Лабораторная работа №3. Проектирование программ линейной структуры

1 Цель и порядок работы

Цель работы – изучить структуру программы на языке С++, операторы присваивания, ввода и вывода данных, используемые при составлении программ линейной структуры.

Порядок выполнения работы:

- ознакомиться с описанием лабораторной работы;
- получить задание у преподавателя, согласно своему варианту;
- написать программу, отладить и решить ее на ЭВМ.

2 Краткая теория

Структура программы на языке С++

Выполнение всех программ, написанных на языке С++, начинается с функции, именуемой main. При запуске программы прежде всего выполняется первое выражение функции main.

Выражение – это строка кодов, представляющая собой отдельную инструкцию для компьютера. (Функция состоит из группы выражений, собранных вместе для решения определенной задачи.) Затем поочередно выполняются все остальные выражения – по одному за раз.

Структура программы выглядит следующим образом:

```
директивы_препроцессора
int main()
{
определения_объектов;
исполняемые операторы;
return 0;
}
```

```
директивы_препроцессора
void main()
{
определения_объектов;
исполняемые операторы;
return;
}
```

У функции есть имя (main), после которого в круглых скобках перечисляются аргументы или параметры функции (в данном случае у функции main аргументов нет). У функции может быть результат или возвращаемое значение. Если функция не возвращает никакого значения, то это обозначается ключевым словом void. В фигурных скобках записывается тело функции – действия, которые она выполняет. Оператор return 0; означает, что функция возвращает результат – целое число 0.

Вслед за заголовком функции main в фигурных скобках размещается тело функции, которое представляет последовательность определений, описаний и исполняемых операторов. Как правило, определения и описания размещаются до исполняемых операторов. Каждое определение, описание и оператор завершается «;».

Переменные используются для представления данных в программе. Например, если нужно запомнить имя пользователя, можно создать переменную Имя. Затем в любой момент, когда потребуется имя пользователя, можно просто сослаться на значение переменной Имя. В процессе выполнения программы значения переменных могут изменяться. Например, можно присвоить переменной Имя значение «Вася», а потом другим выражением присвоить этой же переменной значение «Степан». Но само по себе значение переменной никогда не

меняется – в любом случае вы должны написать какое-нибудь выражение, меняющее одно значение на другое.

Комментарии используются для описания того, что происходит в процессе выполнения программы. Вы можете добавлять их для расшифровки целей, с которыми пишутся те или иные фрагменты кодов, для фиксирования каких-то мыслей и идей, для описания решаемых задач. Добавляя комментарии, вы упрощаете чтение кодов вашей программы пользователями. Для вас комментарии также могут быть очень полезны. Если через некоторое время вы захотите внести изменения в уже набранные коды, вам, скорее всего, трудно будет вспомнить, для чего используется та или иная переменная и зачем нужно было создавать ту или иную функцию. В таких случаях, пожалуй, комментарии могут быть единственным средством, которое поможет вам вспомнить, что же именно вы хотели сделать, набирая эти коды. При преобразовании кодов C++ в машинные коды компилятор игнорирует строки, являющиеся комментариями, и просто пропускает их.

Понятие переменной

Программа оперирует информацией, представленной в виде различных объектов и величин. Переменная – это символическое обозначение величины в программе. Как ясно из названия, значение переменной (или величина, которую она обозначает) во время выполнения программы может изменяться.

С точки зрения архитектуры компьютера, переменная – это символическое обозначение ячейки оперативной памяти программы, в которой хранятся данные. Содержимое этой ячейки – это текущее значение переменной.

В языке C++ прежде чем использовать переменную, ее необходимо объявить. Объявить переменную с именем *x* можно так: *int x;*

В объявлении первым стоит название типа переменной *int* (целое число), а затем идентификатор *x* – имя переменной. У переменной *x* есть тип – в данном случае целое число. Тип переменной определяет, какие возможные значения эта переменная может принимать и какие операции можно выполнять над данной переменной. Тип переменной изменить нельзя, т.е. пока переменная *x* существует, она всегда будет целого типа.

