## Лабораторная работа №3

## Тема: “Линейные алгоритмы. Операторы ввода и вывода в языке программирования Си”

**Цель работы:** изучить операторы ввода и вывода, форматы, используемые в этих операторах. Разработать линейные алгоритмы и реализовать с применением этих операторов.

### 1. Теоретические сведения

Линейный алгоритм – это алгоритм, в котором все действия выполняются в строгом порядке, последовательно, одно за другим. Программа, реализующая линейный алгоритм, называется программой с линейной структурой.

### 1.1 Структура программы

В языке СИ любая программа состоит из одной или более функций, задающих действия, которые нужно выполнить. Выполнение любой программы начинается с функции main. Данная функция может не иметь параметров и может не возвращать результат, но наличие круглых скобок (как и для других функций без параметров) обязательно.

Упрощенная структура программы на языке Си:

<директивы препроцессора>

<определение глобальных переменных>

<функции>

Рассмотрим кратко основные части структуры программ.

Перед компиляцией программа на языке Си обрабатывается специальной программой – препроцессором, который работает под управлением директив. Директивы записываются по следующим правилам:

1) все препроцессорные директивы должны начинаться с символа #;

2) все директивы начинаются с первой позиции;

3) сразу за символом # должно следовать наименование директивы, указывающее текущую операцию препроцессора.

Препроцессор решает ряд задач по предварительной обработке программы, основной из которых является «подключение» к программе так называемых заголовочных файлов (обычных текстов) с декларацией стандартных библиотечных функций, которые используются в программе. Наименование такой директивы: #include (подключить), а общий формат ее использования:

#include <файл 1.h>

...

#include <файл n.h>

#include “файл.h”

препроцессор подставляет на место этих директив тексты файлов:

файл1...n, содержащих прототипы-объявления стандартных библиотечных функций и декларации некоторых дополнительных объектов соответствующей библиотеки.

Если имя файла заключено в угловые скобки < >, то поиск данного файла производится в стандартной директории с этими файлами, если же имя файла заключено в двойные кавычки ” ”, то поиск данного файла производится в текущем директории.

Например:

#include <stdio.h> – подключается файл с объявлениями стандартных функций файлового ввода-вывода;

#include <conio.h> – функции для работы с консолью;

#include <graphics.h> – содержит прототипы графических функций;

#include <math.h> – объявляет прототипы различных математических функций.

Второе основное назначение препроцессора – это обработка макроопределений, которые начинаются с #define (определить). Директива #define предписывает компилятору заменить имя константы на то, что следует за этим именем.

Макроподстановка #define имеет общий вид:

#define <идентификатор значение>

Например:

#define PI 3.14

В ходе препроцессорной обработки программы появление в тексте идентификатора PI заменяется значением 3.14.

Далее идет заголовок функции и текст, который заключается в фигурные скобки.

Пример:

```
Заголовок   #include <stdio.h>    /*директива препроцессора*/
функции        void main()          /*имя функции с аргументами */

                {
Тело              int num;              /*оператор описания*/
функции                num=25 ;          /*операция присваивания*/
                       printf("%d  это  число   \n",num);
                }
```

Таким образом, отличительным признаком имени функции служат круглые скобки ( ), в которые заключается список аргументов. Аргументы могут отсутствовать. Здесь атрибут void – отсутствие значения (используется для нейтрализации значения, возвращаемого функцией, или для декларации указателей на область памяти без назначения типа объекта). А тело функции представляет собой набор операций и операторов, каждый из которых оканчивается символом «;».

## 1.2 Алфавит языка и типы данных

Алфавит языка включает латинские прописные и строчные буквы, цифры и специальные знаки. К последним относятся: . (точка), , (запятая), ‘ (апостроф), : (двоеточие) и др.

Важным понятием языка является идентификатор, который используется в качестве имени объекта, например, переменной, функции и т.п. Идентификатор может содержать до 32 символов и состоит из букв и цифр, но начинается обязательно с буквы. Строчные буквы отличаются от прописных, поэтому идентификаторы SIGMA и sigma считаются разными.

