ГАОУ ВО «Дагестанский государственный университет народного хозяйства»

Кафедра «Информационные технологии и информационная безопасность»

Ахмедова З.А., Шарифова Ц.Г.

Лабораторный практикум

по дисциплине

«Программирование на языке С»

Направление подготовки – 10.03.01 «Информационная безопасность», профиль «Безопасность автоматизированных систем»

Махачкала – 2019

УДК 681.3.06
ББК 32.973.2-018 П784

Составители – Ахмедова З.А. старший преподаватель кафедры «Информационные технологии и информационная безопасность» ДГУНХ, Шарифова Ц.Г., старший преподаватель кафедры «Информационные технологии и информационная безопасность» ДГУНХ.

Внутренний рецензент – Савина Елена Владимировна, кандидат физико-математических наук, доцент кафедры «Информационные технологии и информационная безопасность» ДГУНХ.

Внешний рецензент – Абдурагимов Гусейн Эльдарханович, кандидат физико-математических наук, доцент кафедры "Прикладная информатика" Дагестанского государственного университета.

Представитель работодателя – Зайналов Джабраил Тажутдинович, директор регионального экспертно-аттестационного центра «Экспертиза».

Лабораторный практикум дисциплины «Программирование на языке С» разработана в соответствии с требованиями федерального государственного образовательного стандарта высшего образования по направлению подготовки 10.03.01 «Информационная безопасность», утвержденного приказом Министерства образования и науки Российской Федерации от 1 декабря 2016 г., № 1515, в соответствии с приказом от 5 апреля 2017г., № 301 Министерства образования и науки РФ.

Лабораторный практикум по дисциплине «Программирование на языке С» размещена на сайте www.dgunh.ru

Шарифова Ц.Г., Ахмедова З.А. Лабораторный практикум по дисциплине «Программирование на языке С» для направления подготовки «Информационная безопасность», профиль «Безопасность автоматизированных систем» – Махачкала: ДГУНХ, 2019. – 54 с.

Рекомендовано к утверждению Учебно-методическим советом ДГУНХ. Председатель Учебно-методического совета ДГУНХ, проректор по учебной работе, доктор экономических наук, профессор Казаватова Н.Ю. 29 мая 2019 г.	Одобрено Советом факультета «Информационные технологии и управление 25 мая 2019 г., протокол № 6. Председатель Совета Раджабов К.Я, к.э.н., доцент
Одобрено на заседании кафедры «Информационные технологии и информационная безопасность» 20 мая 2019 г., протокол № 10 Зав.кафедрой, к.ф.-м.н., доцент Галяев В.С.	

Печатается по решению Учебно-методического совета Дагестанского государственного университета народного хозяйства.

Содержание:

Лабораторная работа 1.
Часть 1. Структура программы на языке Си.

Цель работы: познакомиться с современными интегрированными системами программирования на языке Си, приобрести навыки разработки и отладки консольных приложений в этой системе.

Общие сведения:

Программа на языке Си состоит из одной или более подпрограмм, называемых функциями. Язык Си является блочно-структурированным. Каждый блок заключается в фигурные скобки {}.Основным блоком в программе консольного приложения на языке Си является главная функция, имеющая имя main().Каждое действие в языке Си заканчивается символом «точка с запятой» — ; В качестве действия может выступать вызов функции или осуществление некоторых операций.Имя функции — это коллективное имя группы описаний и операторов,заключенных в блок (фигурные скобки). За именем функции в круглых скобках указываются параметры функции.

Комментарии в языке Си

В языке Си для комментариев используются символы

/* — начало комментария;

*/ — конец комментария.

Вся последовательность, заключенная между этими символами, является комментарием.

Это удобно для написания многострочных комментариев:

```
int a;   /* целаяпеременная */
```

Многострочные комментарии также удобно использовать при отладке для сокрытия от выполнения части кода.

В дополнение к этому, для написания коротких комментариев могут использоваться символы //. При этом комментарием является все, что расположено после символов // и до конца строки:

```
float b; // вещественная переменная
```

Главная функция

При выполнении консольного приложения, написанного на языке Си, операционная система компьютера передаёт управление функции с именем main(). Функцию main() нельзя вызывать из других функций программы, она является управляющей.

Следующие за именем функции круглые скобки предназначены для указания параметров (аргументов), которые передаются в функцию при обращении к ней. В данном случае операционная система не передаёт в функцию main() никаких аргументов, поэтому список аргументов в круглых скобках пустой.

Главную функцию можно записать по-разному:

```
int main()
```

void main().

Перед именем функции указывается тип возвращаемого значения. При обращении к главной функции значение возвращается операционной системе. Последняя запись не будет возвращать значения. Однако void main() — не совсем корректная запись, так как сообщает компилятору, что функция main() не возвращает никакого значения.

При этом запись int main() сообщает компилятору о возвращении целочисленного значения, которое необходимо операционной системе и сообщает ей о том, что программа завершилась корректно. Если же это значение не возвращено, то операционная система понимает, что программа завершилась в аварийном режиме. Для возврата целочисленного значения перед завершением функции добавляется строка

return 0; // вещественная переменная

В фигурные скобки заключены описания и операторы. В общем случае программа может содержать несколько функций. Каждая функция имеет список передаваемых в нее параметров, указанный в круглых скобках, и набор операций, заключенных в блок, ограниченный фигурными скобками.

Пример: Вывод на экран сообщения "Hello, world!".

```
#include <stdio.h>  // Подключение библиотеки ввода-вывода
int main()        // Главная функция
{
 printf("Hello, world!"); // Вывод сообщения
 getchar();              // Задержка окна консоли
 return 0;
}
```

Результат работы программы:

Часть2.Линейные алгоритмы и операторы ввода-вывода

Цель работы: познакомиться с основными принципами программирования с операторов ввода-вывода синтаксические особенности определения и работы с операторами ввода-вывода данных различных типов.

Задание: составить программы на языке Си для решения представленных задач.

Задача 1: Поезд едет из пункта А в пункт Б со средней скоростью V. Составить программу для нахождениявремени пути t1, если есть встречный ветер, скорость которого V1, и времени t2, если ветра нет. Расстояниемежду пунктами А и Б считать известным и равным S.

Исходныйкодпрограммы:

```
#include
#include
intmain ()
{ floatt1,t2,v1,v2,s;
printf ("S:");
scanf ("%f",&s);
printf ("V1:");
scanf ("%f",&v1);
printf ("V2:");
scanf ("%f",&v2);
t1=(v1-v2)/s;
t2=v2/s;
printf ("t1=%0.1f,t2=%0.1f",t1,t2);
getchar();
getchar();
return 0;
}
```

Встроенные математические операции языка Си.

+, - плюс
-, - минус
*, - умножение
/, - деление
% - деление с остатком

Основные операции на языке Си.

1) арифметические
2) логические
3) адресные
4) операции отношения
5) операции присваивания

Задача 2: Даны две стороны прямоугольника (sizeofrectangle) (целые числа - integer). Нужно вычислить периметр (perimeter P ofthisrectangle) этого прямоугольника.

Решение для периметра

```
#include<stdio.h>
#include <stdlib.h>
#include <math.h>

// Начало программы
intmain(){
  // int - тип числа, a - название места
int a;        // местодляоднойстороны (to store one size)
int b;        // местодлявторойстороны (to store second size)
intperimetr; // местодляпериметра (to store result - perimeter)

// Значения для a и b:
   a = 10;
   b = 22;
   // Вычислим периметр (count),
   // +  сложение
   // *  умножение
   // -  минус (разность)

perimetr = ( a + b ) * 2;

   // Печать периметра
printf("периметрравен: %d\n", perimetr);
return 0;
}
```

Задача 3: Автомобиль (car) проехал N км за h часов. Написать программу, которая вычисляет скорость этого автомобиля.

```
#include <stdio.h>
#include <stdlib.h>
#include<math.h> // для извлечения коней и т.д.

// Начало программы
intmain(){
   // int - тип числа, dimension - название переменной
floatpath; // путь  (в метрах)
floathour;        // время
floatspeed;      //для метров

   // Пусть программа получает значение path и hour
   // когда будет работать с клавиатуры
   // %f - для чтения дробных чисел
```

```
scanf("%f", &path);

  // Вычислим скорость,
speed = path / hour;

  // Печать скорости
  // %0.2f - печать 2-х знаков после запятой без отступов слева
printf("%0.2f км/час\n", speed);
return 0;
}
```

Задания для самостоятельного выполнения:

1. Составьте программу для вычисления среднего арифметического элементов числовой последовательности, вводимых с клавиатуры.
2. Составьте программу для проверки является ли последовательность геометрической прогрессией.
3. Введите с клавиатуры число x и определите, сколько раз оно встречается в последовательности.
4. Введите с клавиатуры число x и определите порядковый номер первого числа, равного x.
5. Введите с клавиатуры число x и определите порядковый номер последнего числа, равного x.
6. Даны длины рёбер параллелепипеда. Найти площадь боковой поверхности.
7. Вычислить высоту треугольника, опущенную на сторону a, по известным значениям длин его сторон a, b, c.
8. Треугольник задан координатами своих вершин. Найти периметр треугольника.
9. Известна длина окружности. Найти площадь круга, ограниченного этой окружностью.
10. Даны длины рёбер параллелепипеда. Найти его объём.

Лабораторная работа 2. Условные операторы

Цель работы: Получение навыков программирования линейных и разветвляющихся алгоритмов и принципы их программирования на языке Си.