Язык C++ строго типизированный язык. Любая величина, используемая в программе, принадлежит к какому-либо типу. При любом использовании переменных в программе проверяется, применимо ли выражение или операция к типу переменной. Довольно часто смысл выражения зависит от типа участвующих в нем переменных.

Например, если записать *x+y*, где *x* – переменная, то переменная *y* должна быть одного из числовых типов.

Соответствие типов проверяется во время компиляции программы. Если компилятор обнаруживает несоответствие типа переменной и ее использования, он выдаст ошибку (или предупреждение). Однако во время выполнения программы типы не проверяются. Такой подход, с одной стороны, позволяет обнаружить и исправить большое количество ошибок на стадии компиляции, а, с другой стороны, не замедляет выполнения программы.

Основные типы данных

В языке C++ существует несколько стандартных основных типов данных. Основные типы, наиболее непосредственно отвечающие средствам аппаратного обеспечения, такие:

char short int long float double

Первые четыре типа используются для представления целых, последние два – для представления чисел с плавающей точкой. Переменная типа char имеет размер, естественный для хранения символа на данной машине (обычно, байт), а переменная типа int имеет размер, соответствующий целой арифметике на данной машине (обычно, слово). Диапазон целых

чисел, которые могут быть представлены типом, зависит от его размера. В C++ размеры измеряются в единицах размера данных типа char, поэтому char по определению имеет размер единица.

Для определения данных целого типа используются различные ключевые слова, которые определяют диапазон значений и размер области памяти, выделяемой под переменные.

Тип	Размер памяти в байтах	Диапазон значений
char	1	от -128 до 127
int	4	от -2 147 483 648 до 2 147 483 647
short	2	от -32768 до 32767
long	4	от -2 147 483 648 до 2 147 483 647
unsigned char	1	от 0 до 255
unsigned int	4	от 0 до 4 294 967 295
unsigned short	2	от 0 до 65535
unsigned long	4	от 0 до 4 294 967 295

Для переменных, представляющих число с плавающей точкой используются следующие модификаторы-типа: float, double, long double.

Величина с модификатором-типа float занимает 4 байта. Из них 1 байт отводится для знака, 8 бит для избыточной экспоненты и 23 бита для мантиссы. Отметим, что старший бит мантиссы всегда равен 1, поэтому он не заполняется, в связи с этим диапазон значений переменной с плавающей точкой приблизительно равен от 3.14E-38 до 3.14E+38.

Величина типа double занимает 8 бит в памяти. Ее формат аналогичен формату float. Биты памяти распределяются следующим образом: 1 бит для знака, 11 бит для экспоненты и 52 бита для мантиссы. С учетом опущенного старшего бита мантиссы диапазон значений равен от 1.7E-308 до 1.7E+308.

Понятие константы

В программе можно явно записать величину – число, символ и т.п. Например, выражение x + 4 – сложить текущее значение переменной x и число 4. В зависимости от того, при каких условиях будет выполняться программа, значение переменной x может быть различным. Однако целое число четыре всегда останется прежним. Таким образом, явная запись значения в программе – это константа и в данном случае она задаётся своим изображением.

Гораздо чаще используются символические константы. Для определения символической константы используется ключевое слово ***const.*** Например, если записать

```
const int BITS_IN_WORD = 32;
```

то затем имя BITS_IN_WORD можно будет использовать вместо целого числа 32.

Операторы языка программирования

Оператором называется элементарная структурная единица программы. Оператор предназначен как для записи алгоритмических действий по преобразованию данных, так и для задания порядка выполнения других действий. Операторы выполняются в порядке их следования в программе. Операторы отделятся друг от друга точкой с запятой. Операторы делятся на:

- простые (не содержат в себе других операторов);
- составные (включают в себя один или несколько дополнительных операторов).

