы.
6. Ответить на контрольные вопросы.
7. Результаты выполнения практической работы оформить в виде отчета.

Краткие теоретические сведения

Предполагаем, что A и B действительные числа. Вычислить $y = (2 + \cos b) / \cos(a^2 + b)$

Программа линейной структуры на языке Паскаль:

```
{Практическая работа № }  
{Студент гр. .... Фамилия ...}  
PROGRAM prim1;  
  VAR  a, b, y: real;  
  BEGIN  
    Write ('Введите два вещ. числа');  
    Readln (a, b);  
    Y:=(2+cos(b))/cos(sqr(a)+b);  
    Writeln;  
    Writeln ('Значение переменной y= ', y);  
  END.
```

Выделенные заглавными буквами слова являются служебными и их нельзя использовать в качестве имен переменных.

Любая программа начинается со слова «PROGRAM», за которым указывается выбранное программистом имя программы, и заканчивается словом «END».

Все операторы кроме «END» отделяются друг от друга символом « ; ».

За заголовком программы следует раздел описания переменных «VAR». Необходимо описывать все переменные, которые будут использоваться в программе.

Между «BEGIN» и «END» находится последовательность операторов, которые описывают все алгоритмические действия, которые необходимо сделать над исходными данными для получения результата.

Исходные данные при вводе должны отделяться друг от друга хотя бы одним пробелом.

Вопросы для закрепления теоретического материала к практическому занятию:

- 1) Какие типы величин используются в языке программирования Паскаль?
- 2) Указать диапазон значений величин целого и действительного типов.
- 3) Какие имена переменных допустимы в программе?
- 4) Как задать тип переменных в программе?
- 5) Назвать последовательность действий при выполнении оператора присваивания.
- 6) Указать старшинство выполнения операций при вычислении арифметического выражения.

7) Указать средства, имеющиеся в языке программирования Паскаль для управления размещением данных на строке. Как организовать вывод значений, сопровождая выводимое числовое значение именем переменной? Как организовать пропуск одной, двух строк при выводе?

Задания для практического занятия:

1. Выполните программу, описанную в примере.
2. В соответствии с вариантом написать и выполнить программы.
 - 1.1 При выполнении заданий по необходимости нужно формализовать постановку задачи, т. е. построить математическую модель.
 - 1.2 Составить список используемых переменных с указанием их типов.
 - 1.3 Составить блок-схему.
 - 1.4 Составить программу, отредактировать и отладить ее, проанализировать полученные результаты. Если в условии задачи не указаны конкретные данные, то задать их самостоятельно, однако программа должна работать при произвольных в допустимых пределах данных.
2. Ответить на контрольные вопросы.
3. Оформить отчет по практической работе, содержащий тему и цель практического занятия, задания по вариантам, контрольные вопросы с письменными ответами (где это нужно). Отчет по каждому практическому заданию должен содержать:
 - 3.1 Постановку задачи;
 - 3.2 Математическую модель;
 - 3.3 Алгоритм решения задачи (блок – схему);
 - 3.4 Листинг программы;
 - 3.5 Протокол работы программы.

Инструкция по выполнению практической работы

Задание 1.

№ варианта	Функция	Тип данных
1.	$y = \sqrt{x^2 + a^2} \arctg\left(\frac{x}{a}\right) - \ln b$	Действительный
2.	$z = \ln a + \frac{4}{tg\left(\frac{x}{a}\right)}$	Целый
3.	$k = e^{\frac{ax}{\sqrt{2}}} \cos \sqrt{\frac{bx}{2}}$	Действительный
4.	$d = e^{3 \cdot \sin bx } - \frac{x^3}{a}$	Целый
5.	$t = (e^{ax} - e^{-ax}) / 2 * \sin(ax + b)$	Действительный
6.	$r = \frac{x^5}{4 \ln a + \sqrt{x^2 - b^2} }$	Целый
7.	$w = \frac{\sin(x^3 - a) + b x }{\cos(ax + b)}$	Действительный
8.	$k = \frac{a+b}{2 * a - b} a^2 \sin x$	Целый
9.	$l = \frac{a \cos^3 bx}{\sin a + bx \cos b - e^a}$	Действительный
10.	$m = \sin^2 a + \arctg \frac{x-a}{x}$	Целый

11.	$f = \operatorname{tg} x + \frac{ax}{b/a - b}$	Действительный
12.	$p = e^x - (a + b + \sin a) \cos b$	Целый

Задание 2. Вычисления в математических задачах.

1. Пересчитать значение температуры из шкалы Цельсия в шкалы Фаренгейта, Кельвина, Ренкина (Фаренгейт=1,8*Цельсий+32, Кельвин = Цельсий + 273,15; Ренкин = 1,8*Цельсий + 492).
2. Дана величина А, выражающая объем информации в байтах. Перевести А в более крупные единицы измерения информации.
3. Составить программу вычисления объема цилиндра и конуса, которые имеют одинаковую высоту Н и одинаковый радиус основания R.
4. Даны длина, ширина и высота прямоугольного параллелепипеда. Составьте программу вычисления объема, площади полной поверхности прямоугольного параллелепипеда.
5. Задана координаты трех вершин треугольника (x₁,y₁), (x₂,y₂), (x₃, y₃). Найти его периметр и площадь.
6. Найти произведение цифр заданного четырехзначного числа.
7. Даны два числа. Найти среднее арифметическое кубов этих чисел и среднее геометрическое модулей этих чисел.
8. Вычислить корни квадратного уравнения с заданными коэффициентами a,b,c (предполагается, что дискриминант неотрицателен).
9. Найти частное произведений четных и нечетных цифр четырехзначного числа.
10. Найти площадь кольца, внутренний радиус которого равен r, а внешний – R (R>r).
11. Известна длина окружности. Найти площадь круга, ограниченного этой окружностью.
12. Найти расстояние между двумя точками с данными координатами.

Задание 3. Дать письменные ответы:

№ вар.	Задания
1.	1) Записать следующее число без десятичного порядка: -0,00027E+4; 2) Записать на Паскале следующие формулы: а) $[(ax-b)x+c]x-d$; б) $\frac{ab}{c} + \frac{c}{ab}$. 3) Вычислить значение выражения: а) $20 \operatorname{div} 6$; б) $20 \bmod 6$.
2.	1) Записать на Паскале следующие формулы: а) $\frac{x+y}{a1} \cdot \frac{a2}{x-y}$; б) $10^4 \alpha - 3\frac{1}{5} \beta$; 2) Записать в общепринятой форме: $(p+q)/(r+s)-p*q/(r*s)$. 3) Определить операцию div через другие операции и стандартные функции.
3.	1) Записать на Паскале следующие формулы: а) $(1 + \frac{x}{2} + \frac{y}{3}) / (1 + \frac{2}{3+xy})$; б) $(1+x)^2$; 2) Записать операторы присваивания, которые переменной d присваивают: среднее арифметическое чисел x, y, z. 3) Вычислить значение выражения: а) $20 \operatorname{div} 4$; б) $2 \bmod 5$.
4.	1) Записать на Паскале следующие формулы: а) $\sqrt{1+x^2}$; б) $\frac{\beta + \sin^2 \pi^4}{\cos 2 + \operatorname{ctg} \gamma }$; 2) Записать в общепринятой форме: $1E3 + \beta / (x2 - \gamma * \delta)$; 3) Указать порядок выполнения операций в выражении $-a \bmod b + a \operatorname{div} b * c$

5.	1) Записать на Паскале следующие формулы: а) $a + bx$; б) $\frac{xyz - 3.3 x + \sqrt[4]{y} }{10^7 + \sqrt{\lg 4}}$; 2) Записать операторы присваивания, которые переменной d присваивают: расстояние между точками с координатами (x1,y1) и (x2,y2). 3) Вычислить значение выражения: а) $20 \bmod 4$; б) $2 \operatorname{div} 5$.
6.	1) Записать следующее число без десятичного порядка: 666E-3; 2) Записать на Паскале следующие формулы: а) $\cos^2 x^3$; б) 2^{1+x};
7.	1) Записать на Паскале следующие формулы: а) $\sqrt[8]{x^8 + 8^x}$; б) $2^{-x} - \cos x + \sin(2xy)$ 2) Записать в общепринятой форме: $(-b + \sqrt{\operatorname{sqr}(b) - 4*a*c}) / (2*a)$. 3) Поменять местами значение переменных: x, y, и z так, чтобы в x оказалось значение переменной y, в y – значение переменной z, а в z – прежнее значение переменной x.
8.	1) Записать на Паскале следующие формулы: а) $\cos^2(\sin \frac{1}{z})$; б) $(1 + x)^2 / n$; 2) Записать операторы присваивания, которые переменной d присваивают: первую цифру из дробной части положительного вещественного числа x. 3) Вычислить значение выражения: а) $123 \operatorname{div} 0$; б) $3.0 \bmod 3$.
9.	1) Записать на Паскале следующие формулы: а) $\sqrt[3]{1 + x}$. б) $\frac{\ln \cos x }{\ln(1 + x^2)}$ 2) Записать в общепринятой форме: $x1 + \arctan(y2 - \alpha) / 2 * \operatorname{abs}(x4 - \ln(5) * y5) / \exp(-1)$. 3) Вычислить значение выражения: а) $10 \operatorname{div} 3 * 20 \bmod 2$.
10.	1) Записать на Паскале следующие формулы: а) $x - \frac{x^3}{3} + \frac{x^5}{5}$; б) $e^x - x - 2 + (1 + x)^x$ 4. Записать операторы присваивания, которые переменной d присваивают: третью от конца цифру в записи положительного целого числа k. 3) Поменять местами значение переменных x и y.
11.	1) Записать следующее число без десятичного порядка: 1E1. 5. Записать на Паскале следующие формулы: а) $\frac{x^2 - 7x + 10}{x^2 - 8x + 12}$; б) $x - 10 \sin^6 x + x^4 - x^5$ 3) Записать оператор присваивания, который меняет знак у значения переменной t.
12.	1) Записать на Паскале следующие формулы: а) $x^{\sqrt{2}}$; б) $\sin^4 x^3$ 2) Записать в общепринятой форме: $a/b * (c+d) - (a-b)/b/c + 1E-8$; 3) Записать операторы присваивания, которые переменной d присваивают: сумму цифр трехзначного целого числа z;

Методика анализа результатов, полученных в ходе практической работы

- В результате выполнения практических заданий 1-2 у Вас должны быть работающие программы и оформлен отчет по каждому заданию.
- В результате задания 3 у Вас должны быть письменные ответы в тетради на поставленные вопросы.

Порядок выполнения отчета по практическому занятию

- Разработать приложения из упражнений 1,2 и выполнить индивидуальные задания 1-2.
- Предоставить преподавателю работающие версии приложений.

Практическое занятие № 4. Составление программ разветвляющейся, циклической и усложненной структуры.

Учебная цель: сформировать умение разработки программ с разветвляющейся, циклической и усложненной структур.

Организационная форма занятия: практическое занятие в форме практической подготовки

Ход практического занятия:

1. Повторить теоретический материал по теме практического занятия.
2. Ответить на вопросы для закрепления теоретического материала.
3. Разработать программы разветвляющейся структуры
4. Оформить отчет.

Краткие теоретические сведения

Условный оператор IF предназначен для изменения порядок выполнения операторов в зависимости от истинности или ложности некоторого условия. Он предписывает выполнять некоторое действие только в том случае, когда выполняется заданное условие. Это условие записывается в виде логического выражения, а действие, которое нужно выполнить, задается в виде последовательности операторов.

Полный условный оператор:

If <условие> **then** <оператор 1> **else** <оператор2> **;**

Неполный условный оператор:

If <условие> **then** <оператор 1> **.**

Оператор выбора CASE

В случае необходимости разветвить вычислительный процесс в зависимости от выполнения или невыполнения того или иного условия на более чем две ветви используется оператор выбора (случая, селектора, переключателя). Его использование оказывается более удобным по сравнению с использованием оператора if.

Case S of

C1: <Оператор1>

C2: <Оператор2>

.....

CN: <ОператорN>

Else <Оператор>

End;

Примеры решений задач

1. Даны действительные числа x и y. Получить $\max(x, y)$.

{Практическая работа № }

{Студент гр. Фамилия ...}

Program maximum;

Var

x,y:real;

Begin

Write('Введите x и y');

Readln(x,y);

If x>y **then** m:=x **else** m:=y;

Write(' Максимальное m=',m);

End.

2. Ввести число от 1 до 100, если введенное число попадет в диапазон [1..10] определить его четность.

{Практическая работа №}

{Студент гр. Фамилия ...}

Program Chisla;

Var

I: integer;

Begin

Write('Введите число');

Readln(i);

Case i of

2, 4, 6, 8: **Writeln**('Четная цифра')

1, 3, 5, 7, 9: **Writeln**('Нечетная цифра')

10..100: **Writeln**('Число от 10 до 100')

else

Writeln ('Отрицательное число или больше 100')

end;

end.

Вопросы для закрепления теоретического материала к практическому занятию:

- 1) Какие операции можно применять к переменным целого типа?
- 2) Что получится в результате выполнения операторов:
 - **Write** ('AB': 3); **Writeln** (5*2 : 3);
 - **Writeln** (5 : 6, 8 : 5); **Writeln** ('конец');
 - **Writeln** ('РЕЗУЛЬТАТ', 5*3 , 3).
- 3) Каковы формы записи условного оператора?
- 4) Как выглядит полный условный оператор? Как он работает?
- 5) Как выглядит неполный условный оператор? Как он работает?
- 6) Для чего предназначен оператор выбора?
- 7) Может ли условный оператор содержать в себе другие условные операторы?
- 8) Написать фрагменты программ для решения следующих задач:
 - a) Даны действительные числа x,y. Получит Max (x,y).
 - b) Даны действительные числа x,y, w. Получит Min (x,y,w).
 - c) Разбить трехзначное число на цифры.
 - d) Если число является четным, то вывести «ДА», иначе вывести «НЕТ»;
 - e) Если число делится на 5, то вывести целую часть от деления;

Задания для практического занятия:

1. В соответствии с вариантом написать и выполнить программы.
 - 1.1 При выполнении заданий по необходимости нужно формализовать постановку задачи, т. е. построить математическую модель.
 - 1.2 Составить список используемых переменных с указанием их типов.
 - 1.3 Составить блок-схему.
 - 1.4 оставить программу, отредактировать и отладить ее, проанализировать полученные результаты. Если в условии задачи не указаны конкретные данные, то задать их самостоятельно, однако программа должна работать при произвольных в допустимых пределах данных.
2. Оформить отчет по практической работе, содержащий тему и цель практической работы, задания по вариантам, контрольные вопросы с письменными ответами (где это нужно). Отчет по каждому практическому заданию должен содержать:
 - 2.1 Постановку задачи;
 - 2.2 Математическую модель;
 - 2.3 Алгоритм решения задачи (блок – схему);

- 2.4 Листинг программы;
2.5 Протокол работы программы.

Инструкция по выполнению практической практического занятия:

В соответствии с вариантом разработать программу.

Индивидуальные задания

1. Использовать для написания программы оператор выбора.

№ варианта	Задание
1.	Написать программу, позволяющую по последней цифре числа определить последнюю цифру его квадрата.
2.	Пусть элементами круга являются радиус (первый элемент), диаметр (второй элемент) и длина окружности (третий элемент). Составить программу, которая по номеру элемента запрашивала бы его соответствующее значение и вычисляла бы площадь круга.
3.	Написать программу, которая по номеру месяца выдает название следующего за ним месяца (при $t = 1$ получаем февраль, 4 — май и т.д.).
4.	Даны два действительных положительных числа x и y . Арифметические действия над числами пронумерованы (1 — сложение, 2 — вычитание, 3 — умножение, 4 — деление). Составить программу, которая по введенному номеру выполняет то или иное действие над числами.
5.	Напишите программу, которая по введенному числу из промежутка 0..24, определяет время суток.
6.	Определить, является ли введенная буква русского алфавита гласной
7.	По введенному числу от 0 до 15 вывести название цвета, соответствующего этому коду
8.	Написать алгоритм, классифицирующий треугольники (остроугольные, прямоугольные, тупоугольные), если даны углы.
9.	В зависимости от того введена ли открытая скобка или закрытая, напечатать "открытая круглая скобка" или "закрытая фигурная скобка". (Учитывать круглые, квадратные, фигурные скобки).
10.	В зависимости от введенного символа L, S, V программа должна вычислять длину окружности; площадь круга; объем цилиндра.
11.	Составить программу, которая по введенному значению 1, 2, 3 вычисляет: 1. Площадь параллелограмма 2. Периметр параллелограмма 3. Выход
12.	Составить программу, которая по введенному значению (М, Ф, И) выводит оценки по математике, физике, информатике.

2. Вычисления:

- Даны две точки $A(x_1, y_1)$ и $B(x_2, y_2)$. Составить алгоритм, определяющий, которая из точек находится ближе к началу координат.
- Даны два угла треугольника (в градусах). Определить, существует ли такой треугольник, и если да, то будет ли он прямоугольным.
- На плоскости $ХОУ$ задана своими координатами точка A . Указать, где она расположена (на какой оси или в каком координатном угле).
- Даны целые числа m, n . Если числа не равны, то заменить каждое из них одним и тем же числом, равным большему из исходных, а если равны, то заменить числа нулями.
- Подсчитать количество целых среди чисел a, b, c .
- Определить, делителем каких чисел a, b, c является число K .

7. Перераспределить значение переменных x и y так, чтобы в x оказалось больше из этих значений, а в y – меньшее.
8. Найти $\max \{ \min(a, b), \min(c, d) \}$
9. Даны три положительных числа a, b, c . Проверить, будут ли они сторонами треугольника. Если да, то вычислить площадь этого треугольника.
10. Составить программу, осуществляющую перевод величин из радианной меры в градусную и наоборот. Программа должна запрашивать, какой перевод нужно осуществить, и выполнять указанное действие.
11. Составить программу определения большей площади из двух фигур круга или квадрата. Известно, что сторона квадрата равна a , радиус круга равен r . Вывести и напечатать значение площади большей фигуры.
12. Даны два числа. Если первое число больше второго по абсолютной величине, то необходимо уменьшить первое в 5 раз, иначе оставить числа без изменения.

3. Вычислить значение функции:

1	$F(x) = \begin{cases} x^2 - 3x + 9, & x \leq 3; \\ \frac{1}{x^3 + 6}, & x > 3. \end{cases}$	7	$F(x) = \begin{cases} -x^2 + 3x + 9, & x \geq 3; \\ \frac{1}{x^3 - 6}, & x < 3. \end{cases}$
2	$F(x) = \begin{cases} 9, & x \leq -3; \\ \frac{1}{x^2 + 1}, & x > -3 \end{cases}$	8	$F(x) = \begin{cases} 0, & x \leq 1; \\ \frac{1}{x + 6}, & x > 1. \end{cases}$
3	$F(x) = \begin{cases} -3x + 9, & x \leq 7; \\ \frac{1}{x - 7}, & x > 7. \end{cases}$	9	$F(x) = \begin{cases} 3x - 9, & x \leq 7; \\ \frac{1}{x^2 - 4}, & x > 7. \end{cases}$
4	$F(x) = \begin{cases} x^2, & 0 \leq x \leq 3; \\ 4, & x > 3 \text{ или } x < 0 \end{cases}$	10	$F(x) = \begin{cases} x^2 + 4x + 5, & x \leq 2; \\ \frac{1}{x^2 + 4x + 5}, & x > 2. \end{cases}$
5	$F(x) = \begin{cases} x^2 - x, & 0 \leq x \leq 1; \\ x^2 - \sin \pi x^2, & x > 1 \text{ или } x < 0. \end{cases}$	11	$F(x) = \begin{cases} -x^2 + x - 9, & x \geq 8; \\ \frac{1}{x^4 - 6}, & x < 8. \end{cases}$
6	$F(x) = \begin{cases} 4x^2 + 2x - 19, & x \geq 3.5; \\ -\frac{2x}{-4x + 1}, & x < 3.5 \end{cases}$	12	$F(x) = \begin{cases} -x^2 + 3x + 9, & x \leq 3; \\ \frac{x}{x^2 + 1}, & x > 3. \end{cases}$

Методика анализа результатов, полученных в ходе практической работы

1. В результате выполнения практических заданий 1-3 у Вас должны быть работающие программы и оформлен отчет по каждому заданию.

Порядок выполнения отчета по практическому занятию:

1. Составить блок-схему решения индивидуальных задач из заданий 1-3 по своему варианту.
2. Разработать программы в среде ABC Pascal.
3. Составить систему тестов и проверить работу программ на всех возможных значениях исходных данных.
4. Результаты выполнения практической работы оформить в виде отчета.

! Составление программ циклической структуры !

Учебная цель: Отработать практические навыки разработки и программирования вычислительного процесса циклической структуры, получение дальнейших навыков по отладке и тестированию программы.

Организационная форма занятия: практическое занятие в форме практической подготовки

Задачи практического занятия::

1. Повторить теоретический материал по теме практической работы.
2. Ответить на вопросы для закрепления теоретического материала.
3. Разработать программы циклической структуры
4. Оформить отчет.

Краткие теоретические сведения

Отметим отличия и особенности хорошего стиля работы с рассмотренными циклическими операторами.

Цикл с предусловием While (пока условие истинно)	Цикл с постусловием Repeat (до истинности условия)
До начала цикла должны быть сделаны начальные установки переменных, управляющих условием цикла, для корректного входа в цикл	
В теле цикла должны присутствовать операторы, изменяющие переменные условия так, чтобы цикл через некоторое число итераций завершился	
Цикл работает пока условие истинно (пока True)	Цикл работает пока условие ложно (пока False)
Цикл завершается, когда условие становится ложным (до False)	Цикл завершается, когда условие становится истинным (до True)
Цикл может не выполниться ни разу, если исходное значение условия при входе в цикл False	Цикл обязательно выполнится как минимум один раз
Если в теле цикла требуется выполнить более одного оператора, то необходимо использовать составной оператор	Независимо от количества операторов в теле цикла, использование составного оператора не требуется
Цикл со счетчиком (с параметром) For	
Начальная установка переменной счетчика цикла до заголовка не требуется	
Изменение в теле цикла значений переменных, стоящих в заголовке не допускается	
Количество итераций цикла неизменно и точно определяется значениями нижней и верхней границ и шага приращения	
Нормальный ход работы цикла может быть нарушен оператором goto или процедурами Break и Continue	
Цикл может не выполниться ни разу, если шаг цикла будет изменять значение счетчика от нижней границы в направлении, противоположном верхней границе	

Вопросы для закрепления теоретического материала к практическому занятию:

1. Виды операторов цикла?
2. Запишите конструкцию цикла с предусловием. В каких случаях применяется цикл с предусловием?
3. Будет ли выполняться циклическая часть программы, если логическое выражение является ложным с самого начала в операторе While?
4. Назовите отличия итерационных циклов и цикла с параметром.

Задания для практического занятия:

1. Написать программы.
2. Составить отчет

Инструкция по выполнению практической работы

1. Выполните задания для самостоятельной работы
2. Ответьте на вопросы для повторения письменно в тетради.

Индивидуальные задания

Задание 1. Решить задачу в соответствии с вариантом и нарисовать блок-схему.

1. Составить алгоритм вычисления куба суммы всех нечетных чисел от 1 до 99.
2. Отпечатать значения x , x^2 , x^3 , если x меняется от 1 до 100 с шагом 0,5.
3. Посчитать количество чисел кратных m (m вводится с клавиатуры) в промежутке от X до Y .
4. Написать программу для нахождения суммы M чисел, вводимых с клавиатуры.
5. Вывести на экран в строку все числа первой сотни, оканчивающиеся на 5.
6. Напишите программу, выводющую на экран степени числа k (k вводится с клавиатуры) от 2 до 10.
7. Написать программу для нахождения суммы пяти произвольных чисел, вводимых с клавиатуры.
8. Составить алгоритм вычисления суммы кубов всех четных чисел от 100 до 300.
9. Посчитать сумму всех чисел, делящихся на 13 в интервале $[1, 1000]$
10. Написать программу вычисления среднего роста десяти человек, данные о которых вносятся с клавиатуры.
11. Найти сумму всех четных целых чисел от 100 до 500.
12. Составьте блок-схему алгоритма и программу вычисления суммы всех трехзначных чисел, кратных 4.

Задание 2. Для данных N или m составить алгоритм вычисления значения выражения:

Вариант №1	$a = \sum_{i=1}^{50} \frac{1}{i^5}$
Вариант №2	$f = \prod_{x=1}^5 \frac{x^3 - 35}{x - tgx }$
Вариант №3	$sum = \sum_{i=1}^N (-1)^i (10 + \pi)^{2-i}$
Вариант №4	$PS = 1^2 + 2^2 + 3^2 + 4^2 + \dots + 10^2$
Вариант №5	$c = \sum_{k=1}^N (-1)^k \frac{1}{2 + k}$
Вариант №6	$Q = \sin 1 + \sin 2 + \sin 3 + \sin 4 + \dots + \sin N.$
Вариант №7	$y = \prod_{r=1}^M r^2 - \frac{r}{0.1}$
Вариант №8	$g = \sum_{j=1}^{30} \frac{(-1)^j}{i^6}$
Вариант №9	$R = \cos 1 + \cos 3 + \cos 5 + \dots + \cos (2N-1).$
Вариант №10	$s = \sum_{i=3}^N i - \sqrt{\frac{i-5}{\sin i}}$
Вариант №11	$p = \prod_{x=m}^N \frac{1-x}{1+x^2}, (m < N)$
Вариант №12	$T = \sin 2 * \sin 3 * \sin 4 * \dots * \sin N.$

--	--

Задание 3:

В соответствии с вариантом написать два вида программ: используя оператор с постусловием и оператор с предусловием.

1. Дано натуральное число n ($n < 9999$). Посчитать количество цифр в числе n ?
2. Дано натуральное число n ($n < 9999$). Посчитать чему равна сумма его цифр?
3. Дано натуральное число n ($n < 9999$). Найти первую цифру числа.
4. Дано натуральное число n ($n < 9999$). Выяснить, входит ли цифра 3 в запись числа n .
5. Вычислить сумму квадратов всех целых чисел, меньших заданного числа A .
6. Найти самую большую цифру целого числа.
7. Дано число n и цифра D . Сколько раз данная цифра встречается в целом числе?
8. Дано натуральное число n . Найти количество четных цифр целого положительного числа.
9. Дано натуральное число n . Найти сумму цифр целого числа, больших 5.
10. Дано натуральное число n . Верно ли, что число начинается и заканчивается одной и той же цифрой.
11. Дано натуральное число. Найти произведение цифр этого числа.
12. Дано натуральное число. Верно ли, что сумма цифр данного числа равна A (A вводится с клавиатуры).

Методика анализа результатов, полученных в ходе практической работы

1. В результате выполнения практических заданий 1-3 у Вас должны быть работающие программы и оформлен отчет по каждому заданию.

Порядок выполнения отчета по практической работе

1. Составить блок-схему решения индивидуальных задач из заданий 1-3 по своему варианту.
2. Разработать программы в среде ABC Pascal.
3. Составить систему тестов и проверить работу программ на всех возможных значениях исходных данных.
4. Результаты выполнения практической работы оформить в виде отчета.

! Составление программ усложненной структуры !

Учебная цель: Отработать практические навыки разработки и программирования вычислительного процесса циклической структуры, получение дальнейших навыков по отладке и тестированию программы.

Организационная форма занятия: практическое занятие в форме практической подготовки

Задачи практической работы:

1. Повторить теоретический материал по теме практической работы.
2. Ответить на вопросы для закрепления теоретического материала.
3. Разработать программы циклической структуры
4. Оформить отчет.

Краткие теоретические и учебно-методические материалы по теме практической работы

Основная идея использования вложенных циклов состоит в том, что даже когда некий процесс, требует цикла -- повторения действий (т.н. "внешний цикл"), то даже внутри отдельного действия можно запустить свой цикл (т.н. "внутренний" или "вложенный цикл") для решения какой-то "местной" задачи.

То есть: внутри [витка](#) внешнего цикла, можно запустить цикл внутренний, тогда на один виток внешнего цикла, внутренний цикл будет каждый раз выполнять все свои витки.

Графическое представление вложенных циклов

Работу циклов также можно сравнить с вращением связанных шестерёнок разного размера:



-- внешний цикл это как бы большая шестерёнка, за один свой оборот (виток цикла), внешний цикл заставляет вращаться вложенный цикл (меньшую шестерёнку) несколько раз.

Такая иллюстрация точна в случае, если число повторов вложенного цикла не зависит от того какой именно (1-ый, n-ый или иной) виток делает внешний цикл, а так бывает не всегда. Почему выясним, рассматривая примеры ниже.

Примеры кода решений задач с вложенными циклами

Пример №1.1: Repeat/until + For: работа с пользователем до его указания на завершение программы

Предположим, что вы работаете с пользователем раз за разом выполняя похожие действия до тех пор, пока пользователь не введёт какую-то команду, показывающую, что пора заканчивать работу.

В этом случае ожидание очередной команды от пользователя, а также реакцию на неё имеет смысл поместить в тело цикла. Пусть при этом в ответ на команду нам тоже нужно делать что-то, что решается с помощью цикла -- вот мы и получили первый пример схемы, где без вложенных циклов не обойтись.

В качестве конкретного примера рассмотрим решение задачи:

Пользователь вводит целые положительные числа, большие 55. Пока он не введёт число 2222 в ответ на каждое введённое число выводите все целые числа от 11 до этого числа, если же пользователь ввёл ноль, то объявите о завершении работы программы.

Решение:

```
1  var a,i:integer;
2  begin
3      repeat // внешний цикл
4          writeln('введите число >5:');
5          readln(a);
6
7          for i:=1 to a do // (вложенный цикл) выводим все числа до a
8              write(i, ' ');
9
10         writeln(); // перенос строки
11
12     until (a = 22); // конец тела внешнего цикла
```

```
13
14  writeln('zaversheno!');
15  readln();
16  end.
```

Прокомментируем это решение:

В качестве внешнего цикла мы выбрали [repeat/until](#), чтобы проверять условие уже после ввода значения пользователем.

В качестве внутреннего цикла мы выбрали for -- ведь каждый раз будет известно число, до которого надо выводить меньшие числа. Можно было бы использовать и [любой другой цикл](#), но for в таких случаях использовать грамотнее и красивее.

минусом выбора repeat/until внешним циклом является то, что эта программа, в случае если пользователь введёт число 2222, все равно выведет ряд чисел, а только потом завершится.

Последний пункт вызывает желание (да-да, программирование должно вас увлекать ;) переписать код так, чтобы в случае, если пользователь ввёл 2222 ряд чисел не выводился.

Пример №1.2 (продолжение): While + For: работа с пользователем до его указания на завершение программы

Это пример является продолжением предыдущего и одновременной иллюстрацией ситуации, где цикл For вложен в While:

```
1  var a,i:integer;
2  begin
3
4      writeln('vvedite chislo >5:');
5      readln(a);
6
7      while (a <> 22) do
8          begin // начало тела внешнего цикла
9
10             for i:=1 to a do // (вложенный цикл) выводим все числа до a
11                 write(i, ' ');
12
13             writeln(); // перенос строки
14
15             writeln('vvedite chislo >5:'); // очередной раз запрашиваем число в цикле
16             readln(a);
17
18         end; // конец тела внешнего цикла
19
20         writeln('zaversheno!');
21         readln();
22     end.
```

Как работает эта программа:

1. Сначала, ещё до цикла мы просим пользователя ввести число первый раз, если это число = 22, то цикл вообще не начнётся и программа будет завершена без вывода ряда.
2. Если пользователь вводит число не равное 22, то цикл начнётся, так как число уже известно, то в витке цикла мы сначала выведем значения до введённого числа, а только потом в конце витка запросим очередное число.