Общие сведения:

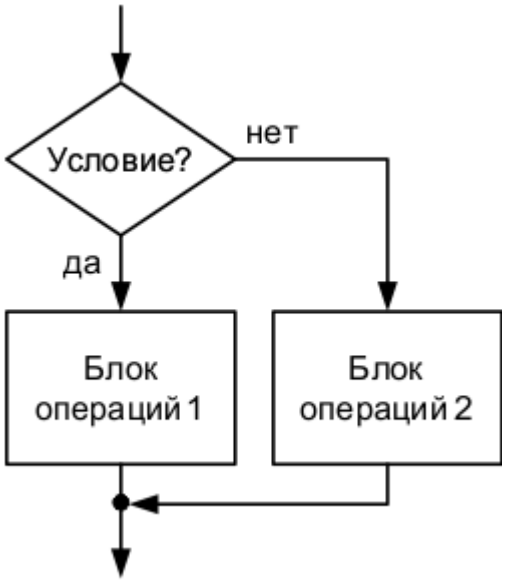
Разветвляющимся называется такой алгоритм, в котором выбирается один из нескольких возможных вариантов вычислительного процесса. Каждый подобный путь называется ветвью алгоритма. Признаком разветвляющегося алгоритма является наличие операций проверки условия. Чаще всего для проверки условия используется условный оператор if.

Условный оператор

Условный оператор if может использоваться в форме полной или неполной развилки.

Неполнаяформа ветвления	Полнаяформа ветвления

if (Условие)
{
БлокОпераций1;
}
	if (Условие)
	{
	БлокОпераций1;
	}
	else
	{
	БлокОпераций2;
	}

В случае неполной развилки если **Условие** истинно, то **БлокОпераций1** выполняется, если **Условие** ложно, то **БлокОпераций1** не выполняется.

В случае полной развилки если **Условие** истинно, то выполняется **БлокОпераций1**, иначе выполняется **БлокОпераций2.**

БлокОпераций может состоять из одной операции. В этом случае наличие фигурных скобок, ограничивающих блок, необязательно.

Операторы отношения, используемые в условии:

Оператор	Значение	Пример
==	равно	if (x == 0.06)
>	больше	if (s > 40)
<	меньше	if (s < 40)
>=	большеилиравно	if (y >= 10000)
<=	меньшеилиравно	if (t <= l)
!=	неравно	if (count != 1)

Примернаязыке СИ

```
#define _CRT_SECURE_NO_WARNINGS //
длявозможностииспользованияscanf
#include <stdio.h>
int main()
{
  int k;          // объявляемцелуюпеременную k
  printf("k= ");   // выводимсообщение
  scanf("%d", &k); // вводимпеременную k
  if (k >= 5)      // если k>5
    printf("%d >= 5", k); // выводим "ЗНАЧЕНИЕ >= 5"
  else            // иначе
    printf("%d < 5", k);  // выводим "ЗНАЧЕНИЕ < 5"
  getchar(); getchar();
  return 0;
}
```

Результатом программы будет

```
K=3
3<5
```

Задание 1. Составить программы на языке Си с использованием условного оператор if

Задача 1. Составить программу вычисления стоимости покупки со скидкой 10 процентов если сумма покупки превышает 1000 рублей.

```
#include<stdio.h>
#include<math.h>
intmain()
{
   inta,b;
   printf ("Введите сумму покупки\n");
   scanf ("%d", &a);
   if (a>1000)
   {
      b = a - a * 0.1;
      printf ("Сумма покупки со скидкой равна %d", b);
   }
   else
   {
   printf ("Сумма покупки %d", a);
   }
   return 0;
}
```

Задача 2.

```
#include <stdio.h>
#include <math.h>
int main() {
int x;
printf("Введитезначениех:");
scanf("%d" , &x);
if (x==0) {
printf("0");
   } else if ((x>0)&&(x%7==0)) {
printf("1");
   } else printf("-1");
return 0;
   }
```

Задача 3.Ввести 2 числа. Если больше будет первое, возвести его в квадрат. Иначе вывести на экран cos этого числа.

```
#include<stdio.h>
#include<math.h>
intmain()
```

```
{
intx,y;
printf("Введите 2 числа \n");
scanf("%d", &x);
scanf("%d", &y);
if(x>y) {

printf("%d", x * x);
}
else {
printf("%f", cos(x));
}
        return 0;
}
```

Задача 4. Даны три действительных числа .Выбрать из них те которые принадлежат интервалу (1,3).

```
#include <stdio.h>
#include <math.h>
int main()
{
inta,b,c;
printf("Введите 3 числа");
scanf("%d",&a);
scanf("%d",&b);
scanf("%d",&c);
if (a>=1 && a<=3){
printf("%2d",a);
}
if (b>=1 && b<=3) {
printf("%2d",b);
}
if(c>=1 && c<=3){
printf("%2d",c);
}
return 0;
}
```

Задание 2.Написатьпрограмму, используявложенныйоператорif

```
#include <stdio.h>
#include <stdlib.h> // дляиспользованияфункцииsystem
int main() {
  int key;  // объявляемцелуюпеременнуюkey
  system("chcp 1251"); // переходимвконсолинарусскийязык
  system("cls");       // очищаемокноконсоли
  printf("Введите номер пункта, 1 или 2: ");
```

```
  scanf("%d", &key);   // вводимзначениепеременнойkey
  if (key == 1)        // еслиkey = 1
    printf("\n Выбран первый пункт"); // выводимсообщение
  else if (key == 2)   // иначееслиkey = 2
    printf("\n Выбран второй пункт"); // выводимсообщение
  else                 // иначе
    printf("\n Первый и второй пункты не выбраны"); // выводимсообщение
  getchar(); getchar();
  return 0;
}
```

Результатом программы будет

```
Введите номер пункта, 1 или 2:1
Выбран первый пункт
Введите номер пункта, 1 или 2:2
Выбран второй пункт
Введите номер пункта, 1 или 2:3
Выбран третий пункт
```

Задания для самостоятельного выполнения:

1. Определить расстояние между двумя точками с заданными координатами
2. Вычислить высоту треугольника, опущенную на сторону a, по известным значениям длин его сторон a,b,c.
3. Известна длина окружности. Найти площадь круга, ограниченного окружностью.
4. По данным сторонам прямоугольника вычислить его периметр, площадь и длину диагонали.
5. Составить программу вычисления среднего арифметического и среднего геометрического четырех чисел.
6. Проверить, является вводимое с клавиатуры число делящимся на 7 и не делящимся на 13.
7. Даны три числа x, y, z. Выберите из них положительные и присвойте им обратные их значения.
8. Даны три действительных числа. Выбрать из них те, которые принадлежат интервалу (1, 3).
9. Определить, какая из двух фигур (круг или квадрат) имеет большую площадь. Известно, что сторона квадрата равна a, радиус круга r. Вывести на экран название и значение площади большей фигуры.
10. Даны действительные числа x, y. Если x, y отрицательны, то каждое значение заменить его модулем; если отрицательное только одно из них, то оба значения увеличить на 0.5; если оба положительны – оставить без изменения.

Лабораторная работа 3. Оператор множественного выбораswitch

Цель работы: Получение навыков алгоритмизации разветвляющихся вычислительных процессов, а также принципов проверки их работоспособности.

Общие сведения:

Оператор if позволяет осуществить выбор только между двумя вариантами. Для того, чтобы производить выбор одного из нескольких вариантов необходимо использовать вложенный оператор if. С этой же целью можно использовать оператор ветвления switch.

Общая форма записи

```
switch (ЦелоеВыражение)
{
 case Константа1: БлокОпераций1;
   break;
 case Константа2: БлокОпераций2;
   break;
 . . .
 case Константаn: БлокОперацийn;
   break;
 default: БлокОперацийПоУмолчанию;
   break;
}
```

Оператор ветвления switch выполняется следующим образом:

- вычисляется ЦелоеВыражение в скобках оператора switch;
- полученное значение сравнивается с метками (Константами) в опциях case, сравнение производится до тех пор, пока не будет найдена метка, соответствующая вычисленному значению целочисленного выражения;
- выполняется Блок Операций соответствующей метки case;
- если соответствующая метка не найдена, то выполнится Блок Операций По Умолчанию, описанный в опции default.

Альтернатива default может отсутствовать, тогда не будет произведено никаких действий.
Опция break; осуществляет выход из оператора switch и переход к следующему за ним оператору. При отсутствии опции break будут выполняться все операторы, начиная с помеченного данной меткой и кончая оператором в опции default.

Константы в опциях case должны быть целого типа (могут быть символами).

Задание: Составить программы на языке Си с использованием оператора switch.

Задача 1. Вывести название дня недели по его номеру.