Присваивание

Переменной можно присвоить какое-либо значение с помощью операции присваивания. Присвоить – это значит установить текущее значение переменной. По-другому можно объяснить, что операция присваивания запоминает новое значение в ячейке памяти, которая обозначена переменной.

```
int x;          // объявить целую переменную x
int y;          // объявить целую переменную y
x = 0;          // присвоить x значение 0
y = x + 1;      // присвоить y значение x + 1,   т.е. 1
x = 1;          // присвоить x значение 1
y = x + 1;      // присвоить y значение x + 1, теперь уже 2
```

Ввод-вывод данных с использованием библиотеки потокового ввода вывода

Механизм для ввода-вывода в C++ называется потоком, так как информация вводится и выводится в виде потока байтов – символ за символом.

Библиотека потоков ввода-вывода (iostream.h) определяет три глобальных объекта: cout, cin и cerr.

Для использования возможностей библиотеки необходимо в начале программы указать директиву ***using namespace*** std;

cout называется стандартным выводом, ***cin*** – стандартным вводом, ***cerr*** – стандартным потоком сообщений об ошибках. cout и cerr выводят на терминал и принадлежат к классу ostream, cin имеет тип istream и вводит с терминала.

Вывод осуществляется с помощью операции <<, ввод с помощью операции >>. Выражение

```
cout << "Пример вывода: " << 34;
```

напечатает на терминале строку "Пример вывода", за которым будет выведено число 34. Выражение

```
int x;
cin >> x;
```

введет целое число с терминала в переменную x. (Разумеется, для того, чтобы ввод произошел, на терминале нужно напечатать какое-либо число и нажать клавишу возврат каретки.)

#include <iostream> подключает библиотеку потокового ввода-вывода. Файл заголовков определяет глобальный объект этого класса cout. Объект называется глобальным, поскольку доступ к нему возможен из любой части программы. Этот объект выполняет вывод на консоль. В функции main мы можем к нему обратиться и послать ему сообщение:

```
#include "stdafx.h"
#include <iostream>

using namespace std;
int main()
{
   cout << "Hello world!" << endl;
   return 0;
}
```

Операция сдвига << определена как "вывести". Таким образом, программа посылает объекту *cout* сообщения "вывести строку Hello world!" и "вывести перевод строки" (*endl* обозначает перевод на новую строку). В ответ на эти сообщения объект *cout* выведет строку "Hello world!" на консоль и переведет курсор на следующую строку.

Подключение заголовочного файла #include "stdafx.h" не является обязательным с точки зрения языка C++, однако среда разработки Visual Studio 2010 требует его

подключения для включения прекомпиляции заголовочных файлов. Данная возможность позволяет ускорить компиляцию и запуск программы.

Манипуляторы и форматирование ввода-вывода

Часто бывает необходимо вывести строку или число в определенном формате. Для этого используются так называемые манипуляторы.

Манипуляторы – это объекты особых типов, которые управляют тем, как обрабатывают последующие аргументы. Некоторые манипуляторы могут также выводить или вводить специальные символы. Манипуляторы позволяют задавать формат вывода чисел.

Таблица 1 – Манипуляторы потокового ввода-вывода

Манипулятор	Назначение
endl	при выводе перейти на новую строку;
ends	вывести нулевой байт (признак конца строки);
flush	вывести и очистить все промежуточные буферы;
dec	выводить числа в десятичной системе (по умолчанию);
oct	выводить числа в восьмеричной системе;
hex	выводить числа в шестнадцатеричной системе счисления;
setw (int n)	установить ширину поля вывода в n символов (n – целое число);
setfill(int n)	установить символ-заполнитель, которым выводимое значение будет дополняться до необходимой ширины;
setprecision(int n)	установить количество цифр после запятой при выводе вещественных чисел;
setbase(int n)	установить систему счисления для вывода чисел; n может принимать значения 0, 2, 8, 10, 16, причем 0 означает систему счисления по умолчанию, т.е. 10.