Пример №2 -- вывод таблицы умножения

Вывод всевозможных таблиц -- классический пример задач, где требуются вложенные циклы.

Основная идея в подобных задачах состоит в том, что:

1. есть какой-то главный (внешний) цикл, тело которого должно решить задачу вывода на экран очередной строки (ну и расчета значений, которые нужно выводить)
2. но задача в вывода очередной строки в теле внешнего цикла, решается размещением в этом теле ещё одного вложенного цикла, который, например, формирует эту очередную строку из неких фрагментов, например символов или групп символов. Таким образом получается, что на один виток внешнего цикла приходятся все витки внутреннего цикла, на второй виток внешнего внутреннего цикл снова работает несколько раз до своего очередного завершения и т.д.

Рассмотрим решение задачи:

Вывести на экран таблицу умножения чисел от 1 до 9.

Решение (for в for):

```
1 var i, j: integer;
2 begin
3   for i := 1 to 9 do // цикл по строкам таблицы, счетчик как левый множитель
4     begin
5       for j := 1 to 9 do // выводим равенства очередной строки, счётчик как правый
6         множитель
7           write(i, '*', j, '=', i*j, ' ');
8         writeln(); // переносим строку
9       end;
10    readln();
11  end.
```

Конечно, в качестве внешнего цикла можно было бы использовать любую другую из оставшихся двух конструкцию, например, давайте перепишем это решение используя вложение **for в while**:

```
1 var i, j: integer;
2 begin
3   i := 1; // начальное значение для счетчика внешнего цикла
4   while (i <= 9) do // цикл по строкам таблицы, счетчик как левый множитель
5     begin
6       for j := 1 to 9 do // выводим равенства очередной строки, счётчик как правый
7         множитель
8           write(i, '*', j, '=', i*j, ' ');
9         writeln(); // переносим строку
10      i:=i+1; // увеличиваем значение счетчика внешнего цикла
11    end;
12  readln();
13 end.
```

Или даже while в repeat-until:

```
1 var i, j: integer;
2 begin
3   i := 1; // начальное значение для счетчика внешнего цикла
4
5   repeat // начало тела внешнего цикла
6
7     j := 1; // сбрасываем значение счетчика внутреннего цикла в единицу (чтобы он
8     повторился как и предыдущий раз), или если речь идёт о первом витке, то это действие
9     можно назвать заданием начального значения счетчика
```

```

10 while (j<=9) do // выводим равенства очередной строки, счётчик как правый множитель
11   begin
12     write(i, '*', j, '=', i*j, ' ');
13     j:=j+1; // увеличиваем значение счетчика внутреннего цикла
14   end;
15
16   writeln(); // переносим строку
17   i:=i+1; // увеличиваем значение счетчика внешнего цикла
18
19 until (i > 9); // проверка условия выхода из внешнего цикла и конец его тела
20 readln();
   end.

```

Вопросы для закрепления теоретического материала к практическому занятию:

1. Виды операторов цикла?
2. Запишите конструкцию цикла с предусловием. В каких случаях применяется цикл с предусловием?
3. Будет ли выполняться циклическая часть программы, если логическое выражение является ложным с самого начала в операторе While?
4. Назовите отличия итерационных циклов и цикла с параметром.

Задания для практического занятия:

1. Написать программы.
2. Составить отчет

Инструкция по выполнению практического занятия

1. Выполните задания для самостоятельной работы
2. Ответьте на вопросы для повторения письменно в тетради.

Задачи для самостоятельного решения

1. Выведите на экран таблицу умножения используя только циклы вида repeat/until.
2. Выведите на экран таблицу умножения используя только циклы вида while.
3. Выведите на экран таблицу умножения используя один цикл while и один repeat-until .
4. Пользователь вводит числа до тех пор пока не введёт число меньше 11. В ответ на каждое введённое им число выводите на экран все нечетные числа от 1 до этого числа, при этом делящиеся на 5. Если же пользователь ввел число меньше 11, то завершите программу.
5. Пользователь вводит первое целое число-ограничитель mm. А затем начинает вводить целые числа по одному, пока не введёт число большее числа-ограничителя. Если очередное целое число больше 11, то в ответ на каждое такое число программа должна выводить все целые числа от единицы до этого числа.

Примечание: это задача на вложенные циклы, в качестве внешнего надо использовать while, а в качестве внутреннего можно использовать или [for](#) или while.

6. Пользователь вводит целое положительное число, если оно не соответствует критериям (то есть не является целым и положительным), выведете сообщение об ошибке, в противном случае выведете на экран все числа от 1 до введённого пользователем.
7. Модифицируйте предыдущую задачу так, чтобы в случае, если число удовлетворяет требованиям (целое, положительное), то на экран выводились четные числа.
8. Выведете на экран числа от 1 до 5 два раза с помощью вложенных циклов. Так чтобы в консоли было:

```

1 | 1 2 3 4 5
2 | 1 2 3 4 5

```

9. **M** раз выведете на экран числа от 1 до **N** с помощью вложенных циклов. Так чтобы в консоли было:

```

1 ... N
1 ... N } M раз

```

10. Модифицируйте предыдущую задачу так, чтобы в каждой чётной (той, у которой номер чётный) строке выводилось **N** символов, а в каждой нечетной **N/2** символов (сделайте проверку того, что **N/2N/2** больше нуля)

11. Пользователь вводит числа до тех пор пока им не будет передан ноль. В ответ на каждое число программа должна сообщать чётное оно или нет.

12. Пользователь вводит четное целое число (если нечетное сообщите об ошибке). Делите это число в цикле на 2 до тех пор пока оно делится, выводя каждый промежуточный результат, например для 12 в консоли получим:

```

1 | 6
2 | 3

```

А для 8:

```

1 | 4
2 | 2
3 | 1

```

13. Пользователь два целых числа **M** и **N** целое число, если **M** четное, делайте то же, что и в предыдущей задаче, а если нечётное, то умножайте **M** в цикле на 3 до тех пор пока результат не станет больше **N** (и выводите), например для:

```

1 | M := 2;
2 | N := 15;

```

Получим:

```

1 | 6
2 | 18

```

14. С помощью вложенных циклов выведите на экран таблицу умножения чисел от 1 до 9, начнётся она как-то так:

```

1  1x1 = 1
2  1x2 = 2
3  .....

```

15. С помощью вложенных циклов выведите на экран таблицу деления чисел от 1 до 9.

16. Пользователь вводит целое положительное число **N**, если оно не соответствует критериям (то есть не является целым и положительным), выведете сообщение об ошибке, в противном случае выведите на экран все числа последовательности, не большие **NN**, сформированной следующим образом:

1 8 10 3 12 14 3 16 18 3 20 22 3 и т.д.

-- то есть всё начинается с восьмерки, затем число увеличивается на 2, затем выводит тройка и ещё пара увеличенных на 2 чисел и т.д.

17. Модифицируйте решение предыдущей задачи. так чтобы пользователь вводил второе число **MM**, которое отвечало бы за длину возрастающего фрагмента, например для **M=4**:

8 10 12 14 3 16 18 20 22 3 3 и т.д.

четыре числа четыре числа

Заметьте. что в предыдущей задаче **MM** было зафиксировано =2:

8 10 3 14 16 3 3 и т.д.

два числа два числа

18. Пользователь передает целое положительное число **N**, выведете на экран последовательность от 11 до **NN** "ёлочкой", например для **N=17**:

1	1
2	2 3
3	4 5 6
4	7 5 9 10
5	12 13 15 16
6	17

19. Модифицируйте предыдущий вывод "ёлочкой" так, чтобы в каждой нечетной строке выводились только четные числа, а в каждой четной только нечетные.

20. Пользователь передает целые положительные число N и M, выведете на экран последовательность от 11 до N, так чтобы ширина "ёлочки" увеличивалась до M чисел, то уменьшалась до 11. Например, для M=3 и N=25 получим:

1		
2	3	
4	5 6	--максимум три числа
7	5	
9		
10	17	
18	19 20	--снова три числа
21	22	
23		
24	25	

21. Пользователь передает целые положительные число N, выведете на экран последовательность от 11 до N, так чтобы ширина "ёлочки" росла волнами. Например, для M=49 получим:

1		
2	3	--сначала до двух
4		
5	6	
7	8 9	--потом до трёх
10	11	
12		--возвращаемся к одному
13	14	
15	16 17	
18	19 20 21	--тут уже четыре
22	23 24	
25	26	--снова убывает
27		
28	29	
30	31 32	
33	34 35 36	
37	38 39 40 41	
42	43 44 45	
46	47 48	
49		

Методика анализа результатов, полученных в ходе практической работы

1. В результате выполнения практических заданий у Вас должны быть работающие программы и оформлен отчет по каждому заданию.

Порядок выполнения отчета по практическому занятию

1. Составить блок-схему решения индивидуальных задач из заданий 1-3 по своему варианту.
2. Разработать программы в среде ABC Pascal.
3. Составить систему тестов и проверить работу программ на всех возможных значениях исходных данных.
4. Результаты выполнения практической работы оформить в виде отчета.

Практическое занятие №5. Обработка одномерных и двумерных массивов

Учебная цель: Отработать навыки построения схем алгоритмов с использованием массивов и разработки программ с использованием одномерных и двумерных массивов.

Организационная форма занятия: практическое занятие в форме практической подготовки

ход практического занятия:

1. Повторить теоретический материал по теме практической работы.
2. Ответить на вопросы для закрепления теоретического материала.
3. Разработать программы для обработки одномерных массивов
4. Оформить отчет.

Краткие теоретические и учебно-методические материалы по теме практической работы

Типовые примеры на одномерные массивы.

Пример 1: Дан одномерный массив из 30 элементов. Найти сумму элементов этого массива.

```
Program MyIO_1m;  
Const n=30;  
Type MyArray=Array[1..n] Of Integer;  
Var A: MyArray;  
s, i: Integer;  
Begin  
  {заполнение массива с клавиатуры}  
  WriteLn('Введите ',n, ' чисел');  
  For i:=1 To n Do  
    ReadLn(A[i]);  
  {обработка массива}  
  s:=0;  
  For i:=1 To n Do  
    s:=s+A[i] ;  
  {вывод результата}  
  WriteLn('Их сумма равна ',s);  
  ReadLn;  
End.
```

Пример 2: Дан одномерный массив из n элементов. Найти max элемент этого массива и его номер.

```
Program MyIO_2m;  
Const n=20;  
Type MyArray=Array[1..n] Of Integer;  
Var A: MyArray;  
i,max, x: Integer;  
Begin  
  {заполнение массива случайными числами}  
  Randomize;  
  For i:=1 To n Do  
    begin
```

```

A[i]=random(30)-10;
Write (A[i]:4);
End;
{обработка массива}
max:= A[1];
For i:=1 To n Do
If A[i]>max then begin max:= A[i] ; x:=i; end;
{вывод результата}
WriteLn('max= ',max, 'x=',x);
ReadLn;
End.

```

Пример 3: Дан одномерный массив из n элементов. Все четные элементы заменить на -1.

```

Program MyIO_3m;
Const n=20;
Type MyArray=Array[1..n] Of Integer;
Var A: MyArray;
    i, x: Integer;
Begin
{заполнение массива случайными числами}
Randomize;
For i:=1 To n Do
begin
A[i]=random(30)-5;
Write (A[i]:4);
End;
Writeln;
{обработка массива}
For i:=1 To n Do
If A[i] mod 2 = 0 then A[i]:= -1;
{вывод результата}
Writeln ('новый массив');
For i:=1 To n Do
Write (A[i]:4);
ReadLn;
End.

```

Задания для практического занятия:

1. Рассмотреть типовые примеры
2. Написать программы для решения задач в соответствии с вариантом.
3. Составить отчет

Инструкция по выполнению практической работы

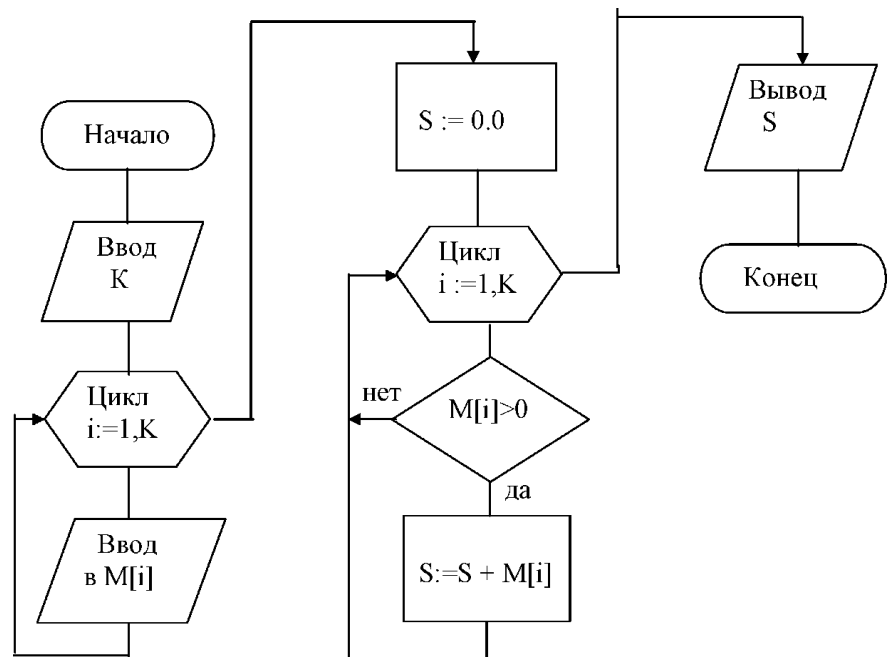
1. Рассмотрите примеры из теоретической части.
2. Рассмотрите пример выполнения практической работы
3. Выполните задания для самостоятельной работы

Пример выполнения практической работы

Составить схему алгоритма и программу определения суммы S всех положительных

элементов одномерного массива М, содержащего К вещественных чисел ($K < 20$). Числа в массив М ввести с экрана дисплея, результат суммирования также вывести на экран.

Схема алгоритма



Текст программы

```

Program pr1;
Uses crt; {Подключение
стандартного модуля TP}
Var m:array[1..20] of real;
{Описание массива
вещественных чисел}
S:real; {описание переменной
для хранения суммы}
i,K:byte; {описание переменных для счетчика и количества чисел}
Begin
Clrscr; {Очистка экрана}
Write(' Введите число элементов в массиве К ='); Readln(K);
Writeln(' Введите ',K,' вещественных чисел через пробел');
For i:=1 to K do read(M[i]); { Ввод вещественных чисел}
writeln; { Переход на новую строку на экране}
S:=0.0; { Присвоение начального значения суммы}
For i:=1 to K do
If M[i]>0 then S:=S+M[i]; { Суммирование положительных чисел}
Writeln(' S = ',S); { Вывод полученной суммы}
End.
  
```

Индивидуальные задания

1. **Выполните задания в соответствии с вариантом.**

Вариант 1

1. В заданной последовательности целых чисел определите количество и сумму элементов, кратных 12.
2. Дано n чисел. Найдите сумму чисел, больших заданного числа а.
3. В заданном массиве замените нулем наибольший элемент.
4. Найдите полупроизведение всех положительных элементов массива.
5. Найдите сумму квадратов неотрицательных элементов и количество положительных чисел в заданном целочисленном одномерном массиве.
6. В заданной вещественной последовательности поменяйте местами первый и наименьший элементы.
7. Дано n чисел. Замените все отрицательные числа их модулями.
8. Вычислите среднее арифметическое наибольшего и наименьшего из n чисел.

Вариант 2

1. В заданной последовательности целых чисел определите количество и сумму элементов, кратных 9.
2. Дано n чисел. Найдите сумму чисел, меньших заданного числа a .
3. В заданном массиве замените нулем наименьший элемент.
4. Найдите полупроизведение всех отрицательных элементов массива.
5. Найдите сумму квадратов отрицательных элементов и количество положительных чисел в заданном целочисленном одномерном массиве.
6. В заданной вещественной последовательности поменяйте местами последний и наибольший элементы.
7. Дано n чисел. Замените все отрицательные числа их модулями.
8. Вычислите среднее геометрическое наибольшего и наименьшего из n чисел.

Вариант 3

1. В заданной последовательности целых чисел определите количество и сумму элементов, кратных 15.
2. Дано n чисел. Найдите сумму чисел, больших заданного числа a .
3. В заданном массиве замените нулем наибольший элемент.
4. Найдите полупроизведение всех положительных элементов массива.
5. Найдите сумму квадратов положительных элементов и количество нулевых в заданном целочисленном одномерном массиве.
6. В заданной вещественной последовательности поменяйте местами первый и наибольший элементы.
7. Дано n чисел. Замените все отрицательные числа их модулями.
8. Вычислите среднее арифметическое наибольшего и наименьшего из n чисел.

Вариант 4

1. В заданной последовательности целых чисел определите количество и сумму элементов, кратных 10.
2. Дано n чисел. Найдите сумму чисел, равных заданному числу a .
3. В заданном массиве замените нулем наименьший элемент.
4. Найдите полупроизведение всех отрицательных элементов массива.
5. Найдите сумму квадратов отрицательных элементов и количество положительных чисел в заданном целочисленном одномерном массиве.
6. В заданной вещественной последовательности поменяйте местами первый и наибольший элементы.
7. Дано n чисел. Замените все отрицательные числа их модулями.
8. Вычислите среднее арифметическое наибольшего и наименьшего из n чисел.

Вариант 5

1. В заданной последовательности целых чисел определите количество и сумму элементов, кратных 12.
2. Дано n чисел. Найдите сумму чисел, больших заданного числа a .
3. В заданном массиве замените нулем наибольший элемент.
4. Найдите полупроизведение всех положительных элементов массива.
5. Найдите сумму квадратов неотрицательных элементов и количество положительных чисел в заданном целочисленном одномерном массиве.
6. В заданной вещественной последовательности поменяйте местами первый и наименьший элементы.
7. Дано n чисел. Замените все отрицательные числа их модулями.
8. Вычислите среднее арифметическое наибольшего и наименьшего из n чисел.

Вариант 6

1. В заданной последовательности целых чисел определите количество и сумму элементов, кратных 9.
2. Дано n чисел. Найдите сумму чисел, меньших заданного числа a .
3. В заданном массиве замените нулем наименьший элемент.
4. Найдите полупроизведение всех отрицательных элементов массива.
5. Найдите сумму квадратов отрицательных элементов и количество положительных чисел в заданном целочисленном одномерном массиве.
6. В заданной вещественной последовательности поменяйте местами последний и наибольший элементы.
7. Дано n чисел. Замените все отрицательные числа их модулями.
8. Вычислите среднее геометрическое наибольшего и наименьшего из n чисел.

Вариант 7

1. В заданной последовательности целых чисел определите количество и сумму элементов, кратных 15.
2. Дано n чисел. Найдите сумму чисел, больших заданного числа a .
3. В заданном массиве замените нулем наибольший элемент.
4. Найдите полупроизведение всех положительных элементов массива.
5. Найдите сумму квадратов положительных элементов и количество нулевых в заданном целочисленном одномерном массиве.
6. В заданной вещественной последовательности поменяйте местами первый и наибольший элементы.
7. Дано n чисел. Замените все отрицательные числа их модулями.
8. Вычислите среднее арифметическое наибольшего и наименьшего из n чисел.

Вариант 8

1. В заданной последовательности целых чисел определите количество и сумму элементов, кратных 10.
2. Дано n чисел. Найдите сумму чисел, равных заданному числу a .
3. В заданном массиве замените нулем наименьший элемент.
4. Найдите полупроизведение всех отрицательных элементов массива.
5. Найдите сумму квадратов отрицательных элементов и количество положительных чисел в заданном целочисленном одномерном массиве.
6. В заданной вещественной последовательности поменяйте местами первый и наибольший элементы.
7. Дано n чисел. Замените все отрицательные числа их модулями.
8. Вычислите среднее арифметическое наибольшего и наименьшего из n чисел.

Вариант 9

1. В заданной последовательности целых чисел определите количество и сумму элементов, кратных 12.
2. Дано n чисел. Найдите сумму чисел, больших заданного числа a .
3. В заданном массиве замените нулем наибольший элемент.
4. Найдите полупроизведение всех положительных элементов массива.
5. Найдите сумму квадратов неотрицательных элементов и количество положительных чисел в заданном целочисленном одномерном массиве.
6. В заданной вещественной последовательности поменяйте местами первый и наименьший элементы.
7. Дано n чисел. Замените все отрицательные числа их модулями.
8. Вычислите среднее арифметическое наибольшего и наименьшего из n чисел.

Вариант 10

1. В заданной последовательности целых чисел определите количество и сумму элементов, кратных 9.
2. Дано n чисел. Найдите сумму чисел, меньших заданного числа a .
3. В заданном массиве замените нулем наименьший элемент.
4. Найдите полупроизведение всех отрицательных элементов массива.
5. Найдите сумму квадратов отрицательных элементов и количество положительных чисел в заданном целочисленном одномерном массиве.
6. В заданной вещественной последовательности поменяйте местами последний и наибольший элементы.
7. Дано n чисел. Замените все отрицательные числа их модулями.
8. Вычислите среднее геометрическое наибольшего и наименьшего из n чисел.

Вариант 11

1. В заданной последовательности целых чисел определите количество и сумму элементов, кратных 15.
2. Дано n чисел. Найдите сумму чисел, больших заданного числа a .
3. В заданном массиве замените нулем наибольший элемент.
4. Найдите полупроизведение всех положительных элементов массива.
5. Найдите сумму квадратов положительных элементов и количество нулевых в заданном целочисленном одномерном массиве.
6. В заданной вещественной последовательности поменяйте местами первый и наибольший элементы.
7. Дано n чисел. Замените все отрицательные числа их модулями.
8. Вычислите среднее арифметическое наибольшего и наименьшего из n чисел.

Вариант 12

1. В заданной последовательности целых чисел определите количество и сумму элементов, кратных 10.
2. Дано n чисел. Найдите сумму чисел, равных заданному числу a .
3. В заданном массиве замените нулем наименьший элемент.
4. Найдите полупроизведение всех отрицательных элементов массива.
5. Найдите сумму квадратов отрицательных элементов и количество положительных чисел в заданном целочисленном одномерном массиве.
6. В заданной вещественной последовательности поменяйте местами первый и наибольший элементы.
7. Дано n чисел. Замените все отрицательные числа их модулями.
8. Вычислите среднее арифметическое наибольшего и наименьшего из n чисел.

Методика анализа результатов, полученных в ходе практической работы

1. В результате выполнения практических заданий у Вас должны быть работающие программы и оформлен отчет по каждому заданию.

Порядок выполнения отчета по практической работе

1. Составить блок-схему решения индивидуальных задач из заданий по своему варианту.
2. Разработать программы в среде ABC Pascal.
3. Составить систему тестов и проверить работу программ на всех возможных значениях исходных данных.
4. Результаты выполнения практической работы оформить в виде отчета.

! Обработка двумерных массивов !

Учебная цель: Отработать стандартные алгоритмы обработки двумерных массивов.

Организационная форма занятия: практическое занятие в форме практической подготовки

Ход практической работы:

1. Повторить теоретический материал по теме практической работы.
2. Ответить на вопросы для закрепления теоретического материала.
3. Разработать программы для обработки одномерных массивов
4. Оформить отчет.

Краткие теоретические и учебно-методические материалы по теме практической работы

Двумерный массив - это структурированный тип данных в виде прямоугольной таблицы. Положение элементов в двумерных массивах описывается двумя индексами.

Например, двумерный массив размером 3*3, то есть в нем будет три строки, а в каждой строке по три элемента:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Каждый элемент имеет свой номер, как у одномерных массивов, но сейчас номер уже состоит из двух чисел – номера строки, в которой находится элемент, и номера столбца. Таким образом, номер элемента определяется пересечением строки и столбца.

A[2,1] - это элемент, стоящий во второй строке и в первом столбце.

Описание двумерного массива (матрицы):

Var A: array[1..100,1..100] of integer;

Ввод двумерного массива

Для последовательного ввода элементов двумерного массива используется цикл for, в котором нужно последовательно изменять номер строки с 1-й до последней и в каждой строке перебирать элементы столбцов с 1-го до последнего.

Рассмотрим пример ввода двумерного массива Паскаля с клавиатуры:

var

 a, : array [1..5, 1..10] of integer;

 i, j: integer; { индексы массива }

begin

 for i :=1 to 5 do {цикл для перебора всех строк}

 for j :=1 to 10 do {перебор всех элементов строки по столбцам}

 readln (a [i , j]); {ввод с клавиатуры элемента, стоящего в i -й строке и j -м столбце}

Двумерный массив можно заполнить случайным образом, т.е. использовать функцию random (N), а также присвоить каждому элементу матрицы значение некоторого выражения. Способ заполнения двумерного массива выбирается в зависимости от поставленной задачи, но в любом случае должен быть определен каждый элемент в каждой строке и каждом столбце.

Вывод двумерного массива Паскаля на экран

Вывод элементов двумерного массива также осуществляется последовательно, необходимо напечатать элементы каждой строки и каждого столбца. При этом нужно, чтобы элементы, стоящие в одной строке, печатались рядом, т.е. в строку, а элементы столбца располагались один под другим. Для этого необходимо выполнить следующую последовательность

действий (рассмотрим фрагмент программы для массива, описанного в предыдущем примере):

```
for i :=1 to 5 do {цикл для перебора всех строк}
begin
  for j :=1 to 10 do {перебор всех элементов строки по столбцам}
    write ( a [ i , j ]:4); {печать элементов, стоящих в i -й строке матрицы в одной экранной строке, при этом для вывода каждого элемента отводится 4 позиции}
    writeln ; {прежде, чем сменить номер строки в матрице, нужно перевести курсор на начало новой экранной строки}
  end ;
```

Типовые примеры на двумерные массивы.

Пример 1. Чтобы заполнить двумерный массив (матрицу) случайными числами, введите программу:

```
Const n=6; m=7;
Var I,j: integer;
    a: array[1..n, 1..m] of integer;
begin
  // заполнение массива
  Randomize;
  for i:=1 to n do begin
    for j:=1 to m do begin
      a[i,j]:=random(10);
      write(a[i,j]:5);
    end;
    writeln;
  end;
end.
```

Пример 2. Решите задачу поиска max числа в массиве и его положения. Const n=6; m=7;

```
Var I,j: integer;
    a: array[1..n, 1..m] of integer;
begin
  // заполнение массива
  Randomize;
  for i:=1 to n do begin
    for j:=1 to m do begin
      a[i,j]:=random(10);
      write(a[i,j]:5);
    end;
    writeln;
  end;
  // нахождение max
  Max:=a[1,1];
  for i:=1 to n do
    for j:=1 to m do
      if a[i,j]>max then begin max:=a[i,j]; x:=I; y:=j; end;
  // вывод результатов
  Writeln ('max=', max);
  Writeln (' строка=' , x, ' столбец=', y);
End.
```

Пример 3. В двумерном массиве заменить все отрицательные элементы на ноль.

```
Const n=10; m=10;
Var I,j: integer;
a: array[1..n, 1..m] of integer;
begin
// заполнение массива
Randomize;
for i:=1 to n do begin
for j:=1 to m do begin
a[i,j]:=random(20)-10;
write(a[i,j]:5);
end;
writeln;
end;
// нахождение отрицательных и замена
for i:=1 to n do
for j:=1 to m do
if a[i,j]<0 then a[i,j]:=0;
// вывод результатов
for i:=1 to m do begin
for j:=1 to n do begin
write(a[i,j]:5);
end;
writeln; end; end.
```

Задания для практического занятия:

1. Рассмотрите примеры и проверьте их работу на ПК.
2. Решите самостоятельно задачу поиска *max* числа в массиве и его положения.
3. Решите самостоятельно задачу замены элементов больших 5 на единицы.
4. Написать программы для решения задач в соответствии с вариантом.
5. Составить отчет

Инструкция по выполнению практической работы

1. Рассмотрите примеры из теоретической части.
2. Выполнить примеры.
3. Выполните задания для самостоятельной работы.

Индивидуальные задания

Вариант 1

1. Составьте программу для ввода с клавиатуры элементов двумерного массива и определения количества элементов, которые превосходят заданное число.
2. Составьте программу для ввода с клавиатуры элементов матрицы 3*3 и вычисления суммы элементов каждого столбца.
3. Составьте программу для ввода с клавиатуры элементов двумерного массива, нахождения всех положительных элементов и распечатки их на экране ЭВМ.
4. Составьте программу для ввода с клавиатуры элементов двумерного массива и получения для массива средней арифметической.

Вариант 2

1. Составьте программу для ввода с клавиатуры элементов двумерного массива и определения количества элементов, которые не превосходят заданное число.
2. Составьте программу для ввода с клавиатуры элементов матрицы 4*4 и вычисления суммы элементов каждой строки.

3. Составьте программу для ввода с клавиатуры элементов двумерного массива, нахождения всех отрицательных элементов и распечатки их на экране ЭВМ.
4. Составьте программу для ввода с клавиатуры элементов двумерного массива и получения для массива средней геометрической.

Вариант 3

1. Составьте программу для ввода с клавиатуры элементов двумерного массива и определения количества элементов, которые равны заданному числу.
2. Составьте программу для ввода с клавиатуры элементов матрицы 3×3 и вычисления суммы элементов каждой строки.
3. Составьте программу для ввода с клавиатуры элементов двумерного массива, нахождения всех положительных элементов и распечатки их на экране ЭВМ.
4. Составьте программу для ввода с клавиатуры элементов двумерного массива и получения для массива средней арифметической.

Вариант 4

1. Составьте программу для ввода с клавиатуры элементов двумерного массива и определения количества элементов, которые превосходят заданное число.
2. Составьте программу для ввода с клавиатуры элементов матрицы 4×4 и вычисления суммы элементов каждого столбца.
3. Составьте программу для ввода с клавиатуры элементов двумерного массива, нахождения всех отрицательных элементов и распечатки их на экране ЭВМ.
4. Составьте программу для ввода с клавиатуры элементов двумерного массива и получения для массива средней геометрической.

Вариант 5

1. Составьте программу для ввода с клавиатуры элементов двумерного массива и определения количества элементов, которые не превосходят заданное число.
2. Составьте программу для ввода с клавиатуры элементов матрицы 3×3 и вычисления суммы элементов каждой строки.
3. Составьте программу для ввода с клавиатуры элементов двумерного массива, нахождения всех нулевых элементов и распечатки на экране ЭВМ их номеров.
4. Составьте программу для ввода с клавиатуры элементов двумерного массива и получения для массива суммы.

Методика анализа результатов, полученных в ходе практической работы

1. В результате выполнения практических заданий у Вас должны быть работающие программы и оформлен отчет по каждому заданию.

Порядок выполнения отчета по практической работе

1. Разработать программы в среде ABC Pascal.
2. Составить систему тестов и проверить работу программ на всех возможных значениях исходных данных.
3. Результаты выполнения практической работы оформить в виде отчета.

Практическое занятие №6 Работа со строковыми переменными.

Учебная цель: Отработать навыки работы со строковыми переменными.

Организационная форма занятия: практическое занятие в форме практической подготовки

Ход практической работы:

1. Повторить теоретический материал по теме практической работы.
2. Ответить на вопросы для закрепления теоретического материала.
3. Разработать программы для обработки строк
4. Оформить отчет.

