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

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению лабораторных работ
по дисциплине «Основы программирования в инфокоммуникационных системах»
для студентов направления подготовки
11.03.02 Инфокоммуникационные технологии и системы связи

Ставрополь 2026

Практикум по дисциплине «Основы программирования в инфокоммуникационных системах» для студентов направления подготовки 11.03.02 Инфокоммуникационные технологии и системы связи

Рассмотрены следующие вопросы: разработка программ линейной, разветвляющейся, циклической и смешанной структур; одномерные и двумерные массивы; структуры и строки; функции, стандартные функции языка C++.

В каждой лабораторной работе указывается ее цель, изложены краткие теоретические сведения, описываются этапы выполнения работ и варианты заданий, а также предлагаются контрольные вопросы, которые позволяют студенту произвести самооценку.

СОДЕРЖАНИЕ

Лабораторное занятие № 1

Программы линейной структуры в языке С++

1. Цель работы: Закрепление знаний об алгоритмах и программах линейной структуры.

2. Основные сведения.

Вначале рассмотрим пример программы линейной структуры на языке С.

Пусть требуется вычислить значение z по формуле

$z = 3(y + 3x) + (x+y)^2$, при этом значения x и y вводятся с клавиатуры. Программа вычисления на языке С с применением форматного ввода-вывода имеет вид:

```
//lab2_1.c  вычисления по формулам
#include<stdio.h>
#include<conio.h>
#include<math.h>
main(){
  float x,y,z;
  printf(«Enter x,y\n»);
  scanf(«%f%f»,&x,&y);
  z = 3*(y+3*x)+(x+y)*(x+y);
  printf("x=%8.2f, y=%8.2f, z=%10.4f",x,y,z);
  getch();
  return 0;
}
```

Дадим краткие пояснения. Первая строка программы – однострочный комментарий, поясняющий назначение программы. Комментарий начинается с двух символов «слэш» подряд и действует до конца строки. Комментарий не влияет на компиляцию программы и предназначен программисту, в том числе и автору программы, т.к. программы довольно быстро забываются.

Хотя комментарий необязателен, настоятельно рекомендуется снабжать комментарием начало программы, чтобы отличить одну программу от другой.

Три следующих строчки программы – директивы включения заголовочных файлов stdio.h (заголовочный файл ввода-вывода), conio.h (от console input-output, заголовочный файл работы с консольным вводом-выводом), math.h (заголовочный файл для работы с математическими функциями).

Следующая строка – описание функции main (главной функции программы). Программа на языке С (и С++) состоит из ряда функций, из которых функция main должна присутствовать обязательно, она служит точкой входа в программу.

Параметры любой функции, в том числе и функции main, заключаются в круглые скобки. Даже если список параметров пуст (как в данном случае), круглые скобки ставятся все равно, чтобы отличить функцию от простой переменной. После списка параметров в круглых скобках записывается тело функции (список выполняемых действий), заключаемое в фигурные скобки.

Строка float x,y,z; содержит описание переменных вещественного типа x, y, z (от float – плавающий, числа с плавающей точкой, могут содержать как целую, так и дробную части). Перед использованием каждой переменной она должна быть описана, т.е. указан ее тип. Переменные типа float (вещественные с одинарной точностью) занимают 4 байта в оперативной памяти и используются в неответственных вычислениях и учебных программах. В ответственных вычислениях используются вещественные переменные с удвоенной точностью типа double, занимающие 8 байтов.

Кроме переменных вещественного типа, в языках С и С++ используются целочисленные переменные типа int. Эти переменные не имеют дробной части и представляются в компьютере точно. Объем занимаемой оперативной памяти для переменных этого типа зависит от реализации. В среде Borland C++ версии 3.1 эти переменные занимают 2

байта, в среде Borland C++ Builder переменные типа int занимают 4 байта. Переменные других типов будут рассмотрены по мере изучения.

Функции printf и scanf содержатся в заголовочном файле stdio.h и служат для вывода на экран и ввода с клавиатуры соответственно. Управляющие символы «\n» (бэкслэш n) в функции printf служат для перевода на новую строку. Последовательности символов, начинающиеся с символа бэкслэш, называются эскейп-последовательностями, так что «\n» – одна из эскейп-последовательностей. Символ «&» (амперсанд) в функции scanf указывает на адрес переменной.

Оператор printf(“Enter x,y”); означает вывод на экран текста «Enter x,y», т.е. приглашения ко вводу x и y, следующая строка scanf(“%f%f”,&x,&y); – ввод с клавиатуры значений x и y и занесение их по адресам этих переменных (для этого используется символ амперсанд).

Оператор z = 3 * (y+3*x) + (x+y)*(x+y); присваивает переменной z значение $3(y + 3x) + (x+y)^2$, звездочка означает знак умножения.

Оператор printf(“x=%8.2f, y=%8.2f, z=%10.4f”,x,y,z); служит для вывода значений x, y, z . В форматной строке, заключенной в кавычки, указываются форматы вывода на экран: %8.2f %8.2f %10.4f переменных x, y, z соответственно, первые 2 переменных выводятся в 8 позициях с двумя знаками после десятичной точки, переменная z в10 позициях с 4-мя знаками после десятичной точки, форматы указываются после символа «процент».

Символ «f» означает вывод в виде числа с плавающей точкой (float). Остальные символы в форматной строке выводятся на экран без изменения. Поэтому вначале будет выведен текст «x=», затем значение x в формате “%8.2f”, далее текст «y=», далее значение y в формате “%8.2f”, затем текст «z=» и наконец значение z в формате “%10.4f”.

Функция getch, содержащаяся в заголовочном файле conio.h, ожидает нажатия любой клавиши, при этом пользователь имеет возможность просмотреть результаты работы программы. Без использования этой функции программа завершит работу, произойдет переход к показу текста

программы, и пользователь не сможет увидеть результаты (экран с результатами покажется лишь на мгновение).

Все функции, кроме функций типа void, возвращают значение определенного типа, для чего служит оператор return a, где a – возвращаемое значение. Если у функции не указан тип (как в данном случае), то по умолчанию функция имеет тип int. Возвращаемое значение 0 означает, что работа программы прошла без ошибок.

Контрольный пример для приведенной программы. Пусть x=3,4; y=2,5. В ответ на приглашение к вводу данных “Enter x,y” вводим 3.4_2.5 и нажимаем ENTER. (Здесь знак подчеркивания означает пробел). Разделителем целой и дробной частей является точка.

Результат работы программы имеет вид:

x=_ _ _ _3.40, _ y=_ _ _ 2.50 , z= _ _ _72.9100

Здесь также знаки подчеркивания означают пробелы.

Этот результат следует проверить с помощью калькулятора Windows.

Все операторы, как выполняющие определенные действия, так и служащие для описания объектов программы (переменных, констант, типов и т.п.), должны завершаться точкой с запятой.

Та же программа на языке C++ с применением потокового ввода-вывода имеет вид:

```
//lab2_2.cpp  вычисления по формулам
#include<iostream.h>
#include<conio.h>
#include<math.h>
main(){
  float x,y,z;
  cout<<”Enter x,y\n”;
  cin>>x>>y;
  z = 3*(y+3*x)+(x+y)*(x+y);
  cout<<”x= “<<x<<”  y=”<<y<<”  z=”<<z<<endl;
```

```
  getch();
  return 0;
}
```

Отличие программы lab2_2.cpp от программы lab2_1.c в том, что в ней применяется потоковый ввод вывод, содержащийся в файле iostream.h. Выходной поток cout – это экран дисплея, входной поток cin – клавиатура. Лексемы “<<” и “>>” указывают направление ввода-вывода: в выходной поток cout из переменных и в переменные из входного потока cin. Потоковый ввод-вывод является бесформатным, компилятор сам определяет формат по типу переменной.

Текстовая константа «endl» (endline, конец строки) означает, как и эскейп-последовательность «\n», переход на новую строку. Если выводится текстовая константа, заканчивающаяся переходом на новую строку, можно применить «\n», если же выводятся значения переменных, после которых следует переход на новую строку, следует применить endl. Ввод-вывод каждого аргумента ввода-вывода должен сопровождаться лексемами “<<” или “>>”.

3. Выполнение работы

3.1. Набрать и откомпилировать оба приведенных варианта программы. Результаты проверить с помощью калькулятора Windows.

3.2. Разработать программы линейной структуры для вычислений по формулам на языках C и C++ согласно вариантам заданий.

4. Варианты заданий

Вычислить значение функции при заданных значениях параметров:

1. $x=2y+3t-z$ при $y=2$; $t=5/(1+y^2)$; $z=4$.
2. $x=3y^2/(4z-2t^2)$ при $t=0.5$; $z=6$; $y=t+2z$.
3. $x=4y^2/(4z-2t^3)$ при $t=1$; $z=3$; $y=2 + t$.
4. $x=4y^3-z/t$ при $t=2$; $z=3$; $y=(t+z)$.
5. $x=6t^2-(z+1)/y^2$ при $y=2$; $z=4$; $t=(2+z)$.
6. $x=(8z^2+1)/(y+t^2)$ при $z=1$; $t=2$; $y=t+z$.