Использовать манипуляторы просто – их надо вывести в выходной поток. Выведем одно и то же число в разных системах счисления:

```
int  x=53;
cout <<"Десятичный вид:    " << dec << x << endl
<< "Восьмиричный вид:      " << oct << x << endl
 <<"Шестнадцатиричный вид:" << hex << x << endl;
```

Аналогично используются манипуляторы с параметрами. Вывод числа с разным количеством цифр после запятой:

```
#include "stdafx.h"
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{ const double   d1 = 1.23456789;
  const double   d2 = 12.3456789;
  const double   d3 = 123.456789;
```

```
  const double   d4 = 1234.56789;
  const double   d5 = 12345.6789;
  cout << endl << "setprecision(" << 3 << ")" << setprecision(3);
  cout << endl << "default display" << endl;
  cout << "d1 = " << d1 << endl;
  cout << "d2 = " << d2 << endl;
  cout << "d3 = " << d3 << endl;
  cout << "d4 = " << d4 << endl;
  cout << "d5 = " << d5 << endl;
  return 0;
}
В результате получим:
d1 = 1,23 d2 = 12,3 d3 = 123 d4 = 1.23e+003 d5 = 1.23e+004
```

Те же манипуляторы (за исключением *endl* и *ends*) могут использоваться и при вводе. В этом случае они описывают представление вводимых чисел. Кроме того, имеется манипулятор, работающий только при вводе, это ws. Данный манипулятор переключает вводимый поток в такой режим, при котором все пробелы (включая табуляцию, переводы строки, переводы каретки и переводы страницы) будут вводиться. По умолчанию эти символы воспринимаются как разделители между атрибутами ввода.

```
int x;
// ввести шестнадцатеричное число
cin >> hex >> x;
```

Ввод вывод с использованием стандартной библиотеки ввода-вывода stdio.h

Все возможности организации ввода-вывода СИ реализованы в библиотечных функциях стандартной библиотеки stdio.h.

Для организации вывода используется функция

printf(форматная_строка,список_аргументов);

Форматная строка ограничивается кавычками «”» и может включать произвольный текст, управляющие символы и спецификации преобразования данных.

Список аргументов может отсутствовать.