В языке СИ существует несколько типов данных. Каждый тип данных определяется одним из следующих ключевых слов:

1. **int** (целый) – целые числа. Диапазон возможных целых значений лежит в пределах от –32768 до 32767, переменная типа int занимает 16 бит;
2. **char** (символьный) – задает значения, которые представляют различные символы;
3. **float** (вещественный) – задает значения, к которым относятся вещественные числа, имеющие дробную часть, отделяемую точкой. Вещественные числа могут быть записаны также в экспоненциальной форме. Например, -1.58e+2 (что равно $-1{,}58 \cdot 10^2$ ). В языке СИ переменная типа float занимает 32 бита. Она может принимать значения в диапазоне от +3.4e-38 до +3.4e+38;
4. **double** (двойная точность) – определяет вещественные переменные, занимающие в два раза больше места, чем переменная типа float. Переменная типа double занимает 64 бита. Она может принимать значения в диапазоне от +1.7e-308 до +1.7e+308.
5. **void** – неопределенный (несуществующий) тип. Объектов типа void не существует. Множество значений этого типа пусто, т.е. нельзя переменной такого типа присвоить какое-нибудь значение. Более того, нельзя даже описать переменную этого типа. С его помощью задаются пустой список аргументов функции или функции, не возвращающие значение.

6.

В свою очередь, данные целого типа могут быть короткими (short),

длинными (long) и беззнаковыми (unsigned).

7. **short** (короткий целый) – соответствующие объекты не могут быть больше, чем int, переменные этого типа занимают 16 бит;

8. **long** (длинный целый) – соответствующие объекты не могут быть меньше, чем int. Переменная типа long занимает 32 бита и позволяет представить целые числа от –2147483648 до 2147483647;

9. **unsigned** (беззнаковый) – в языке СИ можно объявлять некоторые типы (char, short, int, long) беззнаковыми с помощью модификатора unsigned (например, unsigned short). Это значит, что соответствующие переменные не будут иметь отрицательных значений. В результате они могут принимать большие положительные значения, чем переменные знаковых типов. В случае типа int объявления вида «unsigned int a;» можно записать «unsigned a;»;

## 1.3 Ввод и вывод информации

### 1.3.1 Форматный вывод

Вначале рассмотрим функцию, определяющую форматный вывод:

printf("управляющая строка", аргумент1, аргумент2, ... );

Управляющая строка содержит объекты трех типов: обычные символы, которые просто выводятся на экран дисплея, спецификации преобразования, каждая из которых вызывает вывод на экран значения очередного аргумента из последующего списка и управляющие символы-константы.

Каждая спецификация преобразования начинается со знака % и заканчивается некоторым символом, задающим преобразования.

Символ преобразования связан с типом переменных. приведем символы преобразования:

1. d – значением аргумента является десятичное целое число;
2. o – значением аргумента является восьмеричное целое число;
3. x – значением аргумента является шестнадцатеричное целое число;
4. c – значением аргумента является символ;
5. s – значением аргумента является строка символов;
6. e – значением аргумента является вещественное число в экспоненциальной форме;
7. f – значением аргумента является вещественное десятичное число с плавающей точкой;
8. u – значением аргумента является беззнаковое целое число;
9. p – значением аргумента является указатель (адрес).

Если после знака % записан не символ, то он выводится на экран. Функция printf использует управляющую строку, чтобы определить, сколько всего

аргументов и каковы их типы.

Например, в результате работы программы получены переменная i, имеющая значение 100, и переменная j, имеющая значение 25. Обе переменные целого типа. Для вывода этих переменных на экран в виде

**i=100 j=25**

необходимо применить функцию

printf(“i=%d j=%d”,i,j);

Как было описано выше, в кавычках задается формат вывода. перед знаком % записываются символы, которые будут непосредственно выданы на экран. После знака % применена спецификация d, т.к. переменные i и j имеют целый тип. Сами i и j приведены через запятую в списке аргументов. Если результат должен быть представлен в виде

**i=100; j=25**

необходимо применить функцию

printf(“i=%d; j=%d, i, j);

Если после знака % стоит цифра, то она задает поле, в котором будет выполнен вывод числа. Приведем несколько функций printf, которые будут обеспечивать вывод одной и той же переменной S целого типа, имеющей значение 336.

