Лабораторная работа №1

Линейные вычислительные процессы

Цель работы: Изучить правила составления программ на языке СИ: встроенные типы данных, ввод-вывод данных, основные математические функции. Научиться программировать линейные алгоритмы.

Краткие теоретические сведения

Элементы языка

Символы — это основные знаки, с помощью которых пишется весь текст программы:

- прописные и строчные буквы латинского алфавита, и знак подчеркивания;
- арабские буквы от 0 до 9;
- специальные символы:

.	— точка	(	— левая круглая скобка
,	— запятая	)	— правая круглая скобка
;	— точка с запятой	[	— левая квадратная скобка
:	— двоеточие	]	— правая квадратная скобка
?	— вопросительный знак	{	— открывающая фигурная скобка
!	— восклицательный знак	}	— закрывающая фигурная скобка
’	— апостроф	<	— знак «меньше»
|	— вертикальная черта	>	— знак «больше»
/	— дробная черта	=	— знак «равно»
\	— обратная черта	^	— логическое «или»
~	— тильда	&	— логическое «и»
+	— плюс	%	— знак процента
−	— минус	#	— номер
*	— звездочка	"	— кавычки

- четвертую группу символов составляют управляющие последовательности, используемые при вводе-выводе; они начинаются со знака обратной черты и будут рассмотрены ниже.

Лексема образуется из символов и имеет самостоятельный смысл:

- *Идентификаторы* — имена объектов C/C++ — программ. *Идентификатор* — это последовательность латинских букв, цифр и знака подчеркивания. Первым символом должна быть буква или знак подчеркивания (но не цифра). Пробелы и ключевые слова в идентификаторах не допускаются. Прописные и строчные буквы различаются.
- *Ключевые слова* — это зарезервированные идентификаторы, которые имеют специальное значение для компилятора.
- *Знаки операций* — один или более символов, определяющих действия над операндами. Операции бывают унарные, бинарные и тернарные. Все знаки операций, кроме [ ] ( ) и ?, являются отдельными лексемами.
- *Константы* — это неизменные величины.

  Константы бывают строковые ("Hello\t Word\n"), целые (123, 020, OxA), вещественные (5.7, .45, 0.2E6) и символьные (’a’, ’\n’, ’\0’, ’\x07’).
- *Разделители* (скобки, точка, запятая, пробельные символы).

Выражение — это правило вычисления некоторого действия. Оно состоит из операндов, знаков операций и скобок, которые используются для вычисления некоторого значения. *Операнд* — это выражение, константа или переменная.
Оператор — это задание законченного описания действия. Выполнение оператора — это вычисление данного выражения.
Составной оператор — это последовательность операторов, заключенная в фигурные скобки:

```
{
  n++;
  summa+=n;
}
```

Блок — это последовательность операторов, заключенная в фигурные скобки, а также наличие определений переменных:

```
{
  int n=0;
  n++;
  summa+=n;
}
```

Типы данных

Каждый тип данных имеет определенный размер, т. е. сколько байтов выделяется в оперативной памяти для записи переменной данного типа, и диапазон значений.

В языке Си применяются данные двух категорий: *простые* (скалярные) и *сложные* (составные) типы данных.

К *простым типам данных* относятся:

- символы;
- указатели;
- перечисления;
- целые;
- вещественные.

К *сложным типам данных* относятся:

- массив;
- структура;
- объединение;
- битовые поля.

Целый тип данных

Данные целого типа (`int`) могут быть короткими — `short`, длинными — `long`. Размер переменной целого типа зависит от системы, например, в 32-разрядных операционных системах размер переменной `int` равен 4 байтам (32 битам), что позволяет хранить в `int` значения от $-2\,147\,483\,648$ до $2\,147\,483\,648$.

Тип `short` в любой операционной системе имеет размер, равный двум байтам. Диапазон значений типа `short` — от $-32\,768$ до $32\,767$.

Размер типа `long` всегда равен 4 байтам и совпадает с размером типа `int` в случае 32-разрядных систем. Тип `long` может быть описан как `long int` между двумя такими описаниями нет разницы.