Краткие теоретические и учебно-методические материалы по теме практической работы

Строка представляет собой особую форму одномерного массива символов, которая имеет существенное отличие. Массив символов имеет фиксированную длину (количество элементов), которая определяется при описании. Строка имеет две разновидности длины:

- общая длина строки, которая характеризует размер памяти, выделяемый строке при описании;
- текущая длина строки (всегда меньше или равна общей длине), которая показывает количество смысловых символов строки в каждый конкретный момент времени.

Строка в Паскале – упорядоченная последовательность символов. Количество символов в строке называется ее длиной. Длина строки в Паскале может лежать в диапазоне от 0 до 255. Каждый символ строковой величины занимает 1 байт памяти и имеет числовой код в соответствии с таблицей кодов ASCII.

Код ASCII (American Code for Information Interchange – Американский стандартный код для обмена информацией) имеет основной стандарт и его расширение. Основной стандарт использует шестнадцатеричные коды 00-7F, расширение стандарта – 80-FF. Основным стандартом является международным и используется для кодирования управляющих символов, цифр и букв латинского алфавита; в расширении стандарта используются символы псевдографики и буквы национальных алфавитов.

Встроенная функция **Length** позволяет определить фактическую длину текстовой строки, хранящейся в указанной переменной (а не величину предельного размера строки, установленную при декларации).

Строковая константа Паскаля – последовательность символов, заключенная в апострофы. Например, 'строковая константа', '243'. Два следующих друг за другом апострофа (‘’) обозначают пустую строку, т.е. строку с нулевой длиной.

Описание строковой переменной Паскаля

Для описания строковых переменных в Паскале существует предопределенный тип **string**.

В общем виде описание строковой переменной будет выглядеть следующим образом:

Пример описания строковой переменной в Паскале:

```
Var <имя_переменной>: string[<максимальная длина строки>]
```

Например:

```
Var
```

```
s1: string[10];
```

```
s2: string[20];
```

```
smax: string;
```

В приведенном выше описании строковая переменная s1 может содержать не более 10 символов, переменная s2 – не более 20 символов. Если же при описании строки ее

максимальная длина не указывается, то по умолчанию принимается максимально допустимая длина, равная 255 символам (переменная `smax`)..

Символы в строке упорядочены, каждый из них имеет порядковый номер, начиная с первого. Имеется возможность обратиться к любому элементу строки, указав его номер, так же как это делается в одномерных массивах. Например, `s1[2]` позволяет обратиться ко второму символу в строке `s1`, при этом мы можем поменять это значение, выполнив оператор присваивания `s1[2]:= 'r'`, можем вывести на экран это значение или присвоить его другой переменной.

Действия со строками в Паскале

Операция **слияния** (сцепления, конкатенации) применяется для соединения нескольких строк в одну, обозначается знаком «+». Операция слияния применима для любых строковых выражений, как констант, так и переменных. Функция **Concat** выполняет конкатенацию (или сцепление) строк `Str1, Str2, ..., StrN` в том порядке, в каком они указаны в списке параметров. Общее количество символов всех сцепленных строк не должно превышать 255. Операции **отношения** позволяют сравнивать строки на отношение равенства (`=`), неравенства (`<>`), больше (`>`), меньше (`<`), больше или равно (`>=`), меньше или равно (`<=`). В результате сравнения двух строк получается логическое значение (`true` или `false`). Сравнение строк производится слева направо посимвольно до первого несовпадающего символа, большей считается та строка, в которой первый несовпадающий символ имеет больший код в таблице кодировки. Если строки имеют различную длину, но в общей части символы совпадают, считается, что короткая строка меньше. Строки равны, если они имеют равную длину и соответствующие символы совпадают.

Пример действий со строками в Паскале:

'строка' <> 'строки' (верно, т.к. не совпадают последние символы);

'Abc' < 'abc' (отношение истинно, т.к. код символа 'A' равен 65 в десятичной системе счисления, а код символа 'a' – 97);

'год' > 'век' (отношение верно, т.к. буква 'г' в алфавите стоит после буквы 'в', а, следовательно, имеет больший код).

Стандартные функции для работы со строками в Паскале

Copy (S, poz, n) выделяет из строки `S`, начиная с позиции `poz`, подстроку из `n` символов. Здесь `S` – любое строковое выражение, `poz, n` – целочисленные выражения.

Значение S	Выражение	Результат
'строка символов'	<code>Copy(S,3,3)</code>	рок

Функция **Copy** позволяет копировать фрагмент некоторой строки из одной переменной в другую. Вызывая эту функцию, нужно указать следующие параметры:

- имя строки, из которой должен извлекаться копируемый фрагмент,
- позицию в строке, начиная с которой будет копироваться фрагмент,
- число копируемых символов.

Concat (s1, s2,...,sn) выполняет слияние строк `s1, s2,...,sn` в одну строку.

Выражение	Результат
<code>Concat('язык', ' ', 'Pascal')</code>	'язык Pascal'

Length(S) определяет текущую длину строкового выражения `S`. Результат – значение целого типа.

Значение S	Выражение	Результат
'(a+b)*c'	<code>Length(s)</code>	7

Pos(subS, S) определяет позицию первого вхождения подстроки `subS` в строку `S`. Результат – целое число, равное номеру позиции, где находится первый символ искомой подстроки. Если вхождение подстроки не обнаружено, то результат функции будет равен 0. С помощью функции **Pos** можно осуществить поиск некоторого фрагмента в строке. Если заданный фрагмент в строке присутствует, то функция возвращает номер позиции, с которой он начинается. Если фрагмент не найден, то функция возвращает нуль.

Значение S	Выражение	Результат
------------	-----------	-----------

‘предложение’	Pos(‘e’, S)	3
‘предложение’	Pos(‘a’, S)	0

Стандартные процедуры для работы со строками в Паскале

Delete (S, poz, n) удаляет из строки S, начиная с позиции poz, подстроку из n символов. Здесь S – строковая переменная (в данном случае нельзя записать никакое другое строковое выражение, кроме имени строковой переменной, т.к. только с именем переменной связана область памяти, куда будет помещен результат выполнения процедуры); poz, n – любые целочисленные выражения.

Исходное значение S	Оператор процедуры	Конечное значение S
‘abcdefg’	Delete(s, 2, 3)	‘aefg’

Insert(subS, S, poz) вставляет в строку S, начиная с позиции poz, подстроку subS. Здесь subS – любое строковое выражение, S – строковая переменная (именно ей будет присвоен результат выполнения процедуры), poz – целочисленное выражение.

Исходное значение S	Оператор процедуры	Конечное значение S
‘рис. 2’	Insert(‘№’, S, 6)	‘рис. №2’

Процедуры преобразования типов в Паскале

Str(x, S) преобразует число x в строковый формат. Здесь x – любое числовое выражение, S – строковая переменная. В процедуре есть возможность задавать формат числа x. Например, str(x: 8: 3, S), где 8 – общее число знаков в числе x, а 3 – число знаков после запятой.

Оператор процедуры	Значение S
Str (sin(1):6:4, S)	‘0.0175’
Str (3456, S)	‘3456’

Val(S, x, kod) преобразует строку символов S в число x. Здесь S – строковое выражение, x – числовая переменная (именно туда будет помещен результат), kod – целочисленная переменная (типа integer), которая равна номеру позиции в строке S, начиная с которой произошла ошибка преобразования, если преобразование прошло без ошибок, то переменная kod равна 0.

Тип X	Оператор процедуры	Значение X	Значение kod
Real	Val(‘12.34’, x, kod)	12.34	0
Integer	Val(‘12.34’, x, kod)	12	3

Функция Uppcase позволяет преобразовывать символ любой литеры из строчного в прописной. Эта функция рассчитана на обработку отдельного символа, поэтому для обработки строки символов с помощью этой функции приходится организовывать цикл.

Практическая часть

Пример 1. Определение длины строки.

```
Program DemoFunctionLength;
Var
Word : string;
Begin
write ('Введите слово :');
readln(Word);
writeln('Это слово состоит из ',Length (Word),' букв');
End.
```

Пример 2. Преобразование строчных букв в прописные.

```
Program DemoFunctionUppcase;
Var
Word : string;
i : Byte;
```

```

Begin
Word := 'фирма Microsoft';
for i := 1 to Length (Word) do
Word[i] := UpCase (Word[i]);
writeln(Word); {выводится текст 'фирма MICROSOFT'}
End.

```

Пример 3. Копирование фрагмента текста.

```

Program DemoFunctionCopy;
Var
Word : string;
Word1 : string[20];
Begin
Word := 'фирма Microsoft';
writeln(Word); {выводится текст 'фирма Microsoft'}
Word1 := Copy (Word,1,5);
writeln(Word1); {выводится текст 'фирма'}
End.

```

Пример 4. Определение позиции символа, входящего в строку.

```

Program DemoFunctionPos;
Var
Word : string;
SearchWord : string[20];
Position : Byte;
Begin
Word := 'фирма Microsoft';
writeln(Word); {выводится текст 'фирма Microsoft'}
writeln ('Введите искомый текст ');
readln (SearchWord);
Position := Pos(SearchWord, Word);
if Position <> 0 then
begin
write ('Фрагмент <',SearchWord,> содержится в строке <',Word);
writeln ('>, начиная с позиции ',Position );
end
else
writeln('Фрагмент <',SearchWord,> не содержится в строке <',Word, '>');
End.

```

Пример 5. Объединение двух строк.

```

Program DemoFunctionConcat;
Var
Word : string;
Word1, Word2 : string[20];
Begin
Word1 := 'фирмы ';
Word2 := 'Microsoft';
Word := Concat('Компьютеры ',Word1,Word2);
writeln(Word); {выводится текст 'Компьютеры фирмы Microsoft'}
End.

```

Пример: Из набора 10 слов, длина каждого из которых 5 символов, напечатать все слова, отличные от слова "hello".

```

program r2;
typesl=array [1..5] of char;

```

```

msl=array [1..10] of sl;
var
s:msl;
i:integer;
begin
writeln ('введите 10 слов по 5 символов ');
for i:=1 to 10 do
read (s[i]);
writeln;
for i:=1 to 10 do
if s[i]<>'hello' then write (s[i], ' ');
writeln;
end.

```

Тип данных STRING

В Турбо Паскаль предусмотрен тип данных STRING. Переменная типа STRING, в отличие от обычного массива литер, может принимать значения переменной длины. Максимально возможная длина переменной 255 символов. Например:

```
L:STRING[200]; cit:STRING[30]; ouw:STRING[10];
```

В скобках указывается максимальная длина для данной переменной. Для ввода значений типа STRING необходимо использовать READLN, а не READ. За один раз может быть введена только одна строка. Две строки можно сравнивать, используя операции отношения (сначала сравниваются самые левые символы, если они равны, то сравниваются следующие и т.д.). Пример: Распечатать по алфавиту k слов.

```

program In1;
const k=5;
var:
c: integer;
s: 1..k;
ln: array [1..k] of string[20];
begin
for c:=1 to k do readln (ln[c]);
writeln;
for c:=1 to k do begin
s:=1;
for:=2 to k do
If (ln[c]<ln[s]) and (ln[c]<>' ') then s:=c;
Writeln (ln[s]);
Ln[s]:=' ';
End;
end.

```

Инструкция по выполнению практической работы

1. Рассмотрите примеры из теоретической части.
2. Выполнить примеры.
3. Выполните задания для самостоятельной работы.

Индивидуальные задания

№ варианта	Задание
1.	1. Подсчитать, сколько раз в данной строке встречается буква Ch, вводимая с клавиатуры. 2. Подсчитать длину первого слова в строке.

	Раздвинуть заданный текст, вставив заданную последовательность после i-го символа каждого слова.
2.	1. Из заданной строки удалить среднюю букву, если длина строки нечетная, иначе — удалить две средние буквы. 2. Из заданной строки получить новую, заменив в ней все «7» на символ «o». Распечатать полученную строку. 3. В заданном тексте слова разделены пробелами или запятыми. Напечатать список слов, начинающихся с символа, введенного с клавиатуры.
3.	1. Заменить в заданной строке все буквы Ch1 на Ch2 (их значения вводить с клавиатуры). 2. Проверить, является ли заданное слово палиндромом (Например, слова палиндромы: КАЗАК, ШАЛАШ, МАДАМ). 3. Определить наличие слов в заданном тексте содержащих сочетание символов, задаваемое с клавиатуры.
4.	1. Заменить все вхождения подстроки Str1 на подстроку Str2, которые вводятся с клавиатуры. 2. В строке заменить все фразы "что и требовалось доказать " на "ч. т. д.", тем самым сжав ее. 3. Из заданной последовательности слов удалить слова, содержащие числа.
5.	1. В заданной строке после каждой буквы Ch вставить строку Str1. 2. В строке заменить все пробелы, стоящие на четных позициях, на двойные. 3. Подсчитать количество слов, разделенных пробелами, содержащих k гласных букв (k задается с клавиатуры).
6.	1. В заданной строке удвоить каждое вхождение буквы Ch. 2. Подсчитать, сколько слов в заданной строке заканчивается буквой «я», если известно, что после слова обязательно стоит либо точка, либо пробел, либо запятая. 3. Выбрать из заданного текста все слова, которые имеют в своем составе по 2 и более одинаковых символа.
7.	1. В заданной строке удвоить каждое вхождение буквы Ch. 2. Дана строка символов. Удалить из нее слова, начинающиеся с заданной буквы. 3. Вывести на печать список слов, имеющих приставку (несколько букв), задаваемую с клавиатуры.
8.	1. Дана последовательность слов. Напечатать все слова, отличные от слова «hello». 2. Дана строка символов. Найти слово, которое имеет четную длину и начинается с заданного символа. 3. Каждое слово текста преобразовать таким образом, чтобы оно читалось слева направо.
9.	1. Дана последовательность слов. Напечатать все слова в алфавитном порядке. 2. Из заданной строки получить новую, заменив в ней все группы букв «абсд» на «abc». 3. Список фамилий в произвольном порядке через запятую задается с экрана. Упорядочить фамилии по алфавиту.
10.	1. Дана последовательность слов. Напечатать все слова последовательности, которые встречаются в ней по одному разу. 2. Определить, сколько слов в заданной строке начинается на букву "a". 3. Распечатать столбцами слова из текста заканчивающиеся на согласную букву и на гласную букву.
11.	1. Дано предложение. Напечатать все различные слова. 2. Какой процент слов в строке начинается на букву " k ".

	3. Из заданного текста распечатать столбцами отдельно гласные и согласные буквы каждого слова.
12.	1. Дана последовательность слов. Напечатать все слова, предварительно преобразовав каждое из них по следующему правилу: удалить из слова все предыдущие вхождения последней буквы. 2. Вводится строка. Определить, сколько слов начинается с той же буквы, с которой начинается первое слово. 3. Из произвольного текста распечатать слова, повторяющиеся в порядке убывания частоты повторения.

Методика анализа результатов, полученных в ходе практической работы

1. В результате выполнения практических заданий у Вас должны быть работающие программы и оформлен отчет по каждому заданию.

Порядок выполнения отчета по практической работе

1. Разработать программы в среде ABC Pascal.
2. Составить систему тестов и проверить работу программ на всех возможных значениях исходных данных.
3. Результаты выполнения практической работы оформить в виде отчета.

Практическое занятие №7. Работа с данными типа множество.

Учебная цель: отработать навыки работы с типом данных - множество

Учебные задачи:

1. Закрепить теоретические знания о понятии «множество»
2. Овладение навыками алгоритмизации структур, содержащих множества
3. Научиться использовать возможности языка программирования для обработки множеств
4. Закрепить практические навыки по составлению тестов и отладке программы.

Организационная форма занятия: практическое занятие в форме практической подготовки

Задачи практической работы:

1. Повторить теоретический материал по теме практической работы.
2. Ответить на вопросы для закрепления теоретического материала.
3. Разработать программы для обработки одномерных массивов
4. Оформить отчет.

Краткие теоретические и учебно-методические материалы по теме практической работы

Наряду с числом множество является фундаментальным математическим понятием.

Паскаль - один из немногих алгоритмических языков, который имеет встроенные средства для работы с множествами.

В математике под множеством понимается некоторый набор элементов. Например, множество фигур на плоскости (прямоугольник, круг, ромб, квадрат). В математике рассматриваются конечные и бесконечные множества, состоящие из произвольных элементов. В Паскале множества всегда конечные, причем состоят из небольшого числа элементов (в Турбо Паскале - до 255).

Постоянные множества и в математике и в Паскале задаются перечнем их элементов.

Математика	Паскаль
{1,2,3}	[1,2,3]
{'A','K','B','L'}	['A','K','B','L']
∅	пустое множество []
{1,2,...,N}	[1..N]

В квадратных скобках могут находиться не только константы, но любые выражения типа элементов множества, например, $[2++x, 8-3]$.

К множествам применимы следующие операции.

объединение $C = A \cup B$

пересечение $C = A \cap B$

разность $C = A \setminus B$

Под множеством в языке паскаль понимают ограниченный, неупорядоченный набор различных элементов одинакового типа. Можно, например говорить о множестве радиодеталей, транспортных средств, станков и т.д. всему множеству в целом дается имя. Каждый объект множества называется элементом множества. Тип элементов, входящих в множество, называется базовым. В качестве базового типа можно использовать простые типы: стандартный (кроме действительного), перечисляемый и ограниченный.

Базовый тип задается диапазоном или перечислением. Область значений типа множество - набор всевозможных подмножеств, составленных из элементов базового типа. Если базовый тип принимает n значений, то тип множество для него будет иметь 2^n вариантов значений. В выражениях на языке Pascal значения элементов множества указываются в квадратных скобках: $[1, 2, 3, 4]$, $['a', 'b', 'c', 'd']$, $['a'...'z']$.

Множества должны быть объявлены в var или type.

var <имя_ множества>: set of <базовый элемент>;

Например: var god: set of 1880..2000; c: set of char;

type <имя типа>= set of <базовый элемент>;

var <имя множества>: <имя типа>;

set of 'a'..'z' - множество прописных английских букв;

set of 1..100 - множество целых чисел от 1 до 100;

set of (winter, spring, summer, autumn) - множество времен года

set of char - множество всех символов.

В языке паскаль имеются следующие операции над множествами:

+ - объединение множеств;

* - пересечение множеств;

- - вычитание множеств;

=, <> - проверка множеств на равенство, неравенство;

<=, >= - проверка множеств на включение;

in - проверка на принадлежность элемента множеству (c in a).

Операция “равно” (=). Два множества A и B считаются равными, если они состоят из одних и тех же элементов. Порядок следования элементов в сравниваемых множествах значения не имеет.

Значение A	Значение B	Выражение	Результат
[1, 2, 3, 4]	[1, 2, 3, 4]	$A = B$	True
['a'..'2']	['b'..'2']	$A = B$	False
['a'..'2']	['2'..'a']	$A = B$	True

Операция “не равно” (<>). Два множества A и B считаются не равными, если они отличаются по мощности или по значению хотя бы одного элемента.

Значение A	Значение B	Выражение	Результат
[1, 2, 3,]	[3, 1, 2, 4]	$A <> B$	True
['a'..'2']	['6'..'2']	$A <> B$	True

Операция “больше или равно” ($\geq$). Используется для определения принадлежности множества. Результат операции $A \geq B$ равен *True*, если все элементы множества B содержатся во множестве A . В противном случае результат равен *False*.

Значение А	Значение В	Выражение	Результат
[1, 2, 3, 4]	[2, 3, 4]	$A \geq B$	<i>True</i>
['a'..'2']	['b'..'t']	$A \geq B$	<i>True</i>
['z','x','c']	['c','x']	$A \geq B$	<i>True</i>

Операция “меньше или равно” ($\leq$). Используется аналогично предыдущей операции. Результат выражения $A \leq B$ равен *True*, если все элементы множества A содержатся во множестве B . В противном случае результат равен *False*.

Значение А	Значение В	Выражение	Результат
[1, 2, 3,]	[1, 2, 3, 4]	$A \leq B$	<i>True</i>
['a'..'h']	['2'..'a']	$A \leq B$	<i>True</i>
['a'..'v']	['a','n','v']	$A \leq B$	<i>True</i>

Операция *In*. Используется для проверки принадлежности какого-либо значения указанному множеству. Обычно применяется в условных операциях.

Значение А	Выражение	Результат
2	<i>if A in [1, 2, 3] then...</i>	<i>True</i>
'v'	<i>if A in ['a'..'n'] then...</i>	<i>False</i>
x1	<i>if A in [x0, x1, x2, x3] then...</i>	<i>True</i>

Все значения множества представляются в памяти последовательностями битов одинаковой длины. За каждое значение базового типа "отвечает" один бит. Если множество содержит некоторый элемент, в "ответственном" за него бите хранится 1, если не содержит - хранится 0.

Задача №1. Имеются три множества символьного типа, которые заданы своими конструкторами: $Y1=['A','B','D','R','M']; Y2=['R','A','H','D']; Y3=['A','R'];$ Сформировать новое множество: $X=(Y1*Y2)+(Y1\setminus Y2)$. Вывести на печать полученное множество x , проверить, включено ли $y3$ во множество x . (файл MNOG1.PAS)

Решение.

```

program mnog1;
var y1,y2,y3,x: set of char;
c:char;
begin
y1:=['a','b','d','r','m'];
y2:=['r','a','h','d'];
y3:=['a','r'];
x:=(y1*y2)+(y1-y2);
write('множество x=');
for c:= 'a' to 'r' do
if c in x then write(c);
writeln;
if y3<=x then write('y3 включено в x')
else write('y3 не включено в x');
end.

```

Задача №2. Из множества целых чисел 1..20 выделить: множество чисел, делящихся без остатка на 6; множество чисел, делящихся без остатка или на 2, или на 3. (файл MNOG2.PAS).

Решение.

```
program mnog2;
const n=20;
var n2,n3,n6,n23: SET OF 1..N;
k:1..N;
begin
  n2:=[]; n3:=[];
  for k:=2 to n do
  begin
    if (k mod 2)=0 then n2:=n2+[k];
    if (k mod 3)=0 then n3:=n3+[k];
  end;
  n6:=n2*n3;
  n23:=n2+n3;
  writeln('на 6 делятся числа:');
  for k:=1 to n do
    if k in n6 then write(k:3);
  writeln;
  writeln('на 2 или на 3 делятся числа:');
  for k:=1 to n do
    if k in n23 then write(k:3);
  end.
```

Задача №3. Дан текст (например, 5a7233111bge2257cde.) Признаком конца текста является точка. Вывести на экран цифры, которые встречаются в тексте. (файл MNOG3.PAS).

Решение.

```
program mnog3;
var z: SET OF 0..9;
k,i:integer;
si:char;
begin
  readln;
  write('=>');
  z:=[];
  repeat
    read(si);
    k:=ord(si)-ord('0');
    if k in [0..9] then z:=z+[k]
  until si='.';
  if z=[] then writeln ('В тексте нет цифр')
  else begin
    write('В тексте имеются следующие цифры:');
    for i:=1 to 9 do
      if i in z then write (i:2);
    writeln;
  end;
end.
```

Задания для практического занятия:

1. Найдите ответ на вопрос "Какие структурированные типы данных существуют в Паскале?".
2. Найдите ответ на вопрос "Что понимается под множеством в Паскале?".
3. Найдите ответ на вопрос "Как объявляются множества?".
4. Какие типы данных используются в качестве базовых при построении множественных типов.
5. Найдите ответ на вопрос "Как в Паскале обозначаются операции над множествами?".
6. Какие операции определены над переменными множественного типа и каков их приоритет?
7. Чем похожи и чем отличаются множества и массивы,
8. Как представляются множества в памяти.
 - a) Вычислить следующие выражения:
 - a) $[5] \leq [1..5]$
 - b) $['a'..'d', 'k'..'m'] + ['d'..'k'];$
 - c) $[Москва, Минск, Сочи] * [Сочи]$
 - d) $[7, 1, 3..6] = [1..7]$
 - e) $[';', '9', '0', ':'] - [';', ':']$
 - f) $15 \text{ in } [1..10]$
 - g) $['A'..'D', 'K'..'M'] + ['D'..'K'];$
 - h) $[Jan, Feb, Mar] * [Mar];$
 - i) $[2, 1, 3..6] = [1..7];$
 - j) $15 \text{ In } [1..10];$
 - k) $[';', '(', ')', ':'] - [';', ':']$.
9. Дано описание переменной множественного типа: Var Pm: Set of (Red, Grey, Blue, Black). Выписать все допустимые значения этой переменной.
10. Будут ли равны множества:
 - a. $['A'..'D']$ и $['A', 'B', 'C', 'D'];$
 - b. $[White, Black]$ и $[Black, White].$
11. Вычислить выражение: $[1..14] * [5, 12..60] + [4..7] - [2 * 16] * [6].$
12. Упростить данное выражение множественного типа:
 $[11..17] * [2] + [7, 17..40] * [2..17] - [2..8];$
13. Познакомиться с условием задачи № 1.
14. Вызвать с диска файл (mnog1.pas). Запустить программу на выполнение. Записать программу в тетрадь. Ответ примера № 1 записать в тетрадь.
15. Найдите ответ на вопрос "Как организована в программе проверка на включение множества Y3 в множество X?".
16. Познакомиться с условием задачи № 2.
17. Вызвать с диска файл (mnog2.pas). Запустить программу на выполнение. Записать программу в тетрадь. Ответ записать в тетрадь. Ответить на вопрос "Как в программе сформировано множество чисел, делящихся без остатка на 6; множество чисел, делящихся без остатка или на 2, или на 3?"
18. Познакомиться с условием задачи № 3.
19. Вызвать с диска файл (mnog3.pas). Запустить программу на выполнение. Записать программу в тетрадь. Ответ записать в тетрадь. Разобраться как работает данная программа.

Инструкция по выполнению практической работы

1. Рассмотрите примеры из теоретической части.
2. Выполнить примеры.
3. Выполните задания для самостоятельной работы.

Методика анализа результатов, полученных в ходе практической работы

1. В результате выполнения практических заданий у Вас должны отчет по каждому заданию.

Порядок выполнения отчета по практической работе

1. Разработать программы в среде ABC Pascal.
2. Составить систему тестов и проверить работу программ на всех возможных значениях исходных данных.
3. Результаты выполнения практической работы оформить в виде отчета.

Раздел 3. Процедуры и функции

Тема 3.1. Подпрограммы

Практическое занятие №8. Организация и использование процедур.

Цель: Научиться применять различные формы подпрограмм при написании программ.

Организационная форма занятия: практическое занятие в форме практической подготовки

Запустить интегрированную среду ABC Pascal.

Параметры-значения

В окне редактора наберите программу и сохраните файл под именем T051.PAS:

Var X,Y : Integer;	1
Procedure Reverse (A, B : Integer);	2
Begin	3
A := A * 10; B := B + 5;	4
WriteLn (B : 3, A : 3)	5
End;	6
Begin	7
X := 1; Y := 100;	8
(X, Y);	9
Reverse (4, 5);	10
Reverse (4 * X, 5 * Y);	11
End.	12

Начертите в тетради таблицу:

	Фактические параметры		Формальные параметры		Результат вывода на экран
	1-ый параметр	2-ой параметр	1-ый параметр	1-ый параметр	
	1	2	3	4	5
1					
2					
3					

Выполните программу. Обратите внимание на полученный результат.

Выполните программу еще раз, но в пошаговом режиме (клавиша F7). Задайте в окне отладчика просмотр значений переменных X, Y и A, B. Для удобства просмотра измените размер и положение окон (программы и Watches) так, чтобы они не накладывались друг на друга. Обратите внимание, как меняются значения этих переменных.

Какое значение имеют переменные A и B перед обращением к процедуре Reverse и после?

Ответ запишите в тетрадь, заполнив графы с 1 по 5.

Параметры-переменные

В окне редактора наберите программу и сохраните файл под именем T052.PAS:

Var I : Integer;	1
Procedure Count (X : Integer);	2
Begin	3
X := X + 1 ;	4
Write (' X = ' , X);	5
End;	6
Begin	7
I := 5;	8
Write (' I 1 = ' , I);	9

Count (I);	10
Write (' I 2 = ' , I);	11
End.	12

Начертите в тетради таблицу:

	Номер строки программы	Результат вывода на экран для процедуры	
		Count (X : Integer);	Count (Var X: Integer);
	1	2	3
1	9-я строка		
2	5-я строка		
3	11-я строка		

Выполните программу. Результат выполнения программы (какие сообщения выдаются на экран монитора) запишите во второй столбец таблицы

Внесите изменения в программу, заменив строку 2 на следующий текст:

Procedure Count (Var X : Integer);

Выполните программу. Результат выполнения программы (какие сообщения выдаются на экран монитора) запишите в третий столбец таблицы.

Какая разница между полученными результатами во втором и в третьем столбцах? Чем это можно объяснить? Запишите ответ в тетрадь.

Выполните программу еще раз, но в пошаговом режиме с прослеживанием работы процедуры Count (клавиша F7). Задайте для просмотра в окне отладчика переменные I и X. Обратите внимание, как меняются значения этих переменных. Почему получены такие результаты, и как это объяснить?

Локальные и глобальные переменные

В окне редактора наберите программу и сохраните файл под именем T053.PAS:

Var A, B : Integer;	1
Procedure ZB1;	2
Var A: Integer;	3
Begin	4
A := 10; B:= 15;	5
WriteLn ('A2 = ', A, 'B2 = ', B)	6
End;	7
Begin	8
A:= 18; B:=37;	9
WriteLn ('A1 = ',A, 'B1 = ',B);	10
ZB1;	11
WriteLn ('A3 = ',A, 'B3 = ',B)	12
End.	13

Начертите в тетради таблицу:

	Номер строки программы	Результат вывода на экран
	1	2
1	6-я строка	
2	10-я строка	
3	12-я строка	

Выполните программу в пошаговом режиме с прослеживанием работы процедуры ZB1 (клавиша F7).

Задайте для просмотра в окне отладчика переменные А и В. Обратите внимание, как меняются значения этих переменных.

1. Запишите в тетрадь результаты выполнения программы, заполнив таблицу. Почему получены такие результаты, и как это объяснить?

Вопросы для повторения

1. Что такое подпрограмма? Для чего нужна подпрограмма?
2. Какие виды подпрограмм есть в Turbo Pascal? Чем они отличаются друг от друга?
3. Что такое процедура? Ее общий вид.
4. Что называется параметром, и каково его назначение?
5. Правила записи параметров процедуры.
6. Формальные и фактические параметры. Каково отличие параметров-значений от параметров-переменных? Что может выступать в качестве параметров-значений и параметров-переменных?
7. Как вызвать процедуру в программе?
8. Опишите последовательность событий при вызове процедуры с параметрами-значениями.
9. Опишите последовательность событий при вызове процедуры с параметрами-переменными.
10. В чем различие между стандартными и определенными пользователем подпрограммами? Приведите примеры.
11. Чем отличаются локальные переменные от глобальных переменных?
12. Напишите, что выведет на экран следующая программа. Где в программе используется параметр-переменная, а где параметр-значение?

```
Var A,B:Integer;

Procedure TR54 ( Var X,Y:Integer );
Begin Y:=Y+5; X:=2*Y; End;

Procedure TR55(Y:Integer; Var X:Integer);
Begin Y :=Y + 5; X := 2 * Y; End;

Begin
A:=7;B:=-8;
TR54 (A,B); WriteLn( 'A = ', A, 'B = ', B);
TR55 (A,B); WriteLn(' A = ', A, 'B = ', B);
End.
```

13. Напишите, что выведет на экран следующая программа:

```
Var A,B:Integer;

Procedure TR56 (X:Integer; Var Y:Integer);
Begin
If X < 0 Then Y:= - X + 3
Else Y := X * 2
End;

Begin
A := - 2 ; B := 3;
```

TR56 (A + 3, B);	WriteLn ('A = ', A, 'B = ', B);
TR56 (1 - A, B);	WriteLn (' A = ', A, 'B = ', B);
TR56 (7, B);	WriteLn (' A = ', A, 'B = ', B) End.