7. $x=6t^3-3z^2/(y+1)$ при t=2; z=t+1; y=3.

8. $x=8z/(t+3)-y^2$ при t=1; z=t+2; y=4.

9. $x=6y^2+3(z-t)^3$ при t=2; z=t+1; y=3.

10. $x=6t^2-(z+1)/(y+1)$ при t=0.5; z=6; y=t+2z.

5. Контрольные вопросы

1. Что такое алгоритм линейной структуры? программа линейной структуры?

2. В каком заголовочном файле содержатся математические функции?

3.Как вывести вещественное число в поле с заданным числом позиций с заданным числом десятичных знаков после точки?

Лабораторное занятие №2

Условный оператор в языке С++

1. Цель работы: Закрепление знаний о программах разветвляющейся структуры, составление программы с условным оператором и работа с ней.

2. Основные сведения

Алгоритм разветвляющейся структуры - это алгоритм, в котором вычислительный процесс осуществляется либо по одной, либо по другой ветви, в зависимости от выполнения некоторого условия. Программа разветвляющейся структуры реализует такой алгоритм. В программе разветвляющейся структуры имеется один или несколько условных операторов.

Условный оператор в языке С имеет формат:

if (условие) оператор1; else оператор2; (полная форма) или

if (условие) оператор1; (сокращенная форма).

Условие заключается в круглые скобки, при его выполнении исполняется оператор1, при невыполнении - оператор2 (при полной форме условного оператора). Для неполной формы условного оператора при выполнении условия исполняется оператор1, в противном случае оператор1 пропускается и исполняется оператор, следующий за условным оператором.

Оператор1 и оператор2 могут представлять простые операторы (один оператор), в этом случае они не заключаются в скобки. Если же оператор1 и/или оператор2 представляют составной оператор (несколько операторов), то их нужно заключить в фигурные скобки.

В качестве примера приведем программу вычисления функции по различным математическим выражениям на различных интервалах.

```
//razvetvl.c  программа разветвляющейся структуры
#include <stdio.h>
#include <conio.h>
main()
{float x,y;
```

```
printf(”Введите x\n”);
scanf(“%f“,&x);
if (x<=0) {y=x*x+1;goto m;}
else if (x<2) {y=x+1;goto m;}
else y=7-x*x;
m:  printf(“ x=%8.2f y=%8.2f\n”,x,y);
getch();
return 0; }
```

.

3. Выполнение работы

Набрать и откомпилировать приведенную выше программу вычисления заданной функции, исправить выявленные ошибки. Ввести несколько вариантов значений аргумента (в различных интервалах), вычислить функцию вручную и сравнить с полученными по программе результатами.

Составить программу разветвляющейся структуры с применением условного оператора согласно вариантам заданий, откомпилировать ее, ввести исходные данные, проверить полученный результат.

4. Варианты заданий

Вычислить значение функции, заданной различными аналитическими выражениями на различных интервалах:

Варианты 1, 6 y =

$$\begin{cases} x^2 \text{ при } x <= -3, \\ 1+x^3 \text{ при } -3 \le x < 5, \\ x^2/(3+4x) \text{ при } x \ge 5. \end{cases}$$

Варианты 2, 7 y =

$$\begin{cases} 3x + 4 \text{ при } x < 1, \\ x^2/(6+x) \text{ при } 1 \le x < 4, \\ 3x^3 + 8 \text{ при } x \ge 5. \end{cases}$$

Варианты 3, 8 y =

$$\begin{cases} (6 - x^2) / 3 \text{ при } x \le 2, \\ x^2 / (3 + 4x^2) \text{ при } 2 < x \le 4, \\ 6x^3 - 4x \text{ при } x > 4. \end{cases}$$

Варианты 4 , 9 . y =

$$\begin{cases} 6x^2 + 3 / (4 - x) \text{ при } x < -2, \\ x^2 / (3 + x^2) \text{ при } -2 \le x < 3, \\ (2x^3 + 4) / (5x^2 - 2) \text{ при } x \ge 3. \end{cases}$$

Варианты 5, 10 y =

$$\begin{cases} 4x^2 + 6 \text{ при } x <– 1, \\ x^3 / (5 + x) \text{ при } – 1 \le x < 2, \\ 4x^2 – 3 \text{ при } x \ge 2. \end{cases}$$

5. Контрольные вопросы

1. Какой алгоритм является алгоритмом разветвляющейся структуры?

2. По программе раздела 2 составьте выражения для функции на интервалах.

3. Что такое условный оператор? Полная и сокращенная формы.

4. Что такое составной оператор? Формат его записи.

5. Что такое оператор безусловного перехода?

6. Найдите операторы безусловного перехода в примере программы.

7. Можно ли обойтись без операторов безусловного перехода?

Лабораторное занятие №3

Переключатель в языке С++

1. Цель работы: Закрепление знаний о переключателе, составление программы с переключателем и работа с ней.

2. Основные сведения

В ряде случаев разветвление программы в зависимости от значения некоторой переменной требуется произвести не на 2 , а на большее число ветвей. Например, определим время года по введенному номеру месяца.

```
//switch.c   множественный выбор
 #include<stdio.h>
 #include<conio.h>
 int n;
 main() {
 printf(“Введите номер месяца\n”);
 scanf(“%d”,&n);
 printf(“\nВремя года: “);
 switch(n){
 case 1:case 2:case 12:printf(“Зима\n”);break;
 case 3:case 4:case 5:printf(“Весна\n”);break;
 case 6:case 7:case 8:printf(“Лето\n”);break;
 case 9:case 10:case 11:printf(“Осень\n”);break;
 default:printf(“\nНомер месяца неверен\n”);}
 getch();
 return 0;}
```

В приведенном примере программы при вводе номера месяца от 1 до 12 на экране печатается соответствующее время года, если же номер месяца превышает 12, выводится сообщение о неверном вводе месяца, для чего служит зарезервированное слово языка default. После каждой из констант, перечисляющих значение переключателя, ставится двоеточие. Заключительный для каждой ветви оператор break служит для прерывания

цикла проверки и перехода в конец переключателя. В случае отсутствия break переключатель работает неверно - происходит переход на следующую ветвь.

3. Выполнение работы

Набрать и откомпилировать приведенную выше программу с применением переключателя. Исправить выявленные ошибки. Ввести несколько значений номера месяца в этой программе, убедиться в правильности работы программы.

Составить программу с переключателем согласно вариантам заданий, откомпилировать ее, ввести исходные данные, проверить полученный результат.

4. Варианты заданий

1. Вводится число экзаменов N<=20. Напечатать фразу ”Мы успешно сдали N экзаменов”, согласовав слово «экзамен» с числом N.

2. Вводится число - номер месяца. Вывести количество дней в месяце (год невисокосный).

3. Вводится число лет (N<=25).Напечатать фразу “Мне N лет”, согласовав слово «лет» с числом N.

4. Вводится число книг N<=10. Вывести фразу “Я взял из библиотеки N книг”, согласовав слово «книга» с числом N.

5. Вводится число карандашей N<=10. Вывести фразу “Я купил N карандашей”, согласовав слово «карандаш» с числом N.

6. Вводится число версий N<=10.Вывести фразу “Следователь проверил N версий”, согласовав слово «версия» с числом N.

7. Вводится число программ N<=10.Напечатать фразу “Я разработал N программ”, согласовав слово «программа» с числом N.

8. Вводится целое число -9<=c<=9.Вывести величину числа в словесной форме с учетом знака.

9. Вводится число вершин N<=10. Вывести фразу “Мы покорили N вершин”, согласовав слово «вершина» с числом N.

10. Вводится число групп N<=10. Вывести фразу “На факультете N групп”, согласовав слово «группа» с числом N.

5. Контрольные вопросы

1. Что такое множественный выбор?

2. В каких случаях применяется переключатель?

3. Зачем ставится в переключателе оператор break?

4. Зачем в переключателе употребляется зарезервированное слово default?

5. Блок-схема переключателя.

Лабораторное занятие № 4

Циклы. Цикл с параметром

1. Цель работы: Закрепление знаний о программах циклической структуры, составление программы цикла с параметром и работа с ней.

2. Основные сведения

Алгоритм циклической структуры - это алгоритм, в котором происходит многократное повторение одного и того же участка программы. Такие повторяемые участки вычислительного процесса называются циклами.

Программа циклической структуры содержит один или несколько циклов. Различают детерминированные циклы с заранее известным числом повторений и итерационные циклы, в которых число повторений заранее неизвестно.

Для организации цикла необходимо выполнить следующие действия:

1) задать перед циклом начальное значение переменной, изменяющейся в цикле;

2) изменять переменную перед каждым новым повторением цикла;

3) проверять условие повторения цикла;

4) управлять циклом, т.е. переходить к его началу, если он не закончен, или выходить из него по окончании. Изменяющаяся в цикле переменная называется параметром цикла, в одном цикле возможны несколько параметров.

В языке С существует 3 вида циклов: 1) цикл с параметром или цикл типа **for**, 2) цикл с предусловием или цикл типа **while**, 3) цикл с постусловием или цикл типа **do ... while**. В цикле типа for число повторений известно заранее, в циклах типа while и do ... while число повторений цикла заранее неизвестно, производится проверка условия повторения цикла: в цикле типа while - перед циклом, в цикле типа do ... while - после его окончания.