Символьный тип

Размер памяти, занимаемый символьными переменными (`char`), равен 1 байту (8 битам). Каждому символу ставится в соответствие число, называемое кодом символа. Поэтому символьные переменные хранят целые числа, содержащиеся в диапазоне от -128 до 127. Константа типа `char` — это символ, заключенный в одиночные кавычки, например: 'a'. Иногда символьные переменные используют для представления целых чисел, заключенных в указанном диапазоне, но гораздо чаще в таких переменных хранятся ASCII-коды символов. В отличие от символьной константы, строковая константа записывается в двойных кавычках, например, "Слово". Таким образом, строковая константа — это массив символов.

Вещественный тип данных

Переменные вещественного типа хранят числа в десятичной форме представления, например 3.1415927, 0.0000625, -10.2. У таких чисел есть как целая часть, стоящая слева от десятичной точки, так и дробная часть, стоящая справа от нее. Переменные вещественного типа предназначены для хранения вещественных чисел — тех чисел, которыми измеряются непрерывные величины. Вещественные числа, как правило, имеют ненулевую дробную часть.

В C/C++ имеются три вещественных типа: `float`, `double` и `long double`.

Размер типа `float` равен 4 байтам (32 битам), и этот тип способен хранить числа, содержащиеся в интервале от $3.4 \cdot 10^{-38}$ до $3.4 \cdot 10^{38}$, с точностью до семи знаков после запятой.

Тип `double` занимает 8 байтов памяти и хранит значения от $1.7 \cdot 10^{-308}$ до $1.7 \cdot 10^{308}$ с точностью до 15 знаков после запятой.

Характеристики типа `long double` зависят от компилятора, но чаще всего они совпадают с характеристиками типа `double`.

Беззнаковые типы данных

Если исключить из представления целых чисел знак, то полученный тип данных будет собой представлять неотрицательные целые числа с удвоенной верхней границей диапазона представления. Беззнаковые типы данных предваряются словом `unsigned` (нижняя граница диапазона этих данных — 0):

Название	Верхняя граница диапазона	Размер в байтах
`unsigned char`	255	1
`unsigned short`	65 535	2
`unsigned int`	4 294 967 295	4
`unsigned long`	4 294 967 295	4

Определение переменных

Переменные в программе, перед их использованием, необходимо определять. При определении переменные можно инициализировать (т. е. задавать начальные значения).

Определение переменной могут иметь следующие виды:

1. Определение одной переменной заданного типа

   ⟨`Тип_данных`⟩ ⟨`имя_переменной_`1⟩

2. Определение нескольких переменных одного типа

   ⟨`Тип_данных`⟩ ⟨`имя_переменной_`1⟩ . . . ⟨`имя_переменной_`n⟩

3. Определение переменной с инициализацией

   ⟨`Тип_данных`⟩ ⟨`имя_переменной`⟩ = ⟨`начальное_значение`⟩

⟨`Тип_данных`⟩ — это определенный в программе тип данных (`int`, `char`, `double` и т.д.);
⟨`имя_переменной`⟩ — это идентификатор;

⟨`начальное_значение`⟩ — это выражение соответствующего типа.

Например:

```
int j = 10, m = 3, n = j + m;
float c=-1.3, g=-10.23, nl;
char chi = 'a', ch2;
```

Операция присваивания

Операция присваивания выполняется справа налево. Операция присваивания имеет две формы записи (полная и короткая).

Полная форма записи: *переменная* = *выражение*;

Примеры присваивания полной формы:

```
int x;
x = 25;

double y;
y=(x+2)/(3.5*x)-5;

int x,y,z;
x=y=z=5;

int x,y,z;
x = (y=5)-(z=3);
```

Сокращенная форма записи: *переменная* ⟨*операция*⟩= *выражение*; где *операция* — одна из арифметических операций (`+` , `-`, `*`, `/`, `%`),

Примеры присваивания сокращенной формы:

```
x*=5;     ↔  x=x*5;
s+=7;     ↔  s=s+7;
y/=x+3;   ↔  y=y/(x+3);
```

Сокращенная форма операции присваивания применяется, когда переменная используется в обеих частях полной формы данного оператора; левее операций присваивания не могут находиться:

– константы, т. е. `2=x;`

– функции, т. е. `sqrt( )=x;`

– вычисления, т. е. `x+2=y-z;`

В языке C/C++ существуют операции уменьшения (`--`) и увеличения (`++`) значения переменной на 1. Операции могут быть *префиксные* (`++i` и `--i`) и *постфиксные* (`i++` и `i--`).