14. Установите, что будет выведено на экран монитора в результате работы первого и второго вариантов программы. Сравните результаты и сделайте вывод.

<pre> Var X : Real; Procedure TR57; Var X : Real; Begin WriteLn (X) End; Begin X := Pi; TR57 End. </pre>	<pre> Var X : Real; Procedure TR58; Begin WriteLn (X) End; Begin X := Pi; TR58 End. </pre>
--	---

Задачи для решения (на занятии или дома)

Напишите программы с использованием процедур.

- Используя процедуру вычисления степени числа, вычислить значение выражения: $y = a_1 \cdot x^4 + a_2 \cdot x^3 + a_3 \cdot x^2 + a_4 \cdot x + a_5$
- Используя процедуру обмена местами двух значений, упорядочить значения трех переменных в порядке их возрастания.
- Дано натуральное число. Найти все его делители. Посчитать их количество.
- Даны два натуральных числа. Определить, является ли первое число перевертышем второго.

Практическое занятие №9. Организация и использование функций.

Цель работы:

- овладение практическими навыками организации работы в среде ABC PASCAL
- научиться разрабатывать программы с использованием процедур и функций
- приобрести практические навыки по организации подпрограмм

Оборудование:

- программная часть - интегрированная среда ABC Pascal
- задание на выполнение работы

Задания:

1. Используя подпрограммы-функции и написать программу вычисления:

1.	Даны действительные числа a, b, c . Получить: $\frac{\max(a, a+b) + \max(a, b+c)}{1 + \max(a+bc, b, 15)}$
2.	Даны действительные числа a, b . Получить u = min(a, b-a) , y = min(ab, a+b) , k = min(u+v², 3.14) .
3.	Даны действительные числа s, t . Получить: g(1.2,s)+g(t,s)-g(2s-1.5t) , g(ln(s,t+1))-g(t,s) , где $g(a,b) = \frac{a^2 + b^2}{a^2 + 2ab + 3b^2 + 4}$
4.	Даны действительные числа x, y . Получить: f(x,-2y,1.17)+f(2.2,x,x-y) , tg(f(x+y,xy,y-x)+f(3.1,1.4,y-sinx)) , где $f(a,b,c) = \frac{2a - b - \sin c}{5 + c }$
5.	Даны неотрицательные целые числа a, b . Найти F(a, b) , где $F(m,n) = \frac{n! - m!}{(n+m)!}$. (Определить вспомогательную функцию, вычисляющую факториал).
6.	Даны два натуральных числа a, b . Вычислить $\frac{a!! - ab}{a!! + ab}$. Функция x!! Определяется следующим образом: x!! = 1*3*5*...*x , если x нечетно, x!! = 2*4*6*...*x , если x четно.
7.	Даны действительные числа a, b, c . Получить $\frac{\min(a, a+b) + \min(a, b+c)}{1 + \min(a+bc, b)}$
8.	Даны действительные числа a, b . Получить r = max (a, b + a) , d = max (ab, a + b) , s = max (r + d², 3.14) .
9.	Вычислить $A_n^k = \frac{n!}{(n-k)!}$ и P_n = n! . Нахождение факториала числа оформите в виде функции.
10.	Определить количество счастливых билетов на отрезке от N до M (билет счастливый, если сумма первых трех его цифр равна сумме последних трех).
11.	Дано простое число. Составить функцию, которая будет находить следующее за ним простое число.
12.	Написать программу для вычисления суммы факториалов всех нечетных чисел от 1 до m .

1. Написать программу используя процедуры:

Вариант 1.	Задана окружность $(x-a)^2 + (y-b)^2 = R^2$ и точки $P(p_1, p_2)$, $F(f_1, f_2)$, $L(l_1, l_2)$. Выяснить и напечатать, сколько точек лежит внутри окружности. Проверку, лежит ли точка внутри окружности, оформить в виде процедуры.
2.	Используя процедуру вычисления степени числа, вычислить значение выражения: $y=a_1*x^4+a_2*x^3+a_3*x^2+a_4*x+a_5$
3.	Используя процедуру обмена местами двух значений, упорядочить значения трех переменных в порядке их возрастания.
4.	Дано натуральное число. Найти все его делители. Посчитать их количество используя процедуру.
5.	Даны два натуральных числа. Определить, является ли первое число перевертышем второго используя процедуру.
6.	Даны действительные числа S и T. Получить $G(1.2, s) + G(t, s) + G(2*s-l, s*t)$, где $G(A, B) = (A^2 + b^2)/(A^2 + 2*A*b + B^2 + 3*B^2 + 4)$.
7.	Даны действительные числа S, T. Получить $H(s, t) + \max(H(s-t, s*t)^2, H(s-t, s+t)^4) + H(l, l)$, где $H(A, B) = A/(1+A^2) + B/(1+B^2) - (A^2-B^2)$.
8.	Треугольник задан координатами своих вершин. Составить программу для вычисления его площади с использованием подпрограммы
9.	Четыре точки заданы своими координатами $X(x_1, x_2)$, $Y(y_1, y_2)$, $Z(z_1, z_2)$, $P(p_1, p_2)$. Выяснить, какие из них находятся на максимальном расстоянии друг от друга и вывести на печать значение этого расстояния. Вычисление расстояния между двумя точками оформить в виде процедуры.
10.	Задана окружность $(x-a)^2 + (y-b)^2 = R^2$ и точки $P(p_1, p_2)$, $F(f_1, f_2)$, $L(l_1, l_2)$. Выяснить и напечатать, сколько точек лежит внутри окружности. Проверку, лежит ли точка внутри окружности, оформить в виде процедуры.
11.	Четыре точки заданы своими координатами $X(x_1, x_2, x_3)$, $Y(y_1, y_2, y_3)$, $Z(z_1, z_2, z_3)$, $T(t_1, t_2, t_3)$. Выяснить, какие из них находятся на минимальном расстоянии друг от друга и вывести на печать значение этого расстояния. Вычисление расстояния между двумя точками оформить в виде процедуры.
12.	Число является автоморфным, то есть квадрат этого числа заканчивается этим же числом, например, число 6, так как его квадрат – 36 заканчивается на 6 или число 25 – его полный квадрат 625. Найти все автоморфные числа из промежутка от A до B.

Контрольные вопросы:

- 1) В чем разница между процедурами и функциями?
- 2) Описание процедур и функций в программе.
- 3) Обращения к процедурам и функциям.
- 4) Локальные и глобальные переменные языка Паскаль.
- 5) Формальные и фактические параметры.

Организация функций часть 2.

Цель работы: 1) формировать навыки составления программ, используя процедуры и функции;

2) совершенствовать навыки работы в системе Турбо Паскаль.

Организационная форма занятия: практическое занятие в форме практической подготовки

Ход выполнения работы

1. Приступая к разработке программы с функциями, повторите ОК.

ФУНКЦИИ В ПАСКАЛЕ

Повторяющиеся фрагменты программы оформляются в языке программирования в подпрограмме специальным образом. Подпрограмме дается имя, по которому к ней

можно обращаться (вызывать подпрограмму). Использование подпрограмм не только улучшает структуру и внешний вид программы, но и уменьшает вероятность ошибок и облегчает отладку.

В Паскале имеется два вида подпрограмм — процедуры и функции. Их структура очень похожа на структуру основной программы.

Описание ФУНКЦИИ

Заголовок функции состоит из слова Function, за которым указывается имя функции, затем в круглых скобках записывается список формальных параметров, далее ставится двоеточие и указывается тип результата функции.

В теле функции обязательно должен быть хотя бы один оператор присваивания, в левой части которого стоит имя функции, а в правой — ее значение. Иначе значение функции не будет определено.

Общий вид описания функции следующий: Function Имя[(Список формальных параметров)]: Тип результата;

Begin

Тело функции, в котором обязательно должно быть присваивание

Значение:= Имя функции

End;

2. Разобрать и проанализировать пример использования функции в программе.

Задача 1

Составить программу, подсчитывающую число сочетаний без повторения из p элементов по k .

Число сочетаний без повторения вычисляется по формуле:

Решение

Обозначим через n и k переменные для хранения введенных чисел; C — переменную для хранения результата.

Чтобы подсчитать количество сочетаний без повторения, необходимо вычислить $p!$, $(p-k)!$, $k!$

I этап. Опишем функцию для вычисления факториала числа p ($p! = 1 * 2 * \dots * p$).

Function factorial(n:Integer):Longint; {заголовок функции}

Var {описательная часть}

i: Integer;

rez: Longint;

Begin {тело функции}

rez:=1;

For i:=1 To n Do rez:=rez*i;

factorial:=rez; {присваивание значения имени функции}

End;

Первая строчка в описании функции — это ее заголовок. Служебное слово Function (функция) указывает на то, что именем factorial названа функция. В скобках записан список формальных параметров функции, состоящий из одной переменной целого типа. Далее в заголовке указан тип значения функции. В данном примере результат функции factorial — длинное целое число.

За заголовком функции следует описательная часть функции, которая, как и у программы, может состоять из разделов описания переменных, констант, типов. В данном примере имеется только раздел описания переменных. В нем описаны переменные i (счетчик цикла) и rez (для накопления значения факториала).

Далее идет раздел операторов (тело функции). Результат присваивается имени функции, таким образом, функция получает свое значение.

II этап. Составим программу нашего примера с вызовом функции.

В тексте программы описания функций всегда следуют за разделом описания переменных и до начала основной части, как и описания процедур. После того как функция описана, ее можно использовать в программе.

```
Program Example;
Var n,k: Integer;
    a1, a2, a3, c: Longint;
Function factorial(n:Integer):Longint;
Var i: Integer;
rez: Longint;
Begin rez:=1;
For i:=1 To n Do rez:=rez*i;
factorial :=rez;
End;
Begin
writeln ('Ввод n и k:');
{ вычисление n!}
{вычисление k!}
{вычисление (n-k)!}
{результат}
{печать результата}
ReadLn(n,k);
a1:=factorial(n); a2:=factorial(k); a3:=factorial(n-k); c:=a1 div (a2*a3); Writeln(c);
ReadLn;
End.
Program Имя;
```

Описательная часть программы

Function Имя [(Список формальных параметров)]:

Тип результата;

Описательная часть функции

Begin

Тело функции

End;

Begin

Тело программы (функция вызывается по имени factorial для вычисления три раза)

End.

Пусть $n=5$, $k=3$. Когда в программе встречается оператор $a1 := \text{factorial}(n)$, выполняются следующие действия:

выделяется память для переменных, описанных в функции factorial;

формальному параметру присваивается значение фактического: $n := 5$ ($n=5$);

выполняется функция, вычисляется факториал числа 5;

значение функции передается в место обращения к этой функции, то есть присваивается переменной $a1$.

В операторах $a2 := \text{factorial}(k)$ и $a3 := \text{factorial}(n-k)$ еще дважды вызывается функция factorial с параметрами $k=3$ и $n-k=2$. Всего в программе имеется 3 обращения к функции factorial, столько же раз выполняются и описанные выше действия.

Еще раз подчеркнем, что функция — это самостоятельная часть программы, имеющая собственные переменные, которым отводится отдельное место в памяти ЭВМ. Этим объясняется тот факт, что переменные с одинаковыми именами, используемые в функции и в основной программе, являются разными (в рассмотренном примере - переменная n

основной программы и параметр *n* функции). При выполнении программы машина «не путает» имена этих переменных, так как области их действия не совпадают.

1. Проверить работу программы.
2. Ответить на контрольные вопросы:
 1. Что называется подпрограммой в Турбо Паскале?
 2. В каком разделе программы описываются Функции в программе?
 3. В каком случае в программе используется подпрограмма-функция?

Практическое занятие №10. Использование стандартных функций и процедур для работы со строками и массивами.

Учебная цель: отработать навыки применения стандартных процедур и функций для обработки строк и массивов.

Организационная форма занятия: практическое занятие в форме практической подготовки

Задачи практической работы:

1. Повторить теоретический материал по теме практической работы.
2. Ответить на вопросы для закрепления теоретического материала.
3. Разработать программы для обработки строк с использованием стандартных функций и процедур
4. Оформить отчет.

Краткие теоретические и учебно-методические материалы по теме практической работы

Паскалю и означает текстовую переменную. Ее запись:
VAR A:STRING:[N]; где N - максимальная длина текстовой переменной.

Ввод текстовых переменных:

READ (A);

A='ТЕКСТ'

Операции и функции:

1. Две стринговые переменные можно складывать:

```
program ttt;
```

```
var a,b,c: string[40];
```

```
begin
```

```
a:='программирование';
```

```
b:=' на Паскале';
```

```
c:=a+b;
```

```
write (c);
```

```
end.
```

2. LENGTH(переменная), определяет длину символьной переменной k:=length(a);

3. COPY(A,K,L) выделяет из строки заданную подстроку, где A - стринговая переменная;

K - начальная позиция с которой начинается подстрока;

L - количество рассматриваемых позиций.

4. INSERT(X,A,L) вставка подстроки в строку X - вставляемый фрагмент, A - стринговая переменная, куда вставляется фрагмент, L - позиция вставки

5. STR(X,A) преобразует число X в строку A

6. POS(X,A) находит в строке A подстроку X (позицию вхождения). Результат выполнения функции - целое число.

7. DELETE(X,A,L) удаление L символов из строки X, где X - строковая переменная; A - позиция удаления; L - количество удаляемых символов.
8. CONCAT(S1,S2,...,SN) сцепление строк S1,S2,...,SN
9. VAL(A,X,C) преобразование строки A в число переменной X. Если параметр C содержит 0, значит преобразование прошло успешно, в противном случае C содержит номер позиции в строке A, где обнаружен неверный символ.
10. UPCASE(B) функция типа char, преобразует латинские строчные буквы в заглавные.

Задания для практического занятия:

1. Рассмотреть пример.
2. Написать программы для решения задач в соответствии с вариантом.
3. Составить отчет

Инструкция по выполнению практической работы

1. **Запишите программу в тетрадь.**

```
program p1;
uses crt;
var x:real; y:integer; c,z:char;
s,s1,s2,s3,s4,s5:string;
begin
clrscr;
s2:='круговорот';
writeln('длина слова круговорот', length(s2));
s:=concat('12','345');
writeln(s);
s1:=copy(s,3,2);
writeln(s1);
y:=pos('pot',s2);
writeln(y);
z:=upcase(c);
writeln(z);
delete(s2,3,6);
writeln(s2);
insert('abc',s2,3);
writeln(s2);
end.
```

2. **Выполните задания.**

1. Проанализируйте устно, что должно быть на экране после выполнения программы (пример 1)? Ответы запишите карандашом против каждой строки вывода на экран.
2. Загрузите Паскаль.
3. Выполните программу. Сравните полученные вами ответы с ответами на ЭВМ.
4. Выполните задания для самостоятельной работы.

Вариант-1

1. Дана строка символов да точки. Группы символов в ней между пробелами считаются словами. Определить количество слов.
2. Дана строка символов до точки. Определить, является ли она записью десятичного числа кратного двум.
3. Дана строка символов до точки. Вывести ее на экран задом наперед.
4. Составить программу, которая заменяет все окончания “ый” на “ая”.

5. Подсчитать, сколько раз в данной строке встречается буква, вводимая с клавиатуры
6. Заменить две буквы X на Y, причем X и Y вводятся с клавиатуры.

Вариант-2

1. Из строки удалить среднюю букву, если длина строки нечетная, иначе — удалить две средние буквы.
2. Заменить A на AA во всем выражении.
3. На какую букву начинается больше слов в тексте?
4. Дана строка символов. Удалить из нее все знаки препинания.
5. Дана строка символов. Дано слово. Удалить из строки это слово.
6. Дана строка символов. Выделите подстроку между первой и последней точкой.

Вариант-3

1. Дана строка символов. Удалить из нее первый знак препинания.
2. Дана строка символов до точки. Определить, является ли она правильным скобочным выражением. Рассматривать только круглые скобки.
3. Составит программу, которая заменяет все буквы “O” на “OO”.
4. Из строки удалить первую букву, если длина строки нечетная, иначе — удалить среднюю букву.
5. Сколько раз в тексте встречается заданное слово?
6. В данном предложении определите слова, которые начинаются с заданной буквы.

Вариант-4

1. Дана строка символов до точки. Группы символов в ней между пробелами считаются словами. Определить количество слов.
2. Дана строка символов до точки. Определить, является ли она записью десятичного числа кратного двум.
3. Дана строка символов до точки. Вывести ее на экран задом наперед.
4. Составить программу, которая заменяет все окончания “ый” на “ая”.
5. Подсчитать, сколько раз в данной строке встречается буква, вводимая с клавиатуры
6. Заменить две буквы X на Y, причем X и Y вводятся с клавиатуры.

Вариант-5

1. Из строки удалить среднюю букву, если длина строки нечетная, иначе — удалить две средние буквы.
2. Заменить A на AA во всем выражении.
3. На какую букву начинается больше слов в тексте?
4. Дана строка символов. Удалить из нее все знаки препинания.
5. Дана строка символов. Дано слово. Удалить из строки это слово.
6. Дана строка символов. Выделите подстроку между первой и последней точкой.

Вариант-6

1. Дана строка символов. Удалить из нее первый знак препинания.
2. Дана строка символов до точки. Определить, является ли она правильным скобочным выражением. Рассматривать только круглые скобки.
3. Составит программу, которая заменяет все буквы “O” на “OO”.
4. Из строки удалить первую букву, если длина строки нечетная, иначе — удалить среднюю букву.
5. Сколько раз в тексте встречается заданное слово?
6. В данном предложении определите слова, которые начинаются с заданной буквы.

Методика анализа результатов, полученных в ходе практической работы

1. В результате выполнения практических заданий у Вас должны быть работающие программы и оформлен отчет по каждому заданию.

Порядок выполнения отчета по практической работе

1. Разработать программы в среде ABC Pascal.
2. Составить систему тестов и проверить работу программ на всех возможных значениях исходных данных.
3. Результаты выполнения практической работы оформить в виде отчета.

! Использование стандартных функций для работы с массивами !

Учебная цель: отработать навыки использования стандартных функций для работы с массивами

Организационная форма занятия: практическое занятие в форме практической подготовки

Ход практической работы:

1. Повторить теоретический материал по теме практической работы.
2. Ответить на вопросы для закрепления теоретического материала.
3. Разработать программы для обработки одномерных массивов
4. Оформить отчет.

Краткие теоретические и учебно-методические материалы по теме практической работы

При создании функции или процедуры работы с массивами в ее объявление нельзя включать описание индексов.

Пример. объявление **procedure MyProc(A: array[1..10] of Integer);**

будет расценено как синтаксическая ошибка и вызовет соответствующее сообщение компилятора. Правильным будет объявление

type ta = array[1..10] of Integer; procedure MyProc(A: ta);

Функции и процедуры могут воспринимать в качестве параметров не только массивы фиксированного размера, но и так называемые открытые массивы, размер которых неизвестен. В этом случае в объявлении функции или процедуры они описываются как массивы базовых типов без указания их размерности.

Пример:

procedure SumArray(A:array of integer; var B: array of integer);

При таком определении передаваемый в функцию первый массив будет копироваться и с этой копией - массивом A, будет работать процедура. Второй открытый массив определен как **var**. Этот массив передается по ссылке, т.е. он не копируется, и процедура будет работать непосредственно с исходным массивом.

Массив, переданный как открытый, воспринимается в теле процедуры или функции как массив с целыми индексами, начинающимися с 0. Размер массива может быть определен функциями **Length** - число элементов и **High** - наибольшее значение индекса.

Очевидно, что всегда **High = Length - 1**.

При вызове функции или процедуры с параметром в виде открытого массива можно использовать в качестве аргумента конструктор открытого массива, который формирует массив непосредственно в операторе вызова. Список элементов такого конструктора массива заключается в квадратные скобки, а значения элементов разделяются запятыми. Например, функцию Sum, суммирующую элементы числового массива, можно вызвать следующим образом:

Sum([1.2,4.45,0.1]);

Пример:

В качестве примера использования открытых массивов рассмотрим процедуру, которая

принимает два открытых одинакового размера массива целых чисел, суммирует их и заносит результат во второй из переданных массивов.

```
procedure SumArray(A:array of integer; var B: array of integer);  
var i:word;  
begin  
for i:=0 to High(A) do B[i]:=A[i]+B[i];  
end;
```

Вызов этой процедуры может иметь вид:

```
var A1,A2: array [1..3] of integer;  
begin  
  <операторы заполнения массивов>  
  SumArray(A1,A2);  
end;
```

Методика анализа результатов, полученных в ходе практической работы

1.В результате выполнения практических заданий у Вас должны быть работающие программы и оформлен отчет по каждому заданию.

Порядок выполнения отчета по практической работе

1. Разработать программы в среде ABC Pascal.
2. Составить систему тестов и проверить работу программ на всех возможных значениях исходных данных.
3. Результаты выполнения практической работы оформить в виде отчета.

Тема 3.2. Структуризация в программировании

Практическое занятие №11. Работа с файлом последовательного доступа.

Учебная цель: Отработать навыки работы с несколькими файлами последовательного доступа.

Организационная форма занятия: практическое занятие в форме практической подготовки

Задачи практической работы:

1. Повторить теоретический материал по теме практической работы.
2. Ответить на вопросы для закрепления теоретического материала.
3. Разработать программы для обработки одномерных массивов
4. Оформить отчет.

Краткие теоретические и учебно-методические материалы по теме практической работы

Последовательные файлы.

Используются для решения задач, требующих поочередной обработки компонент файла. Работа с последовательными файлами предполагает создание файла, его обработку и корректировку.

Для создания последовательного файла необходимо выполнить следующие шаги :

- 1) связать имя файла с файловой переменной (процедура Assign);
- 2) открыть новый файл (процедура Rewrite);
- 3) подготовить информацию для ввода;
- 4) записать в файл компоненты (процедура Write);
- 5) закрыть созданный файл (процедура Close).

Доступ к компонентам последовательного файла (Пример 2).

Чтобы обработать хранящиеся в файле данные, необходимо получить доступ к компонентам файла. Для этого надо:

- 1) связать имя файла с файловой переменной (процедура Assign);
- 2) открыть уже существующий на диске файл (процедура Reset);
- 3) последовательно считать компоненты до нужного (процедура Read);
- 4) выполнить обработку;
- 5) закрыть файл (процедура Close).

Типовые примеры.

Пример 1. Из файла 'csla.nat' перенести в файл 'chet.cis' все чётные числа. Вывести содержимое файла 'chet.cis' на экран.

```
var  a:integer;
      f1,f2:text;
Begin
  assign(f1,'csla.nat');
  assign(f2,'chet.cis');
  reset (f1);                {открываем файл для чтения}
  rewrite(f2);                {открываем файл для записи}
```

```

while not eof(f1) do           {пока не конец файла}
begin
  readln(f1,a);                {считываем из файла f1}
  if a mod 2 =0 then writeln(f2,a); {если чётное, то сохраняем в файл f2}
end;
close(f1);close(f2);           {закрываем файл}
                                {выводим содержимое файла 'chet.cis'}
reset (f2);                    {открываем файл для чтения}
while not eof(f2) do
begin
  readln(f2,a);                {считываем из файла f2}
  writeln(a);
end;
close(f2);
End.

```

Задания для практического занятия:

1. Рассмотреть типовые примеры
2. Написать программы для решения задач в соответствии с вариантом.
3. Составить отчет

Инструкция по выполнению практической работы

1. Рассмотрите примеры из теоретической части.
2. Выполнить примеры.
3. Выполните задания для самостоятельной работы.

Индивидуальные задания

Вариант 1

1. 1. Дан файл F, компоненты которого являются целыми числами. Найти:
 А) количество элементов;
 Б) наибольшее из значений, если их несколько, то подсчитать количество таких элементов;
 В) среднее арифметическое элементов.
2. Даны символьные файлы R и G. Записать в файл H все компоненты файлов R и G.

Вариант 2

1. Дан файл F, компоненты которого являются целыми числами. Найти:
 А) количество элементов;
 Б) наибольшее из значений, если их несколько, то подсчитать количество таких элементов;
 В) среднее арифметическое элементов.
2. Дан целочисленный файл W. Записать в файл D все четные числа, а в файл T все нечетные..

Вариант 3

1. 1. Дан файл F, компоненты которого являются целыми числами. Найти:
 А) количество элементов;
 Б) наибольшее из значений, если их несколько, то подсчитать количество таких элементов;
 В) среднее арифметическое элементов.
2. Даны два файла A и B. Поменять местами содержимое этих файлов.

Методика анализа результатов, полученных в ходе практической работы

1. В результате выполнения практических заданий у Вас должны быть работающие программы и оформлен отчет по каждому заданию.

Порядок выполнения отчета по практической работе

1. Разработать программы в среде ABC Pascal.
2. Составить систему тестов и проверить работу программ на всех возможных значениях исходных данных.
3. Результаты выполнения практической работы оформить в виде отчета.

Практическое занятие №12. Работа с файлом произвольного доступа.

Учебная цель: Отработать навыки работы с использованием файлов произвольного доступа.

Организационная форма занятия: практическое занятие в форме практической подготовки

Ход практической работы:

1. Повторить теоретический материал по теме практической работы.
2. Ответить на вопросы для закрепления теоретического материала.
3. Разработать программы для обработки одномерных массивов
4. Оформить отчет.

Краткие теоретические и учебно-методические материалы по теме практической работы

Файлы последовательного доступа

Прямой доступ предполагает, что файл представляет собой линейную последовательность блоков. Если файл содержит n блоков, то они нумеруются от 1 через 1 до n . Кроме того, вводится понятие условной границы между блоками, при этом условная граница с номером 0 расположена перед блоком с номером 1, граница с номером 1 расположена перед блоком с номером 2 и, наконец, условная граница с номером n находится после блока с номером n .

Реализация прямого доступа осуществляется с помощью функций и процедур `FileSize`, `FilePos`, `Seek` и `Truncate`.

Функция `FileSize( var f )`: `Longint` возвращает количество блоков в открытом файле `f`.

Функция `FilePos( var f )`: `Longint` возвращает текущую позицию в файле `f`. Позиция в файле - это номер условной границы. Для только что открытого файла текущей позицией будет граница с номером 0. Это значит, что можно записать или прочесть блок с номером 1. После чтения или записи первого блока текущая позиция переместится на границу с номером 1, и можно будет обращаться к блоку с номером 2. После прочтения последней записи значение `FilePos` равно значению `FileSize`.

Процедура `Seek( var f; N: Longint)` обеспечивает назначение текущей позиции в файле (позиционирование). В параметре `N` должен быть задан номер условной границы, предшествующей блоку, к которому будет производиться последующее обращение. Например, чтобы работать с блоком 4, необходимо задать значение `N`, равное 3. Процедура `Seek` работает с открытыми файлами.

Процедура `Truncate( var f )` устанавливает в текущей позиции признак конца файла и удаляет (стирает) все последующие блоки.

Пример. Пусть на НМД имеется текстовый файл `ID.DAT`, который содержит числовые значения действительного типа по два числа в каждой строке - значения аргумента и функции соответственно. Количество пар чисел не более 200. Составить программу, которая читает файл, значения аргумента и функции записывает в одномерные массивы, подсчитывает их количество, выводит на экран дисплея и записывает в файл компонентного типа `RD.DAT`.

```
Program F;  
var  
  rArg, rF: Array[1..200] of Real;  
  inf: Text;  
  outf: File of Real;  
  n, l: Integer;  
begin  
  Assign(inf,'ID.DAT');  
  Assign(outf,'RD.DAT');
```

```

Reset(inf);
Rewrite(outf);
n:=0;
while not EOF(inf) do
begin
  n:=n+1;
  ReadLn(inf,rArg[n],rF[n])
end;
for l:=1 to n do
begin
  WriteLn(l:2,rArg[l]:8:2,rF[l]:8:2);
  Write(outf,rArg[l], rF[l]);
end;
close(outf)
end.

```

Задания для практического занятия:

1. Рассмотреть типовые примеры
2. Написать программы для решения задач в соответствии с вариантом.
3. Составить отчет

Инструкция по выполнению практической работы

1. Рассмотрите примеры из теоретической части.
2. Выполнить примеры.
3. Выполните задания для самостоятельной работы.

Индивидуальные задания

Вариант 1

Записать в файл прямого доступа N действительных чисел. Найти наибольшее из значений модулей компонентов с нечетными номерами.

Методика анализа результатов, полученных в ходе практической работы

1. В результате выполнения практических заданий у Вас должны быть работающие программы и оформлен отчет по каждому заданию.

Порядок выполнения отчета по практической работе

1. Разработать программы в среде ABC Pascal.
2. Составить систему тестов и проверить работу программ на всех возможных значениях исходных данных.
3. Результаты выполнения практической работы оформить в виде отчета.

Практическое занятие №13. Использование стандартных процедур и функций для работы с файлами.

Учебная цель: Отработать навыки работы с использованием файлов-записей.

Организационная форма занятия: практическое занятие в форме практической подготовки

Ход практической работы:

1. Повторить теоретический материал по теме практической работы.
2. Ответить на вопросы для закрепления теоретического материала.
3. Разработать программы для обработки одномерных массивов
4. Оформить отчет.

Краткие теоретические и учебно-методические материалы по теме практической работы

Последовательные файлы.

Используются для решения задач, требующих поочередной обработки компонент файла. Работа с последовательными файлами предполагает создание файла, его обработку и корректировку.

Для создания последовательного файла необходимо выполнить следующие шаги:

- 1) связать имя файла с файловой переменной (процедура Assign);
- 2) открыть новый файл (процедура Rewrite);
- 3) подготовить информацию для ввода;
- 4) записать в файл компоненты (процедура Write);
- 5) закрыть созданный файл (процедура Close).

Доступ к компонентам последовательного файла (Пример 2).

Чтобы обработать хранящиеся в файле данные, необходимо получить доступ к компонентам файла. Для этого надо:

- 1) связать имя файла с файловой переменной (процедура Assign);
- 2) открыть уже существующий на диске файл (процедура Reset);
- 3) последовательно считать компоненты до нужного (процедура Read);
- 4) выполнить обработку;
- 5) закрыть файл (процедура Close).

Типовые примеры.

Пример 1. Из файла 'csla.nat' перенести в файл 'chet.cis' все чётные числа. Вывести содержимое файла 'chet.cis' на экран.

```
var  a:integer;
     f1,f2:text;
Begin
  assign(f1,'csla.nat');
  assign(f2,'chet.cis');
  reset (f1);           {открываем файл для чтения}
  rewrite(f2);          {открываем файл для записи}
  while not eof(f1) do   {пока не конец файла}
  begin
    readln(f1,a);        {считываем из файла f1}
    if a mod 2 =0 then writeln(f2,a); {если чётное, то сохраняем в файл f2}
  end;
  close(f1);close(f2);   {закрываем файл}
  {выводим содержимое файла 'chet.cis'}
```

```

reset (f2);           {открываем файл для чтения}
while not eof(f2) do
  begin
    readln(f2,a);      {считываем из файла f2}
    writeln(a);
  end;
close(f2);
End.

```

Задания для практического занятия:

1. Рассмотреть типовые примеры
2. Написать программы для решения задач в соответствии с вариантом.
3. Составить отчет

Инструкция по выполнению практической работы

1. Рассмотрите примеры из теоретической части.
2. Выполнить примеры.
3. Выполните задания для самостоятельной работы.