В циклах типов for и while повторяющаяся часть (тело цикла) состоит из одного оператора, если требуется выполнить в цикле несколько операторов, они заключаются в фигурные скобки, образуя составной оператор. В цикле типа do ... while тело цикла помещается между зарезервированными словами языка (лексемами) do и while, фигурные скобки также требуются, в названии цикла его тело условно обозначается тремя точками.

Во всех типах циклов условие продолжения цикла заключается в круглые скобки. Для цикла типа for заголовок цикла состоит из трех разделов: инициализации (присваивания начальных значений), проверки условия повторения, модификации (изменения параметров). Разделителем между разделами заголовка цикла типа for служит точка с запятой.

С помощью цикла типа for удобно находить суммы, произведения, искать максимальные и минимальные значения и т.п. При нахождении суммы некоторой переменной, например S присваивается значение 0, затем в цикле к этой переменной прибавляется соответствующий член заданной последовательности. Рассмотрим пример вычисления суммы квадратов натурального ряда чисел от 1 до n.

$S = 1^2 + 2^2 + \ldots + n^2$.

Программа имеет вид:

```
//sum_sq.c  сумма квадратов натурального ряда
 #include<stdio.h>
 #include<conio.h>
 main()
 {   int S,n,i;
   clrscr();
   printf(“Введите n\n”);
   scanf(“%d”,&n);
   for(S=0,i=1;i<=n;i++) S+=i*i;
   printf(“n=%d S=%d\n”,n,S);
```

```
getch();
return 0; }
```

В программе определяются целые переменные S, n, i, в отличие от других языков в языках C и C++ учитывается регистр при определении идентификатора (имени) переменной, так что S и s - разные переменные. Функция clrscr производит очистку экрана, содержится в заголовочном файле conio.h. После ввода переменной n следует цикл типа for. В разделе инициализации присваиваются начальные значения переменным S (в которой накапливается сумма квадратов) и i - параметру цикла, присваивания разделяются запятой, раздел инициализации отделяется точкой с запятой. В разделе проверки условия значение i сравнивается с n, при i меньшем или равном n цикл повторяется, в противном случае происходит выход из цикла. Раздел модификации состоит из оператора инкремента (увеличения на 1) значения параметра цикла i. Запись i++ эквивалентна i=i+1.Аналогично в операции декремента имеем i-- ⇔ i=i - 1.

В цикле выполняется один оператор S+=i*i; это оператор составного присваивания, эквивалентный оператору S=S+i*i, поэтому фигурные скобки для тела цикла не требуются. Операторы инкремента, декремента и составного присваивания придают языку C++ лаконичность , позволяют уменьшить объем исходного текста (C++ - самый лаконичный из языков высокого уровня).

После окончания цикла производится печать результата (оператор printf). Функция getch, содержащаяся в заголовочном файле conio.h, ожидает нажатия любой клавиши, при этом пользователь имеет возможность просмотреть результаты работы программы, не прибегая к просмотру экрана пользователя с помощью комбинации [ALT]+[F5]. Как указывалось выше, при записи функций без параметров clrscr и getch следует ставить круглые скобки даже с пустым списком параметров.

Вычисление произведений производится аналогично, но перед циклом переменной, например S, в которой накапливается значение произведения,

присваивается значение не 0, а 1. Здесь также возможен оператор составного присваивания, который при вычислении факториала имеет вид S*=i; что эквивалентно S=S*i. Вообще, оператор составного присваивания применяется и для других бинарных операций (операций с двумя операндами).

3. Выполнение работы

Набрать и откомпилировать приведенные выше программы с применением цикла типа FOR, исправить выявленные ошибки. Ввести несколько вариантов значений аргумента в этих программах, вычислить результат вручную и сравнить с полученными по программе результатами.

Составить программы циклической структуры согласно вариантам заданий, откомпилировать их, ввести исходные данные, проверить полученный результат.

4. Варианты заданий

1. Вводится целое число N>0.Вычислить сумму N членов последовательности

$$S = \frac{(1+1^2)}{(2+1)} + \frac{(1+2^2)}{(2+2)} + \ldots + \frac{(1+N^2)}{(2+N)} .$$

2. Вводится целое число N>0. Вычислить произведение

$$P = \frac{1}{(1+1^2)} * \frac{1}{(1+2^2)} * \ldots * \frac{1}{(1+N^2)} .$$

3. Вводится целое число N>0. Вычислить произведение

$$P = \frac{(1+1)}{(1+1^2)} * \frac{(1+2)}{(1+2^2)} * \ldots * \frac{(1+N)}{(1+N^2)} .$$

4. Вводится целое число N>0.Вычислить сумму N членов последовательности

$$S = \frac{(1+1)}{(2+1^2)} + \frac{(1+2)}{(2+2^2)} + \ldots + \frac{(1+N)}{(2+N^2)} .$$

5. Вводится целое число N>0.Вычислить сумму N членов последовательности

$$S = \frac{(1+1^2)}{(2+1^3)} + \frac{(1+2^2)}{(2+2^3)} + \ldots + \frac{(1+N^2)}{(2+N^3)} .$$

6. Вводится целое число N>0. Вычислить произведение

$$P = \frac{2}{(1+1^2)} * \frac{2}{(1+2^2)} * \ldots * \frac{2}{(1+N^2)} .$$

7. Вводится целое число N>0. Вычислить произведение

$$P = \frac{(1+1^3)}{(1+1^2)} * \frac{(1+2^3)}{(1+2^2)} * \ldots * \frac{(1+N^3)}{(1+N^2)} .$$

8. Вводится целое число N>0.Вычислить сумму N членов последовательности

$$S = \frac{(2+1^2)}{(1+1^3)} + \frac{(2+2^2)}{(1+2^3)} + \ldots + \frac{(2+N^2)}{(1+N^3)} .$$

9. Вводится целое число N>0. Вычислить произведение

$$P = \frac{(3+1^3)}{(2+1^2)} * \frac{(3+2^3)}{(2+2^2)} * \ldots * \frac{(3+N^3)}{(2+N^2)} .$$

10. Вводится целое число N>0.Вычислить сумму N членов последовательности

$$S = \frac{(3+1^2)}{(2+1^3)} + \frac{(3+2^2)}{(2+2^3)} + \ldots + \frac{(3+N^2)}{(2+N^3)} .$$

5. Контрольные вопросы

1. Какой алгоритм является алгоритмом циклической структуры?

2. Типы циклов в языке С.

3. Какие три раздела записываются в круглых скобках для оператора цикла типа for? Какой разделитель между разделами?

4. Что такое вложенные циклы? Примеры.

Лабораторное занятие № 5

Цикл с предусловием

1. Цель работы: Закрепление знаний о программах цикла с предусловием, составление программы цикла с предусловием и работа с ней.

2. Основные сведения

Не всегда число повторений цикла известно заранее, в этих случаях применяются циклы с предусловием (проверка перед циклом) или с постусловием (проверка после цикла).Приведем примеры.

Программа вычисления факториала числа n с применением цикла с предусловием использует формулу

n! = 1*2* . . . *n

т.е. факториал числа n есть произведение всех целых чисел от 1 до n.

```
//fact.c  вычисление факториала
 #include<stdio.h>
 #include<conio.h>
 main()
 {   int F,n,i;
   clrscr();
   printf(“Введите n\n”);
   scanf(“%d”,&n);
   F=1; i=1;
   while(i<n) {F*=i; i++}
   printf(“n=%d n!=%d\n”,n,F);
   getch();
   return 0;}
```

3. Выполнение работы

Набрать и откомпилировать приведенную выше программу с применением цикла типа WHILE, исправить выявленные ошибки. Ввести несколько вариантов значений аргумента в этой программе, вычислить результат вручную и сравнить с полученным по программе результатом.

Составить программу циклической структуры с применением цикла типа WHILE согласно вариантам заданий лабораторной работы №5 откомпилировать ее, ввести исходные данные, проверить полученный результат.

4. Контрольные вопросы

1. Типы циклов в языке С.
2. Как записывается условие продолжения цикла в цикле типа while ?
3. Что такое вложенные циклы? Примеры.
4. Может ли цикл с предусловием не исполниться ни разу? В каком случае это произойдет?

Лабораторное занятие № 6

Цикл с постусловием

1. Цель работы: Закрепление знаний о программах цикла с постусловием, составление программы цикла с постусловием и работа с ней.

2. Основные сведения

В программах с применением цикла с постусловием проверка условия повторения цикла производится после цикла, поэтому цикл с постусловием выполнится по крайней мере один раз.

Программа вычисления двойного факториала числа n с применением цикла с постусловием использует формулу

n!!=1*3*5* . . . *n при n нечетном или n!!=2*4*6* . . . *n при n четном.

```
//dfact.c  вычисление двойного факториала
 #include<stdio.h>
 #include<conio.h>
 main()
 {int F,n,i;
   clrscr();
   printf("Введите n\n");
   scanf("%d",&n);
   F=1;
   if(n%2==0) i=0; else i=-1;//условие четности
   do   { i=i+2;
 F*=i;}
   while(i<n) ;
   printf("n=%d n!!=%d\n",n,F);
   getch();
   return 0;}
```

3. Выполнение работы

Набрать и откомпилировать приведенную выше программу с применением цикла типа DO…WHILE, исправить выявленные ошибки.