При использовании префиксной операции в выражении сначала выполняется сама операция (изменяется значение `i`), а затем вычисляется выражение.

В случае постфиксной операции — операция применяется после вычисления выражения.

Пример использования постфиксной операции:

```
int x=5,y;
y=x++;
```

В данном примере сначала значение переменной `x` записывается в переменную `y`, а затем значение переменной `x` увеличивается на единицу. Следовательно, в переменной `x` будет храниться значение 6, а в переменной `y` значение 5. Запись `y=x++;` аналогична записям `y=x;  x++;`

Пример использования префиксной операции:

```
int x = 5, y;
y = ++x;
```

В данном примере сначала значение переменной `x` увеличивается на единицу, а затем новое значение переменной `x` присваивается в переменной `y`. Следовательно, в обоих переменных будет храниться значение 6. Запись `y=++x;` аналогична записям `x++; y=x;`

Структура программы

Программа, написанная на языке Си, состоит из директив препроцессора, объявлений глобальных переменных, одной или нескольких функций, среди которых одна главная (main) функция, которая управляет работой всей программы.

Программа на языке Си имеет следующую структуру:

- `#` директивы препроцессора
- определения глобальных переменных
- прототипы функций
- `void main( )` //функция, с которой начинается выполнение программы

```
{
  определения переменных
  операторы присваивания
  вызов функций
  составные операторы
  операторы выбора
  операторы циклов
  операторы перехода
}
```

- описания фунций

Директивы препроцессора

Перед компиляцией программы на языке C/C++ автоматически выполняется препроцессорная обработка текста программы. С помощью директив препроцессора задаются необходимые действия по преобразованию текста программы перед компиляцией.

Директивы начинаются с символа `#`. За символом `#` следует наименование директивы, указывающее текущую операцию препроцессора. Наиболее распространены директивы `#include` и `#define`.

Директива `#include` используется для подключения к программе заголовочных файлов:

```
#include <имя заголовочного файла>
```

Употребление директивы `include` не подключает соответствующую стандартную библиотеку, а только позволяет вставить в текст программы описания из указанного заголовочного файла.

Подключение кодов библиотеки осуществляется после компиляции. Хотя в заголовочных файлах содержатся все описания стандартных функций, в код программы включаются только те функции, которые используются в программе.

Например:
`#include <stdio.h>` — подключение файла с объявлением стандартных функций файлового ввода-вывода;

`#include <conio.h>` — функции работы с консолью;
`#include <math.h>` — математические функции.

Директива `#define` создает макроконстанту и ее действие распространяется на весь файл, например:

```
#define PI 3.1415927
```

В ходе препроцессорной обработки программы идентификатор `PI` заменяется указанным значением 3.1415927.

Функции ввода-вывода информации

Для **вывода** информации чаще всего используется функция форматированного вывода данных:

```
printf ("управляющая строка", список вывода);
```

управляющая строка — указывает компилятору вид выводимой информации и содержит спецификации преобразования, управляющие символы и комментарии.

Спецификация преобразования имеет вид:

```
%<флаг><размер поля.точность> спецификация
```

где `флаг` может принимать следующие значения: − выравнивание влево выводимого числа (по умолчанию — вправо); + выводится знак положительного числа;
`размер поля` — минимальная ширина поля, т. е. длина числа, (при недостаточной ширине поля выполняется автоматическое расширение);
`точность` — количество цифр в дробной части числа;
`спецификация` — спецификаторы формата и управляющие символы;
`список вывода` — печатаемые объекты (константы, переменные или выражения, вычисляемые перед выводом) по количеству, порядку следования и типу должны соответствовать спецификациям преобразования в управляющей строке.

Функция `puts` — предназначена для вывода строки символов с переходом на начало новой строки.

В C++ оператор `cout` вместе с операцией `<<` служат для вывода на экран:

```
cout << variable;
```

Для форматированного **ввода** информации используется функция:

```
scanf ("управляющая строка", список ввода);
```

Для нее, как и для функции `printf`, указывается управляющая строка, и список ввода. Однако для функции `scanf` управляющая строка может содержать только спецификаторы формата без спецификации преобразования, без управляющих символов и без комментариев.