Функция printf(“%2d”, S); выдает на экран:

**336**

В этом примере ширина поля ( она равна двум) меньше, чем число цифр в числе 336, поэтому поле автоматически расширяется до необходимого размера.

Функция printf(“%6d”, S);

выдаст на экран:

**_ _ _336**

(6 позиций)

То есть, в результате работы функции число сдвинуто к правому краю поля, а лишние позиции перед числом заполнены пробелами.

Функция printf(“%-6d”, S);

выдаст на экран:

**336_ _ _**

(6 позиций)

Знак «минус» перед спецификацией приводит к сдвигу числа к левому краю поля.

Рассмотрим вывод вещественных чисел.

Если перед спецификацией f ничего не указано, то выводится число с шестью знаками после запятой. при печати числа с плавающей точкой перед спецификацией f тоже могут находиться цифры.

Рассмотрим на конкретном примере три возможные ситуации:

%6f – печать числа с плавающей точкой в поле из шести позиций;

%.2f – печать числа с плавающей точкой с двумя цифрами после десятичной точки;

%6.2f – печать числа с плавающей точкой в поле из шести позиций и двумя цифрами после десятичной точки.

Например, в результате работы программы получены переменные вещественного типа a=3,687 и b=10,17.

Если для вывода значений использована функция

printf(“%7f %8f”,a,b);

то результат будет представлен в виде строки:

**_ _ 3.687 _ _ _ _10.17**

(7 поз.) (8 позиций)

Как видно из примера, лишние позиции заполняются пробелами. Если для вывода значений использована функция

printf(“%.2f %/2f”, a, b);

то результатом будет строка:

**3.69 10.17,**

из которой следует, что в первом числе третья цифра после десятичной точки отброшена с округлением, т.к. указан формат числа с двумя цифрами после десятичной точки.

Если для вывода значений использована функция

printf(“%7.2f e”,a,b);

то будет выведена строка:

**_ _ _ 3.681.010000e+01**

(7 позиций)

Поскольку для вывода значения переменной b применена спецификация e, то результат выдан в экспоненциальной форме. Следует отметить, что , если ширина поля меньше, чем число цифр в числе, то поле автоматически расширяется до необходимого размера.

Как было отмечено выше, в управляющей строке могут содержаться управляющие символьные константы. Среди управляющих символьных констант наиболее часто используются следующие:

1. \a – для кратковременной подачи звукового сигнала;
2. \b – для перевода курсора влево на одну позицию;
3. \n – для перехода на новую строку;
4. \r – для перевода курсора в начало текущей строки;
5. \t – для горизонтальной табуляции;
6. \v – для вертикальной табуляции.

Предположим, в результате работы программы переменная i получила значение 50. В результате записи инструкции вызова функции

printf(“\t ЭВМ\n%d\n”,i);

сначала выполнится горизонтальная табуляция (\t), т.е. курсор сместится от края экрана на 8 позиций, затем на экран будет выведено слово “ЭВМ”, после этого курсор переместится в начало следующей строки (\n), затем будет выведено целое значение i по формату d, и окончательно курсор перейдет в начало новой строки (\n). Таким образом, результат работы этой функции на экране будет иметь вид:

**_ _ _ _ _ _ _ _ ЭВМ**

**50**

### 1.3.2 Ввод данных

Для форматного ввода данных используется функция

scanf(«управляющая строка», аргумент1, аргумент2,...);

Если в качестве аргумента используется переменная, то перед ее именем записывается символ &.

Управляющая строка содержит спецификации преобразования и используется для установления количества и типов аргументов. спецификации для определения типов аргументов такие же, как и для функции printf. Перед символами d,o,x,f может стоять буква l. В первых трех случаях соответствующие переменные должны иметь тип long, а в последнем double.