Ввести несколько вариантов значений аргумента в этой программе, вычислить результат вручную и сравнить с полученным по программе результатом.

Составить программу циклической структуры с применением цикла типа DO…WHILE согласно вариантам заданий лабораторной работы №5 откомпилировать ее, ввести исходные данные, проверить полученный результат.

4. Контрольные вопросы

1. Какой алгоритм является алгоритмом циклической структуры?
2. Типы циклов в языке С.
3. Как записывается условие продолжения цикла в цикле типа do ... while?
4. Что такое вложенные циклы? Примеры.
5. Может ли цикл с постусловием не исполниться ни разу? Почему?

Лабораторное занятие № 7

Одномерные массивы

1. Цель работы: Закрепление знаний о массивах, составление программ с одномерными массивами.

2. Основные сведения

Массивы - структурированный тип данных с элементами одного типа. Массивы в языке С описываются так же, как простые переменные, но после имени массива в квадратных скобках указывается число его элементов. Например, массив составляют заработные платы сотрудников подразделения предприятия, здесь число элементов равно числу сотрудников, массив образуют максимальные дневные температуры в течение года, число элементов массива - число дней в году. Номер элемента массива называется его индексом, в языках С и С++ индекс элемента массива начинается с нуля, поэтому индекс последнего элемента массива на 1 меньше числа элементов в данном массиве.

Приведем пример программы для нахождения наибольшего элемента в массиве и его индекса.

```
//poiskmax.c  поиск макс. эл-та в массиве.
 #include<stdio.h>
 #include<conio.h>
 #define n 10
 int i,max,k,a[n];//размер массива n
 main() {
 max=a[0];k=0;
 for(i=0;i<n;i++) scanf("%d",&a[i]);
 i=0;
 for(i=0;i<n;i++)
  {if(max<a[i])
   {k=i;max=a[i];}}
 printf("Максимальное число в массиве %d, его индекс %d\n",max,k);
```

```
getch();
return 0;}
```

Здесь директива #define n 10 определяет константу n=10, далее мы описываем массив a с числом элементов n. Индексы элементов массива изменяются от 0 до 9.При задании элемента массива в операторе используется имя массива и значение индекса в квадратных скобках. Первый цикл по i в программе - поэлементный ввод массива a. Затем следует поиск максимального элемента в массиве, также в цикле. Переменная max сравнивается с элементами массива, и если элемент массива больше max, переменной max присваивается значение элемента массива, а переменной k - индекс этого элемента. По окончании цикла переменная max будет иметь значение, равное максимальному элементу массива, а переменная k - значение индекса этого элемента (на 1 меньше его номера в массиве).

3. Выполнение работы

Набрать и откомпилировать приведенную выше программу, исправить выявленные ошибки. Ввести элементы массива, убедиться в правильности нахождения максимального элемента.

Составить программу с применением массивов согласно вариантам заданий, откомпилировать ее, ввести исходные данные, проверить полученный результат.

4. Варианты заданий

1. Ввести массив A из 10 элементов, найти наибольший элемент и переставить его с первым элементом. Преобразованный массив вывести.

2. Ввести массив A из 10 элементов, найти наименьший элемент и переставить его с последним элементом. Преобразованный массив вывести.

3. Ввести массив A из 10 элементов, найти произведение положительных элементов и вывести его на экран.

4. Ввести массив A из 10 элементов, найти произведение отрицательных элементов и вывести его на экран.

5. Ввести массив A из 10 элементов, найти сумму положительных элементов и вывести ее на экран.

6. Ввести массив A из 10 элементов, найти сумму отрицательных элементов и вывести ее на экран.

7. Ввести массив A из 10 элементов, найти сумму элементов, больших 3 и меньших 8 и вывести ее на экран.

8. Ввести массив A из 10 элементов, найти сумму элементов, меньших по модулю 5 и вывести ее на экран.

9. Ввести массив A из 10 элементов, найти сумму элементов, больших 2 и меньше 5, вывести ее на экран.

10. Ввести массив A из 10 элементов, найти сумму элементов, больших 4 и меньших 7 и вывести ее на экран.

5. Контрольные вопросы

1. Что такое массив?
2. Что такое индекс элемента массива?
3. В каких пределах изменяются индексы элементов массива?
4. Для чего нужна директива define?
5. Как ввести и вывести элемент массива?

Лабораторное занятие № 8

Многомерные массивы

1. Цель работы: Закрепление знаний о многомерных массивах, составление программ с многомерными массивами.

2. Основные сведения

Массив может иметь не один, а большее число индексов. Число индексов называется размерностью массива, например, массив с двумя индексами называется двумерным массивом. Таким двумерным массивом является, в частности, матрица системы n линейных алгебраических уравнений с n неизвестными. В то же время столбец свободных членов этой системы является одномерным массивом.

По каждому из индексов устанавливается размер массива, например , описание двумерного массива из четырех строк и трех столбцов имеет вид: float a[4][3]. Такой массив имеет 3*4=12 элементов, первый индекс изменяется от 0 до 3, второй индекс - от 0 до 2.Таким образом, элементы массива суть: a[0][0] , a[0][1], a[0][2] , a[1][0] , ... , a[3][2].

Как правило, при обработке многомерных массивов используются вложенные циклы, т.е. цикл по столбцам располагается внутри цикла по строкам. Таким образом производится ввод и вывод элементов массива, поиск максимальных элементов в строках и т.д.

Приведем пример работы с многомерными массивами. В примере вводится матрица 2x3, элементы 2-го и 3-го столбцов умножаются на 2, преобразованная матрица выводится на экран.

```
//mnog_mas.c многомерные массивы
#include<stdio.h>
 #include<math.h>
 main() {
 int a[2][3],i,j;
 for(i=0;i<2;i++) {
 for(j=0;j<3;j++) {
```

```
printf(“Введите a[%d][%d]=”,i,j);
scanf(“%d”,&a[i][j]);}}
for(i=0;i<2;i++)
for(j=1;j<3;j++) a[i][j]*=2;
for(i=0;i<2;i++) {
for(j=0;j<3;j++)
printf(“%6d”,a[i][j]);
printf(“\n”);}
return 0;}
```

В данном примере при вводе цикл по i является внешним, цикл по j - внутренним, вложенным в цикл по i. При вводе параметр цикла i изменяется от 0 до 2 , внутренний параметр j изменяется от 0 до 3. При умножении i изменяется от 0 до 2 , а параметр j от 1 до 3 , т.к. первый столбец умножать на 2 не нужно.

3. Выполнение работы

Набрать и откомпилировать приведенную выше программу , исправить выявленные ошибки. Ввести элементы массива, убедиться в правильности его обработки.

Составить программу с применением многомерных массивов согласно вариантам заданий, откомпилировать ее, ввести исходные данные , проверить полученный результат.

4. Варианты заданий

1. Ввести матрицу A из 3x3 элементов и вектор B из 3 элементов. Переставить на главную диагональ матрицы элементы вектора, а в вектор - элементы главной диагонали. Преобразованные массивы вывести на экран.

2. Ввести матрицу A из 3x3 элементов, найти наименьший элемент и от элементов всех столбцов, за исключением первого, вычесть наименьший элемент. Преобразованную матрицу вывести.

3. Ввести массив B из 3x4 элементов, найти произведения элементов каждого столбца и вывести их на экран.

4. Ввести матрицу A из 3x3 элементов, найти наибольший элемент и от элементов всех строк, за исключением третьей, вычесть наибольший элемент. Преобразованную матрицу вывести.

5. Ввести массив B из 3x4 элементов, найти суммы элементов каждой строки и вывести их на экран.

6. Ввести матрицы A и B из 3x3 элементов, найти матрицу C, элементы которой равны суммам соответствующих элементов матриц A и B и вывести ее на экран.

7. Ввести матрицы A и B из 2x3 элементов, найти матрицу C, элементы которой равны разностям соответствующих элементов матриц A и B и вывести ее на экран.

8. Ввести массив A из 4 элементов, расположить эти элементы на главной диагонали матрицы B размером 4x4,остальные элементы матрицы B положить равными 1. Вывести матрицу B на экран.

9. Ввести матрицу A из 3x3 элементов, найти наименьший элемент и от элементов всех столбцов, за исключением третьего, вычесть наименьший элемент. Преобразованную матрицу вывести.

10. Ввести массив B из 3x3 элементов, найти суммы элементов каждого столбца и вывести их на экран.

5. Контрольные вопросы

1. Что такое многомерный массив?

2. Сколько индексов может быть у элемента многомерного массива?

3. В каких пределах изменяются индексы элементов массива?

4. Как ввести элементы многомерного массива? Сколько нужно циклов?

5. Где применяются многомерные массивы?

Лабораторное занятие № 9

Символьные данные

1. Цель работы: Закрепление знаний о символьных данных, составление программ с применением символьных данных.

2. Основные сведения

Как известно, данные в компьютере представляются с помощью комбинаций двоичных цифр: нуля и единицы. Как же представить символы, например, буквы русского или английского алфавитов?