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

void main()
{
printf("\nhello!\n");
}
```

Препроцессорная директива *#include* <stdio.h> подключает стандартную библиотеку ввода-вывода. «\n» – перевод строки (управляющий символ).

При организации вывода данных на экран используются спецификации преобразования, которые имеют следующий обобщённый вид:

%флажки ширина_поля.точность модификатор спецификатор

Обязательными являются «%» и спецификатор.

Таблица 2 – Назначение флагов

Флаг	**Назначение**
-	Выравнивание результата по левому краю поля.
+	Результат всегда выводится с указанием знака «+» или «-».

Пробел	Если значение не отрицательное, то вместо плюса выводится пробел, для отрицательных значений выводится «-».
#	Аргументы могут быть преобразованы с использованием альтернативной формы

ширина_поля – целое положительное число, определяющее количество знакомест для вывода значения.

точность – целое положительное число, определяющее количество цифр после десятичной запятой для вывода значения с плавающей точкой.

Возможные модификаторы представлены в таблице 6.

Таблица 3 – Назначение модификаторов

Модификатор	Назначение
N	Для близкого указателя
F	Для дальнего указателя
h	Для значения short int
l	Для значения long
L	Для значения long double

Спецификаторы определяют тип выводимого значения и форму вывода.

Таблица 4 – Назначение спецификаторов

Спецификатор	**Тип аргумента**	**Назначение**
d	Целого типа	Для целых десятичных чисел (int)
i	Целого типа	Для целых десятичных чисел (int)
o	Целого типа	Для беззнаковых восьмеричных целых
u	Целого типа	Для беззнаковых десятичных целых
x	Целого типа	Для беззнаковых шестнадцатеричных целых (a,b,c,d,e,f)
X	Целого типа	Для беззнаковых шестнадцатеричных целых (A,B,C,D,E,F)
Спецификатор	Тип аргумента	Назначение
f	вещественный	Знаковое вещественное число в формате [+/-]ddd.dddd
e	вещественный	Знаковое вещественное число в формате [+/-]d.dddd или в экспоненциальной форме
g	вещественный	Знаковое вещественное число в формате или f, или e (в зависимости от выводимого значения)
E	вещественный	Такое же, как и e
G	вещественный	Такое же, как и g
s	строковый	ввод-вывод строковых данных
c	символьный	ввод-вывод символов

Например:
Printf(“|n summa=%f”,summa);
На экране будет выведено:
Summa=2102.3
После выполнения операторов:
float c=48.3, e=16.33;
int k=-83;
printf(“\nc=%f\tk=%d\te=%e”,c,k,e);
на экране будет выведено
c=48.299999 k=-83 e=1.63300e+01
Для тех же переменных:

printf(“\nc=%5.2f\tk=%5d\te=%8.2f\te=%11.4e”,c,k,e,e);

на экране будет выведено

c=48.30 k= -83 e=16.33 e= 1.6330e+01

В состав строки вывода могут входить управляющие последовательности:

‘*n*’ – перевод строки;

‘*t*’ – горизонтальная табуляция;

‘*r*’ – возврат каретки к началу строки;

‘\\’ – обратная косая черта \;

‘\’’ – апостроф ‘;

‘*0*’ – нулевой символ;

‘*a*’ – сигнал-звонок;

‘*b*’ – возврат на одну позицию;

‘*f*’ – перевод строки;

‘*v*’ – вертикальная табуляция;

‘*?*’ – знак вопроса.

Для организации ввода данных с клавиатуры используется функция

scanf(форматная_строка, список_аргументов);

Эта функция выполняет чтение значений вводимых с клавиатуры и присваивает их последовательно аргументам. Форматная строка представляет собой последовательность спецификаций, управляющих преобразованием вводимых значений.

*%*ширина_поля модификатор спецификатор*

‘’* в настоящее время не используется;

Ширина_поля – целое положительное число, позволяющее определить, какое количество байтов из входного потока соответствует вводимому значению.

модификатор и спецификатор – аналогичны функции *printf().*

Аргументами функции ввода могут быть адреса переменных, которым будут присвоены вводимые значения. Они задаются при помощи операции взятия адреса “**&имя_переменной**”

Например:

scanf(“%d%f%f”,&n,&z,&x);

При организации ввода-вывода данных используются также функции, описанные в стандартной библиотеке ввода-вывода(<*stdio.h*>*):*

puts(const char* Строка); Выводит на экран строку символов и переводит курсор в начало следующей строки экрана. В качестве параметра функции можно использовать строковую константу или строковую переменную.

char *gets(char* s);Вводит с клавиатуры строку символов. Вводимая строка может содержать пробелы.

int putch(int c); Выводит на экран символ.

int getch(void);Возвращает код символа нажатой клавиши. Если нажата служебная клавиша, то функция *getch* возвращает 0. В этом случае, для того, чтобы определить, какая служебная клавиша нажата, нужно обратиться к функции *getch* еще раз**.** Замечание Функция *getch* не выводит на экран символ, соответствующий нажатой клавише.

cputs(const char* Строка); Выводит на экран строку.

Математические функции

Для выполнения математических вычислений в стандартной математической библиотеке **<*math.h*>** описаны следующие функции:

int abs (int к) ; double fabs(double x); Возвращает целое (*abs*) или дробное (*fabs*) абсолютное значение аргумента, в качестве которого можно использовать выражение соответствующего типа.

double acos (double x);
double asin (double x);
double atan (double x);
long double acosl(long double x) ;
long double asinl(long double x);
long double atanl(long double x);

Возвращает выраженную в радианах величину угла, арккосинус, арксинус или арктангенс которого передан соответствующей функции в качестве аргумента. Аргумент функции должен находиться в диапазоне от -1 до 1.

double cos (double x);
double sin (double x);
double tan (double x);
long double cosl(long double x);
long double sinl(long double x);
long double tanl(long double x);

Возвращает синус, косинус или тангенс угла. Величина угла должна быть задана в радианах.