Рассмотрим пример. Требуется ввести значения для переменных i (целого типа) и a (вещественного типа). Эту задачу выполнит функция:

scanf(“%d%f”,&i,&a);

В управляющей строке спецификации трех типов могут быть отделены друг от друга различными знаками, в том числе и пробелом. Следовательно, при занесении значений переменных необходимо использовать указанный разделитель. Если спецификации не отделены одна от другой никакими значениями, то значения переменных заносятся через пробел.

В языке СИ есть две очень удобные функции puts и gets, позволяющие вводить и выводить строку символов. Пример их использования показан ниже:

```
#include<stdio.h>
main()
{
char q[40]; /*объявление строки символов*/
puts(“Введите  строку символов”);
gets(q); /*ввод строки символов*/
puts(q); /*вывод строки символов*/
}
```

В результате работы программы вначале на экране появится текст:

**Введите строку символов,**

после чего следует ввести какую-либо строку символов. Эта информация

при помощи оператора gets будет присвоена элементам символьного массива q. Оператор puts выведет строку символов.

## 1.4 Операторы и выражения

Выражения широко используются в программах на языке СИ и представляют собой формулы для вычисления переменных. Они состоят из операндов (переменные, константы и др.), соединенных знаками операций (сложение, вычитание, умножение и др.). Порядок выполнения при вычислении значения выражения определяется их приоритетами и может регулироваться с помощью круглых скобок. Наиболее часто арифметические выражения используются в операторе присваивания. Этот оператор заменяет значение переменной в левой части оператора на значение выражения, стоящего в правой части, и имеет следующую форму:

переменная = выражение;

В языке СИ может быть использован модификатор const, запрещающий какие бы то ни было переопределения константы: ее уменьшение, увеличение и т.п. Модификатор const, используемый отдельно, эквивалентен const int. Приведем примеры:

```
const float a=3.5;
const j=47;
```

В таблице 1 приведены арифметические операции, используемые в языке СИ.

Таблица 1

| Знак операции | Выполнение действия |
|---|---|
| + | Сложение |
| – | Вычитание |
| * | Умножение |
| / | Деление |
| % | Деление по модулю |

Результатом деления по модулю является остаток от деления. Например, если b=5, c=2, то при выполнении операции

```
a=b%c,
```

переменная a получит значение 1.

Широкое распространение находят также выражения с еще одной нетрадиционной терпарной операцией ?: . В выражении

y=x?a:b,

y=a, если x не равно нулю, и y=b, если x равно нулю. Следующее выражение

y=(a>b)?a:b;

позволяет присвоить переменной y значение большей переменной (a или b), т.е. y=max(a,b).

В таблице 2 приведены некоторые функции, применяемые при программировании на СИ.

Таблица 2

| Математическая запись | Запись на языке СИ |
|---|---|
| \| X \| | int abs(int X) |
| \| X \| | float fabs(float X) |
| arccos X | double acos(double X) |
| arcsin X | double asin(double X) |
| arctg X | double atan(double X) |
| cos X | double cos(double X) |
| sin X | double sin(double X) |
| tg X | double tan(double X) |
| $e^X$ | double exp(double X) |
| ln X | double log(double X) |
| log X | double log10(double X) |
| √X | double sqrt(double X) |
| $X^Y$ | double pow(double X, double Y) |

Перед аргументом и функцией указан допустимый тип (при программировании эта запись типа опускается).

В программах на языке СИ важная роль отводится комментариям, которые повышают наглядность и удобство чтения программ. Они могут быть записаны в любом месте программы и обрамляются символами /* и */.

Рассмотрим пример программы на языке СИ.