Для решения этой задачи разработаны специальные таблицы, называемые кодовыми таблицами. В такой таблице определяется, какой комбинацией единиц и нулей задается определенный символ алфавита.

В США в качестве кодовой таблицы была принята таблица ASCII (American Standard Code for Information Interchange, Американский стандартный код для обмена информацией). В качестве основной единицы представления символьной информацией был принят 1 байт, 8 двоичных разрядов. В этом представлении можно определить $2^8 = 256$ различных символов.

Таблица ASCII состоит из двух частей. В первой части (символы с кодами 0 – 127) представлены цифры, символы латиницы (большие и малые буквы латинского алфавита), знаки препинания, управляющие символы (с кодами 0 – 31), например, переход на новую строку, и другие символы.

Во второй части таблицы ASCII (символы с кодами 128 – 255) представлены символы псевдографики (для графления таблиц), некоторые другие символы, например, символы суммы и интеграла, буквы с диакритическими знаками.

Первая половина кодовой таблицы ASCII используется в качестве международной, вторая половина кодовой таблицы в разных странах используется для представления символов национальных алфавитов, поэтому вторая половина кодовой таблицы может быть различной в разных странах.

В России для представления символов кириллицы (букв русского алфавита) использовались две кодировки: кодировка ГОСТ и альтернативная кодировка. В большинстве случаев применяется альтернативная кодировка. В таблице 1 дана международная часть кодовой таблицы ASCII.

Таблица 1. Международная часть таблицы ASCII.

Код	Символ	Код	Символ	Код	Символ	Код	Символ
32		56	8	80	P	104	h
33	!	57	9	81	Q	105	i
34	“	58	:	82	R	106	j
35	#	59	;	83	S	107	k
36	$	60	<	84	T	108	l
37	%	61	=	85	U	109	m
38	&	62	>	86	V	110	n
39	‘	63	?	87	W	111	o
40	(	64	@	88	X	112	p
41	)	65	A	89	Y	113	q
42	*	66	B	90	Z	114	r
43	+	67	C	91	[	115	s
44	,	68	D	92	\	116	t
45	-	69	E	93	]	117	u
46	.	70	F	94	^	118	v
47	/	71	G	95	_	119	w
48	0	72	H	96	`	120	x
49	1	73	I	97	a	121	y
50	2	74	J	98	b	122	z
51	3	75	K	99	c	123	{
52	4	76	L	100	d	124	|

53	5	77	M	101	e	125	}
54	6	78	N	102	f	126	~
55	7	79	O	103	g	127	

Символы с кодами 128 – 255 в альтернативной кодировке, включая символы кириллицы, приведены в таблице 2.

Таблица 2. Символы альтернативной кодировки.

Код	Символ	Код	Символ	Код	Символ	Код	Символ
128	А	160	а	192	└	224	р
129	Б	161	б	193	┴	225	с
130	В	162	в	194	┬	226	т
131	Г	163	г	195	├	227	у
132	Д	164	д	196	─	228	ф
133	Е	165	е	197	┼	229	х
134	Ж	166	ж	198	╞	230	ц
135	З	167	з	199	╟	231	ч
136	И	168	и	200	╚	232	ш
137	Й	169	й	201	╔	233	щ
138	К	170	к	202	╩	234	ъ
139	Л	171	л	203	╦	235	ы
140	М	172	м	204	╠	236	ь
141	Н	173	н	205	═	237	э
142	О	174	о	206	╬	238	ю
143	П	175	п	207	╧	239	я
144	Р	176	▒	208	╨	240	Ё
145	С	177	▓	209	╤	241	ё
146	Т	178	▓	210	╥	242	≥
147	У	179	│	211	╙	243	≤
148	Ф	180	┤	212	╘	244	⌠
149	Х	181	╡	213	╒	245	⌡
150	Ц	182	╢	214	╓	246	÷
151	Ч	183	╖	215	╫	247	≈
152	Ш	184	╕	216	╪	248	⎝
153	Щ	185	╣	217	┘	249	·
154	Ъ	186	║	218	┌	250	□
155	Ы	187	╗	219	▓	251	√
156	Ь	188	╝	220	▄	252	№
157	Э	189	╜	221	▌	253	¤
158	Ю	190	╛	222	▐	254	▪

159	Я	191	╗	223	■	255	

В операционной системе Windows отсутствуют символы псевдографики, т.к. начертание линий производится другими средствами. В этой операционной системе принята кодовая таблица Windows 1251. Символы кириллицы имеют здесь коды 192 –255 (без букв ё), Ё имеет код 168, ё – код 184:

Таблица 3. Символы кириллицы в кодировке Windows 1251.

Код	Символ	Код	Символ	Код	Символ	Код	Символ
168	Ё	207	П	224	а	241	с
184	ё	208	Р	225	б	242	т
192	А	209	С	226	в	243	у
193	Б	210	Т	227	г	244	ф
194	В	211	У	228	д	245	х
195	Г	212	Ф	229	е	246	ц
196	Д	213	Х	230	ж	247	ч
197	Е	214	Ц	231	з	248	ш
198	Ж	215	Ч	232	и	249	щ
199	З	216	Ш	233	й	250	ъ
200	И	217	Щ	234	к	251	ы
201	Й	218	Ъ	235	л	252	ь
202	К	219	Ы	236	м	253	э
203	Л	220	Ь	237	н	254	ю
204	М	221	Э	238	о	255	я
205	Н	222	Ю	239	п		
206	О	223	Я	240	р		

В кодовой таблице ASCII (и в других кодовых таблицах, в которых на символ отводится 1 байт) можно представить 256 различных символов. Этого достаточно для буквенных алфавитов, но недостаточно для азиатских языков

с иероглифической письменностью (количество иероглифов составляет несколько тысяч). Поэтому была введена кодировка Unicode, в которой на каждый символ выделяется не 1 байт, а 2 байта, 16 двоичных разрядов. В кодировке Unicode можно представить $2^{16} = 65536$ различных символов, что достаточно для всех языков, в том числе и с иероглифической письменностью.

В языке C++ символьные данные имеют типы char или unsigned char и занимают 1 байт оперативной памяти. Кодировка зависит от реализации компилятора.

В среде программирования Borland C++ Builder в режиме консольного приложения используется кодировка ASCII, русские буквы записываются в альтернативной кодировке (таблица 2). В режиме оконного приложения используется кодировка Windows 1251.

Ввод символьных данных в C++ Builder производится в кодировке Windows 1251, а вывод в консольном приложении – в альтернативной кодировке. Поэтому при вводе-выводе символьных и строковых данных в режиме консольного приложения будет иметь место несоответствие введенных и выводимых данных.

Символы латиницы (латинские буквы) имеют одни и те же коды как в альтернативной кодировке, так и в кодировке Windows 1251, поэтому латинские буквы в режиме консольного приложения вводятся и выводятся правильно. Русские же буквы имеют в этих таблицах разные коды, поэтому при воде русских букв на экран выводятся символы псевдографики.

В режиме оконных приложений Windows и ввод и вывод осуществляется по кодовой таблице Windows 1251, поэтому несоответствия вводимых и выводимых данных нет.

В языке C строка определяется как массив символов, в конце строки ставится символ с кодом 0. Такие строки называются нуль-терминальными строками, т.е. строками, в которых признак конца строки – символ с кодом 0. Этот символ добавляет сам компилятор при формировании строки.

В приводимой ниже программе рассматриваются три строки, из которых одна постоянная, задается в программе, и две вводимых строки. Для постоянной строки, определяемой в программе, требуется перекодировка из кодовой таблицы Windows 1251 в альтернативную кодировку. Для вводимых строк и ввод и вывод осуществляется в альтернативной кодовой таблице, перекодировка не требуется.

```
//lab10.cpp  символьные данные
#include<iostream.h>
#include<stdio.h>
#include<conio.h>
char* cyr(char* S){
unsigned char* r=S;
while(*r){if (*r==168) *r=240;
else if(*r==184) *r=241; else
if (*r>=192) if(*r<=239) *r-=64;
else *r-=16; r++;}
return S;} //cyr
main(){
int i,k;
unsigned char j,s0[]="Постоянная строка",s1[20],s2[20],s00[18];
cout<<cyr("s0 без преобразования")<<endl;
cout<<s0<<endl;
strcpy(s00,s0);
cout<<cyr("s0 с преобразованием")<<endl;
cout<<cyr(s00)<<endl;
cout<<cyr("коды s0 без преобразования")<<endl;
strcpy(s00,s0);
for(j=0;j<=strlen(s00);j++)
printf("%u ",s00[j]);printf("\n");
cout<<cyr("коды s0 с преобразованием")<<endl;
```

```
strcpy(s00,s0); cyr(s00);
for(j=0;j<=strlen(s0);j++)
printf("%u ",s00[j]);printf("\n");
cout<<cyr("Введите строку")<<endl;
gets(s1);
cout<<cyr("Вывод введенной строки")<<endl;
cout<<s1<<endl;
cout<<cyr("Вывод кодов введенной строки")<<endl;
for(j=0;j<=strlen(s1);j++)
printf("%u ",s1[j]);printf("\n");
cout<<cyr("Введите строку")<<endl;
gets(s2);
cout<<cyr("Вывод введенной строки")<<endl;
printf("%s\n",s2);
cout<<cyr("Вывод кодов введенной строки")<<endl;
for(j=0;j<=strlen(s2);j++)
printf("%u ",s2[j]);printf("\n");
getch();
return 0;}
```

В приведенной программе для перекодировки из таблицы Windows 1251 в альтернативную кодировку применяется функция cyr (от кириллица). Параметром функции является указатель на строку, после применения этой функции строка оказывается перекодированной.