Список вывода использует указатели на переменные, т. е. их адреса, а не просто имена переменных. Для обозначения указателя перед именем переменной записывается символ &, обозначающий адрес переменной. Ввод данных для функции `scanf` завершается пробелом или нажатием клавиши `enter`. Вводить данные можно как в одной строке через пробел, так и в разных строках.

Функция `gets` используется только для ввода символьных строк. Ввод строки завершается нажатием клавиши Enter.

Функция `getch` используется только для ввода символов. Функция `getch` находится в библиотечном файле `conio.h`; возвращает код нажатой пользователем клавиши, причем вводимый

символ не отображается на экране результатов; для получения расширенного кода функциональных или курсорных клавиш необходимо повторно вызвать данную функцию.

В C++ определено ключевое слово `cin` для работы со стандартным потоком ввода. Этот поток содержит данные, вводимые с клавиатуры (если он не переопределен). `>>` — операция извлечения. Она извлекает данные из потокового объекта, стоящего в левой части, и присваивает эти данные переменной, стоящей в правой части.

```
cin >> variable;
```

Спецификаторы формата

Для функций `printf`, `scanf` существуют спецификаторы формата. Это условные обозначения, позволяющие считать из входящего потока определенную конструкцию и сохранить ее.

Например, следующие спецификаторы:

`%c` — символ,	`%ld` — длинное целое число,
`%s` — строка,	`%f` — вещественное число,
`%d` — целое число,	`%lf` — длинное вещественное число.

Пример.

```
printf("Это %c %s %d", 'я', "купил ", 10);
```

Спецификатор `%c` выводит символ 'я', спецификатор `%s` выводит строку, причем концом строки служит первый найденный символ пробела, а `%d` выводит целое число.

Схожим образом работает и функция `scanf`.

```
char str [80];
printf ("Введите строку: ");
scanf ("\%79s",str);
```

Благодаря спецификатору `%s` функция считывает введенные символы и записывает их в массив символов `str`. Как вы заметили, в спецификаторе присутствует число. Это один из вариантов изменения спецификатора, указывающий длину, которую считывает функцию с этим спецификатором. Так, к примеру, `%5d` считает только первые 5 символов из найденного целого числа.

Отдельной группой стоят ***управляющие спецификаторы***. Они позволяют определенным образом форматировать вывод и добавляют определенные символы:

`\a` — звуковой сигнал,	`\v` — вертикальная табуляция,
`\b` — возврат курсора на одну позицию влево,	`\\` — обратная косая черта,
`\f` — переход на новую страницу,	`\'` — одинарная кавычка,
`\n` — переход на новую строку,	`\"` — двойные кавычки,
`\r` — возврат в начало строки,	`\?` — знак вопроса,
`\t` — горизонтальная табуляция,	`%%` — знак %.

Стандартные математические функции

– находятся в библиотечном файле `math.h`;
– аргументы `x` и `y` имеют тип `double`;
– аргументы тригонометрических функций задаются в радианах ($\pi = 180°$). Для того, чтобы посчитать $30°$, в функцию необходимо передать следующее значение: $30 * (\pi/180)$;
– возвращают значение математических функций типа `double`.

Функция	Описание
`sqrt(x)`	корень квадратный из x
`abs(x)`	абсолютное значение x
`exp(x)`	экспоненциальная функция e^x
`pow(x,y)`	x в степени y
`log(x)`	логарифм натуральный x (по основанию e)
`log10(x)`	логарифм десятичный x (по основанию 10)
`sin(x)`	синус x
`cos(x)`	косинус x
`tan(x)`	тангенс x
`asin(x)`	арксинус x
`acos(x)`	арккосинус x
`atan(x)`	арктангенс x
`fmod(x,y)`	остаток от деления x на y
`ceil(x)`	наименьшее целое $\geqslant x$
`floor(x)`	наибольшее целое $\leqslant x$

ПРИМЕРЫ РЕШЕНИЙ

1. Ввести сторону квадрата a. Найти периметр ($P = 4a$) и площадь ($S = a^2$) квадрата.

```
// подключение заголовочного файла
#include <stdio.h>
void main()
{
// определение переменных
double a, P, S;
// ввод данных
printf(" a= ");
scanf("%lf", &a);
// вычисления
P=4*a;
S=a*a;
// вывод информации
printf("P = 4*%.2ldf = %.2lf\n", a, P);
printf("S = %.2ldf * %.2ldf = %.2lf\n", a, a, S);
}
```