Индивидуальные задания

Вариант 1

Сводная ведомость результатов экзаменационной сессии студенческой группы находится в файле на диске и для каждого студента содержит фамилию, инициалы и оценки по пяти предметам. Количество студентов в группе не превышает 20 человек.

Составить программу, с помощью которой можно корректировать и дополнять список и получать:

- ✓ список студентов;
- ✓ список студентов, сдавших экзамены только на «5»; список студентов,
- ✓ имеющих тройки;
- ✓ список студентов, имеющих двойки. При этом студент, имеющий более чем одну двойку, исключается из списка.

Вариант 2

Предприятие имеет местную телефонную станцию на 20 номеров. Телефонный справочник данного предприятия для каждого номера телефона содержит номер помещения и список служащих, сидящих в данном помещении.

Составить программу, которая:

- ✓ корректирует базу;
- ✓ по номеру телефона выдает номер помещения и список сидящих в нем людей;
- ✓ по номеру помещения выдает номер телефона;
- ✓ по фамилии выдает номер телефона и номер помещения.

Номер телефона — двузначный. В одном помещении может находиться от одного до четырех служащих.

Вариант 3

В гостинице имеется 15 номеров, из них 5 одноместных и 10 двухместных. Составить программу, которая заполняет и (или) корректирует данные о жильцах и по фамилии определяет номер, где проживает жилец. Программа запрашивает фамилию жильца.

- ✓ Если жильца с такой фамилией нет, об этом выдается сообщение.
- ✓ Если жилец с такой фамилией в гостинице единственный, программа выдает фамилию

жильца и номер проживания.

✓ Если в гостинице проживает два или более жильцов с такой фамилией, программа дополнительно запрашивает инициалы.

Вариант 4

В текстовом файле хранится список служащих. Для каждого служащего указаны фамилия и инициалы, название занимаемой должности, год поступления на работу и оклад.

Написать программу, выполняющую следующие действия:

- ✓ корректировку или дополнение списка с клавиатуры;
- ✓ сортировку по фамилии, окладу или году поступления;
- ✓ вывод на экран информации о служащем, фамилия которого введена с клавиатуры;
- ✓ запись списка в файл под тем же или новым именем.

Вариант 5

Расписание электричек хранится в текстовом файле на диске. Каждая запись содержит название пункта назначения, пометки типа «СВ», «ПВ», «КСВ» и время отправления.

Написать программу, выполняющую следующие действия:

- ✓ корректировку или дополнение расписания с клавиатуры;
- ✓ сортировку по станции назначения или по времени отправления;
- ✓ вывод на экран информации о поездах, отходящих после введенного времени;
- ✓ запись расписания в файл под тем же или новым именем.

Вариант 6

В текстовом файле хранится список товаров. Для каждого товара указаны его название, стоимость единицы товара в тыс. руб., количество и единица измерения (например, 100 шт., 20 кг).

Написать программу, выполняющую следующие действия:

- ✓ корректировку или дополнение списка с клавиатуры;
- ✓ сортировку по названию товара или по общей стоимости;
- ✓ вывод на экран информации о товаре, название которого введено с клавиатуры;
- ✓ запись списка в файл под тем же или новым именем

Вариант 7

В текстовом файле хранится список товаров. Для каждого товара указаны его название, название магазина, в котором продается товар, стоимость товара в тыс. руб. и его количество с указанием единицы измерения (например, 100 шт., 20 кг).

Написать программу, выполняющую следующие действия:

- ✓ корректировку или дополнение списка с клавиатуры;
- ✓ сортировку по названию товара или по названию магазина;
- ✓ вывод на экран информации о товаре, название которого введено с клавиатуры;
- ✓ запись списка в файл под тем же или новым именем.

Вариант 8

Список студенческой группы записан на диске как текстовый файл. Каждая строка списка содержит фамилию студента и три экзаменационные оценки, причем список никак не упорядочен. Составить программу, которая корректирует список и сортирует его либо по среднему баллу, либо по алфавиту, либо по оценкам по заданному предмету.

Список записывается в файл либо под старым, либо под новым именем.

Вариант 9

В текстовом файле хранится список товаров. Для каждого товара указаны его название, название магазина, в котором продается товар, стоимость товара в тыс. руб. и его количество с указанием единицы измерения (например, 100 шт., 20 кг).

Написать программу, выполняющую следующие действия:

- ✓ корректировку или дополнение списка с клавиатуры;
- ✓ сортировку по названию товара или по названию магазина;
- ✓ вывод на экран информации о товаре, название которого введено с клавиатуры;
- ✓ запись списка в файл под тем же или новым именем.

Вариант 10

1. Описать запись с именем Route, содержащую следующие поля:

- ✓ название начального пункта маршрута;
- ✓ название конечного пункта маршрута;
- ✓ номер маршрута.

2. Написать программу, выполняющую следующие действия:

- ✓ ввод данных с клавиатуры в массив, состоящий из восьми элементов типа Route; записи должны быть упорядочены по номерам маршрутов;
- ✓ вывод на экран информации о маршруте, номер которого введен с клавиатуры; если таких маршрутов нет, вывести на экран соответствующее сообщение;
- ✓ запись массива в файл под заданным с клавиатуры именем.

Вариант 11

1. Описать запись с именем Route, содержащую следующие поля:

- ✓ название начального пункта маршрута;
- ✓ название конечного пункта маршрута;
- ✓ номер маршрута.

2. Написать программу, выполняющую следующие действия:

- ✓ ввод данных с клавиатуры в массив, состоящий из восьми элементов типа Route; записи должны быть упорядочены по номерам маршрутов;
- ✓ вывод на экран информации о маршрутах, которые начинаются или кончаются в пункте, название которого введено с клавиатуры; если таких маршрутов нет, вывести на экран соответствующее сообщение;
- ✓ запись массива в файл под заданным с клавиатуры именем.

Вариант 12

1. Описать запись с именем Note, содержащую следующие поля:

- ✓ фамилия, имя;
- ✓ номер телефона;
- ✓ день рождения (массив из трех чисел).

2. Написать программу, выполняющую следующие действия:

- ✓ ввод данных с клавиатуры в массив, состоящий из восьми элементов типа Note; записи должны быть упорядочены по трем первым цифрам номера телефона;
- ✓ вывод на экран информации о человеке, чья фамилия введена с клавиатуры; если такого нет, вывести на экран соответствующее сообщение;
- ✓ запись массива в файл под заданным с клавиатуры именем.

Методика анализа результатов, полученных в ходе практической работы

1. В результате выполнения практических заданий у Вас должны быть работающие программы и оформлен отчет по каждому заданию.

Порядок выполнения отчета по практической работе

1. Разработать программы в среде ABC Pascal.
2. Составить систему тестов и проверить работу программ на всех возможных значениях исходных данных.
3. Результаты выполнения практической работы оформить в виде отчета.

Тема 3.3. Модульное программирование

Практическое занятие №14. Программирование модуля.

Учебная цель: Отработать навыки работы с программирования модуля.

Организационная форма занятия: практическое занятие в форме практической подготовки

Задачи практической работы:

1. Повторить теоретический материал по теме практической работы.
2. Ответить на вопросы для закрепления теоретического материала.
3. Разработать программы для обработки одномерных массивов
4. Оформить отчет.

Краткие теоретические и учебно-методические материалы по теме практической работы

Задача. Реализовать в виде модуля набор подпрограмм для выполнения следующих операций над обыкновенными дробями вида P/Q (P — целое, Q — натуральное): 1) сложение; 2) вычитание; 3) умножение; 4) деление; 5) сокращение дроби; 6) возведение дроби в степень N (N — натуральное); 7) функции, реализующие операции отношения (равно, не равно, больше или равно, меньше или равно, больше, меньше).

Дробь представить следующим типом:

```
Type Frac = Record
    P : Integer;
    Q : 1..High(LongInt)
End;
```

Используя этот модуль, решить задачи:

1. Дан массив A — массив обыкновенных дробей. Найти сумму всех дробей, ответ представить в виде несократимой дроби. Вычислить среднее арифметическое всех дробей, ответ представить в виде несократимой дроби.
2. Дан массив A — массив обыкновенных дробей. Отсортировать его в порядке возрастания.

Unit Droby;

Interface

```
Type
Natur = 1..High(LongInt);
Frac = Record
    P : LongInt; {Числитель дроби}
    Q : Natur {Знаменатель дроби}
End;
```

Procedure Sokr(Var A : Frac);

Procedure Summa(A, B : Frac; Var C : Frac);

Procedure Raznost(A, B : Frac; Var C : Frac);

Procedure Proizvedenie(A, B : Frac; Var C : Frac);

Procedure Chastnoe(A, B : Frac; Var C : Frac);

Procedure Stepen(A : Frac; N : Natur; Var C : Frac);

Function Menshe(A, B : Frac) : Boolean;

Function Bolshe(A, B : Frac) : Boolean;

Function Ravno(A, B : Frac) : Boolean;

Function MensheRavno(A, B : Frac) : Boolean;

Function BolsheRavno(A, B : Frac) : Boolean;

Function NeRavno(A, B : Frac) : Boolean;

{Раздел реализации модуля}

Implementation

{Наибольший общий делитель двух чисел - вспомогательная функция, ранее не объявленная}

Function NodEvklid(A, B : Natur) : Natur;

Begin

While A <> B Do

 If A > B Then

 If A Mod B <> 0 Then A := A Mod B Else A := B

 Else

 If B Mod A <> 0 Then B := B Mod A Else B := A;

 NodEvklid := A

End;

Procedure Sokr; {Сокращение дроби}

Var M, N : Natur;

Begin

 If A.P <> 0 Then

 Begin

 If A.P < 0 Then M := Abs(A.P)

 Else M := A.P; {Совмещение типов, т.к. A.P - LongInt}

 N := NodEvklid(M, A.Q); A.P := A.P Div N; A.Q := A.Q Div N

 End

End;

Procedure Summa; {Сумма дробей}

Begin

 {Знаменатель дроби} C.Q := (A.Q * B.Q) Div NodEvklid(A.Q, B.Q);

 {Числитель дроби} C.P := A.P * C.Q Div A.Q + B.P * C.Q Div B.Q;

 Sokr(C)

End;

Procedure Raznost; {Разность дробей}

Begin

 {Знаменатель дроби} C.Q := (A.Q * B.Q) Div NodEvklid(A.Q, B.Q);

 {Числитель дроби} C.P := A.P * C.Q Div A.Q - B.P * C.Q Div B.Q;

 Sokr(C)

End;

Procedure Proizvedenie;

Begin

 {Знаменатель дроби} C.Q := A.Q * B.Q;

 {Числитель дроби} C.P := A.P * B.P;

 Sokr(C)

End;

Procedure Chastnoe;

Begin

 {Знаменатель дроби} C.Q := A.Q * B.P;

 {Числитель дроби} C.P := A.P * B.Q;

 Sokr(C)

End;

Procedure Stepen; {Степень}

Var I : Natur;

Begin

 C.Q := 1; C.P := 1; Sokr(A);

 For I := 1 To N Do Proizvedenie(A, C, C)

End;

```

Function Menshe;
Begin Menshe := A.P * B.Q < A.Q * B.P End;
Function Bolshe;
Begin Bolshe := A.P * B.Q > A.Q * B.P End;
Function Ravno;
Begin Ravno := A.P * B.Q = A.Q * B.P End;
Function BolsheRavno;
Begin BolsheRavno := Bolshe(A, B) Or Ravno(A, B) End;
Function MensheRavno;
Begin MensheRavno := Menshe(A, B) Or Ravno(A, B) End;
Function NeRavno;
Begin NeRavno := Not Ravno(A, B) End;
{Раздел инициализации модуля}
Begin
End.

```

Дадим некоторые рекомендации по разработке модулей:

- 1) спроектировать модуль, т.е. выделить основные и вспомогательные подпрограммы, другие ресурсы;
- 2) каждую подпрограмму целесообразно отладить отдельно, после чего «вклеить» в текст модуля.

Сохраним текст разработанной программы в файле DROBY.PAS и откомпилируем наш модуль. Для этого можно воспользоваться внешним компилятором, поставляемым вместе с Turbo Pascal. Команда будет выглядеть так: TPC DROBY.PAS. Если в тексте нет синтаксических ошибок, получим файл DROBY.TPU, иначе будет соответствующее сообщение с указанием строки, содержащей ошибку. Другой способ компиляции модуля — в среде программирования Turbo Pascal выбрать в пункте меню Run подпункты Make или Build (при этом должна быть включена компиляция на диск).

Теперь можно подключить модуль к программе, где планируется его использование.

Для примера решим задачу суммирования массива дробей.

```

Program Sum;
Uses Droby;
Var A : Array[1..100] Of Frac;
    I, N : Integer;
    S : Frac;
Begin
    Write('Введите количество элементов массива: ');
    ReadLn(N);
    S.P := 0; S.Q := 1; {Первоначально сумма равна нулю}
    For I := 1 To N Do {Вводим и суммируем дроби}
        Begin
            Write('Введите числитель ', I, '-й дроби: '); ReadLn(A[I].P);
            Write('Введите знаменатель ', I, '-й дроби: '); ReadLn(A[I].Q);
            Summa(A[I], S, S);
        End;
        WriteLn('Ответ: ', S.P, '/', S.Q)
    End.

```

Задания для практического занятия:

1. Рассмотреть типовые примеры
2. Написать программы для решения задач в соответствии с вариантом.
3. Составить отчет

Инструкция по выполнению практической работы

1. Рассмотрите примеры из теоретической части.
2. Выполнить примеры.
3. Выполните задания для самостоятельной работы.

Методика анализа результатов, полученных в ходе практической работы

1. В результате выполнения практических заданий у Вас должны быть работающие программы и оформлен отчет по каждому заданию.

Порядок выполнения отчета по практической работе

1. Разработать программы в среде ABC Pascal.
2. Составить систему тестов и проверить работу программ на всех возможных значениях исходных данных.
3. Результаты выполнения практической работы оформить в виде отчета.

Практическое занятие №15. Создание библиотеки подпрограмм.

Учебная цель: Отработать навыки работы создания библиотеки подпрограмм.

Организационная форма занятия: практическое занятие в форме практической подготовки

Ход практической работы:

1. Повторить теоретический материал по теме практической работы.
2. Ответить на вопросы для закрепления теоретического материала.
3. Разработать программы для обработки одномерных массивов
4. Оформить отчет.

Краткие теоретические и учебно-методические материалы по теме практической работы

Стандартный язык Pascal не располагает средствами разработки и поддержки библиотек программиста (в отличие, скажем, от языка Fortran и других языков программирования высокого уровня), которые компилируются отдельно и в дальнейшем могут быть использованы как самим разработчиком, так и другими. Если программист имеет достаточно большие наработки, и те или иные подпрограммы могут быть использованы при написании новых приложений, то приходится эти подпрограммы целиком включать в новый текст. В Pascal это ограничение преодолевается за счет, во-первых, введения внешних процедур, во-вторых, разработки и использования модулей. В настоящей публикации на примерах рассмотрим работу с теми и другими программными единицами. Начнем с внешних подпрограмм. Такой механизм предусматривает, что исходный текст каждой процедуры или функции хранится в отдельном файле и при необходимости с помощью специальной директивы компилятора включается в текст создаваемой программы.

Покажем это на примере задач целочисленной арифметики, где аргументы, результаты и промежуточные величины являются целыми (Integer, Word, LongInt и т.д.). Вот несколько таких задач.

1. Дано натуральное число n. Найти сумму первой и последней цифры этого числа.
2. Дано натуральное число n. Переставить местами первую и последнюю цифры этого числа.
3. Дано натуральное число n. Допisać к нему цифру k в конец и в начало (если это возможно, т.е. результат не выйдет за диапазон допустимых значений), или сообщить о невозможности выполнения операции.
4. Найти наибольшую цифру в записи данного натурального числа.
5. Дано натуральное число n. Переставить его цифры так, чтобы образовалось максимальное число, записанное теми же цифрами.

При решении каждой из этих задач может быть использована функция, возвращающая количество цифр в записи натурального числа. Вот возможный вариант такой функции:

```
Function Digits(N : LongInt) : Byte;
```

```
Var Kol : Byte;
```

```
Begin Kol := 0;
```

```
While N <> 0 Do
```

```
  Begin Kol := Kol + 1;
```

```
  N := N Div 10 End;
```

```
Digits := Kol
```

```
End;
```

Сохраним этот текст в файле с расширением .inc (это расширение внешних подпрограмм в Pascal), например, digits.inc.

Еще необходима функция возведения натурального числа в натуральную степень.

```
Function Power(A, N : LongInt) : LongInt; {файл power.inc}
```

```
Var I, St : LongInt;  
Begin St := 1;  
For I := 1 To N Do St := St * A;  
Power := St End;
```

Попробуем использовать функции при решении задачи номер один.

Program Example1;

```
Var N, S : LongInt; {$I digits.inc} {подключаем внешнюю функцию digits.inc,  
возвращающую количество цифр в записи числа}  
{$I power.inc} {внешняя функция, выполняющая возведение числа A в степень N}
```

```
Begin
```

```
Write('Введите натуральное число: ');
```

```
ReadLn(N); {для определения последней цифры числа N берем остаток от деления этого  
числа на 10, а для определения первой делим N на 10 в степени на единицу меньшую, чем  
количество цифр в записи числа (нумерация разрядов начинается с 0)}
```

```
S := N Mod 10 + N Div Power(10, Digits(N) - 1);
```

```
WriteLn('Искомая сумма: ', S)
```

```
End.
```

Внешние процедуры создаются и внедряются в использующие их программы аналогично функциям, и мы не будем подробно на этом останавливаться. Далее речь пойдет о модулях: их структуре, разработке, компиляции и использовании.

Модуль - это набор ресурсов (функций, процедур, констант, переменных, типов и т.д.), разрабатываемых и хранимых независимо от использующих их программ. В отличие от внешних подпрограмм модуль может содержать достаточно большой набор процедур и функций, а также других ресурсов для разработки программ. Обычно каждый модуль содержит логически связанные между собой программные ресурсы.

В основе идеи модульности лежат принципы структурного программирования. Существуют стандартные модули Pascal, которые обычно описываются в литературе по данному языку.

Модуль имеет следующую структуру:

```
Unit <имя модуля>; {заголовок модуля} Interface {интерфейсная часть}
```

```
Implementation {раздел реализации}
```

```
Begin {раздел инициализации модуля}
```

```
End.
```

После служебного слова Unit записывается имя модуля, которое (для удобства дальнейших действий) должно совпадать с именем файла, содержащего данный модуль. Поэтому (как принято в MS DOS) имя не должно содержать более 8 символов.

В разделе Interface объявляются все ресурсы, которые будут в дальнейшем доступны программисту при подключении модуля.

Для подпрограмм здесь указывается лишь полный заголовок. В разделе Implementation реализуются все подпрограммы, которые были ранее объявлены. Кроме того, здесь могут содержаться свои константы, переменные, типы, подпрограммы и т.д., которые носят вспомогательный характер и используются для написания основных подпрограмм. В отличие от ресурсов, объявленных в разделе Interface, все, что дополнительно объявляется в Implementation, уже не будет доступно при подключении модуля. При написании основных подпрограмм достаточно указать их имя (т.е. не нужно полностью переписывать весь заголовок), а затем записать тело подпрограммы.

Наконец, раздел инициализации (который часто отсутствует) содержит операторы, которые должны быть выполнены сразу же после запуска программы, использующей модуль. Приведем пример разработки и использования модуля.

Поскольку рассмотренная ниже задача достаточно элементарна, ограничимся листингом программы с подробными комментариями.

Задача. Реализовать в виде модуля набор подпрограмм для выполнения следующих операций над обыкновенными дробями вида P/Q (P - целое, Q - натуральное): 1) сложение; 2) вычитание; 3) умножение; 4) деление; 5) сокращение дроби; 6) возведение дроби в степень N (N - натуральное); 7) функции, реализующие операции отношения (равно, не равно, больше или равно, меньше или равно, больше, меньше).

Дробь представить следующим типом:

Type Frac = Record P : Integer; Q : 1.. High(LongInt) End;

Используя этот модуль, решить задачи: 1. Дан массив A - массив обыкновенных дробей. Найти сумму всех дробей, ответ представить в виде несократимой дроби. Вычислить среднее арифметическое всех дробей, ответ представить в виде несократимой дроби. 2. Дан массив A - массив обыкновенных дробей. Отсортировать его в порядке возрастания.

Unit Droby;

Interface Type Natur = 1..High(LongInt); Frac = Record P : LongInt; {Числитель дроби} Q : Natur {Знаменатель дроби} End;

Procedure Sokr(Var A : Frac);

Procedure Summa(A, B : Frac; Var C : Frac); Procedure Raznost(A, B : Frac; Var C : Frac);

Procedure Proizvedenie(A, B : Frac; Var C : Frac);

Procedure Chastnoe(A, B : Frac; Var C : Frac);

Procedure Stepen(A : Frac; N : Natur; Var C : Frac);

Function Menshe(A, B : Frac) : Boolean;

Function Bolshe(A, B : Frac) : Boolean;

Function Ravno(A, B : Frac) : Boolean; Function MensheRavno(A, B : Frac) : Boolean;

Function BolsheRavno(A, B : Frac) : Boolean;

Function NeRavno(A, B : Frac) : Boolean; {Раздел реализации модуля}

Implementation {Наибольший общий делитель двух чисел - вспомогательная функция, ранее не объявленная}

Function NodEvklid(A, B : Natur) : Natur;

Begin While A <> B Do If A > B Then If A Mod B <> 0 Then A := A Mod B Else A := B

Else If B Mod A <> 0 Then B := B Mod A Else B := A; NodEvklid := A End;

Procedure Sokr; {Сокращение дроби}

Var M, N : Natur;

Begin If A.P <> 0 Then Begin If A.P < 0 Then M := Abs(A.P) Else M := A.P; {Совмещение типов, т.к. A.P - LongInt}

N := NodEvklid(M, A.Q); A.P := A.P Div N; A.Q := A.Q Div N End End;

Procedure Summa; {Сумма дробей}

Begin {Знаменатель дроби} C.Q := (A.Q * B.Q) Div NodEvklid(A.Q, B.Q); {Числитель дроби}

C.P := A.P * C.Q Div A.Q + B.P * C.Q Div B.Q; Sokr(C) End;

Procedure Raznost; {Разность дробей} Begin {Знаменатель дроби} C.Q := (A.Q * B.Q) Div NodEvklid(A.Q, B.Q); {Числитель дроби} C.P := A.P * C.Q Div A.Q - B.P * C.Q Div B.Q; Sokr(C) End;

Procedure Proizvedenie; Begin {Знаменатель дроби} C.Q := A.Q * B.Q; {Числитель дроби} C.P := A.P * B.P; Sokr(C) End;

Procedure Chastnoe; Begin {Знаменатель дроби} C.Q := A.Q * B.P; {Числитель дроби} C.P := A.P * B.Q; Sokr(C) End;

Procedure Stepen; {Степень} Var I : Natur; Begin C.Q := 1; C.P := 1; Sokr(A); For I := 1 To N Do Proizvedenie(A, C, C) End; Function Menshe; Begin Menshe := A.P * B.Q < A.Q * B.P End; Function Bolshe; Begin Bolshe := A.P * B.Q > A.Q * B.P End; Function Ravno; Begin Ravno := A.P * B.Q = A.Q * B.P End;

Function BolsheRavno; Begin BolsheRavno := Bolshe(A, B) Or Ravno(A, B) End;

Function MensheRavno; Begin MensheRavno := Menshe(A, B) Or Ravno(A, B) End;

Function NeRavno; Begin NeRavno := Not Ravno(A, B) End;

{Раздел инициализации модуля} Begin End.

Дадим некоторые рекомендации по разработке модулей: 1) спроектировать модуль, т.е. выделить основные и вспомогательные подпрограммы, другие ресурсы; 2) каждую подпрограмму целесообразно отладить отдельно, после чего «вклеить» в текст модуля.

Сохраним текст разработанной программы в файле DROBY.PAS и откомпилируем наш модуль. Для этого можно воспользоваться внешним компилятором, поставляемым вместе с Pascal. Команда будет выглядеть так: TPC DROBY.PAS. Если в тексте нет синтаксических ошибок, получим файл DROBY.TPU, иначе будет соответствующее сообщение с указанием строки, содержащей ошибку. Другой способ компиляции модуля - в среде программирования Pascal выбрать в пункте меню Run подпункты Make или Build (при этом должна быть включена компиляция на диск).

Теперь можно подключить модуль к программе, где планируется его использование. Для примера решим задачу суммирования массива дробей.

```
Program Sum; Uses Droby;
```

```
Var A : Array[1..100] Of Frac; I, N : Integer; S : Frac;
```

```
Begin
```

```
Write('Введите количество элементов массива: ');
```

```
ReadLn(N); S.P := 0; S.Q := 1; {Первоначально сумма равна нулю}
```

```
For I := 1 To N Do {Вводим и суммируем дроби}
```

```
Begin Write('Введите числитель ', I, '-й дроби: ');
```

```
ReadLn(A[I].P);
```

```
Write('Введите знаменатель ', I, '-й дроби: ');
```

```
ReadLn(A[I].Q);
```

```
Summa(A[I], S, S); End;
```

```
WriteLn('Ответ: ', S.P, '/', S.Q) End.
```

Вторую задачу предлагаем решить читателю самостоятельно. Как видно из примера, для подключения модуля используется служебное слово USES, после чего указывается имя модуля и происходит это сразу же после заголовка программы. Если необходимо подключить несколько модулей, они перечисляются через запятую. При использовании ресурсов модуля совсем не нужно знать, как работают его подпрограммы. Достаточно обладать информацией, как выглядят их заголовки и какое действие эти подпрограммы выполняют. По такому принципу осуществляется работа со всеми стандартными модулями. Поэтому, если программист разрабатывает модули не только для личного пользования, ему необходимо сделать полное описание всех доступных при подключении ресурсов. В таком случае возможна полноценная работа с таким продуктом.

Ещё несколько слов о видимости объектов модуля. Если в программе, использующей модуль, имеются идентификаторы, совпадающие с точностью до символа с идентификаторами модуля, то они «перекрывают» соответствующие ресурсы модуля. Тем не менее, даже в такой ситуации доступ к этим ресурсам модуля может быть получен таким образом: <имя модуля>.<имя ресурса>.

Индивидуальные задания

I. Реализовать в виде модуля набор подпрограмм для выполнения следующих операций над комплексными числами: 1) сложение; 2) вычитание; 3) умножение; 4) деление; 5) вычисление модуля комплексного числа; 6) возведение комплексного числа в степень n (n - натуральное). Комплексное число представить следующим типом: Type Complex = Record R, M : Real; {действительная и мнимая часть числа} End; Используя этот модуль, решить задачи: 1. Дан массив A - массив комплексных чисел. Получить массив C, элементами которого будут модули сумм рядом стоящих комплексных чисел.

2. Дан массив A[M] - массив комплексных чисел. Получить матрицу B[N, M], каждая строка которой получается возведением в степень, равную номеру этой строки, соответствующих элементов данного массива A. II. Реализовать в виде модуля набор подпрограмм для выполнения следующих операций с квадратными матрицами: 1) сложение двух матриц; 2)

умножение одной матрицы на другую; 3) нахождение транспонированной матрицы; 4) вычисление определителя матрицы. Матрицу описать следующим образом: Const NMax = 10; Type Matrica = Array [1..NMax, 1..Nmax] Of Real; Используя этот модуль, решить следующие задачи: 1. Решить систему линейных уравнений N-го порядка ($2 \leq N \leq 10$) методом Крамера. 2. Задан массив величин типа Matrica. Отсортировать этот массив в порядке возрастания значений определителей матриц.

III. Реализовать в виде модуля набор подпрограмм для выполнения следующих операций над векторами на плоскости: 1) сложение; 2) вычитание; 3) скалярное умножение векторов; 4) умножение вектора на число; 5) длина вектора. Вектор представить следующим типом: Type Vector = Record X, Y : Real End; Используя этот модуль, решить задачи: 1. Дан массив A - массив векторов. Отсортировать его в порядке убывания длин векторов. 2. С помощью датчика случайных чисел сгенерировать 2N целых чисел. N пар этих чисел задают N точек координатной плоскости. Вывести номера тройки точек, которые являются координатами вершин треугольника с наибольшим углом.

IV. Реализовать в виде модуля набор подпрограмм для выполнения следующих операций над натуральными числами в P-ичной системе счисления ($2 \leq P \leq 9$): 1) сложение; 2) вычитание; 3) умножение; 4) деление; 5) перевод из десятичной системы счисления в P-ичную; 6) перевод из P-ичной системы счисления в десятичную; 7) логическая функция проверки правильности записи числа в P-ичной системе счисления; 8) функции, реализующие операции отношения (равно, не равно, больше или равно, меньше или равно, больше, меньше). P-ичное число представить следующим типом: Type Chislo = Array [1..64] Of 0..8; Используя этот модуль, решить задачи: 1. Возвести число в степень (основание и показатель степени записаны в P-ичной системе счисления). Ответ выдать в P-ичной и десятичной системах счисления. 2. Дан массив A - массив чисел, записанных в P-ичной системе счисления. Отсортировать его в порядке убывания. Ответ выдать в P-ичной и десятичной системах счисления.

V. Реализовать в виде модуля набор подпрограмм для выполнения следующих операций над натуральными числами в шестнадцатеричной системе счисления: 1) сложение; 2) вычитание; 3) умножение; 4) деление; 5) перевод из двоичной системы счисления в шестнадцатеричную; 6) перевод из шестнадцатеричной системы счисления в десятичную; 7) функция проверки правильности записи числа в шестнадцатеричной системе счисления; 8) функции, реализующие операции отношения (равно, не равно, больше или равно, меньше или равно, больше, меньше). Используя этот модуль, решить задачи: 1. Возвести число в степень (основание и показатель степени записаны в шестнадцатеричной системе счисления). Ответ выдать в шестнадцатеричной и десятичной системах счисления. 2. Дан массив A - массив чисел, записанных в шестнадцатеричной системе счисления. Отсортировать его в порядке убывания. Ответ выдать в шестнадцатеричной и десятичной системах счисления.

VI. Определим граф как набор точек, некоторые из которых соединены отрезками, подграф - граф, подмножество данного графа. Реализовать в виде модуля набор подпрограмм, определяющих: 1) число точек в графе; 2) число отрезков в графе; 3) число изолированных подграфов в графе (подграфов, не соединенных отрезками); 4) диаметр графа - длину максимальной незамкнутой линии в графе (длина каждого звена - единица); 5) граф - объединение двух графов; 6) подграф - пересечение двух графов; 7) подграф - дополнение данного графа до полного (графа с тем же количеством вершин, что и в заданном, и с линиями между любыми двумя вершинами); 8) число отрезков, выходящих из каждой вершины графа; 9) при запуске должны инициализироваться переменные: Full_Graph - полный граф с числом вершин NumberOfVertex, Null_Graph - граф без отрезков с числом вершин NumberOfVertex. Граф представить как объект Const NumberOfVertex = 50; Type Graph = Array [1..NumberOfVertex, 1..NumberOfVertex] Of Boolean; Используя модуль, решить задачу: найти все правильные графы из N вершин (граф правилен, если из всех вершин выходит равное количество отрезков).