Из сравнения таблиц 2 и 3 вытекает алгоритм перекодировки в функции cyr: буквы Ё и ё, имеющие в кодировке 1251 коды 168 и 184, заменяются кодами 240 и 241 соответственно, все большие буквы русского алфавита и первая половина маленьких букв (коды 192 – 239) заменяются кодами, меньшими на 64, вторая половина маленьких букв (коды 240 – 255) заменяется кодами, меньшими на 16.

Строка s0, определенная в программе как «Постоянная строка», при непосредственном выводе (без перекодировки) превращается в набор бессмысленных символов. Если же строку s0 скопировать в строку s00 с помощью стандартной функции strcpy, произвести перекодировку строки s00 с помощью функции cyr и вывести s00 на экран, получим правильный вывод «Постоянная строка».

Выводим коды преобразованной и не преобразованной строки s0, убеждаемся, что не преобразованная строка имеет кодировку 1251, преобразованная – альтернативную кодировку.

Теперь введем строки s1 и s2 с клавиатуры, например, «Моя строка1» и «Моя строка2» с помощью функции gets. Эта функция вводит строку целиком, вместе с пробелами. Выводим строки s1 и s2 с помощью оператора вывода cout и функции.printf. Далее выводим коды символов этих строк. Убеждаемся, что и ввод, и вывод осуществляются в альтернативной кодировке, поэтому применять функцию cyr для преобразования нет необходимости.

3. Выполнение работы

Набрать и откомпилировать приведенную выше программу, исправить выявленные ошибки. Основное внимание обратить на выяснение типа кодовой таблицы при различных определениях строки. Ввести несколько вариантов строк и выяснить полученную кодировку.

Модифицировать приведенную выше программу с заданием различных постоянных и вводимых строк. Описать полученные результаты.

4. Контрольные вопросы

1. Как представляются в компьютере символьные данные?
2. Что такое кодовая таблица?
3. Какие типы кодовых таблиц вы знаете?
4. Для чего нужна перекодировка? В каких случаях она необходима?
5. Принципы построения функции перекодировки.

6. Зачем при выводе на экран сообщения «Вывод введенной строки» применяется функция cyr?

7. Почему при выводе введенной строки не применятся функция cyr?

8. Как вывести коды символов введенной строки?

9. Зачем строка s0 копируется в строку s00?

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

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по организации и проведению самостоятельной работы
по дисциплине «Основы программирования в инфокоммуникационных системах»
для студентов направления подготовки
11.03.02 Инфокоммуникационные технологии и системы связи

Ставрополь 2025

ОБЩАЯ ХАРАКТЕРИСТИКА ДИСЦИПЛИНЫ

Цель и задачи дисциплины

Целью дисциплины «Основы программирования в инфокоммуникационных системах» является формирование из набора компетенций будущего бакалавра. Изучение дисциплины обеспечивает реализацию требований Федерального Государственного образовательного стандарта высшего образования по направлению 11.03.02 Инфокоммуникационные технологии и системы связи

Задачи дисциплины: изучить основы алгоритмизации и программирование на примере языка программирования С++.

Место дисциплины в структуре образовательной программы

Дисциплина относится обязательной части. Ее освоение происходит в 3 семестре.

Перечень планируемых результатов обучения по дисциплине (модулю), соотнесённых с планируемыми результатами освоения образовательной программы

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов
ОПК-3. Способен применять методы поиска, хранения, обработки, анализа и представления в требуемом формате информации из различных источников и баз данных, соблюдая при этом основные требования информационной безопасности	ИД-3 $_{ОПК-3}$ решает задачи обработки данных с помощью средств вычислительной техники.	решает задачу обработки данных с помощью средств вычислительной техники, используя язык С++.
ОПК-4. Способен понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности	ИД-4 $_{ОПК-4}$ использует возможности вычислительной техники и программного обеспечения для решения задач управления и алгоритмизации процессов обработки информации.	использует возможности вычислительной техники и программного обеспечения для создания приложений, использует язык С++ для решения задач обработки информации
ОПК-5. Способен разрабатывать алгоритмы и компьютерные программы, пригодные для практического применения	ИД-2 $_{ОПК-5}$ читает коды программных продуктов, написанных на освоенных языках программирования, и вносит требуемые изменения	читает коды, понимает и вносит требуемые изменения в программный код
	ИД-3 $_{ОПК-5}$ разрабатывает оригинальные алгоритмы и компьютерные программы, пригодные для практического применения	разрабатывает оригинальные приложения, пригодные для практического применения

2. ХАРАКТЕРИСТИКА САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Все виды самостоятельной работы по дисциплине «Технологии программирования и алгоритмизация» приведены в таблице «Технологическая карта самостоятельной работы». В основу самостоятельной работы студентов по дисциплине положено конспектирование лекций и рекомендуемых источников, подготовка к компьютерному тестированию знаний, оформление и подготовка к защите отчетов по лабораторным работам, выполнению разноуровневых индивидуальных заданий в ходе лабораторных работ. По каждому виду самостоятельной работы предусмотрены определённые формы отчетности, которые студенты предъявляют преподавателю в ходе лабораторных занятий или в часы индивидуальных консультаций.

Таблица 1 - Технологическая карта самостоятельной работы

Код оцениваемой компетенции, индикатора (ов)	Этап формирования компетенции (№ темы) *(в соответствии с рабочей программой дисциплины)*	Средства и технологии оценки	Вид контроля, аттестация *(текущий/промежуточный)*	Тип контроля *(устный, письменный или с использованием технических средств)*	Наименование оценочного средства
ИД-3 $_{ОПК-3}$ ИД-4 $_{ОПК-4}$ ИД-2 $_{ОПК-5}$ ИД-3 $_{ОПК-5}$	Тема 1-9	Вопросы для собеседования по лабораторным работам	текущий	устный	Вопросы для собеседования по лабораторным работам
ИД-3 $_{ОПК-3}$ ИД-4 $_{ОПК-4}$ ИД-2 $_{ОПК-5}$ ИД-3 $_{ОПК-5}$	Тема 1-9	Вопросы для собеседования	текущий	устный	Вопросы для собеседования

Для успешного освоения дисциплины, необходимо выполнить указанные в таблице виды самостоятельной работы, используя рекомендуемые источники информации.

1. Перечень основной литературы:

1. Тюльпинова, Н. В. Алгоритмизация и программирование Электронный ресурс: Учебное пособие / Н. В. Тюльпинова. - Саратов : Вузовское

образование, 2019. - 200 с. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-4487-0470-3, экземпляров неограничено

2. Бедердинова, О. И Основы алгоритмизации и структурного программирования Электронный ресурс / Бедердинова О. И. : учебное пособие. - Архангельск: САФУ, 2017. - 88 с. - ISBN 978-5-261-01227-6, экземпляров неограничено

3. Забихуллин, Ф. З. Структурное программирование на С++ Электронный ресурс / Забихуллин Ф. З. : учебное пособие. - Уфа : БГПУ имени М. Акмуллы, 2019. - 45 с. - ISBN 978-5-907176-11-9, экземпляров неограничено

4. Конова, Е. А. Алгоритмы и программы. Язык С++ Электронный ресурс / Конова Е. А., Поллак Г. А. : учебное пособие для впо. - 5-е изд., стер. - Санкт-Петербург: Лань, 2020. - 384 с. - Допущено УМО по образованию в области прикладной информатики в качестве учебного пособия для студентов, обучающихся по направлению «Прикладная информатика». - ISBN 978-5-8114-5431-0, экземпляров неограничено

Белева, Л.Ф. Программирование на языке С++ Электронный ресурс: учебное пособие / Л.Ф. Белева. - Саратов : Ай Пи Эр Медиа, 2018. - 81 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-4486-0253-5, экземпляров неограниченно

2. Перечень дополнительной литературы:

1. Белоцерковская, И. Е. Алгоритмизация. Введение в язык программирования С++ / И.Е. Белоцерковская; Н.В. Галина; Л.Ю. Катаева. - 2-е изд., испр. - Москва: Национальный Открытый Университет «ИНТУИТ», 2016. - 197 с., экземпляров неограничено

2. Седжвик, Р. Алгоритмы на С++ / Р. Седжвик. - 2-е изд., испр. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 1773 с., экземпляров неограниченно

3. Лубашева, Т. В. Основы алгоритмизации и программирования: учебное пособие / Т.В. Лубашева, Б.А. Железко. - Минск : РИПО, 2016. - 378 с.: ил. - http://biblioclub.ru/. - Библиогр. в кн. - ISBN 978-985-503-625-9, экземпляров неограничено

