Урок 2. Линейные программы на C/C++

Цель работы: приобрести навыки в написании простейших линейных программ.

Структура программы. Общая структура программы на C/C++ следующая:

```
директивы_препроцессора
определение_функции_1
определение_функции_2
……
определение_функции_N
```

Среди функций обязательно присутствует главная функция с именем main. Простейшая программа содержит только главную функцию и имеет следующую структуру:

```
директивы_препроцессора
void  main( )
{
      определения_объектов;
      ислолняемые_операторы;
}
```

Рассмотрим все необходимые средства языка для составления линейных вычислительных программ. В качестве опорного примера рассмотрим программу для вычисления площади треугольника по формуле Герона.

Пример 1. Дано: a, b, c — стороны треугольника. Вычислить S — площадь треугольника по формуле Герона:

$$S = \sqrt{p(p-a)(p-b)(p-c)},$$

где p — полупериметр треугольника.

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

void  main( )
{
setlocale( LC_ALL, "RUS" );
float a, b, c, p, s;
printf("\na =");  scanf("%f", &a);
printf("\nb =");  scanf("%f", &b);
printf("\nc =") ; scanf("%f", &c);
p = (a + b + c)/2;
s = sqrt( p * (p - a) * (p - b) * (p - c));
printf("\nПлощадь треугольника = %f", s);
}
```

Как уже было сказано, программа состоит из одной главной функции со стандартным именем `main`. Слово `void` обозначает отсутствие какого-либо возвращаемого этой функцией результата, а пустые скобки — отсутствие у нее аргументов. Все, что следует после заголовка функции и заключено в фигурные скобки, можно назвать **телом функции**. Первая строка — объявление используемых **переменных**. Все они плавающего типа `double`. Обратите внимание на то, что объявление переменных заканчивается точкой с запятой.

Дальнейшая часть программы — исполняемые **операторы**. Среди них операторы вывода на экран, ввода данных с клавиатуры, операторы присваивания.

Понятие «оператор» в С трактуется следующим образом: любое выражение, после которого стоит точка с запятой, воспринимается компилятором как отдельный оператор. Оператор определяет законченное действие на очередном шаге выполнения программы. С точки зрения данного выше определения следующая конструкция в программе является оператором:

```
i++;
```

Его называют ***оператором-выражением***. Если вычисление выражения заканчивается присваиванием, то его можно назвать ***оператором присваивания***. В рассматриваемой программе присутствуют два оператора присваивания: вычисления полупериметра (p) и вычисления площади треугольника (S).

В выражении для вычисления площади используется библиотечная функция `sqrt( )` — квадратный корень. Данная функция относится к библиотеке математических функций. Для подключения этой библиотеки к нашей программе используется директива препроцессора `#include <math.h>`. Здесь `math.h` — имя заголовочного файла этой библиотеки. В табл. 4.3 даны описания некоторых наиболее часто используемых функций математической библиотеки С.

Таблица 1. Математические функции (заголовочный файл math. h)

Обращение	**Тип аргумента**	**Тип результата**	**Функция**
`abs( x )`	`int`	`int`	абсолютное значение целого числа
`acos( x )`	`double`	`double`	арккосинус (радианы)
`asin( x )`	`double`	`double`	арксинус (радианы)
`atan(x)`	`double`	`double`	арктангенс (радианы)
`ceil( x )`	`double`	`double`	ближайшее целое, не меньшее x
`cos( x )`	`double`	`double`	косинус (x в радианах)
`exp( x )`	`double`	`double`	e^x — экспонента от x
`fabs( x )`	`double`	`double`	абсолютное значение вещественного x
`floor( x )`	`double`	`double`	наибольшее целое, не превышающее x
`fmod( x, y )`	`double`	`double`	остаток от деления нацело x на y
`log( x )`	`double`	`double`	логарифм натуральный — ln x
`log10( x )`	`double`	`double`	логарифм десятичный — lg x
`pow( x, y )`	`double` `double`	`double`	x в степени y — x^y
`sin( x )`	`double`	`double`	синус (x в радианах)
`sinh( x )`	`double`	`double`	гиперболический синус
`sqrt( x )`	`double`	`double`	корень квадратный (положительное значение)

Обращение	Тип аргумента	Тип результата	Функция
tan(x)	double	double	тангенс (x в радианах)
tanh(x)	double	double	гиперболический тангенс

В рассматриваемой программе операторы printf( ); и scanf( ) ; реализуют соответственно вывод на экран и ввод исходных данных с клавиатуры. Они осуществляют обращение к соответствующим функциям стандартной библиотеки ввода-вывода С, заголовочный файл которой имеет имя stdio.h.

Форматированный вывод на экран. Оператор вызова функции printf( ) имеет следующую структуру:

```
printf( форматная_строка,  список_аргументов );
```

Форматная строка ограничена двойными кавычками (т.е. является текстовой константой) и может включать в себя произвольный текст, управляющие символы и спецификаторы формата. Список аргументов может отсутствовать или же состоять из выражений, значения которых выводятся на экран (в частном случае из констант и переменных). В примере 1 оператор printf ("\na ="); содержит текст ("a =") и управляющие символы ("\n"). Текст выводится на экран в том виде, в котором он записан. Управляющие символы влияют на расположение на экране выводимых знаков. В результате выполнения этого оператора на экран с новой строки выведутся символы a =.
Признаком управляющего символа является значок \. Ниже приводится их список:

```
\n  – перевод строки;
\t  – горизонтальная табуляция;
\r  – возврат курсора к началу новой строки;
\a  – сигнал-звонок;
\b  – возврат на один символ  (одну позицию);
\f  – перевод  (прогон)  страницы;
\v  – вертикальная табуляция.
```

Оператор printf ("\nПлощадь треугольника =%f", s); содержит все виды параметров функции printf. Список аргументов состоит из одной переменной s. Ее значение выводится на экран.

Пара символов %f является **спецификацией формата** выводимого значения переменной s. Значок % — признак формата, а буква f указывает на то, что выводимое число имеет вещественный (плавающий) тип и выводится на экран в форме с фиксированной точкой. Например, если в результате вычислений переменная s получит значение 32,435621, то на экран выведется:

```
Площадь  треугольника = 32.435621
```

Спецификатор формата определяет форму внешнего представления выводимой величины. Вот некоторые спецификаторы формата:

```
%c  – символ;
%s  – строка;
%d  – целое десятичное число (тип  int);
%u  – целое десятичное число без знака (тип unsigned);
%f  –  вещественные числа в форме с фиксированной точкой;
```

```
%e — вещественные числа в форме с плавающей точкой (с мантиссой и
     порядком).
```

Например, после выполнения следующих операторов

```
float  m, p;
int  k ;
m = 84.3;  k =-12; p = 32.15;
printf("\nm =%f\tk =%d\tp =%e", m, k, p);
```

на экран выведется строка:

```
m = 84.299999  k =-12  p = 3.21500e+01
```

Здесь дважды используемый управляющий символ табуляции \t отделил друг от друга выводимые значения. Из этого примера видно, что соответствие между спецификаторами формата и элементами списка аргументов устанавливается в порядке их записи слева направо.

К спецификатору формата могут быть добавлены числовые параметры: ширина поля и точность. Ширина — это число позиций, отводимых на экране под величину, а точность — число позиций под дробную часть (после точки). Параметры записываются между значком % и символом формата и отделяются друг от друга точкой. Внесем изменения в оператор вывода для рассмотренного выше примера.

```
printf ("\nm = %5.2f\tk = %5d\tp = %8.2e\tp = %ll.4e", m, k, p, p);
```

В результате на экране получим:

```
m = 84.30  k = -12  p = 32.15  p = 3.2150e + 01
```

Если в пределы указанной ширины поля выводимое значение не помещается, то этот параметр игнорируется и величина будет выводиться полностью.

К спецификаторам формата могут быть добавлены модификаторы в следующих вариантах:

```
%ld  — вывод  long  int;
%hu  — вывод  short  unsigned;
%Lf  — вывод  long  double.
```

Форматированный ввод с клавиатуры. Оператор вызова функции scanf ( ) имеет следующую структуру:

```
scanf( форматная_строка, список_аргументов );
```

Данная функция осуществляет чтение символов, вводимых с клавиатуры, и преобразование их во внутреннее представление в соответствии с типом величин. В функции scanf( ) форматная строка и список аргументов присутствуют обязательно. В программе из примера 1 имеется оператор scanf("%f", &a);
Здесь "%f" - форматная строка; &a — список аргументов, состоящий из одного элемента. Этот оператор производит ввод числового значения в переменную a.

Символьную последовательность, вводимую с клавиатуры и воспринимаемую функцией scanf( ), принято называть входным потоком. Функция scanf( ) разделяет этот поток на отдельные вводимые величины, интерпретирует их в соответствии с указанным типом и форматом и присваивает переменным, содержащимся в списке аргументов.

Список аргументов — это перечень вводимых переменных, причем перед именем каждой переменной ставится значок &. Это знак операции «взятие адреса переменной». Подробнее смысл этого действия будет объяснен позже, а пока примем это правило формально.

Форматная строка заключается в кавычки (как и для printf) и состоит из списка ***спецификаций***. Каждая спецификация начинается со знака %, после которого могут следовать

```
*ширина_поля модификатор спецификатор
```

Из них обязательным элементом является лишь спецификатор.
Для ввода числовых данных используются следующие спецификаторы:

```
d  — для  целых десятичных чисел  (тип  int);
u  — для целых десятичных чисел без знака (тип unsigned  int);
f  — для вещественных чисел (тип float) в форме с фиксированной точкой;
e  — для  вещественных  чисел  (тип  float)  в форме  с плавающей  точкой.
```

Звездочка в спецификации позволяет пропустить во входном потоке определенное количество символов. Ширина поля — целое положительное число, позволяющее определить число символов из входного потока, принадлежащих значению соответствующей вводимой переменной. Как и в спецификациях вывода для функции printf ( ), в спецификациях ввода функции scanf ( ) допустимо использование модификаторов h f 1, L. Они применяются при вводе значений модифицированных типов:

```
hd  — для  ввода значений  типа  short  int;
ld  — для  ввода  значений  типа  long  int;
lf, le — для ввода значений типа double в форме с фиксированной  и плавающей точкой;
Lf, Le — для ввода значений типа long double в форме с фиксированной и
         плавающей точкой.
```

В программе из примера 1 все три величины a, b, c можно ввести одним оператором:

```
scanf("%f%f%f", &a, &b, &c);
```

Если последовательность ввода будет такой:

```
5  3.2  2.4  <Enter>
```

то переменные получат следующие значения: a = 5,0, 6 = 3,2, c = 2,4. Разделителем в потоке ввода между различными значениями может быть любое количество пробелов, а также другие пробельные символы: знак табуляции, конец строки. Только после нажатия на клавишу Enter вводимые значения присвоятся соответствующим переменным. До этого входной поток помещается в буфер клавиатуры и может редактироваться.

Потоковый ввод-вывод в C++. Программируя на языке C++, можно пользоваться средствами ввода-вывода стандартной библиотеки Си, подключаемой с помощью заголовочного файла stdio.h, как это делалось выше. Однако в C++ имеются свои специфические средства ввода-вывода. Это библиотека классов, подключаемая к программе с помощью файла iostream.h. В этой библиотеке определены в качестве объектов стандартные символьные потоки со следующими именами:

```
cin  — стандартный  поток ввода  с  клавиатуры;
cout — стандартный  поток  вывода  на  экран.
```

Ввод данных интерпретируется как извлечение из потока cin и присваивание значений соответствующим переменным. В C++ определена операция извлечения из стандартного потока, знак которой >> . Например, ввод значений в переменную x реализуется оператором cin >> x;

Вывод данных интерпретируется как помещение в стандартный поток cout выводимых значений. Выводиться могут тексты, заключенные в двойные кавычки, и значения выражений. Знак операции помещения в поток << . Примеры использования потокового вывода:

```
cout << a + b;
cout << "\nРезультат =" << У;
cout << " x = " << x << "  y = " << y << "  z = " << z << endl;
```

Из приведенных примеров видно, что в выходном потоке можно использовать управляющие символы, как и при использовании функции printf ( ); перед каждым элементом вывода нужно ставить знак операции << . Элемент вывода endl является так называемым манипулятором, определяющим перевод курсора на новую строку (действует аналогично управляющему символу \n).

В процессе потокового ввода-вывода происходит преобразование из формы внешнего символьного представления во внутренний формат и обратно. Тип данных и необходимый формат определяются автоматически. Стандартные форматы задаются специальными флагами форматирования, которые устанавливаются с помощью функции setf( ) . Кроме того, на формат отдельных выводимых данных можно влиять путем применения специальных манипуляторов. Здесь мы не будем подробно разбирать эти вопросы.

Перепишем программу из примера 1 в варианте с использованием потокового ввода-вывода C++.

```
#include <iostream.h>
#include <cmath>
#include <clocale>

using namespace std;

void  main( )
{
      setlocale(LC_ALL,"RUS");
      float a, b, c, p, s;
      cout << "\na ="; cin << a;
      cout << "\nb ="; cin >> b;
      cout << "\nc ="; cin >> c;
      p = (a + b + c) /2;
      s = sqrt(p * (p - a)*(p - b) * (p - c));
      cout << "\nПлощадь  треугольника =" << s;
}
```

Упражнения

1. При выполнении программы

```
#include <stdio.h>
void main( )
{
 float a, b, c;
```

```
int m, n;
scanf("%f%d%f%d%f", &a, &m, &b, &n, &c);
printf ( "\na = % 8.3f  d = % 7.2f  c = %12.3e ", a, b, c ) ;
printf("\nm = %10  dn = %5d", m, n );
}
```

с клавиатуры была введена следующая символьная последовательность:

```
32.4 87 0.05 4567 2314.45 <Enter>
```

Как будет выглядеть на экране результат работы программы?

2. Приведенная ниже программа решает следующую задачу: идет к-я секунда суток. Определить, сколько целых часов (Я) и целых минут (М) прошло с начала суток. Например, если

```
к =  13257 = 3 x 36000 + 40 x 60 + 57, то Я = 3, М = 40.
```

Вывести на экран фразу: «Это... часов... минут». Вместо многоточий поставить вычисленные значения Ч М.

```
#include <stdio.h>
void  main( )
      {
      long k;
      int h, m;
      printf("Введите  текущее время в секундах:");
      scanf("%ld", &k) ;
      h = k/3600;
      m = ( k%3600)/ 60;
      printf("Это %d часов %d минут.\n", h, m) ;
      }
```

Разобрать, как работает данная программа. Переписать ее с использованием потокового ввода-вывода C++.

3. Составить программу решения обратной задачи по отношению к предыдущей: дано количество часов и минут, прошедших от начала суток. Определить количество секунд.
4. Составить программу вычисления объема и площади поверхности куба по данной длине ребра.
5. Составить программу для вычисления корней квадратного уравнения.

Литература

1. **Семакин И.Г., Шестаков А.П.** Основы программирования: Учебник.- М.: Мастерство, 2002.- 432 с.