VII. Реализовать в виде модуля набор подпрограмм для работы с длинными целыми числами (числами, выходящими за диапазон допустимых значений любого целого типа): 1) сложение; 2) вычитание; 3) умножение; 4) нахождение частного и остатка от деления одного числа на другое; 5) функции, реализующие операции отношения (равно, не равно, больше или равно, меньше или равно, больше, меньше). Длинное число представить следующим типом: Type Tsifra = 0..9; Chislo = Array [1..1000] Of Tsifra; Используя этот модуль, решить задачи: 1. Возвести число в степень (основание и показатель степени - длинные числа). 2. Дан массив длинных чисел. Упорядочить этот массив в порядке убывания.

VIII. Реализовать в виде модуля набор подпрограмм для выполнения операций с многочленами от одной переменной (первый многочлен степени m, второй - степени n): 1) сложение; 2) вычитание; 3) умножение; 4) деление с остатком; 5) операции отношения (равно, не равно); 6) возведение в натуральную степень k одного из многочленов; 7) вычисление производной от многочлена; 8) вычисление значения в точке x0. Многочлен представить следующим типом: Type Mnogochlen = Array [1..500] Of Integer; Используя этот модуль, решить задачи: 1. Найти наибольший общий делитель многочленов P(x) и Q(x). 2. Вычислить: $P_s(x) - Q_r(x)$ (s, r - натуральные).

IX*. Реализовать в виде модуля набор подпрограмм для работы с длинными действительными числами (числами, выходящими за диапазон допустимых значений любого действительных типа или не представленных в памяти ЭВМ): 1) сложение; 2) вычитание; 3) умножение; 4) нахождение частного от деления одного числа на другое с заданным количеством знаков после запятой; 5) функции, реализующие операции отношения (равно, не равно, больше или равно, меньше или равно, больше, меньше); 6) тригонометрические функции, где аргументом и значениями являются длинные действительные числа (указание: использовать разложение соответствующей функции в ряд). Длинное действительное число представить следующим типом: Type Tsifra = 0..9; Chislo = Array [1..1000] Of Tsifra; LongReal = Record Znak : 0..1; {0 - "плюс", 1 - "минус"} Ts, Dr : Chislo {целая и дробная части} End; Используя этот модуль, решить задачи: 1. Возвести число в степень (основание - длинное действительное, показатель степени - длинное целое число). 2. Дан массив длинных действительных чисел. Упорядочить этот массив в порядке возрастания

Задания для практического занятия:

1. Рассмотреть типовые примеры
2. Написать программы для решения задач в соответствии с вариантом.
3. Составить отчет

Инструкция по выполнению практической работы

1. Рассмотрите примеры из теоретической части.
2. Выполнить примеры.
3. Выполните задания для самостоятельной работы.

Методика анализа результатов, полученных в ходе практической работы

1. В результате выполнения практических заданий у Вас должны быть работающие программы и оформлен отчет по каждому заданию.

Порядок выполнения отчета по практической работе

1. Разработать программы в среде ABC Pascal.
2. Составить систему тестов и проверить работу программ на всех возможных значениях исходных данных.
3. Результаты выполнения практической работы оформить в виде отчета.

Раздел 4. Основные конструкции языков программирования

Тема 4.1 Указатели

Практическое занятие №16. Использование библиотеки подпрограмм.

Учебная цель: получить навыки разработки подпрограмм и функций.

Организационная форма занятия: практическое занятие в форме практической подготовки

Ход работы:

Задача 1: составить процедуру (подпрограмму) вычисления степени числа **a**:
 a^n , где **a**, **n** – целые числа.

Составим программу вычисления степени числа **a** на языке программирования Паскаль ABC:

```
Program STEPEN;  
Var p,a,n,i:integer;  
Begin  
  p:=1; n:=7;  
  writeln ('a=');  
  read (a)  
  for i:=1 to n do p:=p*a;  
  write ('степень числа a = ', p);  
end.
```

Пояснения

В практике программирования часто складываются ситуации, когда одну и ту же группу операторов, реализующих определенную цель, требуется повторить без изменений в нескольких местах программы. Для избавления от столь нерациональной траты времени было предложено использовать подпрограммы.

Подпрограмма – именованная, логически законченная группа операторов языка, которую можно вызвать для выполнения любое количество раз из различных мест программы. В языке Паскаль ABC существует 2 вида подпрограмм: **процедуры** и **функции**.

Главное отличие процедур от функций заключается в том, что результатом последних является одно единственное значение.

Каждая новая процедура или функция должна быть предварительно описана в разделе описаний процедур и функций. Для использования процедуры необходимо написать оператор вызова. Описание процедуры состоит из заголовка процедуры и ее тела. Заголовок процедуры включает служебное слово **procedure**, имя процедуры и заключенный в круглые скобки список формальных параметров **с указанием их типов**:

Procedure <имя> (<список формальных параметров>);

Например:

Procedure a17 (a:real; b,c:real; var x1,x2:real; var k:integer);

Формальные параметры отделяются точкой с запятой. После заголовка идут разделы описаний (констант, типов, переменных, процедур и функций, используемых в процедуре) и операторы языка Паскаль ABC, реализующие алгоритм процедуры. Формальные параметры **нельзя** описывать в разделе описания процедуры.

Для обращения к процедуре необходимо использовать оператор вызова процедуры. Он имеет вид:

<имя процедуры> (<список_фактических_параметров>);

Например:

STEPEN (b, m, j, h);

Фактические параметры в списке отделяются друг от друга запятой. Механизм применения формальных – фактических параметров обеспечивает замену формальных параметров фактическими, что позволяет выполнять процедуру с различными данными. Между фактическими параметрами в операторе вызова процедуры и формальными параметрами в заголовке процедуры устанавливается взаимно однозначное соответствие.

Количество, типы и порядок следования формальных и фактических параметров должны совпадать.

Пример использования процедуры в программе приведен ниже:

Program PROCED1;

Var b, m, j, h:integer;

Procedure s (a,n,i:integer; var p:integer);

Begin

p:=1; n:=7;

for i:=1 to n do p:=p*a;

end;

begin

writeln ('b=');

read (b)

s (b, m, j, h)

write ('степень числа b = ', h);

end.

Функции. В Турбо Паскале есть стандартные функции (sin, ln, exp . . .), кроме них программист может воспользоваться собственными функциями, предварительно описав их в программе.

Формат описания функции:

function<имя функции> (<имя формального параметра 1>:<Тип>;
<имя формального параметра 2>:<Тип>):тип результата;

Обращение к функции осуществляется по имени с указанием списка фактических параметров. Количество, типы и порядок следования формальных и фактических параметров должны совпадать:

<имя функции> (<список_фактических_параметров>);

Пример описания функций: **function tan (x:real):real;**

function max (x,y:real):real;

ПОРЯДОК ВЫПОЛНЕНИЯ ЗАДАНИЯ НА КОМПЬЮТЕРЕ.

1. Запустить среду программирования Паскаль ABC, щелкнув 2 раза ЛКМ на ярлыке.
2. В окне редактора введите текст программы **PROCED1**.
3. Сохранить текст программы, для добавиться автоматически.
4. Выполнить компилирование. Если в программе есть ошибки, то появится сообщение об ошибке, код ошибки, а текстовый курсор покажет место ошибки.

5. Запустить программу на выполнение, для чего выбрать команду **Run – Run** (или нажать одновременно клавиши **Ctrl и F9**).

Откроется пользовательский экран, в котором нужно ввести исходные данные:

1). **B = 2;** 2). **B = 5;** 3). **B = 12.**

Для возврата к окну с текстом программы используется клавиша **Esc**.

6. Показать результат выполнения программы преподавателю, **записать результат в тетрадь**.

Задача 2: составить процедуру (подпрограмму) вычисления суммы двух чисел **a** и **b**:
 $c = a + b$, где **a, b, c** – вещественные числа.

Составим на языке программирования Паскаль ABC программу вычисления суммы двух чисел с использованием процедуры:

```
Program PROCED2;  
Var x,y,z:real;  
Procedure SUM(a,b:real;var c:real);  
Begin  
    c:=a+b;  
end;  
begin  
    writeln ('x=');  
    read (x)  
    writeln ('y=');  
    read (y);  
    SUM(x,y,z);  
    writeln ('z = ',z);  
end.
```

ПОРЯДОК ВЫПОЛНЕНИЯ ЗАДАНИЯ НА КОМПЬЮТЕРЕ.

1. Запустить среду программирования Паскаль ABC, щелкнув 2 раза ЛКМ на ярлыке.
2. В окне редактора введите текст программы **PROCED1**.
3. Сохранить текст программы, для добавиться автоматически.
4. Выполнить компилирование. Если в программе есть ошибки, то появится сообщение об ошибке, код ошибки, а текстовый курсор покажет место ошибки.
5. Запустить программу на выполнение, для чего выбрать команду **Run – Run** (или нажать одновременно клавиши **Ctrl и F9**).

Откроется пользовательский экран, в котором нужно ввести исходные данные:

x = 457; y = 525.

Для возврата к окну с текстом программы используется клавиша **Esc**.

6. Показать результат выполнения программы преподавателю, **записать результат в тетрадь**.

Задача 3: разработать на языке программирования Паскаль ABC функцию вычисления суммы двух чисел **a** и **b**:
 $c = a + b$, где **a, b, c** – вещественные числа.

Составим программу вычисления функции суммирования двух чисел на языке программирования Паскаль ABC:

```

Program FUNC1;
Var x,y:real;
Function SUM(a,b:real):real;
Begin
    SUM:=a+b;
    end;
    begin
    writeln ('x=');
    read (x)
    writeln ('y=');
    read (y);
    writeln (SUM(x,y));
end.

```

ЗАДАНИЕ 2 (для самостоятельного выполнения)

Цель задания: *закрепить навыки программирования задач с использованием процедур и функций, а также навыки работы в среде программирования Паскаль ABC.*

Задача 4: составить процедуру (подпрограмму) вычисления произведения трех чисел **a, b, c**:

$z = a * b * c$, где **a, b, c** – вещественные числа.

ПОРЯДОК ВЫПОЛНЕНИЯ ЗАДАНИЯ

1. Самостоятельно составить на языке программирования Паскаль ABC программу вычисления произведения трех чисел с использованием процедуры.
2. Ввести текст программы, сохранить программу.
3. Выполнить программу, введя следующие исходные данные:

a (k) = 25.7; b (m) = 12.5; c (n) = 33.

4. Показать результат выполнения программы преподавателю, **записать результат в тетрадь.**

Задача 5: разработать на языке программирования Паскаль ABC функцию вычисления произведения трех чисел **a, b, c**:

$z = a * b * c$, где **a, b, c** – вещественные числа.

ПОРЯДОК ВЫПОЛНЕНИЯ ЗАДАНИЯ

1. Самостоятельно составить на языке программирования Паскаль ABC программу вычисления произведения трех чисел с использованием функции.
2. Ввести текст программы, сохранить программу.
3. Выполнить программу, введя следующие исходные данные:

a (k) = 25.7; b (m) = 12.5; c (n) = 33.

4. Показать результат выполнения программы преподавателю, **записать результат в тетрадь.**

Задача 6: разработать на языке программирования Паскаль ABC функцию вычисления $\text{ctg}(x)$.

ПОРЯДОК ВЫПОЛНЕНИЯ ЗАДАНИЯ

1. Самостоятельно составить на языке программирования Паскаль ABC программу вычисления **ctg(x)** с использованием функции.
2. Ввести текст программы, сохранить программу.
3. Выполнить программу, введя следующие исходные данные:

$$c(x) = 30.$$

1. Показать результат выполнения программы преподавателю, **записать результат в тетрадь.**

Контрольные вопросы

1. Что называется подпрограммой?
2. Какие виды подпрограмм используются в языке программирования Паскаль ABC?
3. Чем отличается процедура от функции?
4. Как описывается процедура?
5. Как описывается функция?
6. Опишите оператор вызова процедуры?
7. Опишите оператор вызова функции?

Раздел 5. Объектно-ориентированное программирование

Тема 5.1 Основные принципы объектно-ориентированного программирования (ООП)

Практическое занятие №17. Классы объектов. Компоненты и их свойства.

Учебная цель: научиться создавать приложения с использованием стандартных компонент.

Организационная форма занятия: практическое занятие в форме практической подготовки

Ход работы:

Задание 1. Создайте новое приложение, которое запрашивает вес и рост, вычисляет оптимальное значение, сравнивает его с реальным весом и выводит соответствующее сообщение.

1. Создайте новый проект. Сохраните созданный проект.
2. Перенесите на форму нужные компоненты. Задайте необходимые свойства. Создайте обработчик события:

```
procedure Form1.button1_Click(sender: Object; e: EventArgs);
```

```
var
```

```
  w: real; { вес }
```

```
  h: real; { рост }
```

```
  opt: real; { оптимальный вес }
```

```
  d: real; { отклонение от оптимального веса }
```

```
begin
```

```
  w := StrToFloat(textBox1.Text);
```

```
  h := StrToInt(textBox2.Text);
```

```
  opt := h - 100;
```

```
  if w = opt then
```

```
    Label3.Text := 'Вы в хорошей форме!'
```

```
  else
```

```
    if w < opt then
```

```
      begin
```

```
        d := opt - w;
```

```
        Label3.Text := 'Вам надо поправиться, на '
```

```
        + FloatToStr(d) + 'кг.';
```

```
      end
```

```
    else
```

```
      begin
```

```
        d := w - opt;
```

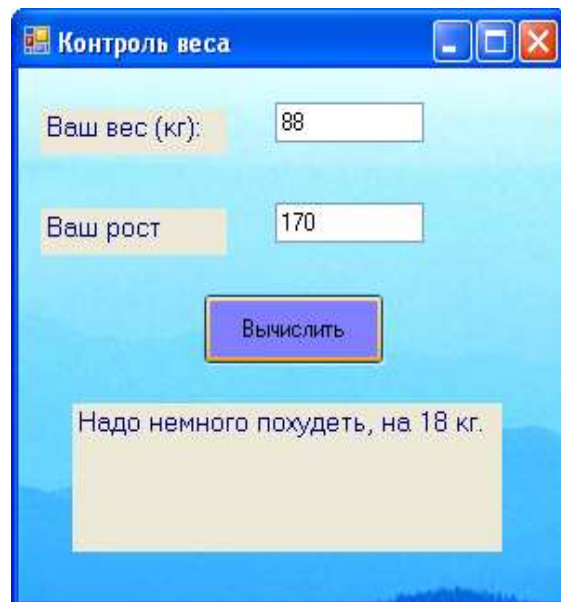
```
        Label3.Text := 'Надо немного похудеть, на '
```

```
        + FloatToStr(d) + 'кг.';
```

```
      end;
```

```
    end;
```

```
end.
```

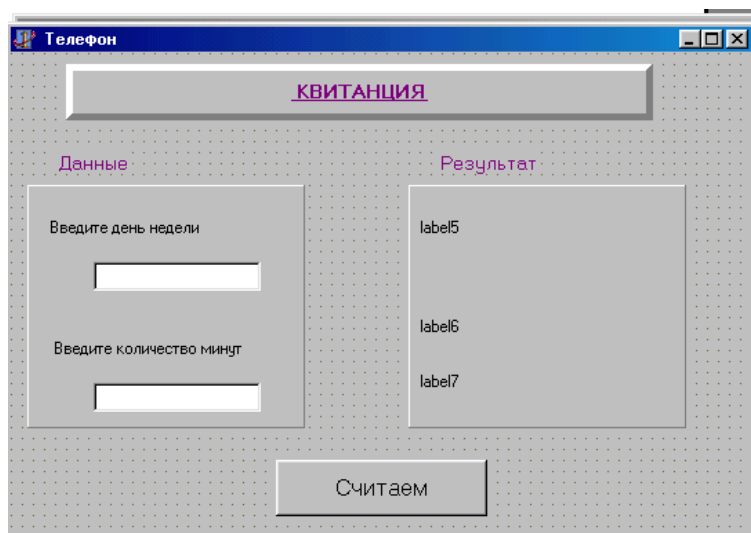


3. Запустите проект на выполнение. Измените свойства компонентов. Сохраните проект.

Задание 2. Создайте новое приложение, с помощью которого можно вычислить стоимость телефонного разговора. Стоимость разговора по телефону в воскресные дни ниже, чем в обычные дни. Программа должна запрашивать длительность разговора и день недели, затем вычислять стоимость разговора. Если день недели суббота или воскресенье, то стоимость уменьшается на величину скидки. Цена минуты разговора и величина скидки должны задаваться в тексте программы как константы.

Для ввода исходных значений (длительность разговора, номер дня недели) используются поля редактирования, для вывода результата и пояснительного текста — поля меток.

1. Создайте новый проект.
2. Разместите компоненты, как показано на рисунке.



3. Свяжите событие OnClick с кнопкой Считаем. Создайте обработчик этого события.
4. Запустите проект на исполнение и протестируйте его.

Задание 3. Создайте новое приложение, решающее задачу из вашего варианта.

Варианты.

1. Вычислить значение функции $y = x^2 + 8x - 5$ в заданной точке.
2. Вычислить стоимость 10 дисков, если стоимость одного диска 15 рублей, а при наличии дисконтной карты - скидка 15%.
3. Вывести название дня недели в зависимости от его номера.
4. Вычислить значение функции $y = \sin(x)$ в заданных точках.
5. Определить является ли введенное число кратным 3.
6. Даны три действительных положительных числа. Найти среднее геометрическое этих чисел.
7. Даны три действительных положительных числа. Найти среднее арифметическое этих чисел.
8. Даны катеты прямоугольного треугольника. Найти его гипотенузу и площадь.
9. Мальчик купил несколько тетрадей по 8 рублей и несколько обложек по 12 рублей. Составить программу, которая могла бы подсчитать стоимость всей покупки.
10. Заданы три числа X, Y, Z. Найти наибольшее из этих чисел.
11. Заданы три числа X, Y, Z. Найти наименьшее из этих чисел.
12. Вычислить площадь круга.
13. Вычислить стоимость поездки для семьи, в которой есть дети (скидка на детские билеты).
14. Вычислить значение функции $y = 12x - 18$ в заданной точке.

Контрольные вопросы.

1. Что такое ООП?
2. Что такое объект?
3. Что такое абстракция?
4. Как задать свойства объекта? Пример.
5. Как выделить группу компонентов? Зачем это нужно?
6. Какие операции можно производить с группой компонентов?
7. Что такое проект? Как создать проект? Как сохранить проект?
8. Что такое приложение?
9. Что такое форма? Как задать свойства формы?
10. Что такое событие? Обработчик события?
11. Как создать обработчик события?

Практическое занятие №18. Событийно-управляемая модель программирования. Компонентно-ориентированный подход.

Учебная цель: получить навыки о СОП.

Организационная форма занятия: практическое занятие в форме практической подготовки

Событийно-ориентированное программирование, СОП, создание событийно-управляемых программ, event-driven programming - парадигма программирования, в которой выполнение программы определяется событиями - действиями пользователя (клавиатура, мышь), сообщениями других программ и потоков, событиями операционной системы (например, поступлением сетевого пакета).

СОП можно также определить, как способ построения компьютерной программы, при котором в коде (как правило, в головной функции программы) явным образом выделяется главный цикл приложения, тело которого состоит из двух частей: выборки события и обработки события.

Как правило, в реальных задачах оказывается недопустимым длительное выполнение обработчика события, поскольку при этом программа не может реагировать на другие события. В связи с этим при написании событийно-ориентированных программ часто применяют автоматное программирование.

Сфера применения событийно-ориентированного программирования

Событийно-ориентированное программирование, как правило, применяется в следующих случаях:

- построение пользовательских интерфейсов (в том числе графических);
- создание серверных приложений;
- моделирование сложных систем;
- параллельные вычисления;
- автоматические системы управления, SCADA;
- программирование игр, в которых осуществляется управление множеством объектов.

Применение событийно-ориентированного программирования в серверных приложениях

Событийно-ориентированное программирование применяется в серверных приложениях для решения проблемы масштабирования на 10 тыс. одновременных соединений и более. Обычно это делается в среде версионных, а не транзакционных систем.

В серверах, построенных по модели «один поток на соединение», проблемы с масштабируемостью возникают по следующим причинам:

- слишком велики накладные расходы на структуры данных операционной системы, необходимые для описания одной задачи (сегмент состояния задачи, стек);
- слишком велики накладные расходы на переключение контекстов.

Серверное приложение при событийно-ориентированном программировании реализуется на системном вызове, получающем события одновременно от многих дескрипторов (мультиплексирование). При обработке событий используются исключительно неблокирующие операции ввода-вывода, чтобы ни один дескриптор не препятствовал обработке событий от других дескрипторов.

Мультиплексирование в событийно-ориентированного программирования

Для мультиплексирования соединений могут быть использованы следующие средства операционной системы:

- `select` (большинство UNIX систем). Плохо масштабируется, из-за того, что список дескрипторов представлен в виде битовой карты;
- `poll` и `epoll` (Linux);
- `kqueue` (FreeBSD);
- `/dev/poll` (Solaris);
- `IO completion port` (Windows);
- `POSIX AIO` на текущий момент только для операций дискового ввода-вывода;
- `io submit` и `eventfd` для операций дискового ввода-вывода.

Примеры реализаций событийно-ориентированного программирования

- Веб-серверы: `Node.js`, `nginx`, `lighttpd`
- Прокси-серверы: `Squid`

Применение событийно-ориентированного программирования в настольных приложениях

В современных языках программирования события и обработчики событий являются центральным звеном реализации графического интерфейса пользователя. Рассмотрим, к примеру, взаимодействие программы с событиями от мыши. Нажатие правой клавиши мыши вызывает системное прерывание, запускающее определенную процедуру внутри операционной системы. В этой процедуре происходит поиск окна, находящегося под курсором мыши. Если окно найдено, то данное событие посылается в очередь обработки сообщений этого окна. Далее, в зависимости от типа окна, могут генерироваться дополнительные события. Например, если окно является кнопкой (в Windows все графические элементы являются окнами), то дополнительно генерируется событие нажатия на кнопку. Отличие последнего события в том, что оно более абстрактно, а именно, не содержит координат курсора, а говорит просто о том, что было произведено нажатие на данную кнопку.

Обработчик события может выглядеть следующим образом (на примере C#):

```
private void button1_Click(object sender, EventArgs e)
```

```
{
    MessageBox.Show("Была нажата кнопка");
}
```

Здесь обработчик события представляет собой процедуру, в которую передается параметр `sender`, как правило, содержащий указатель на источник события. Это позволяет использовать одну и ту же процедуру для обработки событий от нескольких кнопок, различая их по этому параметру.

Языки событийно-ориентированного программирования

В языке C# события реализованы как элемент языка и являются членами классов. Механизм событий здесь реализует шаблон проектирования `Publisher/Subscriber`. Пример объявления события:

```
public class MyClass
```

```
{
    public event EventHandler MyEvent;
}
```

Здесь **EventHandler** — делегат, определяющий тип процедуры обработчика событий. Подписка на событие производится следующим образом:

```
myClass.MyEvent += new EventHandler(Handler);
```

Здесь **myClass** — экземпляр класса **MyClass**, **Handler** — процедура-обработчик. Событие может иметь неограниченное количество обработчиков. При добавлении обработчика события он добавляется в специальный стек, а при возникновении события вызываются все обработчики по их порядку в стеке. Отписка от события, то есть удаление обработчика производится аналогично, но с использованием оператора «-=».

Разные языки программирования поддерживают СОП в разной степени. Наиболее полной поддержкой событий обладают следующие языки (неполный список):

- Perl (события и демоны DAEMON, и их приоритеты PRIO),
- PHP
- Java,
- Delphi (язык программирования),
- ActionScript 3.0,
- C# (события event),
- JavaScript (действия пользователя).

Остальные языки, в большей их части, поддерживают события как обработку исключительных ситуаций.

Компонентное программирование.

Пожалуй, наиболее существенным нововведением идеологии Microsoft .NET является компонентно-ориентированный подход к программированию.

Во-первых, следует отметить то обстоятельство, что компонентно-ориентированный подход к проектированию и реализации программных систем и комплексов является в некотором смысле развитием объектно-ориентированного и практически более пригоден для разработки крупных и распределенных систем (например, корпоративных приложений).

Прежде всего, сформулируем основополагающее для рассматриваемого подхода определение компонента. Под компонентом будем далее иметь в виду независимый модуль программного кода, предназначенный для повторного использования и развертывания. Как видно из определения, применение компонентного программирования призвано обеспечить более простую, быструю и прямолинейную процедуру первоначальной инсталляции прикладного программного обеспечения, а также увеличить процент повторного использования кода, т.е. усилить основные преимущества ООП.

Говоря о свойствах компонентов, следует прежде всего отметить, что это существенно более крупные единицы, чем объекты (в том смысле, что объект представляет собой конструкцию уровня языка программирования). Другими отличиями компонентов от традиционных объектов являются возможность содержать множественные классы и (в большинстве случаев) независимость от языка программирования.

Заметим, что, автор и пользователь компонента, вообще говоря, территориально распределены и используют разные языки. Вполне возможно, что они не только пишут программы, но и говорят на разных языках.

Получив представление о компонентах и их отличиях от традиционных объектов ООП, рассмотрим существующие подходы к моделированию вариаций компонентного подхода в современной практике проектирования и реализации программных комплексов и систем.

Прежде всего, поскольку основной вычислительной средой для исследования в данном курсе является Microsoft .NET, обсудим особенности модели для компонентно-ориентированной разработки программного обеспечения, предложенной корпорацией Microsoft. Эта модель называется компонентной объектной моделью (или Component Object Model, COM) и является изначальным стандартом, принятым для компонентной разработки приложений в корпорации Microsoft.

Компонентная модель COM определяет протокол для конкретизации (т.е. создания экземпляров) и использования компонент (по аналогии с классами и объектами) как внутри одного и того же процесса, так и между различными процессами или компьютерами,

предназначенными для выполнения того или иного программного проекта, основанного на компонентной технологии. Модель COM является достаточно универсальной и используется в качестве фундамента для таких технологий проектирования и реализации программного обеспечения, как ActiveX, OLE и целого ряда других технологий. Приложения для COM-модели могут создаваться средствами таких языков и сред разработки как Visual Basic, C++, .NET и т.д.

Другим известным стандартом для компонентной модели является стандарт компании Sun Microsystems, известный как JavaBeans, который не обладает свойством независимости от языка программирования. Еще один широко используемый стандарт компонентного программирования - архитектура объектных запросов CORBA (несмотря на поддержку многоязычной разработки приложений, существуют сложности, связанные с отображением одного языка реализации в другой; для этой цели применяется достаточно громоздкий интерфейс, основанный на специальном языке описания IDL).

В рамках компонентного подхода сборкой называется логическая единица, содержащая множество модулей, необходимых для осуществления инсталляции программного обеспечения. Сборка характеризуется уникальностью, которая обеспечивается идентификатором версии сборки и цифровой подписью автора. Сборка является самодостаточной единицей для установки программного обеспечения и не требует никаких дополнений. Возможно, как индивидуальное, так и коллективное (сетевое) использование сборки на основе компонентной технологии. Сборка обеспечивает простой и удобный механизм инсталляции и экономит средства, необходимые для развертывания программного обеспечения, сводя к минимуму затраты времени и сил на установку.

Описание сборки содержится в так называемом манифесте, где хранятся метаданные о компонентах сборки, идентификация автора и версии, сведения о типах и зависимостях, а также режим и политика использования. Метаданные типов манифеста исчерпывающе описывают все типы, определенные в сборке, а именно, свойства, методы, аргументы, возвращаемые значения, атрибуты, базовые классы и т.д.

Заметим, что промежуточный язык IL всегда компилируется в естественный (native) код до выполнения программы.

Для более эффективного манипулирования системой типизации компонент создаваемого программного обеспечения в рамках модели COM, концепция .NET предусматривает механизм пространств имен (namespace).

Пространством имен будем называть механизм среды Microsoft .NET, предназначенный для идентификации типов объектов языков программирования и среды реализации.

Описания пространств имен по аналогии с описаниями типов данных размещаются в файлах. Перечислим основные свойства, которыми характеризуются пространства имен в среде Microsoft .NET. Прежде всего, пространства имен могут как объединять различные сборки, так и быть вложенными друг в друга. Кроме того, файлы с описаниями могут содержать множественные пространства имен. Важно отметить, что между пространствами имен и файлами не существует однозначного соответствия. Наконец, полное имя типа должно содержать все необходимые пространства имен.

Тема 5.2. Интегрированная среда разработчика

Практическое занятие №19. Изучение интегрированной среды разработчика.

Учебная цель: изучить интегрированную среду Lazarus.

Организационная форма занятия: практическое занятие в форме практической подготовки

Lazarus это бесплатный инструмент разработки с открытым кодом. Он представляет собой среду с графическим интерфейсом для быстрой разработки программ, аналогичную Delphi, и базируется на оригинальной кроссплатформенной библиотеке визуальных компонентов Lazarus Component Library, совместимых с Delphi. В состав входят и не визуальные компоненты. Такого набора достаточно для создания программ с графическим интерфейсом и приложений, работающих с базами данных и Интернетом. В среде Lazarus используются собственный формат управления пакетами и свои файлы проектов.

Lazarus это стабильная богатая возможностями среда разработки для создания самостоятельных графических и консольных приложений. В настоящее время она работает на Linux, FreeBSD и Windows и предоставляет настраиваемый редактор кода и визуальную среду создания форм вместе с менеджером пакетов, отладчиком и графическим интерфейсом, полностью интегрированным с компилятором FreePascal.

Чтобы создать проект в среде Lazarus, необходимо выбрать пункт «Создать проект» меню «Проект». Выбрав пункт «Приложение» (графическая программа с поддержкой LCL), Вы создадите типичное оконное приложение. Общий вид среды после создания пустого проекта показан на рисунке 3.

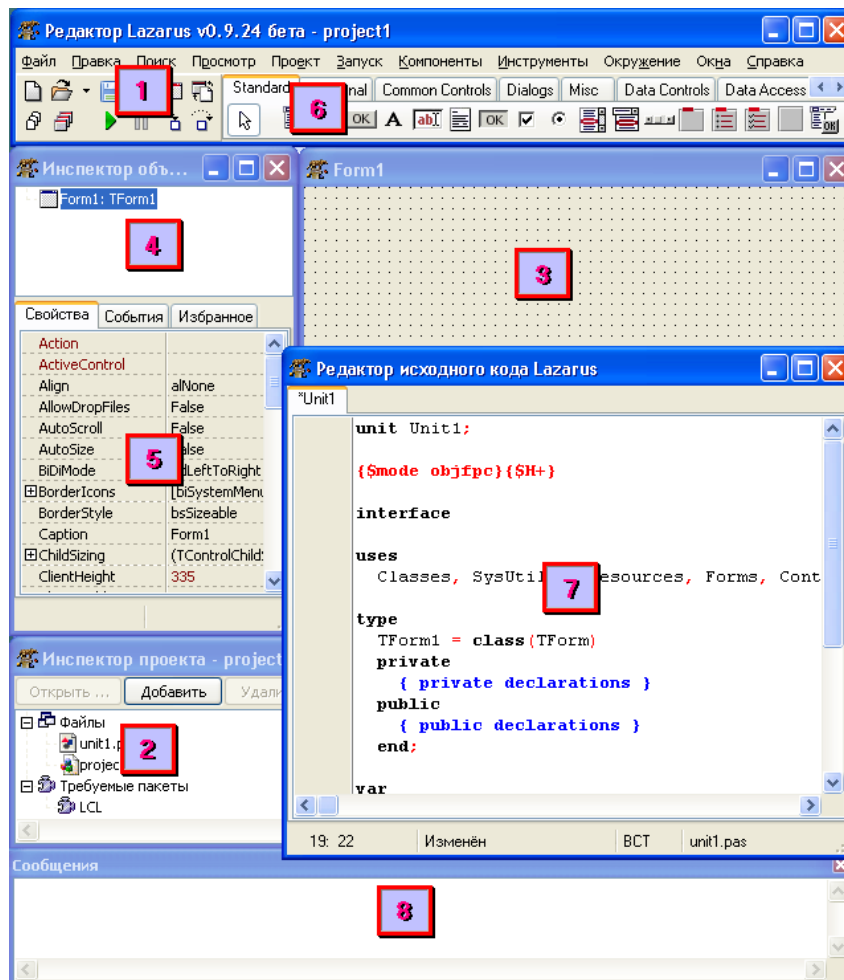


Рисунок 1. Основные окна среды Lazarus

Рассмотрим назначение каждой из отмеченных областей:

1. Основная панель инструментов и меню

Меню содержит все основные команды среды Lazarus, сгруппированные по их предназначению (работа с файлами, отладка, компиляция, справка и т. п.). На панель инструментов вынесены наиболее часто используемые пункты меню для упрощения доступа к ним – это команды сохранения и открытия, запуска, останова, трассировки и пр.