Программа представляет собой сопоставление числового значения и соответствующей ему текстовой строки. Для реализации таких конструкций чаще всего используется оператор ветвления switch.

```
#include <stdio.h>
int main()
{
  int day;
  printf ("Введите номер дня недели: ");
 scanf("%d", &day);
  switch (day)
  {
    case 1: printf("понедельник"); break;
    case 2: printf("вторник"); break;
    case 3: printf("среда"); break;
    case 4: printf("четверг"); break;
    case 5: printf("пятница"); break;
    case 6: printf("суббота"); break;
    case 7: printf("воскресенье"); break;
    default: printf("Неверно введен день недели"; break;
  }
  cin.get(); cin.get();
  return 0;
}
```

Результатом программы будет

Введите номер дня недели : 4

Четверг

Введите номер дня недели : 8

Неверно введен день недели

Задача 2. Составить программу нахождения значения функции, используя оператор CASE

```
#include <stdio.h>
#include<math.h>
int main() {
intk,x;
float z;
printf("Введите k \n");
scanf("%d \n", &k);
printf("Введите x \n");
scanf ("%i \n", &x);
switch(k) {
```

```
case 3:
z = cos(x);
printf("%f", z);
break;
case 0:
z = sin(x);
printf("%f", z);
break;
case 2:
z = 2;
printf ("%d", z);
break;
case 1:
z = 1;
printf("%d", z);
break;
default:
printf("0");
}
return 0;
}
```

Задача 3.Создать программу для нахождения значения функции, используя оператора выбора.

```
     X^2, при k=2
S=   tgx, при k=3
     E^x, при k=1
```

```
#include <stdio.h>
int main ()
{
 intn=0;
 printf ("введите номер задания");
 n=getchar ();
   switch (n)
   {
      case '1' :
      n=2*2; printf( "%d", n); break;
      case '2' :
      n= tan(3); printf ("%d", n); break;
      case '3' :
      n= exp(1); printf ("%d", n); break;
   }
   printf ("\n");
```

```
    return 0;
}
```

Задача 4.

```
#include <stdio.h>
int main ()
{
int n=0;
printf ("введитеномерзадания");
n=getchar ();
switch (n)
{
case '1' :
n=2*2; printf ( "%d", n); break;
case '2' :
n= tan(3); printf ("%d", n); break;
case '3' :
n= exp(1); printf ("%d", n); break;
}
printf ("\n");
return 0;
}
```

Задания для самостоятельного выполнения:

1. Составить программу нахождения корней квадратного уравнения.
2. Используя, оператор безусловного перехода, составьте программу нахождения n!.
3. Составьте программу нахождения значения функции.

$$Y=\begin{cases}\sin x^2, & \text{если } x \geq 2\\ tgx, & \text{если } -2 < x < 2\\ |x|-3, & \text{если } x \leq -1.\end{cases}$$

4. Заданы три числа. Опередить, могут ли они представлять собой стороны одного треугольника.
5. Используя условный оператор и оператор перехода, составьте программу нахождения НОД двух натуральных чисел a и b.

6. Заданы длины трех сторон треугольника m, n, h. Определить, является ли треугольник прямоугольным.
7. Пусть населенные пункты обозначаются номерами от 1 до 8. Стоимость одного билета к пункту k определяется так:

$$\text{Cena}=\begin{cases}22, & при \quad k=1\\ 25, & при \quad k=2..4\\ 30, & при \quad k=5,6\\ 35, & в\ остальных\ случаях.\end{cases}$$

Сколько стоит m билетов к населенному пункту, номер которого пользователь должен ввести с клавиатуры.

8. Составить программу нахождения значения функции, используя оператор CASE.

$$S=\begin{cases}tgx, & при \quad k=3\\ x^2, & при \quad k=2\\ e^x, & при \quad k=1\\ 0 & в\ остальных\ случаях.\end{cases}$$

9. Написать программу, которая запрашивает у пользователя номер дня недели и выводит одно из сообщений: «Рабочий день», «Суббота» или «Воскресенье».

10. Написать программу, которая после введенного с клавиатуры числа (в диапазоне от 1 до 999), обозначающее денежную единицу, дописывает слово «рубль» в правильной форме. Например, 5 рублей, 21 рубль и т.д.

Лабораторная работа 4. Операторы циклической структуры.

Цель работы: Изучение алгоритмов работы циклов и принципы их программирования на языке Си.

Общие сведения:Циклом называется блок кода, который для решения задачи требуется повторить несколько раз.Каждый цикл состоит из:

- блока проверки условия
- повторения цикла
- тела цикла

Цикл выполняется до тех пор, пока блок проверки условия возвращает истинное значение.

Тело цикла содержит последовательность операций, которая выполняется в случае истинного условия повторения цикла. После выполнения последней операции тела цикла снова выполняется операция проверки условия повторения цикла. Если это условие не выполняется, то будет выполнена операция, стоящая непосредственно после цикла в коде программы.

В языке Си следующие виды циклов:

while — цикл с предусловием;

do…while — цикл с постусловием;

for — параметрический цикл (цикл с заданным числом повторений).

Общий вид Цикла с предусловием while

while (Условие)

```
{
 БлокОпераций;
}
```

Если **Условие** выполняется (выражение, проверяющее **Условие**, не равно нулю), то выполняется **БлокОпераций**, заключенный в фигурные скобки, затем **Условие** проверяется снова.Последовательность действий, состоящая из проверки **Условия** и выполнения **БлокаОпераций,** повторяется до тех пор, пока выражение, проверяющее **Условие**, не станет ложным (равным нулю). При этом происходит выход из цикла, и производится выполнение операции, стоящей после оператора цикла.

Задание: Составить программы на языке Си с использованием операторов цикла.

Задача 1. Посчитать сумму чисел от 1 до введенногоk

```
#include <stdio.h>
int main() {
  int k;  // объявляем целую переменную key
  int i = 1;
  int sum = 0; // начальное значение суммы равно 0
```

```
  printf("k = ");
  scanf("%d", &k);   // вводим значение переменной k
  while (i <= k)     // пока i меньше или равно k
  {
    sum = sum + i; // добавляем значение i к сумме
    i++;           // увеличиваем i на 1
  }
  printf("sum = %d\n", sum); // вывод значения суммы
  getchar(); getchar();
  return 0;
}
```

Результат работы программы

К=5

Sum = 15

При построении цикла while, в него необходимо включить конструкции, изменяющие величину проверяемого выражения так, чтобы в конце концов оно стало ложным (равным нулю). Иначе выполнение цикла будет осуществляться бесконечно (бесконечный цикл).

Пример бесконечного цикла

```
while (1)
{
  БлокОпераций;
}
```

while — цикл с предусловием, поэтому вполне возможно, что тело цикла не будет выполнено ни разу если в момент первой проверки проверяемое условие окажется ложным.

Например, если в приведенном выше коде программы ввести k=-1, то получим результат

К=-1

sum = 0

Задача 2. Проверка, что пользователь ввел число от 0 до 10

```
#include <stdio.h>
#include <stdlib.h> // для использования функции system()
int main() {
  int num;            // объявляем целую переменную для числа
  do {
    printf("Введите число от 0 до 10: "); // приглашение пользователю
    scanf("%d", &num); // вводчисла
  } while ((num < 0) || (num > 10)); // повторяемциклпоканum<0 илинum>10
  printf("Вы ввели число %d", num); // выводимвведенноезначениенum - от
0 до 10
  getchar(); getchar();
```

```
  return 0;
}
```

Результат работы программы

```
Введите число от 0 до 10: 20
Введите число от 0 до 10: 34
Введите число от 0 до 10: -1
Введите число от 0 до 10: -10
Введите число от 0 до 10: 6
Вы ввели число 6
```

Задания для самостоятельного выполнения:

1. Составить программу нахождения суммы натуральных двухзначных четных чисел.
2. Написать программу, вычисляющую сумму и среднее арифметическое 50 первых натуральных чисел.
3. Составить программу, выводящую на экран квадраты первых 20 натуральных чисел.
4. Составить программу, определяющую количество элементов кратных 5, в последовательности 100 натуральных чисел.
5. Составить программу нахождения среднего арифметического чисел, кратных 3 для ряда из 100 первых натуральных чисел.

Лабораторная работа 5. Оператор цикла с параметром

Цель работы: Изучение алгоритмов работы циклов с параметром и принципы их программирования на языке Си.

Общие сведения:

Оператор цикла с параметром (Параметрический цикл)

Общий вид оператора

for (Инициализация; Условие; Модификация)

```
{
  БлокОпераций;
}
```

for — параметрический цикл (цикл с фиксированным числом повторений). Для организации такого цикла необходимо осуществить три операции:

Инициализация - присваивание параметру цикла начального значения;

Условие - проверка условия повторения цикла, чаще всего - сравнение величины параметра с некоторым граничным значением;

Модификация - изменение значения параметра для следующего прохождения тела цикла.

Эти три операции записываются в скобках и разделяются точкой с запятой**;**Как правило, параметром цикла является целочисленная переменная.

Инициализация параметра осуществляется только один раз — когда цикл for начинает выполняться.Проверка **Условия** повторения цикла осуществляется перед каждым возможным выполнением тела цикла. Когда выражение, проверяющее **Условие** становится ложным (равным нулю), цикл завершается. **Модификация** параметра осуществляется в конце каждого выполнения тела цикла. Параметр может как увеличиваться, так и уменьшаться.

Задание: Составить программы на языке Си с использованием оператора цикла с параметром.