4. Зоткин, С. П. Программирование на языке высокого уровня С/С++ Электронный ресурс / Зоткин С. П. : конспект лекций. - 3-е изд. - Москва: МИСИ – МГСУ, 2018. - 140 с. - ISBN 978-5-7264-1810-0, экземпляров неограничено

5. Каширская, Е. Н. Процедурное программирование: Практикум Электронный ресурс / Каширская Е. Н. - Москва : РТУ МИРЭА, 2020. - 75 с., экземпляров неограничено

6. Баженова, И.Ю. Введение в программирование Электронный ресурс : учебное пособие / В.А. Сухомлин / И.Ю. Баженова. - Введение в программирование,2020-07-28. - Москва, Саратов: Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. - 327 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-4487-0073-6, экземпляров неограниченно

3. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ИЗУЧЕНИЮ ТЕОРЕТИЧЕСКОГО МАТЕРИАЛА

При самостоятельном изучении тем лекционных занятий и заданной литературы студентам рекомендуется:

- в день проведения занятий проработать, изученный на занятии учебный материал по конспекту. При этом необходимо выделить вопросы, на которые следует обратить большее внимание при дальнейшем изучении текущей темы. После проработки учебного материала необходимо ответить на, рекомендуемые контрольные вопросы;
- с целью качественного закрепления изученного материала в последующем следует более детально проработать учебный материал с использованием рекомендованной литературы. Студентам рекомендуется выделить вопросы, требующие более детальной проработки с помощью преподавателя;
- накануне проведения лекции студенту рекомендуется закрепить ранее изученный теоретический материал, подготовить вопросы, требующие уточнения у преподавателя.

Рекомендации по организации работы с литературой

Для овладения, закрепления и систематизации знаний студентам рекомендуется самостоятельная работа с рекомендованной литературой и конспектом лекций. При самостоятельной работе с рекомендованной литературой студентам рекомендуется проводить конспектирование учебного материала, изложенного в рекомендованных источниках.

Конспектирование источников проводится при детальном изучении темы, при этом студентам рекомендуется:

- дополнять конспект лекционного занятия путем его расширения по наиболее важным вопросам, рассмотренным на занятии.

Контроль этого вида самостоятельной работы осуществляется на учебных занятиях путем проверки преподавателем конспекта лекций и собеседования.

При конспектировании источников студенты должны исходить из рационального объема, конспектируемого материала. Конспект должен быть кратким и понятным при его использовании после изучения текущей темы.

Для успешного освоения дисциплины, необходимо самостоятельно детально изучить представленные темы по рекомендуемым источникам информации.

Работа по конспектированию источников оценивается удовлетворительно если:

- конспект имеет достаточный объем детализации по, изучаемой теме, обеспечивающий заданный уровень освоения теоретического материала дисциплины;

- конспект оформлен аккуратно, изучаемый материал изложен последовательно в соответствии с порядком изучения дисциплины, содержит обобщающие выводы по каждой теме.

4. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

При выполнении практикума по дисциплине (лабораторные работы) для достижения базового уровня сформированности компетенций студент должен уметь применять базовые знания по разделам дисциплины, владеть навыками самостоятельного решения учебных задач базового уровня.

При выполнении практикума по дисциплине (лабораторные работы) для достижения повышенного уровня сформированности компетенций студент должен уметь применять повышенные знания по разделам дисциплины, владеть навыками самостоятельного решения учебных задач повышенного уровня.

Методические указания при подготовке к лабораторной работе

Подготовку к лабораторной работе рекомендуется проводить в следующей последовательности:

- уяснить тему и цель, предстоящей лабораторной работы;
- изучить теоретический материал в соответствии с темой лабораторной работы (рекомендуется использовать рекомендованную литературу, конспект лекций, учебное пособие (практикум по лабораторным работам);
- ознакомиться с оборудованием и материалами, используемыми на лабораторной работе (при использовании специализированного оборудования или программного обеспечения необходимо изучить порядок и правила его использования).

Методические указания при выполнении лабораторной работы

При выполнении лабораторной работы студенты должны строго соблюдать, установленные требования безопасности.

При выполнении лабораторной работы студентам рекомендуется:

- уяснить цель, выполняемых заданий и способы их решения;
- задания, указанные в лабораторной работе выполнять в той последовательности, в которой они указаны в лабораторном практикуме;
- при выполнении практического задания и изучении теоретического материала использовать помощь преподавателя;
- оформить отчет по лабораторной работе;
- ответить на контрольные вопросы.

Методические указания при подготовке к защите лабораторной работы

При подготовке к защите лабораторной работы студентам рекомендуется:

- подготовить отчет по лабораторной работе;
- подготовить обоснование, сделанных выводов;

- закрепить знания теоретического материала по теме лабораторной работы (рекомендуется использовать контрольные вопросы);
- знать порядок проведения расчетов (проводимых исследований);
- уметь показать и пояснить порядок исследований при использовании специализированного оборудования или программного обеспечения.

Защита лабораторной работы осуществляется путем собеседования студента с преподавателем. При собеседовании студент представляет на проверку отчет по лабораторной работе. Собеседование со студентом, претендующим на оценку «отлично» проводится по вопросам и практическим заданиям повышенного уровня обучения. Собеседование со студентом, не претендующим на оценку «отлично» проводится по вопросам и практическим заданиям базового уровня обучения.

5. ВОПРОСЫ ДЛЯ СОБЕСЕДОВАНИЯ ПО САМОСТОЯТЕЛЬНОМУ ИЗУЧЕНИЮ ЛИТЕРАТУРЫ

Вопросы для собеседования

Вопросы (задача, задание) для проверки уровня обученности

1. Для чего нужны диаграммы деятельности UML?
2. Что такое состояние действия и состояние деятельности?
3. Какие нотации существуют для обозначения переходов и ветвлений в диаграммах деятельности?
4. Какой алгоритм является алгоритмом разветвляющейся структуры?
5. Чем отличается разветвляющийся алгоритм от линейного?
6. Что такое условный оператор? Какие существуют его формы?
7. Что такое составной оператор? Каков формат его записи?
8. Какие операторы сравнения используются в Си?
9. Что называется простым условием? Приведите примеры.
10. Что такое составное условие? Приведите примеры.
11. Какие логические операторы допускаются при составлении сложных условий?
12. Может ли оператор ветвления содержать внутри себя другие ветвления?
13. Что такое множественный выбор?
14. В каких случаях применяется переключатель?
15. Зачем ставится в переключателе оператор break?
16. Зачем в переключателе употребляется зарезервированное слово default?
17. Какие UML-диаграммы показывают работу переключателя?
18. Какой алгоритм является алгоритмом циклической структуры?
19. Типы циклов в языке Си.

20. Назовите основные параметры цикла.

21. Что представляют собой операторы инкремента и декремента, в каких формах они используются?

22. Как записывается условие продолжения цикла в циклах типа while и do ... while?

23. Основная форма цикла do … while.

24. Какие три раздела записываются в круглых скобках для оператора цикла типа for? Какой разделитель между разделами?

25. Что такое вложенные циклы? Примеры.

26. Как образуется бесконечный цикл и как выйти из него?

27. Схема приведения цикла while к эквивалентному ему циклу for.

28. Назовите операторы, способные изменять естественный порядок выполнения вычислений.

29. Для чего нужен оператор break?

30. Где употребляется оператор continue и для чего он используется?

31. Для чего используется оператор return?

32. Какой тип должно иметь выражение?

33. Как объявляются одномерные массивы в языке C++?

34. Какими должны быть размерности при описании статического массива в языке C++?

35. Каков диапазон изменения индекса массива в языке C++?

36. Каким образом производится инициализация массива в языке C++?

37. Чем является идентификатор массива?

38. Для чего используется управляющая последовательность \t?

39. Как представляется в C++ двумерный массив?

40. Где и каким образом хранится двумерный массив?

41. В каких пределах изменяются индексы двумерного массива?

42. Какими способами можно описать двумерный массив?

43. Какие действия выполняются для каждой строки при вычислении количества положительных элементов?

44. В каком месте программы следует записывать операторы инициализации накапливаемых в цикле величин?

45. Каким образом в динамической области памяти можно создавать двумерные массивы?

46. Способы создания динамического массива.

47. Каким образом производится обращение к элементам динамических массивов?

48. Что представляет собой функция в C++? Что нужно для ее использования?

49. Приведите пример заголовка функции.

50. Что включает в себя определение функции?

51. В чем отличие функции от других программных объектов?

52. Каким образом происходит вызов функции?

53. Охарактеризуйте существующие способы решения проблемы получения из подпрограммы признака ее аварийного завершения.

54. Каков механизм передачи параметров в функцию?
55. Способы передачи входных данных.
56. Что представляет собой средство C++, называемое значениями параметров по умолчанию?
57. Каким образом происходит передача в функцию имени функции?
58. Каким образом происходит передача одномерных массивов в функцию?
59. Каким образом происходит передача строк в функцию?
60. Каким образом происходит передача двумерных массивов в функцию?
61. Каким образом происходит передача структур в функцию?
62. Какая функция называется рекурсивной? Преимущества и недостатки рекурсии.
63. В чем смысл использования шаблонов?
64. Как записать параметр шаблона?
65. Перечислите основные свойства параметризованных классов?
66. Что такое ассоциация?
67. Что такое наследование?
68. Что такое агрегация?
69. Что такое зависимость?
70. Назовите и охарактеризуйте три основных принципа в использовании классов.