2. Инспектор проекта

Это окно содержит инструменты для управления файловым составом проекта. С точки зрения среды Lazarus, любое создаваемое приложение является проектом. Проект – это файл, содержащий в себе информацию, необходимую для компиляции сложной программы, состоящей из нескольких модулей. Проект Lazarus имеет расширение LPR. В нашем случае проект состоит только из двух файлов – файла проекта и файла формы Form1. В серьёзных решениях число файлов в проекте может быть очень большим. Помимо прочего, инспектор проекта также позволяет показать его зависимости от

внешних пакетов. В нашем случае, project1 требует наличия LCL – библиотеки оконных компонентов.

3. Макет пользовательской формы

Это – дизайнер форм, позволяющий визуально создавать графический интерфейс пользователя (GUI) для Вашей программы. Дизайнер форм имеет ряд удобных функций, таких как, например, выравнивание элементов управления относительно друг друга.

4. Дерево объектов инспектора объектов

Эта область инспектора объектов отображает иерархию элементов управления пользовательской формы. Она может использоваться для наглядного представления объектной структуры и для выделения объектов, расположенных на форме так, что выделить их мышью непосредственно не удаётся.

5. Редактор свойств выбранного элемента управления

Эта область инспектора объектов позволяет программисту задавать или просматривать свойства выбранного элемента во время проектирования приложения (design-time). Следует отметить, что редактор свойств позволяет просматривать и задавать свойства, имеющие агрегатный тип (шрифт, рисунок, дочерний объект и т.п.). Также редактор свойств используется для назначения обработчиков событий элементам управления. Наиболее часто используемые свойства и события для элементов управления для удобства вынесены на особую вкладку – «Избранное». Высоту строк списка свойств можно менять в настройках среды Lazarus (меню «Окружение»/«Настройки окружения»/«Инспектор объектов»/«Высота элемента»).

6. Палитра элементов управления

Эта область IDE содержит набор элементов управления, размещаемых на пользовательских формах, распределённых по логическим группам в соответствии с их функциональным назначением.

7. Редактор кода

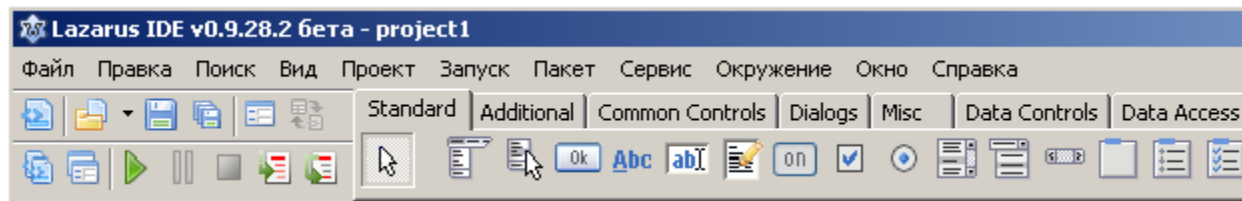
Это окно позволяет редактировать исходный код Вашего приложения. Редактор кода в Lazarus весьма удобен. К числу полезных особенностей можно отнести наличие возможности скрывать код процедур (и вообще содержимое операторных скобок begin..end), что существенно упрощает поиск нужного кода.

8. Окно сообщений

Это окно содержит информацию о результатах компиляции проекта. Здесь Вы можете получить список ошибок в коде Вашего приложения, а также список предупреждений и подсказок компилятора.

Рассмотрим основные элементы среды разработки Lazarus подробнее. Среда Lazarus состоит из нескольких, вообще говоря, не связанных окон.

1. Главное окно.

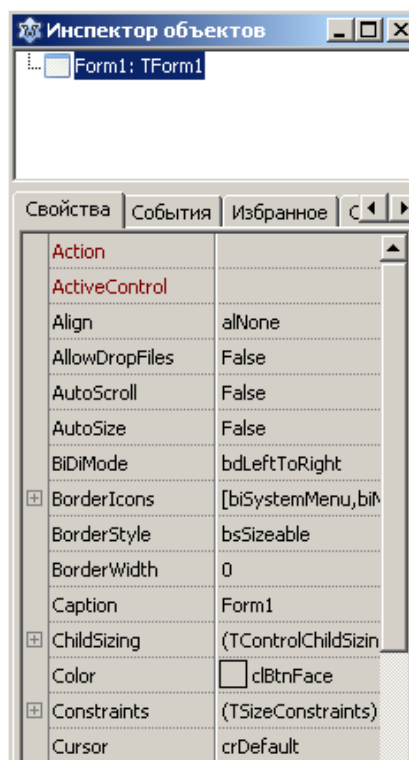


С помощью этого окна можно управлять процессом разработки приложения. В нем предусмотрены команды управления файлами, компиляцией, редактированием, окнами и т.д. Окно разбито на три функциональных блока: Главное меню. В нем расположены команды управления файлами, команды управления компиляцией и свойствами всего приложения, команды управления окнами и настройками среды и многое другое. Меню располагается в верхней части основного окна.

Панель инструментов. Панель инструментов предоставляет быстрый доступ к основным командам главного меню. Она расположена в левой части главного окна, под главным меню.

Палитра компонентов. Предоставляет доступ к основным компонентам среды разработки, например: поле ввода, надпись, меню, кнопка и т.п.

2. Инспектор объектов.



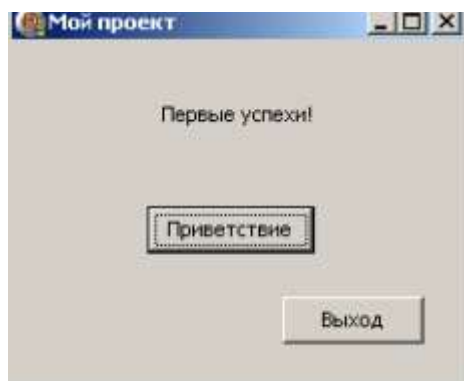
В верхней части окна показывается иерархия объектов, а снизу, расположены три вкладки: "Свойства", "События", "Избранное". Назначение инспектора объекта – это

просмотр всех свойств и методов объектов. На вкладке "Свойства" перечисляются все свойства выбранного объекта. На вкладке "События" перечисляются все события для объекта. На вкладке "Избранное" избранные свойства и методы.

Практическое занятие №20. Создание простого проекта.

Учебная цель: создать программу, выполняющую следующие действия. После запуска программы по щелчку мышью на кнопке «Приветствие» появляется сообщение «Первые успехи!». Для выхода из программы необходимо щелкнуть мышью на кнопке «Выход».

Организационная форма занятия: практическое занятие в форме практической подготовки



Ход работы:

1. Откройте новый проект.
2. Разместите на форме экземпляры компонентов: метку Label и две кнопки Button.
3. Выделите метку Label, перейдите в Object Inspector (инспектор объектов) на вкладку Properties (Свойства), найдите свойство Caption (заголовок) и очистите Label1.
4. Выделите кнопку Button2, перейдите в Object Inspector на вкладку Properties, найдите свойство Caption и измените Button2 на заголовок «Выход». Дважды щелкните по кнопке, чтобы перейти в окно редактора исходного кода для данного объекта. Оказавшись в коде программы, точнее, в заготовке процедуры TForm1.Button2Click, напишите лишь одну команду Close; (обязательно поставьте точку с запятой после «Close»!).
5. Сохраните код программы и проект (для каждой программы необходимо создавать отдельную папку!).
6. Запустите программу, затем закройте окно проекта, щелкнув на кнопке «Выход».
7. Выделите форму, в свойстве Caption окна Object Inspector замените слово Form1 на «Мой проект». Это и будет заголовком основного окна программы.

8. Выделите кнопку Button1, в свойстве Caption окна Object Inspector замените слово Button1 на название кнопки «Приветствие». При необходимости увеличьте длину кнопки.

9. Перейдите в окно редактора кода для кнопки Button1. Введите следующий код:
Label1.Caption:= 'Первые успехи!';

Кавычки обязательно должны быть одиночными (на английской раскладке клавиша с буквой Э).

Эта команда присваивает метке Label1 заголовок «Первые успехи!».

10. Сохраните проект окончательно, запустите и протестируйте его.

Задание для самостоятельного выполнения.

1. Сделайте шрифт выводимой реплики «Первые успехи!» отличным от стандартного по виду, цвету и размеру.

Подсказка. В Object Inspector дважды щелкните на кнопку справа от названия свойства Font (шрифт), откроется окно выбора шрифта, его цвета и стиля.

2. Замените вид кнопки «Выход» на более привлекательный.

Подсказка. Для замены кнопки надо удалить существующую, а другую кнопку найдите в палитре компонентов на вкладке Additional. Она называется BitBtn. Затем измените ее вид с помощью свойства Kind.

3. Сделайте так, чтобы после нажатия кнопки «Приветствие» на экране появлялось сообщение «Первые и не последние!».

Подсказка. Измените значение свойства Caption метки Label1 при реакции кнопки Button1 на нажатие.

4. Запустите исполняемый файл не в среде Lazarus, а в Windows.

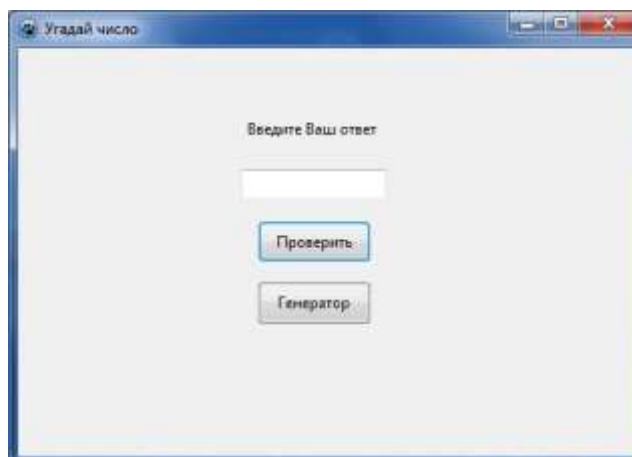
Подсказка. Выйдите из Lazarus в Windows. Используйте диспетчер программ или проводник Windows.

Тема 5.3. Визуальное событийно-управляемое программирование

Практическое занятие №21. Создание проекта с использованием кнопочных компонентов и компонентов для работы с текстом.

Учебная цель: создать программу, которая будет генерировать случайные числа от нуля до десяти и сверять их с введенными с клавиатуры.

Организационная форма занятия: практическое занятие в форме практической подготовки



Ход работы:

1. Откройте новый проект.
2. Разместите на форме экземпляры компонентов: две метки Label, окно редактирования Edit, и две кнопки Button.
3. Выделите метку Label1, перейдите в Object Inspector на вкладку Properties, найдите свойство Caption и измените Label1 на заголовок «Введите Ваш ответ».
4. Очистите содержимое свойства Caption у метки Label2.
5. Выделите окно редактирования Edit, перейдите в Object Inspector на вкладку Properties, найдите свойство Text и очистите его содержимое.
6. Аналогичным образом у кнопки Button1 в свойстве Caption прописать «Сгенерировать», у Button2 в свойстве Caption прописать «Проверить».
7. Перейдите в окно редактора кода для кнопки Button1. Введите следующий код:
Randomize;
a:=random(10)+1;
8. Перейдите в окно редактора кода для кнопки Button2. Введите следующий код:
b:=StrToInt(Edit1.Text);
if b>a then Label2.Caption:='Меньше';

```
if b<a then Label2.Caption:='Больше';  
if b=a then Label2.Caption:='Угадал';  
Edit1.Text:= ' ';
```

9. В окне редактора исходного кода найдите оператор описания переменных `var` и пропишите переменные `b`, `a`, присвоив им целочисленный тип данных (`Integer`):

```
var  
Form1: TForm1;  
a, b: Integer;
```

10. Выделите форму, в свойстве `Caption` окна `Object Inspector` замените слово `Form1` на «Угадай число».

11. Сохраните проект в новой папке, запустите и протестируйте его.

Задание для самостоятельного выполнения.

Используя генератор случайных чисел, напишите программу «Орел и решка», в которой при нажатии на кнопку в окне редактора `Edit` будет выдаваться «орел» или «решка».

Подсказка. Можно в генераторе сократить числовой интервал до 1 (0 и 1) и каждой цифре присвоить значение «орел» или «решка».

Практическое занятие №22. Создание проекта с использованием компонентов ввода и отображения чисел, дат и времени, и компонентов стандартных диалогов и системы меню.

Учебная цель: создать проект с использованием компонентов ввода и отображения чисел, дат и времени, и компонентов стандартных диалогов и системы меню.

Организационная форма занятия: практическое занятие в форме практической подготовки

Теоретическая часть:

Тип `TDateTime`

Для работы с датой и временем в Lazarus предусмотрен тип `TDateTime`. Это универсальный тип, он позволяет одновременно определить и дату, и время. `TDateTime` занимает 8 байт оперативной памяти и фактически, является вещественным числом (`TDateTime = Double`): целая часть этого числа содержит дату, а десятичная - время. Можно также для работы с датой и временем использовать типы соответственно, `TDate` и `TTime`, но это просто синонимы того же типа `TDateTime`.

В справочнике Lazarus говорится, что минимальное возможное значения даты-времени равно -693593.0, а максимальное +2958465.99999. Целая часть такого числа обозначает количество суток, а дробная - часть суток, прошедших с 0 часов. Прибавление или вычитание единицы из целой части числа равносильно прибавлению или вычитанию

одного дня. Число -693593 соответствует дате 01.01.0001, а число +2958465 соответствует дате 31.12.9999, в этом диапазоне вы и сможете работать с датами в Lazarus.

Компоненты для работы с датой-временем

На "Экранная заставка" нам уже довелось работать со временем, для вывода текущего времени на экран мы использовали простую метку TLabel. Но в Lazarus для ввода и вывода даты-времени имеются специальные компоненты. Знакомиться с этими компонентами будем сразу на примерах. Загрузите Lazarus с новым проектом, сразу же сохраните его в папку **12-01**.

Компоненты для работы с датой и временем располагаются на вкладке **Misc Палитры компонентов**. Вообще, очень интересные компоненты находятся на этой вкладке, и с некоторыми из них мы обязательно еще познакомимся. Сейчас же нас интересуют всего два компонента - календарь TCalendar и строка редактирования даты TDateEdit:

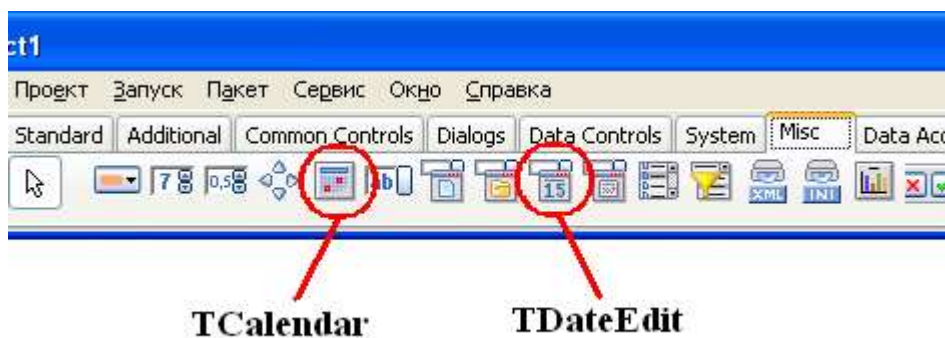


Рис. 12.1. Компоненты для работы с датой-временем TCalendar и TDateEdit

Правда, нужно заметить, компоненты эти еще сырые: некоторые свойства есть, но не работают, некоторые работают лишь частично, некоторых и вовсе нет. Так, в платном аналоге Lazarus - Delphi в компоненте TDateEdit есть возможность вводить как дату, так и время. В Lazarus можно вводить только дату. Для ввода времени, вероятно, придется использовать обычный TMaskEdit.

Поскольку Lazarus - проект новый, будем надеяться, что эти недоработки в будущем будут исправлены.

TCalendar

Вначале познакомимся с компонентом TCalendar. Установите его на форму, и он сразу примет вид календаря, причем текущая дата будет выделена. Выделение будет как красной окантовкой, так в календаре выделяется текущая дата, так и синей заливкой - так в календаре выделяется выбранная дата. А поскольку выбрать можно любую дату, то они не всегда совпадают. Однако из формы управлять календарем не очень удобно. Сохраните проект и запустите его. В рабочем приложении календарем можно управлять: щелкая по синим стрелкам в левой правой частях заголовка календаря можно листать месяцы, щелкнув по названию месяца можно открыть *список*, в котором выбрать другой месяц, щелкнув по году, можно выбрать или ввести вручную другой год.

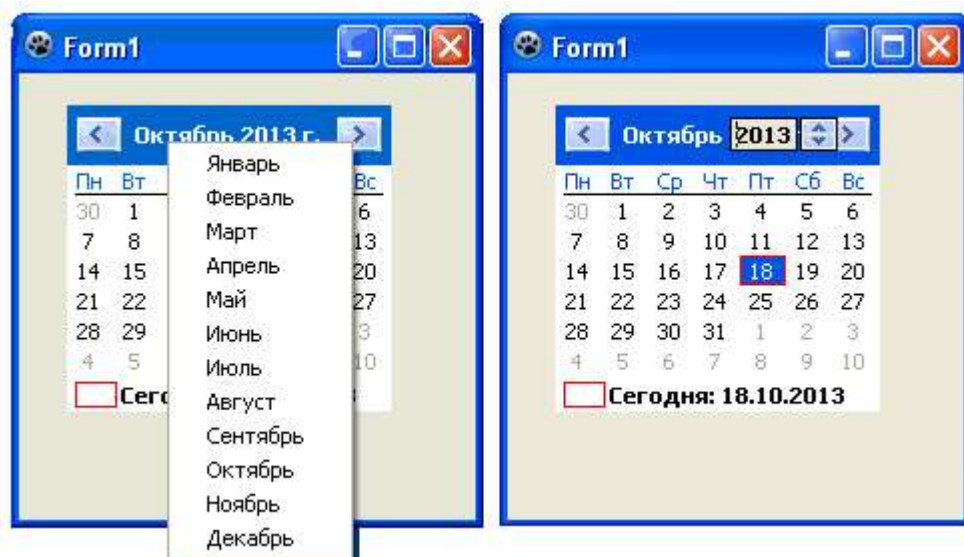


Рис. 12.2. Выбор месяца и года в календаре

Закройте запущенную программу и вернитесь в проект. Познакомимся с некоторыми полезными свойствами компонента TCalendar.

- Align** - выравнивание компонента относительно формы (или другого контейнера). С этим свойством вы уже знакомы, но в случае с календарем применять его не имеет смысла - если еще можно прижать календарь к одной из сторон, то изменить его размеры все равно не удастся.
- BorderSpacing** - раскрывающееся сложное свойство. По идее, оно должно настраивать параметры обрамления: ширину обрамления вокруг календаря или по одной из его сторон, но, увы, это свойство тоже не работает - как я не пытался что-то изменить, вид календаря остался прежним.
- Date и DateTime** - установленная в календаре дата. При изменении одного из этих свойств, изменится и другое, но есть и отличия: Date имеет тип String, а DateTime - TDateTime. Таким образом, если вам нужно получить дату в виде строки, обращайтесь к Date. Если надо работать с датой в форме TDateTime, а это часто бывает нужно, например, для расчета дат, то обращайтесь к TDateTime. Присвоить календарю новую дату можно как через первое, так и через второе свойство, но если вы будете менять дату с помощью Date, вы должны быть уверены, что дата в строке написана правильно.
- DisplaySettings** - свойство должно управлять видом календаря, но тоже почему-то не изменяет его. Имеет следующие подсвойства, разрешающие (при True) или запрещающие (при False) различные действия:
- dsNoMonthChange - Не разрешать изменять месяц
 - dsShowDayNames - Показывать имя дня недели
 - dsShowHeadings - Показывать заголовок
 - dsShowWeekNumbers - Показывать номер недели, в году их 52 (это подсвойство установить можно).
 - dsStartMonday - Начинать неделю с Понедельника (в английских и американских календарях неделю принято начинать с Воскресенья).

Перейдем к событиям компонента (переключитесь на вкладку **События** в **Инспекторе объектов**). Здесь нам могут понадобиться следующие события:

- OnChange - событие наступает при любом изменении в компоненте: изменили год, месяц или день.
- OnClick - щелкнули по компоненту мышью.
- OnDayChanged - изменился день.
- OnDbClick - дважды щелкнули по компоненту мышью.
- OnMonthChanged - изменился месяц.
- OnYearChanged - изменился год.

TDateEdit

Теперь изучим *компонент* TDateEdit. Установите его на форму, где-нибудь под календарем. Как видите, фактически *компонент* состоит из двух компонентов: поля для редактирования даты и кнопки, при нажатии на которую появляется календарь:



Рис. 12.3. Компонент TDateEdit

Теперь перейдите на вкладку **Свойства** в **Инспекторе объектов**. У компонента имеются следующие полезные свойства:

ButtonOnlyWhenFocused	- Показывать кнопку выбора даты только когда фокус ввода находится на компоненте. По умолчанию False, то есть, кнопку видно всегда. Честно говоря, не вижу смысла прятать эту кнопку.
ButtonWidth	- ширина кнопки. По умолчанию равна 23, но можно и изменить. Правда, опять непонятно зачем.
CalendarDisplaySettings	- аналогично свойству DisplaySettings календаря, и предназначено для того же: для изменения внешнего вида календаря, который появляется при нажатии на кнопку.
CancelCaption	- заголовок для кнопки Отмена в диалоговом окне Календарь . Надо полагать, речь идет о привязанном к компоненту диалоге TCalendarDialog (диалоги мы будем изучать в другой лекции). Мне лично не удалось найти, как диалог привязать к компоненту, поиски информации в Интернете также были безуспешны. Будем надеяться, в следующих версиях Lazarus этот компонент заработает, как надо.
DateOrder	- формат вывода даты. При выборе формата, отличного от doNone (по умолчанию), появляется маска даты. Может иметь следующие значения: • doDMY - последовательность даты: День, Месяц, Год. • doMDY - последовательность даты: Месяц, День, Год. • doNone - последовательность даты соответствует системной. Для русской Windows это обычно День, Месяц, Год. Маски в этом формате не предусмотрено, если пользователь вводит дату сам, не пользуясь выбором в календаре, то легко может ошибиться. Поэтому рекомендую использовать значение doDMY, соответствующее русскому представлению даты. • doYMD - последовательность даты: Год, Месяц, День.

DefaultToday	- если True, то в работающей программе компонент по умолчанию отображает текущую дату, если False, то не отображает ничего.
OKCaption	- заголовок для кнопки ОК в диалоговом окне Календарь . Проблемы те же, что и у CancelCaption.

В **Инспекторе объектов** почему то не отображаются следующие необходимые свойства:

Date - имеет тип TDateTime, содержит выбранную или введенную пользователем дату.
Text - имеет тип TCaption, который совместим с обычной строкой. Тут выбранная или введенная дата хранится в виде строки.

Этими свойствами часто придется пользоваться, чтобы узнать, какую же дату выбрал или ввел *пользователь*, так что запомните их.

Основные события, которые вам могут понадобиться, это OnChange (при изменении), OnClick (щелкнули мышью), OnKeyPress (нажали любую клавишу). События эти работают так же, как в других, уже известных вам компонентах, так что задерживаться тут мы не будем.

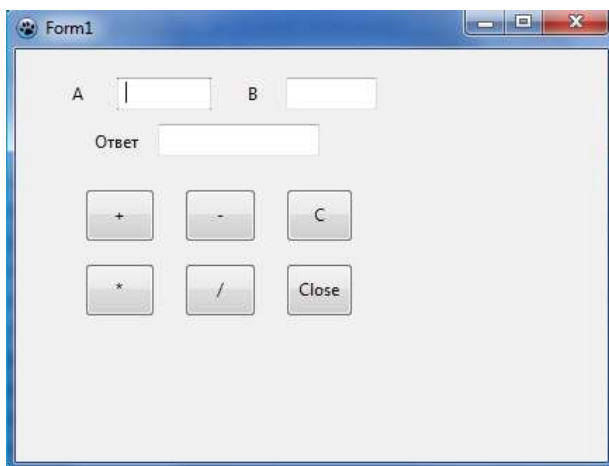
Задание: повторить процесс создания компонентов в IDE Лазарус!

Тема 5.4 Разработка оконного приложения

Практическое занятие №23. Разработка оконного приложения.

Учебная цель: создать программу, в которой по нажатию на кнопки будут производиться вычисления.

Организационная форма занятия: практическое занятие в форме практической подготовки



Ход работы:

1. Откройте новый проект.
 2. Разместите на форме экземпляры компонентов: 6 кнопок Button, 3 метки Label, 3 окна редактирования Edit.
 3. Переименуйте метки и кнопки согласно картинке выше и удалите весь текст в окнах редактирования.
- Напоминание. У объектов Button и Label за это отвечает свойство Caption, у объекта Edit – свойство Text.
4. В окне редактора исходного кода в операторе описания переменных Var пропишите переменные a, b, c и присвойте им тип extended.
 5. Выделите объект Button, который подписан как «+». Перейдите в окно редактора кода и пропишите следующий код:

```
a:=StrToFloat(edit1.Text);
```

```
b:=StrToFloat(edit2.Text);
```

```
c:=a+b;
```

```
edit3.Text:=FloatToStr(c);
```

Операция StrToFloat переводит строковый формат в вещественный. В этом коде мы присваиваем вводимым с клавиатуры исходным значениям вещественный тип данных,

выполняем операцию сложение, переводим результат операции обратно в строковый тип и выводим его в третьем окне редактирования.

6. Аналогичным образом прописываем операции вычитания, деления и умножения. *При делении на ноль программа будет ломаться. Поэтому необходимо задать условие, при котором будет выводиться сообщение (showmessage) об ошибке «На ноль делить нельзя». Помните, что русскоязычный текст в программе вводится через одиночные кавычки ' '.*

7. Выделите объект Button, который подписан как «С». Это кнопка очистки окон редактирования. Перейдите в окно редактора кода и пропишите следующий код:

```
Edit1.Text:=' ';
```

```
Edit2.Text:=' ';
```

```
Edit3.Text:=' ';
```

Объектам Edit присваивается пустое значение.

8. Пропишите код для кнопки закрытия формы.

9. Выделите форму, в свойстве Caption окна Object Inspector замените слово Form1 на «Калькулятор».

10. Сохраните проект в новой папке, запустите и протестируйте его.

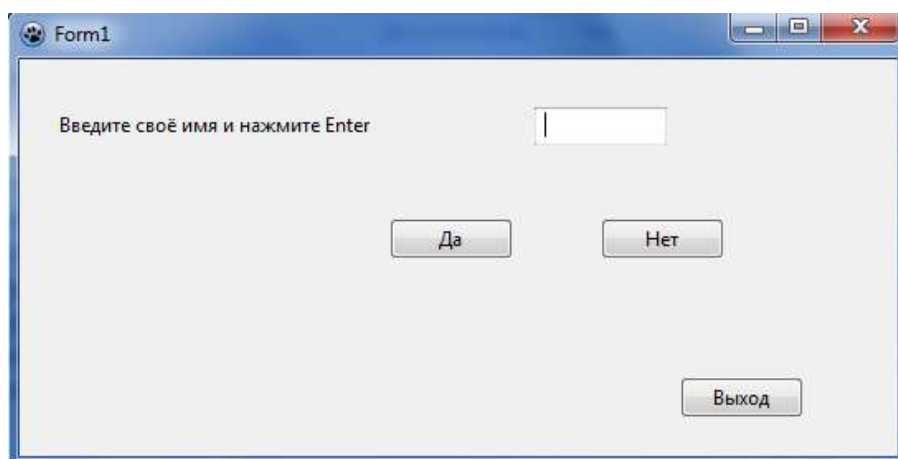
Задание для самостоятельного выполнения.

Добавьте в калькулятор кнопки для вычисления квадратного корня, возведения в квадрат, нахождения Sin и Cos.

Практическое занятие №24. Разработка оконного приложения с несколькими формами.

Учебная цель: создать программу, в которой будет происходить диалог компьютера с пользователем по нажатию на кнопки и Enter.

Организационная форма занятия: практическое занятие в форме практической подготовки



Ход работы:

1. Откройте новый проект.
2. Разместите на форме экземпляры компонентов: три метки Label, окно редактирования Edit, и три кнопки Button.
3. Выделите метку Label1 и в свойстве Caption измените заголовок на «Введите свое имя и нажмите Enter».
4. Очистите содержимое свойства Caption у меток Label2 и Label3.
5. Выделите окно редактирования Edit и очистите его содержимое.
6. Аналогичным образом у кнопки Button1 в свойстве Caption прописать «Да», у Button2 в свойстве Caption прописать «Нет», у Button3 прописать «Закреть».
7. Для кнопки Button1 пропишите код, чтобы по нажатию на нее в объекте Label3 появлялась надпись 'Молодец!'.
8. Для кнопки Button2 пропишите код, чтобы по нажатию на нее в объекте Label3 появлялась надпись 'Почему же? Надо читать!'.
9. Выделите окно редактирования Edit. В инспекторе объектов перейдите на вкладку «События». Дважды кликните в событии OnKeyPress.
10. Задайте код, по которому программа считывает введенное в окно редактирования имя и задает вопрос, обращаясь к пользователю: ', ты любишь читать?'

Ключ нажатия кнопки Enter – #13.

11. Пропишите у третьей кнопки закрытие формы
12. Выделите форму, в свойстве Caption окна Object Inspector замените слово Form1 на «Диалог».
13. Сохраните проект в новой папке, запустите и протестируйте его.

Тема 5.5 Этапы разработки приложений

Практическое занятие №25 Изучение этапов разработки

Учебная цель: Ознакомиться с основами разработки приложений с использованием среды Lazarus.

Организационная форма занятия: практическое занятие в форме практической подготовки.

Теоретическая часть

1. Что такое Lazarus?

Lazarus — это интегрированная среда разработки (IDE), предназначенная для создания кроссплатформенных приложений на языке Pascal (Object Pascal). Она обеспечивает визуальный интерфейс для проектирования и разработки приложений.

2. Основные компоненты Lazarus

- ****Форма****: Основной элемент, который служит контейнером для других компонентов.
- ****Компоненты****: Элементы управления, такие как кнопки, текстовые поля, метки и т.д.
- ****События****: Определенные действия, которые могут произойти в приложении (например, нажатие кнопки).

Практическая часть

Шаг 1: Установка Lazarus

1. Скачайте и установите Lazarus с официального сайта: [Lazarus](<https://www.lazarus-ide.org/>).
2. Следуйте инструкциям по установке и настройке среды.