Задача 1. Посчитать сумму чисел от 1 до введенногок

```
#include <stdio.h>
int main() {
  int k;  // объявляем целую переменную key
  int sum = 0; // начальное значение суммы равно 0
  printf("k = ");
```

```
  scanf("%d", &k);   // вводим значение переменной k
  for(int i=1; i<=k; i++) // цикл для переменной i от 1 до k с шагом 1
  {
    sum = sum + i; // добавляем значение i к сумме
  }
  printf("sum = %d\n", sum); // вывод значения суммы
  getchar(); getchar();
  return 0;
}
```

Результатом работы программы будет:
K=5
Sum=15

В записи цикла for можно опустить одно или несколько выражений, но нельзя опускать точку с запятой, разделяющие три составляющие цикла.
Код предыдущего примера можно представить в виде

```
#define _CRT_SECURE_NO_WARNINGS // для возможности
использования scanf
#include <stdio.h>
int main() {
  int k;  // объявляем целую переменную key
  int sum = 0; // начальное значение суммы равно 0
  printf("k = ");
  scanf("%d", &k);   // вводим значение переменной k
  int i=1;
  for(; i<=k; i++) // цикл для переменной i от 1 до k с шагом 1
  {
    sum = sum + i; // добавляем значение i к сумме
  }
  printf("sum = %d\n", sum); // вывод значения суммы
  getchar(); getchar();
  return 0;
}
```

В цикле for может использоваться операция **запятая - , -** для разделения нескольких выражений. Это позволяет включить в спецификацию цикла несколько инициализирующих или корректирующих выражений. Выражения, к которым применяется операция **запятая**, будут вычисляться слева направо.

```
#include <stdio.h>
int main() {
  int k;  // объявляем целую переменную key
```

```
  printf("k = ");
  scanf("%d", &k);   // вводим значение переменной k
  for(int i=1, j=2; i<=k; i++, j+=2) // цикл для переменных
  {                              // (i от 1 до k с шагом 1) и (j от 2 с шагом 2)
    printf("i = %d   j = %d\n", i, j); // выводим значения i и j
  }
  getchar(); getchar();
  return 0;
}
```

Результатом работы программы будет

```
K=5
I=1   J=2
I=2   J=4
I=3   J=6
I=4   J=8
I=5   J=10
```

Задача 2. Метод сортировки одномерного массива “Пузырька”, вывод чисел по возрастанию

```
#include <stdio.h>
int main (void) {
int n=3;
int a[3];
inti,j,temp;
printf("введите последовательность чисел\n");
for(i=0;i<n;i++) scanf("%d", &a[i]);
for (i=1; i<n; i++)
{
temp=a[i];
for(j=i-1; j>=0 && temp<a[j]; j--)
a[j+1]=a[j];
a[j+1]=temp;
}
for (i=0; i<n; i++)
{
printf("%d ", a[i]);
}
return(0);
}
```

Задача 3. Посчитать сумму всех четных чисел от 0 до 200.

```
#include <stdio.h>
int main()
{
```

```
int i;
int sum = 0;
for(i = 0; i <= 6; i += 2)
sum += i;
printf("Суммачетныхчиселот 0 до 200 : %d", sum);
        return 0;
}
```

Задача 4. Имитировать на экране обратный отсчет – выводить последовательно 20,19, итд до 0.

```
#include<unistd.h>
#include<stdio.h>
int main()
{
int i;
int m;
for (int i=20; i>=0;i--)
   {
printf("%d\n",i);
      m=sleep(1);
}
printf("Пуск");
getchar();
return 0;
```

Задача 5.Составить программу нахождения суммы чисел i в диапазоне от 1до 50 s= $\sum_{i=1}^{50} i2$

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
int i;
float s;
for (i=1;i<=50;i++)
   s=i*i;
   i=i+1;
   {
      s=4+s;
   }
printf("s=%4.2f\n", s);
getchar();
return 0;
}
```

Задания для самостоятельного выполнения:

1. $s = 4 * \sum_{i=1}^{40} 1 + 7i$
2. $s = 4 + \sum_{i=1}^{50} i^2$
3. $s = 8 \sum_{i=1}^{20} (i+1)^2$
4. $s = 5{,}3 \sum_{i=1}^{20} i^3$
5. $s = \sum_{i=1}^{20} (2i + i^2)$
6. $s = \sum_{i=1}^{20} (i^2 - 3i + 1)$
7. $s = 9 - \sum_{i=1}^{21} \frac{i}{i+1}$
8. $s = 4 \sum_{i=1}^{8} (i^2 + 5)^{-1}$
9. $Y = 7 \sum_{i=1}^{20} \frac{\sin(\alpha_i)}{\cos(\alpha_i) + 2}$
10. $Y = 2 \sum_{i=1}^{20} (2i^2 - 5/i)$

Лабораторная работа 6. Вложенные циклы

Цель работы: Получение навыков алгоритмизации вложенных циклов, а также принципов проверки их работоспособности.

Общие сведения:

В Си допускаются вложенные циклы, то есть когда один цикл находится внутри другого:

Пример

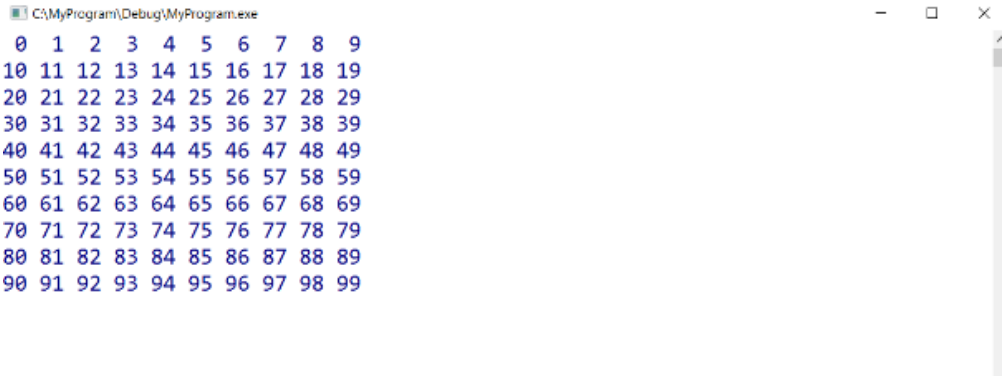
```
for (i = 0; i<n; i++)  // внешний цикл - Цикл1
{
  for (j = 0; j<n; j++)   // вложенный цикл - Цикл2
  {
    ;       // блок операций Цикла2
  }
  // блок операций Цикла1;
}
```

Задание: Составить программы на языке Си с использованием вложенных циклов.

Задача 1. Вывестичислаот 0 до 99, по 10 вкаждойстроке

```
#include <stdio.h>
int main() {
  for(int i=0; i<10; i++) // циклдлядесятков
  {
    for (int j = 0; j < 10; j++) // циклдляединиц
    {
      printf("%2d ", i * 10 + j); // выводимвычисленноечисло (2 знакоместа)
ипробел
    }
    printf("\n"); // вовнешнемциклепереводимстроку
  }
  getchar(); // scanf() неиспользовался,
  return 0;  // поэтомуконсольможноудержатьоднимвызовомgetchar()
}
```

Результатомработыпрограммыбудет



Операторы прерывания и продолжения цикла break и continue

В теле любого цикла можно использовать операторы прерывания цикла - break и продолжения цикла - continue. Оператор break позволяет выйти из цикла, не завершая его. Оператор continue позволяет пропустить часть операторов тела цикла и начать новую итерацию.

Задача 2. Вывести числа от 0 до 99 ниже главной диагонали

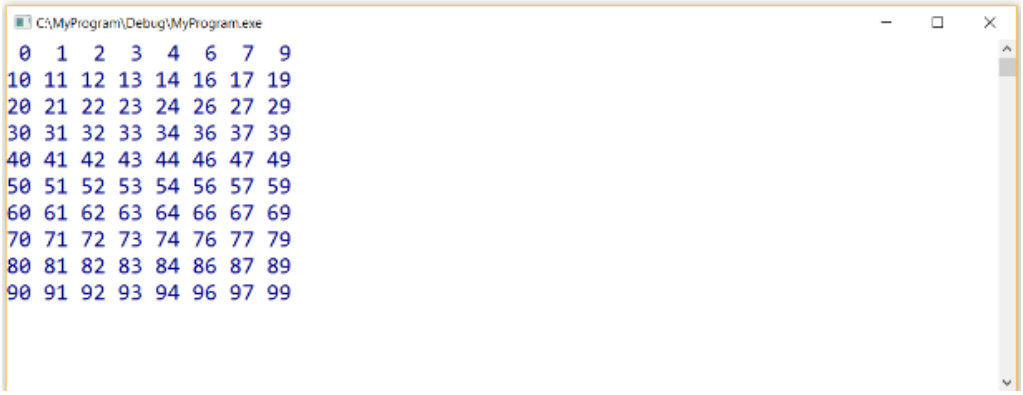
```
#include <stdio.h>
int main() {
  for(int i=0; i<10; i++) // цикл для десятков
  {
    for (int j = 0; j < 10; j++) // цикл для единиц
    {
      if (j > i) // если число единиц больше числа десятков в числе
        break; // выходим из вложенного цикла и переходим к новой строке
      printf("%2d ", i * 10 + j); // выводим вычисленное число (2 знакоместа) и пробел
    }
    printf("\n"); // во внешнем цикле переводим строку
  }
  getchar(); // scanf() не использовался,
  return 0;  // поэтому консоль можно удержать одним вызовом getchar()
}
```

Результатомвыполненияпрограммыбудет

Задача 3. Вывести числа от 0 до 99 исключая числа, оканчивающиеся на 5 или 8