```
#include "stdafx.h"
#include <stdio.h>
#include <math.h>
#include <iostream>
using namespace std;

int main(void)
{
   double result;
   double x = 0.5;
   result = cos(x);
   printf("Косинус числа  %lf -  %lf\n", x, result);
   return 0;
}
```

double exp(double x); long double exp(long double lx); Возвращает значение, равное экспоненте аргумента (*e**, где *e* — основание натурального логарифма).

double pow (double x, double y); long double powl(long double (x), long double (y)); Возвращает значение, равное x^y.

```
#include "stdafx.h"
#include <stdio.h>
#include <math.h>
#include <iostream>
using namespace std;

int main(void)
{
   double result;
   double x = 4.0;
   result = exp(x);
   printf("'e'  в степени  %lf (e ^ %lf) = %lf\n",  x, x, result);
   return 0;
}
```

double sqrt(double κ);

Возвращает значение, равное квадратному корню из аргумента.

double log(double x);
double log10(double x);
long double logl(long double (x));
long double log10l(long double (x));

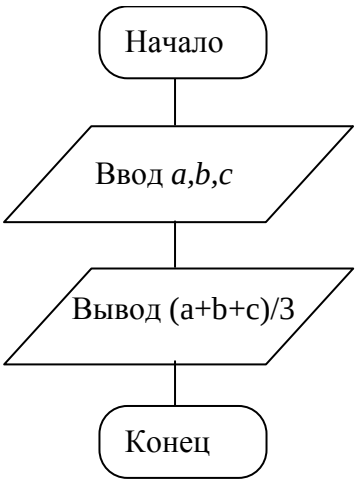
log, logl – возвращают значение натурального логарифма аргумента. *log10, log10l*– возвращают значение логарифма аргумента по основанию *10*.

В библиотеке <*stdlib.h*> описаны генераторы случайных чисел.

int rand(void); Возвращает случайное целое число в диапазоне от 0 до *RAND_MAX*. Перед первым обращением к функции *rand* необходимо инициализировать генератор случайных чисел. Для этого надо вызвать функцию *srand*. ***void srand(unsigned к);*** Инициализирует генератор случайных чисел. Обычно в качестве параметра функции используют переменную, значение которой предсказать заранее нельзя, например это может быть текущее время.

3 Примеры программ

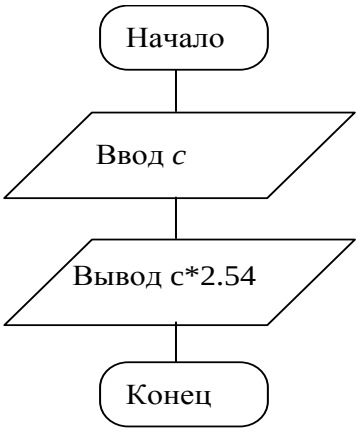
3.1 Программа нахождения среднего арифметического из двух целых чисел и одного вещественного числа:




```
#include "stdafx.h"
#include <iostream>

using namespace std;
void main()
{
  int a,b;
  float c;
  cout<<"Input 3 nambers"<<endl;
  cin>>a>>b>>c;
  cout<<"Rezult="<<(a+b+c)/3;
}
```

3.2 Программа перевода дюймов в сантиметры (1 дюйм = 2,54 см).

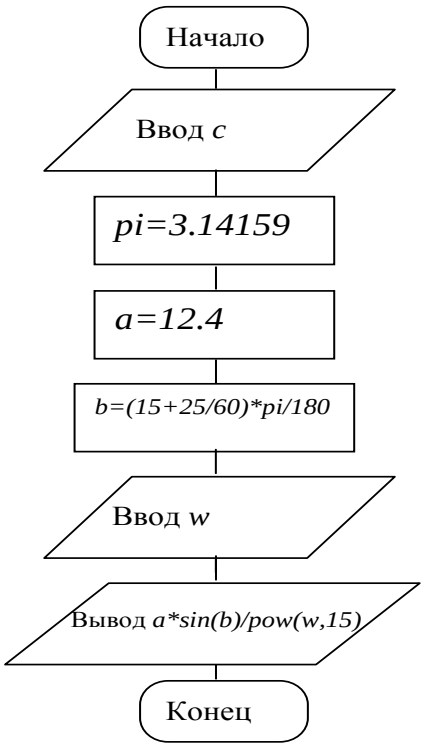



```
#include "stdafx.h"
#include <iostream>

using namespace std;
void main()
{
  float c;
  cout<<"Input  nambe"<<endl;
  cin>>c;
  cout<<"Rezult="<< c*2.54;
}
```

3.3 Программа вычисления значения выражения: $Y = \dfrac{a \cdot \sin(b)}{w^{15}}$

a = 12.4, b =15° 25', a w – вводится с клавиатуры. Для возведения в степень используется функция pow заголовочного файла math.h.




```
#include "stdafx.h"
#include <iostream>
#include <math.h>

using namespace std;