Требуется вычислить:

Для работы с математическими функциями необходимо перед функцией main поместить строку:

#include <math.h>

Программа на СИ имеет вид:

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

```
{
float z,f,k; /*объявление вещественных переменных z,f,k*/
double y,a,b,c,d,x; /*объявление переменных y,a,b,c,d,x переменными двойной точности*/
scanf("%f %f %f %lf %lf", &z, &f, &k, &d, &x); /* ввод с клавиатуры переменных z,f,k,d,x*/
a=log(x)+(z+f)/k;
b=sin(x)+tan(x);
c=pow(d+exp(x),1./5);
y=(a+b)/c;
printf("%lf %lf %ef %lf", a, b, c, y); /*вывод на экран значений переменных a,b,c,y*/
}
```

Следует обратить внимание на то, что при вычислении переменной c, выражение, стоящее в правой части, представлено как $\sqrt[5]{(d+e^x)}$, поэтому применена функция pow.

**Еще одно замечание!!!** Следует осторожно подходить к делению целых чисел. Если оба операнда целые, то результат тоже будет целым, а дробная часть отбрасывается. таким образом, при выполнении операции 1/5, результат будет равен нулю. Для того чтобы сохранить дробную часть, хотя бы один из операндов должен быть вещественным. Это условие выполнено при вычислении 1./5 или 1/5f.

**Задание** выбирается согласно варианту (одно из группы А и одно из группы Б). Необходимо составить блок-схему линейного алгоритма по каждому заданию и написать программу для вычисления выражений на языке СИ.

Задание А

1. $t = \dfrac{2\cos(x-\pi/6)}{0{,}5+\sin^2 y}\left(1+\dfrac{z^2}{3-z^2/5}\right)$.

При $x$ = 14.26, $y$ = –1.22, $z = 3.5\times10^{-2}$ : **0.564846.**

2. $u = \dfrac{\sqrt[3]{8+|x-y|^2+1}}{x^2+y^2+2} - e^{|x-y|}(tg^2 z+1)^x$ .

При $x$ = –4.5, $y = 0.75\times10^{-4}$, $z = 0.845\times10^{2}$ : **–55.6848.**

3. $v = \dfrac{1+\sin^2(x+y)}{\left|x-\dfrac{2y}{1+x^2y^2}\right|}x^{|y|} + \cos^2[arctg(1/z)]$ .

При $x = 3.74\times10^{-2}$, $y$ = –0.825, $z = 0.16\times10^{2}$ : **1.0553.**

4. $w = \left|\cos x - \cos y\right|^{(1+2\sin^2 y)}\left(1+z+\dfrac{z^2}{2}+\dfrac{z^3}{3}+\dfrac{z^4}{4}\right)$.

При $x = 0.4\times10^{4}$, y = –0.875, $z = -0.475\times10^{-3}$ : **1.9873**.

5. $\alpha = \ln(y^{-\sqrt{|x|}})(x - y/2) + \sin^2 arctg(z).$

При $x = -15.246$, $y = 4.642\times10^{-2}$, $z = 20.001\times10^{2}$ : **–182.036**.

6. $\beta = \sqrt{10(\sqrt[3]{x} + x^{y+2})} \cdot (\arcsin^2 z - |x - y|)$

При $x = 16.55\times10^{-3}$, $y = -2.75$, $z = 0.15$ : **40.630694**.

7. $\gamma = 5arctg(x) - \frac{1}{4}\arccos(x)\frac{x + 3|x - y| + x^2}{|x - y|z + x^2}.$

При $x = 0.1722$, $y = 6.33$, $z = 3.25\times10^{-4}$ : **–205.305571**.

8. $\varphi = \frac{e^{|x-y|}|x - y|^{x+y}}{arctg\ x + arctg\ z} + \sqrt[3]{x^6 + \ln^2 y}.$

При $x = -2.235\times10^{-2}$, $y = 2.23$, $z = 15.221$ : **39.374**.

9. $\psi = |x^{y/x} - \sqrt[3]{y/x}| + (y - x)\frac{\cos y - z/(y - x)}{1 + (y - x)^2}.$

При $x = 1.825\times10^{2}$, $y = 18.225$, $z = -3.298\times10^{-2}$ : **1.2131**.