```
#include <stdio.h>
int main() {
  for(int i=0; i<10; i++) // цикл для десятков
  {
    for (int j = 0; j < 10; j++) // цикл для единиц
    {
      if ((j == 5) || (j == 8)) // если число единиц в числе равно 5 или 8,
        continue;             // переходим к следующей итерации цикла
      printf("%2d ", i * 10 + j); // выводим вычисленное число (2 знакоместа) и пробел
    }
    printf("\n"); // во внешнем цикле переводим строку
  }
  getchar(); // scanf() не использовался,
  return 0;  // поэтому консоль можно удержать одним вызовом getchar()
}
```

Результатомвыполненияпрограммыбудет



При вложенных циклах действия операторов break и continueраспространяется только на самую внутреннюю структуру, в которой они содержатся.

Задания для самостоятельного выполнения:

1. В введенном промежутке натуральных чисел найти те, количество делителей у которых не меньше введенного значения. Для найденных чисел вывести на экран количество делителей и все делители.
2. Вывести на экран, из каких простых множителей состоит введенное натуральное число.
3. Найти все совершенные числа до 10000. Совершенное число - это такое число, которое равно сумме всех своих делителей, кроме себя самого. Например, число 6 является совершенным, т.к. кроме себя самого делится на числа 1, 2 и 3, которые в сумме дают 6.
4. Вывести какой-либо символ по диагоналям воображаемого квадрата.
5. Среди натуральных чисел, которые были введены, найти наибольшее по сумме цифр. Вывести на экран это число и сумму его цифр.
6. Вывести на экран "прямоугольник", образованный из двух видов символов. Контур прямоугольника должен состоять из одного символа, а в "заливка" - из другого.

Доказать гипотезу Сиракуз на диапазоне чисел. Гипотеза Сиракуз утверждает, что любое натуральное число сводится к единице в результате повторения следующих действий над самим числом и результатами этих действий.

- Если число четное следует разделить его на 2.
- Если нечетное, то умножить его на 3, прибавить 1 и разделить на 2.

7. Посчитать, сколько раз встречается определенная цифра в введенной последовательности чисел. Количество вводимых чисел и цифра, которую необходимо посчитать, задаются вводом с клавиатуры.
8. Вводятся десять натуральных чисел больше 2. Посчитать, сколько среди них простых чисел.
9. Вывести на экран таблицу умножения (от 1 до 9).

Лабораторная работа 7. Массивы.

Цель работы: познакомиться с организацией массивов в языке Си, изучить принципы работы с массивами, освоить работу с массивами через указатели, научиться грамотно выделять и освобождать память в процессе работы программы.

Общие сведения:

При решении задач с большим количеством данных одинакового типа использование переменных с различными именами, не упорядоченных по адресам памяти, затрудняет программирование. В подобных случаях в языке Си используют объекты, называемые массивами.

Массив — это непрерывный участок памяти, содержащий последовательность объектов одинакового типа, обозначаемый одним именем.

Массив характеризуется следующими основными понятиями:

Элемент массива (значение элемента массива) – значение, хранящееся в определенной ячейке памяти, расположенной в пределах массива, а также адрес этой ячейки памяти.

Каждый элемент массива характеризуется тремя величинами:

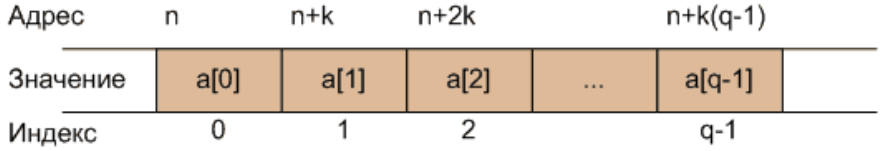
- адресом элемента — адресом начальной ячейки памяти, в которой расположен этот элемент;
- индексом элемента (порядковым номером элемента в массиве);
- значением элемента.

Адрес массива – адрес начального элемента массива.

Имя массива – идентификатор, используемый для обращения к элементам массива.

Размер массива – количество элементов массива
Размер элемента – количество байт, занимаемых одним элементом массива.

Графически расположение массива в памяти компьютера можно представить в виде непрерывной ленты адресов.



Представленный на рисунке массив содержит q элементов с индексами от 0 до q-1. Каждый элемент занимает в памяти компьютера k байт, причем расположение элементов в памяти

последовательное.
Адреса i-го элемента массива имеет значение

n+k·i

Адрес массива представляет собой адрес начального (нулевого) элемента массива. Для обращения к элементам массива используется порядковый номер (индекс) элемента, начальное значение которого равно 0. Так, если массив содержит q элементов, то индексы элементов массива меняются в пределах от 0 до q-1.
Длина массива – количество байт, отводимое в памяти для хранения всех элементов массива.

ДлинаМассива = РазмерЭлемента * КоличествоЭлементов

Для определения размера элемента массива может использоваться функция

int sizeof(тип);

Задание: Составить программы на языке Си по заданным условиям.

Задача1. Обращение к элементам массива

```
#include <stdio.h>
int main()
{
 int a[] = { 5, 4, 3, 2, 1 }; // массив a содержит 5 элементов
 printf("%d %d %d %d %d\n", a[0], a[1], a[2], a[3], a[4]);
 getchar();
 return 0;
}
```

Результатом программы будет

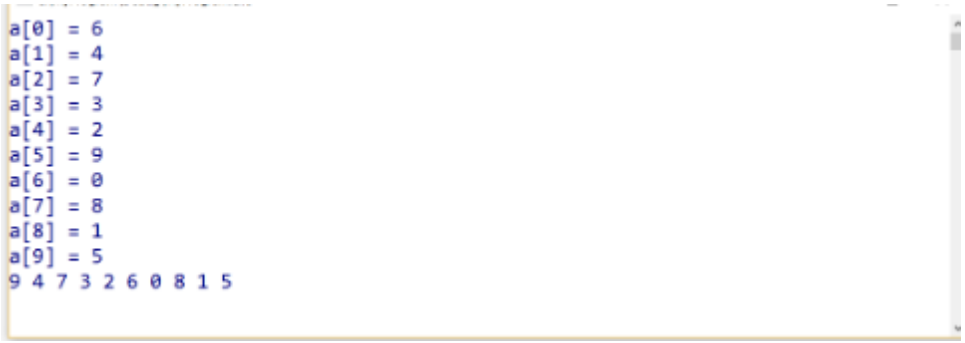
5 4 3 2 1

Задача 2. Дан массив из 10 элементов. Поменять местами наибольший и начальный элементы массива. Для операций поиска максимального элемента и обмена использовать функцию.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
// Функция обмена
void change(int *x, int n)
{
 // x - указатель на массив (адрес массива)
 // n - размер массива
 int i;
 int max, index;
 max = x[0];
 index = 0;
 // Поиск максимального элемента
```

```
  for (i = 1; i<n; i++)
  {
    if (x[i]>max)
    {
      max = x[i];
      index = i;
    }
  }
  // Обмен
  x[index] = x[0];
  x[0] = max;
}
// Главная функция
int main()
{
  int a[10];
  int i;
  for (i = 0; i<10; i++)
  {
    printf("a[%d] = ", i);
    scanf("%d", &a[i]);
  }
  change(a, 10);  // вызов функции обмена
        // Вывод элементов массива
  for (i = 0; i<10; i++)
    printf("%d ", a[i]);
  getchar();
  getchar();
  return 0;
}
```

Результатомвыполненияпрограммыбудет



Задача 3. Дан случайный одномерный массив из 10 целых чисел. Задать с помощью генератора случайных чисел из промежутка от -20 до +20, найти среднеарифметическое элементов массива.

```
#include <stdio.h>
```

```
#include <time.h>
#include <stdlib.h>
int main()
{
constint
   n = -20,
   m = 20,
   k = 10;
int a[k-1];
inti,sum;
srand (time(NULL));
printf("Получившейся массив случайных чисел \n");
for (int i = 1; i<=k; i++){
a[i]=rand()%((m-n)+n);
printf("a[i]=%d\n" , a[i]);
   }
sum = 0;
for (int i = 1; i <=k; i++){
sum = sum + a[i];
   }
printf("Среднееарифметическоечиселравно = %4.2d" , sum/k);
return 0;
}
```

Задача 4.Сложениедвухматриц

```
#include<stdio.h>
intmain()
{
floatA[3][2], B[3][2], C[3][2];
intn,m;
floatmax=0;
floatprum;
printf("ВведитезначенияматрицыA:\n");
for (n = 0; n< 3; n++) for (m = 0; m< 2; m++) {
 printf("A[%d][%d] = ", n, m);
 scanf("%f", &A[n][m]);
 if ((A[n][m] < 0) || (A[n][m] > 9.9)) { printf("НеверноеA, повторите\n"); m--;
}
}
printf("ВведитезначенияматрицыB:\n");
for (n = 0; n< 3; n++) for (m = 0; m< 2; m++) {
 printf("B[%d][%d] = ", n, m);
 scanf("%f", &B[n][m]);
 if ((B[n][m] < 0) || (B[n][m] > 9.9)) { printf("НеверноеB, повторите\n"); m--;
```

```
}
}
printf("Результатсложения:\n");
for (n = 0; n< 3; n++) for (m = 0; m< 2; m++) {
 C[n][m] = A[n][m] + B[n][m];
 printf("C[%d][%d] = %f\n", n, m, C[n][m]);
}
return 0;
}
```

Задания для самостоятельного выполнения:

1. Дан массив A(15). Найти наименьший из его положительных элементов.
2. Подсчитать и вывести на экран количество элементов массива, равных заданному значению. Заданное значение вводится с клавиатуры.
3. Составить и вывести на экран новый массив со значениями элементов исходного массива, которые не равны заданному значению. Заданное значение вводится с клавиатуры.
4. Поменять местами максимальный и минимальный элементы массива. Вывести измененный массив на экран.
5. Посчитать и вывести на экран количество отрицательных, положительных и нулевых элементов массива.
6. Дана матрица 4x4. Написать программу подсчета элементов больших нуля в заданной матрице.
7. Дана матрица 5x5. Все нулевые элементы матрицы заменить на –1.
8. Дана матрица 4x4. Найти номер максимального элемента в i-ой строке.
9. Дана матрица 4x4. Найти номер максимального элемента в i-ом столбце.
10. Дана матрица 4x4. Найти количество нулевых элементов, стоящих: выше главной диагонали.