1. Критерии оценивания компетенций (*в соответствии с результатами освоения дисциплины***)**

Оценка «**отлично**» выставляется студенту, если студент дал полный, развернутый ответ на поставленные вопросы, при этом показана совокупность осознанных знаний по дисциплине, доказательно раскрыты основные положения вопросов; в ответе прослеживается четкая структура, логическая последовательность, отражающая сущность раскрываемых понятий. Знание демонстрируется на фоне понимания его в системе дисциплины. Ответ изложен литературным языком с использованием технической терминологии. Могут быть допущены недочеты в определении понятий, исправленные студентом самостоятельно в процессе ответа, продемонстрировал высокое умение применять полученные знания на практике через решение конкретной задачи, свободно без затруднений справился с поставленной задачей, показав владение разносторонними приемами и навыками ее выполнения, не допустил ошибок и неточностей.

Оценка «**хорошо**» выставляется студенту, если студент дал полный, развернутый ответ на поставленные вопросы, при этом показано умение выделить существенные и несущественные признаки, причинно-следственные связи. Ответ четко структурирован, логичен, изложен литературным языком с использованием технической терминологии. Могут быть допущены 2-3 неточности или незначительные ошибки, исправленные студентом с помощью преподавателя, продемонстрировал умение применять

полученные знания на практике через решение конкретной задачи, справился с поставленной задачей, показав владение необходимыми приемами и навыками ее выполнения, при этом допустил не более одной ошибки.

Оценка «**удовлетворительно**» выставляется студенту, если студент дал недостаточно полные и недостаточно развернутые ответы по вопросам билета. Логика и последовательность изложения имеют нарушения. Допущены ошибки в раскрытии понятий, употреблении терминов. Студент не способен самостоятельно выделить существенные и несущественные признаки и причинно-следственные связи. В ответе отсутствуют выводы. Умение раскрыть значение обобщенных знаний не показано. Речевое оформление требует поправок, коррекции, продемонстрировал посредственное умение применять полученные знания на практике через решение конкретной задачи, с трудом справился с поставленной задачей, при этом допустил не более двух ошибок.

Оценка «**неудовлетворительно**» выставляется студенту, если ответ представляет собой разрозненные знания с существенными ошибками по вопросу. Присутствуют фрагментарность, нелогичность изложения. Студент не осознает связь обсуждаемых вопросов с другими объектами дисциплины. Отсутствуют выводы, конкретизация и доказательность изложения. Речь неграмотная, гистологическая терминология не используется. Дополнительные и уточняющие вопросы преподавателя не приводят к коррекции ответа студента, ответ на вопросы полностью отсутствует или студент отказался от ответа, не продемонстрировал умение применять полученные знания на практике через решение конкретной задачи, не справился с поставленной задачей или допустил при ее решении три и более серьезные ошибки.

3. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Процедура проведения данного оценочного мероприятия включает в себя:

- отдельное собеседование по каждой из указанных тем;
- составление конспекта.

Предлагаемые задания позволяют проверить компетенции **ОПК-3, ОПК-4, ОПК-5**

Для подготовки к каждой теме следует изучить и законспектировать рекомендованные источники и научную литературу и на их основании подготовиться к указанным в методических материалах вопросам для обсуждения.

При проверке задания, оцениваются

- увереное владение материалом;
- умение четко и логично излагать свои мысли:
- умение творчески подходить к решению основных вопросов темы;
- самостоятельность мышления,
- полное соответствие конспекта установленным требованиям;

- самостоятельная устная речь, полностью раскрывающая суть суть проблематики собседования.

7. ВОПРОСЫ ДЛЯ СОБЕСЕДОВАНИЯ ПРИ ПРОВЕРКЕ ГОТОВНОСТИ К ЛАБОРАТОРНЫМ РАБОТАМ

Вопросы для защиты лабораторных работ

Тема: Форматы представления констант и данных, виды операций, ввод-вывод в языке C++

1. Форматы представления констант и данных, виды операций, ввод-вывод в языке C++

Тема: Исследование программирования алгоритмов разветвляющейся структуры

1. Операторы if и switch языков Си и C++.

Тема: Исследование программирования алгоритмов циклической структуры

1. Операторы while, do … while и for языков Си и C++.

Тема: Исследование программирования алгоритмов с использованием статических массивов

1. Определение и использование статических массивов в языках программирования Си и C++.

Тема: Исследование программирования алгоритмов с использованием динамических массивов

1. Связь между массивами и указателями в языках Си и C++.
2. Выделение и освобождение динамической памяти в языках Си и C++.
3. Обращение к элементам динамического массива.

Тема: Исследование программирования алгоритмов с использованием многомерных массивов

1. Определение и использование статических и динамических многомерных массивов в языках Си и C++.

Тема: Исследование средств обработки строк в C++

1. Определение и операции со строками стандартной библиотеки C++.

Тема: Исследование программирования алгоритмов обработки ASCIIZ строк

1. Определение и использование ASCIIZстрок в языках Си и C++.
2. Исследование средств стандартной библиотеки для обработки ASCIIZстрок.

Тема: Исследование возможностей повторного использования кода в структурном программировании

1. Определение и использование обычных и встраиваемых функций в языках Си и C++.

Тема: Исследование средств препроцессора

1. Средства препроцессора в языках Си и C++.
2. Определение и использование макроопределений в языках Си и C++.
3. Сравнение макроопределений и функций.

Тема: Исследование средств для создания многомодульных программ

1. Директивы условной компиляции.
2. Создание подключаемых файлов.
3. Статических и динамические библиотеки.
4. Компиляция многомодульных программ с использованием компиляторов GCCи Clang.

Тема: Исследования средств создания классов и объектов в языке C++

1. Определение класса.
2. Использование класса.
3. Определение методов класса.
4. Вложенные классы.

Тема: Исследование средств инициализации объектов и перегрузка операций в языке C++

1. Перегрузка операций.
2. Конструкторы и деструктор.
3. Константы в классе.
4. Поля-массивы в классе.
5. Статические элементы класса.

Тема: Наследование классов в языке C++

1. Простое открытое наследование.
2. Конструкторы и деструкторы при наследовании.
3. Поля и методы при наследовании.
4. Статические элементы класса при наследовании.
5. Вложенные классы и наследование.
6. Операторы присваивания и принцип подстановки.
7. Функции-операции преобразования.
8. Закрытое и защищенное наследование.
9. Виртуальные и абстрактные методы.
10. Абстрактные классы.

Тема: Исследование средств организации ввода-вывода в языке C++

1. Классификация потоков.
2. Подключение потоков.
3. Операции ввода-вывода.
4. Состояние потока.
5. Форматирование ввода-вывода.
6. Файловые потоки.
7. Буферизация.
8. Строковые потоки.
9. Позиционирование в потоке.
10. Широкие потоки.

1. Критерии оценивания компетенций

оценка «отлично» выставляется, если студент продемонстрировал высокое умение применять полученные знания на практике через решение конкретной задачи с применением алгоритмов и языков программирования, свободно без затруднений справился с поставленной задачей, показав владение разносторонними приемами и навыками ее выполнения, не допустил ошибок и неточностей;

оценка «хорошо» выставляется, если студент продемонстрировал умение применять полученные знания на практике через решение конкретной задачи с применением алгоритмов и языков программирования, справился с поставленной задачей, показав владение необходимыми приемами и навыками ее выполнения, при этом допустил не более одной ошибки;

оценка «удовлетворительно» выставляется, если студент продемонстрировал посредственное умение применять полученные знания на практике через решение конкретной задачи с применением алгоритмов и языков программирования, с трудом справился с поставленной задачей, при этом допустил не более двух ошибок;

оценка «неудовлетворительно» выставляется, если студент не продемонстрировал умение применять полученные знания на практике через решение конкретной задачи с применением алгоритмов и языков программирования, не справился с поставленной задачей или допустил при ее решении три и более серьезные ошибки.

3. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Процедура проведения данного оценочного мероприятия включает в себя:

- собеседование;
- защита лабораторных работ.

Предлагаемые задания позволяют проверить компетенции **ОПК-3, ОПК-4, ОПК-5**

К лабораторному занятию студент должен подготовить ответы на вопросы, проработать дополнительные научные источники, сделать конспект теоретического материала (если это предусмотрено).

Максимальное количество баллов студент получает, если он выполнит лабораторное задание, владеет материалом, умеет логично и четко излагать мысли, показвает самостоятельность выполнения задания.

Основанием для снижением оценки являются:

- слабое знание темы и неполностью выполненное задание;
- невыполненное задание к лабораторной работе;
- отсутствие умения применить теоретические знания для решения лабораторных задач.

При проверке задания, оцениваются

- активное участие в работе;
- увереное владение материалом;
- умение четко и логично излагать свои мысли:
- умение творчески подходить к решению заданий;
- самостоятельность мышления,
- самостоятельная устная речь, полностью раскрывающая суть суть проблематики собседования.