Шаг 2: Создание нового проекта

1. Откройте Lazarus.
2. Выберите в меню ****File**** -> ****New**** -> ****Application****.
3. У вас появится новая форма (Form1) в редакторе.

Шаг 3: Проектирование интерфейса

1. На панели компонентов (Component Palette) выберите необходимые элементы управления:
 - ****TButton**** (кнопка)
 - ****TLabel**** (метка)
 - ****TEdit**** (текстовое поле)
2. Перетащите компоненты на форму и разместите их по своему усмотрению.
3. Настройте свойства компонентов:
 - Выделите компонент и в инспекторе объектов (Object Inspector) измените свойства (например, `'Caption'` для кнопки и метки).

Шаг 4: Написание кода

1. Дважды щелкните по кнопке на форме, чтобы создать обработчик события `'OnClick'`.
2. В открывшемся редакторе кода добавьте следующий код для изменения текста метки в зависимости от введенного в текстовое поле значения:

```
``pascal
procedure TForm1.Button1Click(Sender: TObject);
```

```
begin  
Label1.Caption := 'Вы ввели: ' + Edit1.Text;  
end;  
...
```

Шаг 5: Компиляция и тестирование приложения

1. Сохраните проект (File -> Save All).
2. Нажмите на кнопку ****Run**** (или нажмите F9) для компиляции и запуска приложения.
3. Введите текст в текстовое поле и нажмите кнопку, чтобы увидеть результат.

Шаг 6: Завершение работы

1. После успешного тестирования приложения закройте Lazarus.
2. Обсудите, что было сделано, какие ошибки возникали, и как они были решены.

Заключение

В данной практической работе вы изучили этапы разработки приложения в Lazarus. Вы научились создавать пользовательский интерфейс, обрабатывать события и компилировать приложение. Это даст вам основы для дальнейшего изучения программирования на Pascal и разработки более сложных приложений.

Тема 5.6 Иерархия классов

Практическое занятие №26. Объявление класса, создание экземпляров класса.

Учебная цель: Ознакомиться с объявлением класса, созданием экземпляров класса.

Организационная форма занятия: практическое занятие в форме практической подготовки.

Основой объектно-ориентированного программирования является *объект*. *Объект* состоит из трёх основных частей:

1. Имя (например, автомобиль);
2. Состояние, или переменные состояния (например, марка автомобиля, цвет, масса, число мест и т. д.);
3. Методы, или операции, которые выполняют некоторые действия над объектами и определяют, как объект взаимодействует с окружающим миром.

Для работы с объектами во FreePascal введено понятие класса. *Класс* — сложная структура, включающая в себя описание данных, процедуры и функции, которые могут быть выполнены над объектом.

Классом называется составной *тип данных*, членами (элементами) которого являются функции и переменные (поля). В основу понятия *класс* положен тот факт, что "над объектами можно совершать различные *операции*". Свойства объектов описываются с помощью переменных (полей) классов, а действия над объектами описываются с помощью подпрограмм, которые называются методами класса. Объекты называются экземплярами класса.

Объектно-ориентированное *программирование (ООП)* представляет собой технологию разработки программ с использованием объектов. В объектно-ориентированных языках есть три основных понятия: *инкапсуляция*, *наследование* и *полиморфизм*. Инкапсуляцией называется *объединение* в классе данных и подпрограмм для их обработки. *Наследование* — это когда любой *класс* может быть порождён другим классом. Порождённый *класс* (наследник) автоматически наследует все поля, методы, свойства и события. *Полиморфизм* позволяет использовать одинаковые имена для методов, входящих в различные классы.

Для объявления класса используется конструкция:

type

<название класса> = *class* (<имя класса родителя>)

<поля и методы класса>

private

<поля и методы, доступные только в пределах модуля>

protected

<поля и методы, доступные только в классах-потомках>

public

<поля и методы, доступные из других модулей>

published

<поля и методы, видимые в инспекторе объектов>

end;

В качестве имени класса можно использовать любой допустимый в FreePascal *идентификатор*. *Имя класса* родителя — *имя класса*, наследником которого является данный *класс*, это необязательный *параметр*, если он не указывается, то это означает, что данный *класс* является наследником общего из предопределённого класса **TObject**.

Структуры отличаются от классов тем, что поля структуры доступны всегда. При использовании классов могут быть члены, доступные везде — публичные (описатель *public*), и приватные (описатель *private*), *доступ* к которым возможен только с помощью публичных методов. Это также относится и к методам класса. Поля, свойства и методы секции *public* не имеют ограничений на видимость. Они доступны из других функций и методов объектов как в данном модуле, так и во всех прочих, ссылающихся на него. При обращении к публичным полям вне класса используется оператор "." (точка).

Поля, свойства и методы, находящиеся в секции *private*, доступны только в методах класса и в функциях, содержащихся в том же модуле, что и описываемый *класс*. Это позволяет полностью скрыть детали внутренней реализации класса. Вызов приватных методов осуществляется из публичных.

Публикуемый (**published**) — это раздел, содержащий свойства, которые *пользователь* может устанавливать на этапе проектирования и которые доступны для редактирования в инспекторе объектов.

Защищенный (*protected*) — это раздел, содержащий поля и методы, которые доступны внутри класса, а также любым его классам-потомкам, в том числе и в других модулях.

Поля и методы, описанные до указания имени секции, являются публичными (**public**).

При программировании с использованием классов, программист должен решить, какие члены и методы должны быть объявлены публичными, а какие приватными. Общим принципом является следующее: "Чем меньше публичных данных о классе используется в программе, тем лучше". Уменьшение количества публичных членов и методов позволит минимизировать количество ошибок.

Поля могут быть любого типа, в том числе и классами. Объявление полей осуществляется так же, как и объявление обычных переменных:

поле1 : тип_данных;

поле2 : тип_данных;

...

Методы в классе объявляются так же, как и обычные подпрограммы:

function метод1 (*список параметров*) : тип результата;

procedure метод2 (*список параметров*);

Описание процедур и функций, реализующих методы, помещается после слова **implementation** того модуля, где объявлен *класс*, и выглядит так:

```
function имя_класса.метод1(список параметров) : тип результата;
```

```
begin
```

```
  тело функции;
```

```
end;
```

```
procedure имя_класса.метод2(список параметров);
```

```
begin
```

```
  тело функции;
```

```
end;
```

Объявление *переменной* типа **class** называется созданием (инициализацией) объекта (экземпляра класса). Экземпляр класса объявляется в блоке описания переменных:

```
var имя_переменной : имя_класса;
```

После описания переменной в программе можно обращаться к полям и методам класса аналогично обращению к полям структуры, используя оператор **"."**. Например:

```
имя_переменной.поле1:=выражение;
```

```
имя_переменной.метод1(список параметров);
```

```
...
```

Также можно использовать оператор **With**:

```
With имя_переменной do
```

```
begin
```

```
  поле1:=выражение;
```

```
  метод1(список параметров);
```

```
...
```

```
end;
```

В *Free Pascal* имеется большое количество классов, с помощью которых описывается форма приложения и её компоненты (кнопки, поля, флажки и т. п.). В процессе конструирования формы в текст программы автоматически добавляются программные объекты. Например, при добавлении на форму компонента формируется описание класса для этого компонента, а при создании подпрограмм обработки событий в описание класса добавляется объявление методов. Рассмотрим это на примере проекта с формой, на которой есть кнопка **Button1**.

```
unit Unit1;
```

```
interface
```

```
uses
```

```
Classes, SysUtils, LResources, Forms, Controls, Graphics,
```

```

Dialogs, StdCtrls;
type
{ TForm1 }
//объявление класса формы TForm1
TForm1 = class (TForm)
//объявление компонента кнопки Button1
Button1 : TButton;
//объявление метода обработки события -//щелчка по кнопке Button1
procedure Button1Click ( Sender : TObject );
private
{ private declarations }
public
{ public declarations }
end;
var
//описание переменной класса формы TForm1
Form1 : TForm1;
implementation
{ TForm1 }
//описание метода обработки события - щелчка по кнопке Button1
procedure TForm1. Button1Click ( Sender : TObject );
begin
//Текст процедуры обработки события
end;
initialization
{$I unit1.lrs}
end.

```

Во *Free Pascal* класс (*объект*) — это *динамическая* структура. В отличие от статической она содержит не сами данные, а ссылку на них. Поэтому программист должен сам позаботиться о выделении памяти для этих данных.

Конструктор — это специальный метод, создающий и инициализирующий *объект*. Объявление конструктора имеет вид:

```

constructor Create;

```

Описывают *конструктор* так же, как и другие методы, после ключевого слова **implementation** того модуля, в котором объявлен *класс*.

```

constructor имя_класса. Create;

```

```

begin

```

```

поле1:=выражение1;

```

```

поле2:=выражение2;

```

```

...

```

```

inherited Create;

```

```

end;

```

В результате работы конструктора инициализируются все *поля класса*, при этом порядковым типам в качестве начальных значений задаётся 0, а строки задаются пустыми.

Деструктор — это специальный метод, уничтожающий *объект* и освобождающий занимаемую им *память*. Объявляется *деструктор* следующим образом:

destructor Destroy;

Если в программе какой-либо *объект* больше не используется, то оператор

имя_переменной_типа_класс. free;

с помощью метода **free** вызывает *деструктор* и освобождает *память*, занимаемую полями объекта **имя_переменной_типа_класс**.

Рассмотрим всё описанное на примере класса "комплексное число"¹. Назовём этот класс **Tcomplex**. В нём будут члены класса: **x** — действительная часть комплексного числа — и **y** — мнимая часть комплексного числа. Также в классе будут методы:

- конструктор **Create**, который будет записывать в действительную и мнимую части значение 0;
- **Modul()** — функция вычисления модуля комплексного числа;
- **Argument()** — функция вычисления аргумента комплексного числа;
- **ComplexToStr()** — функция, представляющая комплексное число в виде строки для вывода.

Создадим новый проект, на форму поместим кнопку **Button1**, два поля — **Edit1** и **Edit2** — для ввода действительной и мнимой частей, для вывода результатов разместим компонент **Memo1**. При щелчке *по* кнопке будет создаваться экземпляр класса "Комплексное число", затем будет вычисляться его *модуль* и *аргумент*. В компонент **Memo1** выведем результаты: число в алгебраической форме, его *аргумент* и *модуль*. Ниже приведём текст модуля с комментариями, который демонстрирует работу с этим классом.

```
unit Unit1;
{$mode objfpc} {$H+}
interface
uses
Classes, SysUtils, LResources, Forms, Controls, Graphics,
Dialogs, StdCtrls;
type
{ TForm1 }
TForm1 = class (TForm)
Button1 : TButton;
Edit1 : TEdit;
Edit2 : TEdit;
Label1 : TLabel;
Label2 : TLabel;
Label3 : TLabel;
Memo1 : TMemo;
procedure Button1Click ( Sender : TObject );
private
{ private declarations }
public
{ public declarations }
end;
type
```

```

//описание класса - комплексное число
TComplex = class
private
    x : real; //действительная часть
    y : real; //мнимая часть
public
    constructor Create; //конструктор
    function Modul ( ) : real; //метод вычисления модуля
    function Argument ( ) : real; //метод вычисления аргумента
    //метод записи комплексного числа в виде строки
    function ComplexToStr ( ) : String;
end;
var
Form1 : TForm1;
//объявление переменной типа класс "комплексное число"
chislo : TComplex;
implementation
//описание конструктора
constructor TComplex. Create;
begin
x := 0; y := 0;
//Вызов родительского конструктора
inherited Create;
end;
//Описание метода вычисления
//модуля комплексного числа.
function TComplex. Modul ( ) : real;
begin
modul:= sqrt ( x * x+y * y );
end;
//Описание метода вычисления
//аргумента комплексного числа.
function TComplex. Argument ( ) : real;
begin
argument := arctan ( y/x ) * 180/ pi;
end;
//Описание метода записи комплексного
//числа в виде строки.
function TComplex. ComplexToStr ( ) : String;
begin
if y>=0 then
    ComplexToStr := FloatToStrF ( x, ffFixed,5,2)+ '+' +
FloatToStrF ( y, ffFixed,5,2)+ ' i '
else
ComplexToStr := FloatToStrF ( x, ffFixed,5,2)+
FloatToStrF ( y, ffFixed,5,2)+ ' i '
end;
{Обработчик кнопки: создание экземпляра класса "Комплексное число }
{вычисление его модуля и аргумента, вывод числа в алгебраической форме,}
{его аргумента и модуля.}
procedure TForm1. Button1Click ( Sender : TObject );
var Str1 : String;

```

```

begin
//создание объекта (экземпляра класса) типа "комплексное число"
chislo :=TComplex. Create;
//ввод действительной и мнимой частей комплексного числа
chislo. x:= StrToFloat ( Edit1. Text );
chislo. y:= StrToFloat ( Edit2. Text );
Str1 := ' Kompleksnoe  chislo  '+c h i s l o. ComplexToStr ( );
//вывод на форму в поле Memo1 построчно комплексного числа,
Memo1. Lines. Add( Str1 );
//его модуля
Str1 := ' Modul  chisla  '+
FloatToStrF ( chislo. Modul ( ), ffFixed, 5, 2 );
Memo1. Lines. Add( Str1 );
//и аргумента
Str1 := ' Argument  chisla  '+
FloatToStrF ( chislo. Argument ( ), ffFixed, 5, 2 );
Memo1. Lines. Add( Str1 );
//уничтожение объекта
chislo. Free;
end;
initialization
{$I unit1.lrs}
end.

```

Результат работы программы приведён на [рис. 9.1](#).

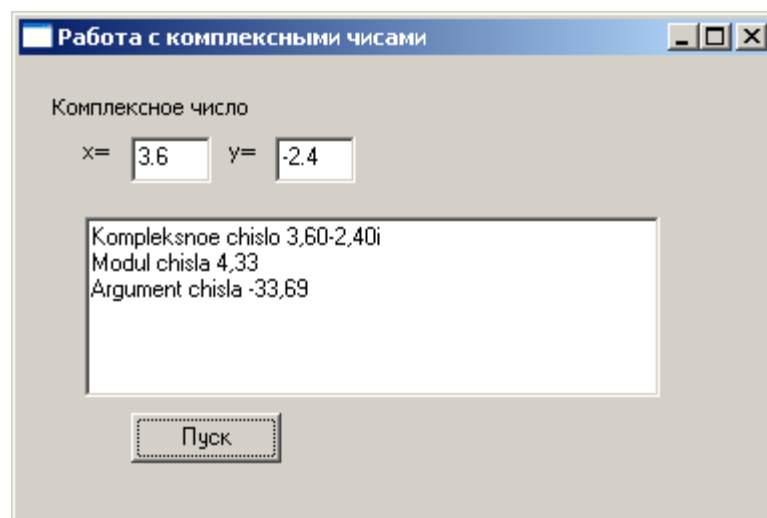


Рис. 1. Результаты работы программы работы с классом "Комплексное число"

Практическое занятие №27. Создание наследованного класса.

Учебная цель: создать наследованные классы в Лазарус.

Организационная форма занятия: практическое занятие в форме практической подготовки.

Для того чтобы использовать файлы как ресурсы Lazarus, нужно пересоздать их при помощи lazres. Lazres можно найти в папке "Tools", расположенной в установочной директории Lazarus (C:\Lazarus\Tools\).

Теперь можно скомпилировать файлы ресурсов Lazarus (*.lrs) при помощи командной строки. Синтаксис lazres:

```
lazres <имя файла ресурсов> <включаемые файлы (file1 file2 file3 ...)>
```

Пример:

```
lazres mylazarusresource.lrs image.jpg
```

Для использования файла ресурсов Lazarus в вашем проекте, подключите его используя директиву компилятора \$I в разделе **initialization** вашего модуля.

Вы можете получить доступ к данным ресурса сразу или с использованием метода переменных LoadFromLazarusResource, которые после этого будут хранить данные. LoadFromLazarusResource принимает на вход один строковый параметр, который указывает, какой объект должен быть загружен из файла ресурсов. Пример:

```
procedure ExampleProc;  
var  
    Image: TImage  
begin  
    Image := TImage.Create;  
    Image.Picture.LoadFromLazarusResource('image'); // Обратите внимание, что расширение  
    файла не указывается  
end;  
  
initialization  
    {$I mylazarusresource.lrs}
```

Этот код подключает файл mylazarusresource.lrs к проекту. В процедуре ExampleProc создан объект TImage и в него загружено изображение "image" из ресурса. Файл, который был встроен в ресурс, возможно был назван image.jpg.

Все класс, наследованные от TGraphic, имеют процедуру LoadFromLazarusResource.

Lazarus Resource Form File

Lazarus генерирует .LRS файлы с файлами формы .LFM.

Когда файл формы (*.lrs) отсутствует, FPC выдает следующее сообщение об ошибке: ERROR: unit1.pas(193,4) Fatal: Can't open include file "unit1.lrs"

Чтобы решить эту проблему вы можете:

- использовать lazres: c:\lazarus\tools\lazres.exe unit1.lrs unit1.lfm
- (проще): немного изменить дизайн формы и сохранить его; это пересоздаст .lrs файл без необходимости запускать lazres.

Получение данных из LRS ресурса

Вы можете извлекать данные из ресурса так:

```
uses ...LResources...;

...
procedure TForm1.FormCreate(Sender: TObject);
var
  r: TLRResource;
  Data: string;
begin
  r := LazarusResources.Find('datafile1');
  if r = nil then
    raise Exception.Create('Resource datafile1 is missing');

  Data := r.Value;
  //...что-то делать с данными...
end;
```

4. FPC ресурсы

Начиная с версии FPC 2.4 вы можете использовать стандартные .rc (Resource Script) файлы для подключения ресурсов в ваших проектах. Для превращения .rc скрипта в бинарный ресурс (.res файл), FPC запускает внешний компилятор (windres или GoRC). Поэтому, компилятор ресурсов должен быть установлен и в переменной окружения PATH, должен быть прописан путь к нему. Для получения дополнительной информации см.: [FPC Programmer's guide, chapter 13 "Using Windows resources"](#)

Для упрощения процесса компиляции, можно использовать уже скомпилированные файлы ресурсов (.res). Вы можете предварительно скомпилировать ресурсы любым доступным компилятором - windres (доступен на Unix и Windows), GoRC (только Windows), Microsoft Resource Compiler (rc.exe входит в Visual Studio), Borland Resource Compiler (brcc32.exe входит в Delphi, C++ Builder или продукты Rad Studio) или любой другой.

Используйте директиву {\$R filename.rc} или {\$R filename.res} для включения файла ресурсов в исполняемый файл. FPC RTL предоставляет как низкоуровневые функции, так и высокоуровневые классы для работы с ресурсами.

Низкоуровневые функции:

- EnumResourceTypes
- EnumResourceNames
- EnumResourceLanguages
- FindResource
- FindResourceEx
- LoadResource
- SizeofResource

- LockResource
- UnlockResource
- FreeResource

Они совместимы с Windows API функциями [Resource Functions](#).

Базовым классом, работающим с ресурсами, является TResourceStream. LCL использует его для загрузки встроенных растровых изображений, иконок, и форм. Посмотрите на TGraphic.LoadFromResourceID или TIcon.LoadFromResourceHandle чтобы понять, как они используются в LCL.

Добавление ресурсов в программу

Давайте рассмотрим ситуацию, когда нужно сохранить некоторые данные в исполняемый файл и при выполнении программы необходимо извлечь их.

Сначала мы должны сообщить компилятору, какие файлы включить в ресурс. Делаем это в .rc файле:

Файл mydata.rc

```
MYDATA    RCDATA "mydata.dat"
```

Здесь MYDATA это имя ресурса, RCDATA это тип ресурса (подробнее смотрите [Resource Types](#)) и "mydata.dat" это файл с данными.

Давайте скажем компилятору подключить ресурс к программе:

```
program mydata;

{$R mydata.rc}
begin
end.
```

Незаметно, FPC указывает компилятору ресурсов, распространяемому вместе с ним, компилировать .rc скрипт в бинарный файл ресурсов .res. После, компоновщик включит его в исполняемый файл. Хотя это не заметно для программиста, но вы можете, если хотите, создать свой собственный .res файл при помощи, например, Borland Resource Compiler. Тогда, вместо использования

```
{$R mydata.rc}
```

нужно использовать

```
{$R mydata.res}
```

.

Установлен ли windres

На Linux/OS X, вам нужно убедиться, что компилятор ресурсов windres, который входит в состав инструментов mingw32, у вас установлен. Например на Debian вы можете выполнить:

```
aptitude install mingw32-binutils
```

```
# this installs e.g. /usr/bin/i586-mingw32msvc-windres
# but FPC expects just windres, so:
ln -s /usr/bin/i586-mingw32msvc-windres /usr/bin/windres
# alternatively (and probably more "official") use -FCi586-mingw32msvc-windres in your calls
to fpc or
# add -FCi586-mingw32msvc-windres to your fpc.cfg
```

На Windows, компиляторы ресурсов windres и gorc устанавливаются вместе с FPC и Lazarus.

Использование ресурсов в вашей программе

Давайте сейчас извлечем сохраненный ресурс в файл, например mydata.dat:

```
program mydata;
uses
  SysUtils, Windows;
{$R mydata.res}
var
  S: TResourceStream;
  F: TFileStream;
begin
  // create a resource stream which points to our resource
  S := TResourceStream.Create(HInstance, 'MYDATA', RT_RCDATA);
  // Please be aware of writing an apostrophes in resource type - source will not be atracted!!!
  try
    // Создание файла mydata.dat в папке приложения
    F := TFileStream.Create(ExtractFilePath(ParamStr(0)) + 'mydata.dat', fmCreate);
    try
      F.CopyFrom(S, S.Size); // Копирование данных из ресурса в файл
    finally
      F.Free; // destroy the file stream
    end;
  finally
    S.Free; // destroy the resource stream
  end;
end.
```

Практическое занятие №28 Перегрузка операций.

Учебная цель: изучить перегрузку операций в Паскаль.

Организационная форма занятия: практическое занятие в форме практической подготовки.

Теоретический материал:

Перегрузка операций - это средство языка, позволяющее вводить операции над типами, определяемыми пользователем. В **PascalABC.NET** можно использовать только predefined значки операций. Перегрузка операций для типа **T**, являющегося классом или записью, осуществляется при помощи статической (классовой) функции-метода со специальным именем **operator** *ЗнакОперации*. Перегрузка специальных операций **+=**, **-=**, ***=**, **/=** осуществляется с помощью статической процедуры-метода, первый параметр которой передается по ссылке.

Например:

```
type
Complex = record
  re,im: real;
static function operator+(a,b: Complex): Complex;
begin
  Result.re := a.re + b.re;
  Result.im := a.im + b.im;
end;
static function operator=(a,b: Complex): boolean;
begin
  Result := (a.re = b.re) and (a.im = b.im);
end;
end;
```

Для перегрузки операций действуют следующие правила:

1. Перегружать можно все операции за исключением **@** (взятие адреса), **as**, **is**, **new**. Кроме того, можно перегружать специальные бинарные операции **+=**, **-=**, ***=**, **/=**, не возвращающие значений.
2. Перегружать можно только еще не перегруженные операции.
3. Тип по крайней мере одного операнда должен совпадать с типом класса или записи, внутри которого определена операция.
4. Перегрузка осуществляется с помощью статической функции-метода, количество параметров которой совпадает с количеством параметров соответствующей операции (2 - для бинарной, 1 - для унарной).
5. Перегрузка операций **+=**, **-=**, ***=**, **/=** для соответствующих операторов осуществляется с помощью статической процедуры-метода, первый параметр которой передается по ссылке и имеет тип записи или класса, в котором определяется данная операция, второй - передается по значению и совместим по присваиванию с первым. Перегрузка остальных операций осуществляется с помощью статических функций-методов.
6. Операции приведения типа задаются статическими функциями, у которых вместо имени используется **operator implicit** (для неявного приведения типа) или **operator explicit** (для явного приведения типа).

Например:

type

Complex = **record**

...

static function operator implicit(d: real): Complex;

begin

Result.re := d;

Result.im := 0;

end;

static function operator explicit(c: Complex): string;

begin

Result := Format('{{0}},{{1}}',c.re,c.im);

end;

static procedure operator +=(var c: Complex; value: Complex);

begin

c.re += value.re;

c.im += value.im;

end;

static function operator +(c,c1: Complex): Complex;

begin

Result.re := c.re + c1.re;

Result.im := c.im + c1.im;

end;

end;

Можно перегружать операции с помощью [методов расширения](#) - в этом случае при описании подпрограммы не следует писать слово class. Например, так в системном модуле реализовано добавление числа к строке:

function operator +(str: string; n: integer): string; **extensionmethod;**

begin

result := str + n.ToString;

end;

Список информационных источников

Основные печатные издания

1. Давыдова, Н. А. Программирование: учебное пособие / Н. А. Давыдова, Е. В. Боровская. — 5-е изд. — Москва: Лаборатория знаний, 2025. — 239 с. — ISBN 978-5-93208-831-9. — Текст: электронный // Электронный ресурс цифровой образовательной среды СПО PROОбразование : [сайт]. — URL: <https://profspo.ru/books/147044>
2. Маляров, А. Н. Объектно-ориентированное программирование: учебник для СПО / А. Н. Маляров. — Саратов: Профобразование, 2021. — 331 с. — ISBN 978-5-4488-1238-5. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: <https://www.iprbookshop.ru/106837.html>
3. Теория и реализация языков программирования: учебное пособие для СПО / В. А. Серебряков, М. П. Галочкин, Д. Р. Гончар, М. Г. Фуругян. — Саратов: Профобразование, 2021. — 372 с. — ISBN 978-5-4488-1013-8. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: <https://www.iprbookshop.ru/102206.html>

Основные электронные издания

1. Павловская, Т. А. Программирование на языке высокого уровня Паскаль: учебное пособие для СПО / Т. А. Павловская. — Саратов: Профобразование, 2021. — 153 с. — ISBN 978-5-4488-1008-4. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: <https://www.iprbookshop.ru/102201.html>

Дополнительные источники

1. www.edu.ru — Федеральный образовательный портал «Российское образование». Форма доступа:
2. <http://iit.metodist.ru> — Информатика и информационные технологии: сайт лаборатории информатики МИОО
3. <http://www.intuit.ru> — Интернет-университет информационных технологий (ИНТУИТ.ру)
4. <http://www.osp.ru> — Открытые системы: издания по информационным технологиям

7. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

7.1.1. Методическая литература:

- методические указания для практических занятий;
- методические указания к выполнению курсовой работы.

7.1.2. Программное обеспечение:

1. Microsoft Visual Studio 2022

7.1.3. Материально-техническое обеспечение дисциплины

Мультимедийное оборудование для чтения лекций. Компьютерный класс, оснащенный компьютерами, с установленной программой Microsoft Visual Studio 2022.

7.2.1. Основные печатные издания

1. Давыдова, Н. А. Программирование: учебное пособие / Н. А. Давыдова, Е. В. Боровская. — 5-е изд. — Москва: Лаборатория знаний, 2025. — 239 с. — ISBN 978-5-93208-831-9. — Текст: электронный // Электронный ресурс цифровой образовательной среды СПО PROФобразование: [сайт]. — URL: <https://profspo.ru/books/147044>

Маляров, А. Н. Объектно-ориентированное программирование: учебник для СПО / А. Н.

2. Маляров. – Саратов: Профобразование, 2021. – 331 с. – ISBN 978-5-4488-1238-5. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: <https://www.iprbookshop.ru/106837.html>

3. Теория и реализация языков программирования: учебное пособие для СПО / В. А. Серебряков, М. П. Галочкин, Д. Р. Гончар, М. Г. Фуругян. – Саратов: Профобразование, 2021. – 372 с. – ISBN 978-5-4488-1013-8. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: <https://www.iprbookshop.ru/102206.html>

7.2.2. Основные электронные издания

1. Павловская, Т. А. Программирование на языке высокого уровня Паскаль: учебное пособие для СПО / Т. А. Павловская. – Саратов: Профобразование, 2021. – 153 с. – ISBN 978-5-4488-1008-4. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: <https://www.iprbookshop.ru/102201.html>

7.2.3. Дополнительные источники

1. www.edu.ru – Федеральный образовательный портал «Российское образование». Форма доступа:

2. <http://iit.metodist.ru> – Информатика и информационные технологии: сайт лаборатории информатики МИОО

3. <http://www.intuit.ru> – Интернет-университет информационных технологий (ИНТУИТ.ру)

4. <http://www.osp.ru> – Открытые системы: издания по информационным технологиям

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего
образования
«Северо-Кавказский федеральный университет»

Колледж СКФУ в г. Ставрополе

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к самостоятельной работе

ОП.06 Основы алгоритмизации и программирования

Специальность 09.02.11 Разработка и управление программным обеспечением

Форма обучения очная

Ставрополь

1. Пояснительная записка

Методические указания для самостоятельной работы студентов при изучении учебной дисциплины ОП.06 Основы алгоритмизации и программирования предназначены для обучающихся по специальности 09.02.11 Разработка и управление программным обеспечением.

При организации СР 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.

По учебному плану специальности 09.02.07 Информационные системы и программирование/ в учебной дисциплине ОП.04 Основы алгоритмизации и программирования на внеаудиторную самостоятельную работу обучающихся отводится 8 часов.

1. План-график выполнения СРС

№ п/п	Наименование разделов и тем дисциплины, их краткое содержание; вид самостоятельной работы	Форма контроля	Объем часов
1	Тема 4.1 Указатели	составление конспекта	2
2	Тема 5.1 Основные принципы объектно-ориентированного программирования (ООП)	работа с учебной литературой	2
3	Тема 5.5 Этапы разработки приложений	составление конспекта	2
	Итого		6

3. Методические рекомендации по выполнению самостоятельной работы

3.1. 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.

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.

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.

3.2. 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.

Список информационных источников

Основные печатные издания

1. Давыдова, Н. А. Программирование: учебное пособие / Н. А. Давыдова, Е. В. Боровская. — 5-е изд. — Москва: Лаборатория знаний, 2025. — 239 с. — ISBN 978-5-93208-831-9. — Текст: электронный // Электронный ресурс цифровой образовательной среды СПО PROОбразование : [сайт]. — URL: <https://profspo.ru/books/147044>
2. Маляров, А. Н. Объектно-ориентированное программирование: учебник для СПО / А. Н. Маляров. — Саратов: Профобразование, 2021. — 331 с. — ISBN 978-5-4488-1238-5. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: <https://www.iprbookshop.ru/106837.html>
3. Теория и реализация языков программирования: учебное пособие для СПО / В. А. Серебряков, М. П. Галочкин, Д. Р. Гончар, М. Г. Фуругян. — Саратов: Профобразование, 2021. — 372 с. — ISBN 978-5-4488-1013-8. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: <https://www.iprbookshop.ru/102206.html>

Основные электронные издания

1. Павловская, Т. А. Программирование на языке высокого уровня Паскаль: учебное пособие для СПО / Т. А. Павловская. — Саратов: Профобразование, 2021. — 153 с. — ISBN 978-5-4488-1008-4. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: <https://www.iprbookshop.ru/102201.html>

Дополнительные источники

1. www.edu.ru — Федеральный образовательный портал «Российское образование». Форма доступа:
2. <http://iit.metodist.ru> — Информатика и информационные технологии: сайт лаборатории информатики МИОО
3. <http://www.intuit.ru> — Интернет-университет информационных технологий (ИНТУИТ.ру)
4. <http://www.osp.ru> — Открытые системы: издания по информационным технологиям