Лабораторная работа 8. Указатели.

Цель работы:познакомиться с адресацией памяти, научиться правильно использовать указатели различных типов.

Общие сведения:

Указатель — переменная, содержащая адрес объекта. Указатель не несет информации о содержимом объекта, а содержит сведения о том, где размещен объект.

Указатели широко используются в программировании на языке Си. Указатели часто используются при работе с массивами.

Память компьютера можно представить в виде последовательности пронумерованных однобайтовых ячеек, с которыми можно работать по отдельности или блоками.

Каждая переменная в памяти имеет свой адрес — номер первой ячейки, где она расположена, а также свое значение. Указатель — это тоже переменная, которая размещается в памяти. Она тоже имеет адрес, а ее значение является адресом некоторой другой переменной. Переменная, объявленная как указатель, занимает 4 байта в оперативной памяти (в случае 32-битной версии компилятора).

Указатель, как и любая переменная, должен быть объявлен. Общая форма объявления указателя

Например

тип *ИмяОбъекта;

Тип указателя— это тип переменной, адрес которой он содержит.

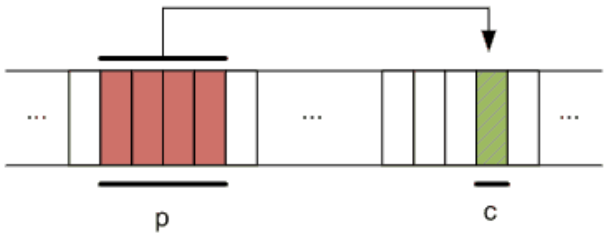
Для работы с указателями в Си определены две операции:

операция * (звездочка) — позволяет получить значение объекта по его адресу — определяет значение переменной, которое содержится по адресу, содержащемуся в указателе;

операция & (амперсанд) — позволяет определить адрес переменной.

Например,

```
char c;   // переменная
char *p; // указатель
p = &c;  // p = адрес c
```



Задание: Составить программы на языке Си по заданным условиям.

Задача 1. Использование указателей в Си

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
 int a, *b;
 system("chcp 1251");
 system("cls");
 a = 134;
 b = &a;
 // %x = вывод числа в шестнадцатеричной форме
 printf("\n Значение переменной a равно %d = %x шестн.", a,a);
 printf("\n Адрес переменной a равен %x шестн.", &a);
 printf("\n Данные по адресу указателя b равны %d = %x шестн.", *b,*b);
 printf("\n Значение указателя b равно %x шестн.", b);
 printf("\n Адрес расположения указателя b равен %x шестн.", &b);
 getchar();
 return 0;
}
```

Результатомвыполненияпрограммыбудет

```
Значение переменной a равно 134 = 86 шестн.
Адрес переменной a равен 93f7b8 шестн.
Данные по адресу указателя b равны 134 = 86 шестн.
Значение указателя b равно 93f7b8 шестн.
Адрес расположения указателя b равен 93f7ac шестн.
```

Переменные разных типов занимают разное количество байт. Например, целые числа, типа int занимают 4 байта, а символы типа charзанимают только один байт.

Задача 2. Можно написать программу для проверки сколько байт занимают те или иные переменные.

#include <stdio .h > #include <stdlib .h

```
// Запуск программы н а ч и н а е т с я с main
int   main (){
int   a;          //          целые  числа
float    fa ;     //          дробные      числа
char     z ;      //          символы
longlongal;//               длинные      целые  числа
```

```
doubledf;       //          длинные дробные числа
int  lin ;      //          для размера числа
// Для определения размера воспользуемся sizeof :
lin  = s i z e o f (a );
printf (" размерint:  %d\n" ,  lin );
lin  = s i z e o f ( long      long);
printf (" размер  long  long :  %d\n" ,                  lin );
}
```

Результат работы программы:

```
>./ check_ lin
>      размер  int :  4
>      размер  longlong :8
```

Задания для самостоятельного выполнения:

1. Определите переменную x как массив указателей на функцию,имеющую два параметра типа int и возвращающую результат типа указатель на
2. Определите переменную y как указатель на массив указателей нафункцию без параметров, возвращающую результат типа указатель на функциюс одним параметром типа int и результатом типа float.
3. Описать функцию, определяющую упорядочены ли строго по возрастанию элементы целочисленного массива из n элементов.
4. Описать функцию, определяющую индекс первого элемента целочисленного массива из n элементов, значение которого равно заданному числух. Если такого элемента в массиве нет, то считать номер равным –1.
5. Описать функцию, вычисляющую значение x0 + x0∗x1 + x0∗x1∗x2 +…+ x0∗x1∗x2 ∗… ∗xm, где xi - элементы вещественного массива x из n элементов,m - индекс первого отрицательного элемента этого массива либо число n-1, еслитакого элемента в массиве нет.
6. Описать функцию, вычисляющую значение max(x0 + xn-1, x1 + xn-2,x2 + xn-3,…, x(n-1)/2 + xn/2), где xi - элементы вещественного массива x из n элемен-
7. Описать функцию, вычисляющую значение min(x0 ∗ x1, x1 ∗ x2 ,x2 ∗ x3,…, xn-3 ∗ xn-2, xn-2 ∗ xn-1), где xi - элементы вещественного массива x из nэлементов.
8. Описать функцию, вычисляющую значение x0∗y0+x1∗y1+ …+xk∗yk, где xi – отрицательные элементы вещественного массива a из n элементов,взятые в порядке их следования; yi – положительные элементы этого массива,взятые в обратном порядке; k = min(p,q), где p – количество положительных элементов массива a, q – количество отрицательных элементов этого массива.

Лабораторная работа 9. Работа со строками

Цель работы: изучить способы хранения строковой информации в программах на языке Си, функции обработки строковой информации.

Общие сведения:

В программе строки могут определяться следующим образом:

- как строковые константы;
- как массивы символов;
- через указатель на символьный тип;
- как массивы строк.

Кроме того, должно быть предусмотрено выделение памяти для хранения строки.

Любая последовательность символов, заключенная в двойные кавычки «», рассматривается как ***строковая константа.***

Для корректного вывода любая строка должна заканчиваться нуль-символом '\0', целочисленное значение которого равно 0. При объявлении строковой константы нуль-символ добавляется к ней автоматически. Так, последовательность символов, представляющая собой строковую константу, будет размещена в оперативной памяти компьютера, включая нулевой байт.

Под хранение строки выделяются последовательно идущие ячейки оперативной памяти. Таким образом, строка представляет собой массив символов. Для хранения кода каждого символа строки отводится 1 байт. Для помещения в строковую константу некоторых служебных символов используются символьные комбинации. Так, если необходимо включить в строку символ двойной кавычки, ему должен предшествовать символ «обратный слеш»: ‘\»‘.

Строковые константы размещаются в статической памяти. Начальный адрес последовательности символов в двойных кавычках трактуется как адрес строки. Строковые константы часто используются для осуществления диалога с пользователем в таких функциях, как printf().

При определении ***массива символов*** необходимо сообщить компилятору требуемый размер памяти.

```
char m[82];
```

Компилятор также может самостоятельно определить размер массива символов, если инициализация массива задана при объявлении строковой константой:

```
char m2[]="Горные вершины спят во тьме ночной.";
char m3[]={'Т','и','х','и','е',' ','д','о','л','и','н','ы',' ','п','о','л','н','ы',' ','с','в','е','ж','е','й',' ','м','г','л','о','й','\0'};
```

Функции работы со строками

Давайте вначале посмотрим, что нам может дать библиотека stdio.h, которая содержит парочку функций работы со строками:

1. intgetchar()

возвращает значение символа, введенного вами с клавиатуры. А вот и вывод этого числа:

```
printf("%d", getchar());
```

2. char *gets (char *s)

функция просит вводить пользователя строку, которую она помещает в массив s, пока пользователь не нажмет 'Enter':

```
charstr[7] = "";
gets(str) ;
```

3. intputchar (int c)

печатает символ, который имеет код 'c':

```
putchar(97);//напечатает символ a
```

4. intputs (char *s)-----печатает строку s и переводит курсор на новую строку:

```
charstr[7] = "sergey";
puts(str);
```

5. intsprintf (char *s, char *format, ...)

Выполняет тоже, что и функция printf, за тем исключением, что записывает данные в массив s:

```
charstr[37] = "";
sprintf (str,"chislo:%d, month: %s",10,"desember");
//Вмассивебудет: chislo:10, month: desember
```

6. intsscanf (char *s, char *format, ...)

происходит ввод значений не с клавиатуры, а из массива s:

```
charstr[37] = "sergey", s[100]="";
sscanf (str,"%s", s);
```

Основные функции стандартной библиотеки string.h

Функция	**Описание**
char *strcat(char *s1, char *s2)	присоединяет s2 к s1, возвращает s1
char *strncat(char *s1, char *s2, int n)	присоединяет не более n символов s2 к s1, завершает строку символом '\0', возвращает s1
char *strcpy(char *s1, char *s2)	копирует строку s2 в строку s1, включая '\0', возвращает s1
char *strncpy(char *s1, char *s2, int n)	копирует не более n символов строки

Функция	**Описание**
	s2 в строку s1, возвращает s1;
int strcmp(char *s1, char *s2)	сравнивает s1 и s2, возвращает значение 0, если строки эквивалентны
int strncmp(char *s1, char *s2, int n)	сравнивает не более n символов строк s1 и s2, возвращает значение 0, если начальные n символов строк эквивалентны
int strlen(char *s)	возвращает количество символов в строке s
char *strset(char *s, char c)	заполняет строку s символами, код которых равен значению c, возвращает указатель на строку s
char *strnset(char *s, char c, int n)	заменяет первые n символов строки s символами, код которых равен c, возвращает указатель на строку s

Задание: Составить программы на языке Си по заданным условиям.

Задача. Посчитать количество введенных символов во введенной строке.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main() {
  char s[80], sym;
  int count, i;
  system("chcp 1251");
  system("cls");
  printf("Введите строку : ");
  gets_s(s);
  printf("Введите символ : ");
  sym = getchar();
  count = 0;
  for (i = 0; s[i] != '\0'; i++)
  {
    if (s[i] == sym)
      count++;
  }
  printf("В строке\n");
  puts(s);      // Выводстроки
```

```
  printf("символ ");
  putchar(sym); // Выводсимвола
  printf(" встречается %d раз", count);
  getchar(); getchar();
  return 0;
}
```

Задания для самостоятельного выполнения:

1. Дан текст. А) Подсчитать количество слов в данной строке. Б) Подсчитать количество букв а в последнем слове данной строки. В) Найти количество слов, начинающихся сбуква а. г) Найти количество слов, у которой первый и последний символы совпадают между собой. Д) Найти длину самого короткого слова.
2. Дан текст. Найти его перевертыш.
3. Введите текст с клавиатуры. Замените буквы на цифры в соответствии с их расположение в алфавите.
4. Написать программу для подсчета суммы мест, на которых в словах текста стоит заданная буква.
5. Составить таблицу слов данного текста, начинающихся с буквы “А”, с указанием числа повторений каждого слова.
6. Составить программу циклической перестановки букв в словах текста так, что i-я буква слова становится i+1-ой, а последняя - первой.
7. В каждом слове текста замените "а" на букву "е", если "а" стоит на четном месте, и заменить букву "б" на сочетание "ак", если "б" стоит на нечетном месте.
8. Вводится строка. Удалить из строки пару слов, которые имеют одинаковую длину.
9. В заданном тексте поменять местами слово, имеющее максимальное количество символов, и слово, имеющее минимальное количество символов.
10. Определить количество слов, содержащий заданный символ k раз. Заданный символ и число k вводятся дополнительно.

Лабораторная работа 10. Функции в языке Си.

Цель работы:изучить основные принципы структурного подхода к решению задач на ЭВМ, ознакомиться с синтаксисом определения и вызова функций с различными параметрами на языке Си, получить навыки программирования на языке Си с использованием функций.

Общие сведения:

Функция — это самостоятельная единица программы, которая спроектирована для реализации конкретной подзадачи. Функция является подпрограммой, которая может содержаться в основной программе, а может быть создана отдельно (в библиотеке). Каждая функция выполняет в программе определенные действия.

Сигнатура функции определяет правила использования функции. Обычно сигнатура представляет собой описание функции, включающее имя функции, перечень формальных параметров с их типами и тип возвращаемого значения.

Семантика функции определяет способ реализации функции. Обычно представляет собой тело функции.

Определение функции

Каждая функция в языке Си должна быть определена, то есть должны быть указаны:

- тип возвращаемого значения;
- имя функции;
- информация о формальных аргументах;
- тело функции.

Пример: Посчитать сумму двух чисел.

```
#define _CRT_SECURE_NO_WARNINGS // для возможности
использования scanf
#include <stdio.h>
// Функция вычисления суммы двух чисел
int sum(int x, int y) // в функцию передаются два целых числа
{
  int k = x + y;  // вычисляем сумму чисел и сохраняем в k
  return k;       // возвращаем значение k
}
int main()
{
  int a, r;      // описание двух целых переменных
  printf("a= ");
  scanf("%d", &a); // вводим a
  r = sum(a, 5);    // вызов функции: x=a, y=5
```

```
  printf("%d + 5 = %d", a, r); // вывод: a + 5 = r
  getchar(); getchar(); // мы использовали scanf(),
  return 0;  // поэтому getchar() вызываем дважды
}
```

Результат выполнения

Задание: Составить программы на языке Си с использованием заданных функций

Задача 1.Имя функции - truncate . Она получает дробную переменную a (аргумент) , а результатом работы функции должно быть округленное по математическим правилам целое число.

```
#include <stdio .h > #include <stdlib .h >
int truncate ( float a) // интерфейсфункции

{
//      переменнаяfloat a уже описана как аргумент и является

//      внутренней( локальной ) переменной.

//      имя a - это локальное имя , видимое только внутри функции

//      все переменные с таким же именем , описанные в другом месте

//      функция trunc не видит

//      Во время передачи пара метра будет передаваться только значение переменной.
intrez ;// локальная переменнаярez
//инструкции
rez = a + 0.5;
//      вернули результат returnrez ;
```

```
};
//      Запуск      программы начинается  с  main
intmain (){
floatz ;//  дробноечисло
intresult ;//  целое  число  для  результата
scanf (" % f " ,&z );
//      вызов  функции

//      вычесленное  значение  ( возвращаемое)  присвоим  result

//      значение  переменной  z  будет  присвоен о  локальной
переменной a
result  =  truncate ( z );
printf (" result =% d \ n " , result );
}
```

Задача 2. Имя функции - add. Она получает два аргумента - целых числа, а р е з у л ь т а т о м работы функции должна быть сумма чисел

```
#include <stdio .h >
#include <stdlib.h>
int  add (  int a ,  int b)  //  интерфейсфункции
{
//      переменныеа  и  b  -  локальные .
//      Передаются  значения  этих  переменных

intrez ;  //  локальна я  п е р е м е н н а я  rez
//      инструкции
rez  =  a  +  b;
//      вернули результат returnrez ;

};
//      Запуск      программы начинаются с  main
intmain (){
int x ,  y ;//  дробное  число
intresult ;//  целое  число  для  результата
scanf (" % d % d " ,&x ,& y );
//      вызов  функции
//      вычисленное  значение  ( возвращаемое)  присвоим  result
//      значения  переменных  x  и  y  будет  присвоен о  локальным
//      переменным a  и  b  соответственно
result  =  add ( z );
printf (" result =% d \ n " , result );

}
```

Задания для самостоятельного выполнения:

1. Даны три символьные матрицы.
а) ту матрицу, где есть хотя бы одна гласная – транспонировать;
b) в той матрице, на главной диагонали которой все цифры, найти наименьшую и удалить соответствующую строку.

2. Написать программу вычисления выражения $\left(\frac{A}{B+C}-\frac{C}{A-C}\right)*\frac{E}{F}$ в виде правильной дроби, где A, B, C, E, F – целые числа. Воспользуйтесь функцией сложения дробей.

3. Описать функцию Stepen (x,n) от вещественного x и целого n, вычисляющую (посредством умножения) величину x^n, и использовать её для вычисления $b=2.7^k+(a+1)^5$.

4. Даны длины a, b и c сторон некоторого треугольника. Найти медианы треугольника, сторонами которого являются медианы исходного треугольника. Длина медианы, проведенной к стороне a, равна

5. Даны координаты вершин двух треугольников. Определить, какой из них имеет большую площадь.

6. Даны координаты вершин треугольника и координаты некоторой точки внутри него. Найти расстояние от данной точки до ближайшей стороны треугольника. (При определении расстояний учесть, что площадь треугольника вычисляется и через три его стороны, и через основание и высоту.).

7. Запишите функцию, вычисляющую sh(x), и для заданного значения переменной x вычислите следующие выражение: sh(x) = tg(x+1) – tg^2(2+sh(x–1))

8. Определите наибольший общий делитель для трех чисел, написав функцию для определения наибольшего делителя NOD(x,y), используя алгоритм Евклида.

9. Вычислите выражение z(x) = (sign(x)+sign(y))*sign(x+y). При решении задачи определите и используйте функцию sign:

$$sign(x)=\begin{cases}-1, & x<0,\\ 0, & x=0,\\ 1, & x>0.\end{cases}$$

10. Напишите программу сортировки одномерного массива целых чисел в порядке возрастания.

Лабораторная работа 11. Работа с файлами.

Цель работы:изучение принципов работы с файлами через механизм потокового ввода-вывода на языке Си, приобретение практических навыков работы с файлами в Си.

Общие сведения:

Файл – именованная область внешней памяти, выделенная для хранения массива данных. Данные, содержащиеся в файлах, имеют самый разнообразный характер: программы на алгоритмическом или машинном языке; исходные данные для работы программ или результаты выполнения программ; произвольные тексты; графические изображения и т. п.

Каталог (***папка***, ***директория***) – именованная совокупность байтов на носителе информации, содержащая название подкаталогов и файлов, используется в файловой системе для упрощения организации файлов.

Файловой системой называется функциональная часть операционной системы, обеспечивающая выполнение операций над файлами. Примерами файловых систем являются FAT (FAT – FileAllocationTable, таблица размещения файлов), NTFS, UDF (используется на компакт-дисках).

Для программиста открытый файл представляется как последовательность считываемых или записываемых данных. При открытии файла с ним связывается ***поток ввода-вывода***. Выводимая информация записывается в поток, вводимая информация считывается из потока.

Когда поток открывается для ввода-вывода, он связывается со стандартной структурой типа FILE, которая определена в stdio.h. Структура FILE содержит необходимую информацию о файле.