void main()
{
 const float pi=3.14159;
 const float a=12.5;
 const float b=(15+25/60)*pi/180;
 float w;
 cout<<"Input w"<<endl;
 cin>>w;
 cout<<"Rezult="<<a*sin(b)/pow(w,15);
}
```

3 Контрольные вопросы

1. Опишите структуру программы на языке С++.
2. Какие группы символов входят в алфавит языка С++.
3. Какие символы содержатся вы знаете.
4. Что такое управляющие последовательности, и каким образом они задаются?
5. Как задаются идентификаторы?
6. Перечислите ключевые слова языка С++.
7. Перечислите и опишите основные типы данных.
8. Как определить константу?
9. Опишите возможности ввода-вывода данных с помощью библиотеки потокового ввода вывода.
10. Опишите известные вам манипуляторы ввода-вывода.
11. Как производится ввод-вывод с использованием стандартной библиотеки ввода-вывода stdio.h.
12. Какие модификаторы и спецификаторы поддерживает функция printf.
13. Как осуществляется ввод при помощи стандартной библиотеки stdio.h.
14. Как подключить библиотеку с математическими функциями.
15. Какие стандартные математические функции содержит библиотека math.h.
16. Как получить случайное число.

4 Задание

1. Написать программу в соответствии с вариантом задания из пункта 5. Вариант определяется по последней цифре в номере студента в общем списке группы. И два на выбор задания из пункта 6.

0 вариант – 1, 11, 21
1 вариант – 2, 12, 22
2 вариант – 3, 13, 23
3 вариант – 4, 14, 24
4 вариант – 5, 15, 25
5 вариант – 6, 16, 26
6 вариант – 7, 17, 27
7 вариант – 8, 18, 28
8 вариант – 9, 19, 29
9 вариант – 10, 20, 30

2. Проверить работоспособность программы.
3. Отладить и протестировать программу.

5 Задания к лабораторной работе:

1) вычислить $y=\sin(x)*\cos(x)-3x^2$;
2) определить время падения камня на поверхность Земли с высоты h.
3) вычислить $y=|x-tg(x)|$;
4) вычислить площадь треугольника по стороне и высоте;
5) даны значения a и b, найти их среднее арифметическое, среднегеометрическое;
6) вычислить $y=tg(x)+5x^3-4x^2$;
7) вычислить площадь квадрата;
8) вычислить высоту треугольника, зная две стороны треугольника и угол между ними;
9) вычислить $y=|x-\cos(x)|$;
10) ввести сторону квадрата a. Вычислить радиус вписанной окружности;
11) Задается длина окружности. Найти площадь круга, ограниченного этой окружностью.
12) вычислить углы треугольника, зная его стороны;
13) вычислить площадь трапеции;
14) вычислить $y=\cos|x^3-x^2|$;
15) вычислить длину гипотенузы прямоугольного треугольника, зная длины двух катетов;
16) вычислить корень квадратный от $(x^5-x^4+|x^3|)$;
17) вычислить корень квадратный от $(\sin(x)+\cos(x))$;
18) вычислить объем цилиндра, зная радиус основания и высоту;
19) вычислить объем конуса;
20) вычислить сторону треугольника, зная две другие стороны и угол между ними;
21) вычислить площадь ромба, зная длину стороны и угол;
22) вычислить площадь треугольника, зная длины всех сторон и радиус описанной окружности;
23) вычислить $y= tg(x^3)+|x^2-x^5|$;
24) вычислить высоту равностороннего треугольника, зная длину всех сторон;
25) вычислить $y = x^2-\sin(x)+\cos(x)$;
26) вычислить $y = x^3-x^5+|x-\sin(x)|+x^2$;
27) вычислить длину отрезка, зная координаты его концов;

28) вычислить y = x/cos (x) + x2/sin (x);
29) вычислить среднее арифметическое четырех вводимых чисел;
30) вычислить среднее геометрическое пяти вводимых чисел.

6 Дополнительные задания

1) Найти корни квадратного уравнения Ax^2-Bx+C=0 для A,B и C, вводимых с клавиатуры.

2) Для A, вводимого с клавиатуры вычислить B=A^10 за четыре операции.

3) Вычислите значение выражения $y = 15x^2 + 8x - 9$.

4) Используя стандартные функции, вычислите значение выражения. $y = \cos\left(x^7 + \sqrt{x + x^4}\right) - \left|\sin\left(\frac{1+\sqrt[3]{x}}{x}\right)\right|$

5) Используя стандартные функции, вычислите значение выражения $y = \ln\left(x^5 + x - \sqrt[3]{x^7}\right) - \frac{1+x}{1-x}$.

6) Вычислить за минимальное количество операций следующее выражение $y = x^5$.

7) Вычислить за минимальное количество операций следующее выражение $y = x^6$.

8) Вычислить за минимальное количество операций следующее выражение $y = x^8$.

9) Вычислить за минимальное количество операций следующее выражение $y = x^{14}$.

10) Поменяйте местами значения переменных x и y с использованием дополнительной переменной.

11) Поменяйте местами значения переменных x и y без использования дополнительной переменной.

12) Используя стандартные функции, найдите значение выражения $y = |x - 2| + 3x^8$, при x=-2; x=1.

13) С помощью логической переменной выяснить является ли трехзначное число числом Армстронга (сумма его цифр, возведенных в степень равную количеству цифр в числе, равна самому числу, например: $153 = 1^3 + 5^3 + 3^3$).

14) Найдите сумму цифр четырехзначного числа.

15) Найдите первую и среднюю цифру пятизначного числа.

16) Припишите по 1 в начало и в конец записи трехзначного числа n.

17) Поменяйте местами первую и последнею цифру в записи четырехзначного числа.

18) Поменяйте порядок цифр в записи четырехзначного числа (Например: было 1234, стало 4321).

19) С помощью логической переменной определите: есть ли в записи пятизначного числа четные цифры.

20) С помощь логической переменной определить: есть ли в записи четырехзначного числа цифра n (цифра n вводиться с клавиатуры).

21) С помощью логической переменной определить: является ли шестизначное натуральное число палиндромом (т.е. десятичная запись которого читается одинаково слева направо и справа налево).