10. $a = 2^{-x}\sqrt{x + \sqrt[4]{|y|}}\ \sqrt[3]{e^{x-1/\sin z}}.$

При $x = 3.981\times10^{-2}$, $y = -1.625 \times \times10^{3}$, $z = 0.512$ : **1.26185**.

11. $b = y^{\sqrt[3]{|x|}} + \cos^3 y\ \frac{|x - y| \cdot \left(1 + \frac{\sin^2 z}{\sqrt{x + y}}\right)}{e^{|x-y|} + x/2}.$

При $x = 6.251$, $y = 0.827$, $z = 25.001$ : **0.7121**.

12. $c = 2^{y^x} + (3^x)^y - \frac{y \cdot (arctgz - \pi/6)}{|x| + \frac{1}{y^2 + 1}}.$

При $x = 3.251$, $y = 0.325$, $z = 0.466\times10^{-4}$ : **4.251433**.

13. $f = \frac{\sqrt[4]{y + \sqrt[3]{x - 1}}}{|x - y|(\sin^2 z + tg\ z)}.$

При $x = 17.421$, $y = 10.365\times10^{-3}$, $z = 0.828\times10^{5}$ : **0.33056**.

14. $g = \frac{y^{x+1}}{\sqrt[3]{|y - 2|} + 3} + \frac{x + y/2}{2|x + y|}(x + 1)^{-1/\sin z}.$

При $x = 12.3\times10^{-1}$, $y = 15.4$, $z = 0.252\times10^{3}$ : **82.825623**.

15. $h = \frac{x^{y+1} + e^{y-1}}{1 + x|y - tgz|}(1 + |y - x|) + \frac{|y - x|^2}{2} - \frac{|y - x|^3}{3}.$

При $x = 2.444$, $y = 0.869\times10^{-2}$, $z = -0.13\times10^{3}$ : **–0.49871**.

16. $w = \sqrt[3]{x^6 + \ln^2 y} + \frac{e^{|x-y|}|x - y|^{x+y}}{arctg(x) + arctg(z)}.$

При $x = -2.235\times10^{-2}$, $y = 2.23$, $z = 15.221$ : **39.374**.

Задание Б

1. Дана величина A , выражающая объем информации в байтах. Перевести A в другие единицы измерения информации (биты, мегабайты, килобайты, мегабайты).

2. Даны два действительных числа x и y . Вычислить их сумму, разность, произведение и частное.

3. Дано x . Получить значения $-2x+3x^2-4x^3$ и $1+2x+3x^2+4x^3$ с наименьшим числом произведенных операций.

4. Дано действительное число x . Не пользуясь никакими другими арифметическими операциями, кроме умножения, сложения и вычитания, вычислить за минимальное число операций $2x^4-3x^3+4x^2-5x+6$ .

5. Даны два действительных числа x и y . Вычислить их сумму, разность, произведение и частное.

6. Ввести любой символ и определить его порядковый номер, а также указать предыдущий и последующий символы.

7. Найти сумму членов арифметической прогрессии, если известны ее первый член, знаменатель и число членов прогрессии.

8. Найти сумму и произведение последних двух цифр заданного числа.

9. Даны три цифры A, B, C, сформировать из них действительное число вида A,BC.

10. Ввести два любых символа и определить сумму их порядковых номеров.

11. Дана величина D , выражающая длину в метрах. Перевести D в другие единицы измерения длины (километры, сантиметры, миллиметры).

12. Даны два целых числа x и y . Вычислить их сумму, разность, произведение и частное.

13. Даны три цифры, сформировать из них трехзначное число.

14. Дана величина L , выражающая объем в $м^3$. Перевести L в другие единицы измерения объема ($дм^3$, $см^3$).

**Содержание отчета:**

1. Титульный лист с темой работы
2. Цель работы
3. Вариант выполнения заданий
4. Задание по варианту
5. Схемы алгоритмов решения задач
6. Тексты программ
7. Результаты выполнения (не менее двух с разными исходными данными по каждому заданию)
8. Выводы