Открытие файла осуществляется с помощью функции fopen(), которая возвращает указатель на структуру типа FILE, который можно использовать для последующих операций с файлом.

```
FILE *fopen(name, type);
```

name – имя открываемого файла (включая путь),
type — указатель на строку символов, определяющих способ доступа к файлу:

- "r" — открыть файл для чтения (файл должен существовать);
- "w" — открыть пустой файл для записи; если файл существует, то его содержимое теряется;
- "a" — открыть файл для записи в конец (для добавления); файл создается, если он не существует;

- "r+" — открыть файл для чтения и записи (файл должен существовать);
- "w+" — открыть пустой файл для чтения и записи; если файл существует, то его содержимое теряется;
- "a+" — открыть файл для чтения и дополнения, если файл не существует, то он создаётся.

Возвращаемое значение — указатель на открытый поток. Если обнаружена ошибка, то возвращается значение NULL.

Функция fclose() закрывает поток или потоки, связанные с открытыми при помощи функции fopen()файлами. Закрываемый поток определяется аргументом функции fclose().

Задание: Составить программы на языке Си по заданным условиям.

Задача 1.Возвращаемое значение: значение 0, если поток успешно закрыт; константа EOF, если произошла ошибка.

```
#include <stdio.h>
int main() {
  FILE *fp;
  char name[] = "my.txt";
  if ((fp = fopen(name, "r")) == NULL)
  {
    printf("Не удалось открыть файл");
    getchar();
    return 0;
  }
  // открыть файл удалось
  ...      // требуемые действия над данными
  fclose(fp);
  getchar();
  return 0;
}
```

Задача 2. Ввести число и сохранить его в файле s1.txt. Считать число из файла s1.txt, увеличить его на 3 и сохранить в файле s2.txt.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  FILE *S1, *S2;
  int x, y;
  system("chcp 1251");
  system("cls");
  printf("Введите число : ");
```

```
 scanf("%d", &x);
 S1 = fopen("S1.txt", "w");
 fprintf(S1, "%d", x);
 fclose(S1);
 S1 = fopen("S1.txt", "r");
 S2 = fopen("S2.txt", "w");
 fscanf(S1, "%d", &y);
 y += 3;
 fclose(S1);
 fprintf(S2, "%d\n", y);
 fclose(S2);
 return 0;
}
```

Задача 3. В данном файле символы каждой строки упорядочить по алфавиту.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
   FILE* txt;
charstrtxt[80][80], d[1];
int i, j, k, count = 0, len, max = 0;
if (!(txt = fopen("test.txt", "r"))){
printf("Error opening file!");
return 0;
   }
while (fgets(strtxt[count], 80, txt))
count++;
fclose(txt);
txt = fopen("test.txt", "w");
for (k = 0; k < count; k++) {
len = strlen(strtxt[k]) - 1;
for (i = 0; i <len; i++){
for (j = 0; j <len-i; j++)
if ((int)strtxt[k][j] > (int)strtxt[k][max])
max = j;
d[0] = strtxt[k][max];
```

```
strtxt[k][max] = strtxt[k][len-i-1];
strtxt[k][len-i-1] = d[0];
max = 0;
      }
fputs(strtxt[k], txt);
   }
printf("OK!");
fclose(txt);
return 0;
}
```

Задания для самостоятельного выполнения:

1. Даны текстовые файлы f1 и f2. Переписать с сохранением порядка следования компоненты файла f1 в файл f2, а компоненты файла f2 в файл f1. Использовать вспомогательный файл h.
2. Дан текстовый файл f. Записать в файл g компоненты файла f в обратном порядке.
3. Даны тестовые файлы f и g. Записать файл h сначала компоненты файла f, затем - компоненты файла g с сохранением порядка.
4. Дан файл f, компоненты которого являются целыми числами. Получить в файле g все компоненты файла f: а) являющимися чётными числами; б) делящиеся на 3 и не делящиеся на 7; в) являющимися точными квадратами.
5. Дан символьный файл f. Группы символов, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами. Удалить из файла все однобуквенные слова и лишние пробелы. Результат записать в файл g.
6. Сведения об автомобиле состоят из его марки, номера и фамилии владельца. Дан файл f, содержащий сведения о нескольких автомобилях. Найти: а) фамилии владельцев и номера автомобилей данной марки; б) количество автомобилей каждой марки. Найденные данные записать в файл g.
7. Дан файл f, содержащий различные даты. Каждая дата - это число, месяц и год. Найти: а) год с наименьшим номером; б) все весенние даты; в) самую позднюю дату. Найденные данные записать в файл g.
8. Дан файл f, содержащий сведения о книгах. Сведения о каждой из книг - это фамилия автора, название и год издания. 1) Найти названия книг данного автора, изданных с 1960 г. 2) Определить, имеется ли книга с названием "Информатика". Если да, то сообщить фамилию автора и год издания. Если таких книг несколько, то сообщить имеющиеся сведения обо всех книгах.
9. Дан файл f, содержащий сведения о кубиках: размер каждого кубика (длина ребра в сантиметрах), его цвет (красный, зеленый, желтый или

синий) и материал (деревянный, металлический, картонный). Найти: а) количество кубиков каждого из перечисленных цветов и их суммарный объем; б) количество деревянных кубиков с ребром 3 см и количество металлических кубиков с ребром, большим 5 см.

10.Дан файл f, содержащий сведения о веществах: указывается название вещества, его удельный вес и проводимость (проводник, полупроводник, изолятор). 1) Найти удельные веса и названия всех полупроводников. 2) Выбрать данные о проводниках и упорядочить их по убыванию удельных весов.

Рекомендуемая литература:

№ п/п	Автор	Название основной и дополнительной учебной литературы, необходимой для освоения дисциплины	Выходные данные	Количество экземпляров в библиотеке ДГУНХ
I. Основная учебная литература				
1.	Царев Р. Ю. http://biblioclub.ru	Программирование на языке Си: учебное пособие	М.:Национальный Открытый Университет «ИНТУИТ»,2016. - 179с.	15000 в соответствии с договором № 149-09/2018 на оказание услуг по предоставлению доступа к электронным изданиям от 1.10.2018 г.
2.	Давыдова Н. А., Боровская Е. В. http://biblioclub.ru/	Программирование	М.:БИНОМ. Лаборатория знаний,2015. -241с.	15000 в соответствии с договором № 149-09/2018 на оказание услуг по предоставлению доступа к электронным изданиям от 1.10.2018 г.
3.	Котов О. М. http://biblioclub.ru	Язык C#: краткое описание и введение в технологии программирования: учебное пособие	Екатеринбург: Издательство Уральского университета, 2014	15000 в соответствии с договором № 149-09/2018 на оказание услуг по предоставлению доступа к электронным

				изданиям от 1.10.2018 г.
II. Дополнительная учебная литература				
А) Дополнительная учебная литература				
1.	Павловская Т.А. http://biblioclub.ru	Программирование на языке С++	Москва, ИНТУИТ, 2010 г.	15000 в соответствии с договором № 149-09/2018 на оказание услуг по предоставлению доступа к электронным изданиям от 1.10.2018 г.
2.	Подбельский В.В. http://biblioclub.ru	Язык С#. Базовый курс: учебное пособие	Финансы и статистика 2013 г.	15000 в соответствии с договором № 149-09/2018 на оказание услуг по предоставлению доступа к электронным изданиям от 1.10.2018 г.
3.	Березин Б. И., Березин С. Б http://biblioclub.ru	Начальный курс С и С++	учебное пособие [Электронный ресурс] / М.:Диалог-МИФИ,2012. -280с.	15000 в соответствии с договором № 149-09/2018 на оказание услуг по предоставлению доступа к электронным изданиям от 1.10.2018 г.
4.	Пахомов Б.И.	С|С++ MS	Санкт-Петербург:	25 экз.

		Visual C++ 2010 для начинающих.	БХВ-Петербург. 2012.	
Б) Официальные издания: сборники законодательных актов, нормативно-правовых документов и кодексов РФ				
1.	ГОСТ Р ИСО/МЭК 12119-2000. Информационная технология. Пакеты программ. Требования к качеству и тестирование. 2005 г.www.standartgost.ru			
2.	Федеральный закон от 27 июля 2006 г. N 149-ФЗ "Об информации, информационных технологиях и о защите информации" (с изменениями и дополнениями).			
3.	ГОСТ Р ИСО/МЭК ТО 12182-2002. Информационная технология. Классификация программных средств. 2002 г.www.standartgost.ru			
4.	ГОСТ 28195-89. Оценка качества программных средств. Общие положения. 2001 г.www.standartgost.ru			
5.	ГОСТ 34.320-96.Информационные технологии. Система стандартов по базам данных. Концепции и терминология для концептуальной схемы и информационной базы. 2001 г.www.standartgost.ru			
6.	ГОСТ Р ИСО/МЭК 15910-2002. Информационная технология. Процесс создания документации пользователя программного средства. 2002 г.www.standartgost.ru			
В) Периодические издания				
1.	Журнал для пользователей персональных компьютеров «Мир ПК»			
2.	Открытые системы			
3.	Научный журнал «Прикладная дискретная математика»			
4.	Научный журнал «Информатика и ее применение»			
5.	Информатика и безопасность			
6.	Журнал о компьютерах и цифровой технике «ComputerBild»			
7.	Рецензируемый научный журнал «Информатика и система управления»			
8.	Рецензируемый научный журнал «Проблемы информационной безопасности»			
9.	Рецензируемый научный журнал «Прикладная информатика»			
Г) Справочно-библиографическая литература				
1.	Краткий энциклопедический словарь по информационной безопасности http://biblioclub.ru/			