2. Дано трехзначное целое число. В нем зачеркнули первую цифру слева и приписали ее справа. Вывести полученное число.

```
// подключение заголовочного файла
#include <stdio.h>
void main()
{
// определение переменных
double n1, n;
// ввод данных
printf(" n= ");
scanf("%d", &n);
// получение нового числа
```

```
n1 = (n%100)*10 + (n/100);
// вывод информации
printf("n1 = %d\n", n1);
}
```

3. Вычислить выражение $z = \dfrac{\sqrt{2b + 2\sqrt{b^2 - 4}}}{\sqrt{b^2 - 4} + b + 2}$.

```
#include <stdio.h>
#include <math.h>
#include <conio.h>
void main()
{
   double z, a, b;
   printf("Введите число не равное -2:   ");
   scanf("%lf", &b);
   a = sqrt( pow(b,2)-4);
   z = sqrt(2*b+2*a)/(a+b+2);
printf("\n Ответ: rezult=%lf\n", z);
printf("\n Press any key... \n");
getch();
}
```

Задачи для выполнения

1. Дан диаметр окружности d. Найти ее длину $L = \pi d$. Взять $\pi = 3.14$.
2. Даны катеты прямоугольного треугольника a и b. Найти его гипотенузу c и периметр P: $c = \sqrt{a^2 + b^2}$, $P = a + b + c$.
3. Дана длина L окружности. Найти радиус R и площадь S круга, ограниченного этой окружностью, учитывая, что $L = 2\pi R$, $S = \pi R^2$. Положить $\pi = 3.14$.
4. Дана площадь S окружности. Найти его диаметр D и длину L окружности, ограничивающей этот круг, учитывая, что $L = 2\pi R$, $S = \pi R^2$. Взять $\pi = 3.1416$.
5. Даны координаты трех вершин треугольника: $(x1, y1)$, $(x2, y2)$, $(x3, y3)$. Найти его периметр и площадь. Для нахождения сторон треугольника a, b, c использовать формулу для расстояния между точками на плоскости. Для нахождения площади треугольника, со сторонами a, b, c, использовать формулу Герона: $S = \sqrt{p(p-a)(p-b)(p-c)}$, где $p = (a + b + c)/2$ — полупериметр.
6. Даны переменные A, B, C. Изменить их значения, переместив содержимое A в B, B в C, C в A, и вывести новые значения переменных A, B, C.
7. Найти значение функции $y = 3x^6 - 6x^2 - 7$ при данном значении x.
8. Дано двузначное целое число. Найти сумму и произведение его цифр.
9. Ввести числа a, b, c, которые являются коэффициентами квадратного уравнения вида: $ax^2 + bx + c = 0$, где $a \neq 0$. Вывести полученное квадратное уравнение. Например, если в переменную a ввели 1, в переменную b ввели 0, а в переменную c ввели 67.45, то на экране результатов должны увидеть следующее:

   «Решаем следующее квадратное уравнение: x^2 + 67.45 = 0»

10. Найти решение системы линейных уравнений вида: $\begin{cases} A_1x + B_1y = C_1, \\ A_2x + B_2y = C_2, \end{cases}$ где A_1, B_1, C_1, A_2, B_2, C_2 — коэффициенты. Если известно, что данная система имеет единственное решение, то можно воспользоваться формулами: $x = (C_1 \cdot B_2 - C_2 \cdot B_1)/(A_1 \cdot B_2 - A_2 \cdot B_1)$, $y = (C_2 \cdot A_1 - C_1 \cdot A_2)/(A_1 \cdot B_2 - A_2 \cdot B_1)$.

11. Известно, что X килограмм шоколадных конфет стоит A рублей, а Y килограмм ирисок стоит B рублей. Определить, сколько стоит один килограмм шоколадных конфет, один килограмм ирисок, а также во сколько раз шоколадные конфеты дороже ирисок.

12. Даны два неотрицательных числа a и b. Найти их среднее геометрическое, т. е. квадратный корень из их произведения.

13. Дано значение угла в градусах $(0, 360°)$. Определить значение этого же угла в радианах, учитывая, что, $180° = \pi$ радиан.

ВАРИАНТЫ ЗАДАНИЙ

Составить программу для расчета значений z_1 и z_2 (результаты должны совпадать).

1. $z_1 = 2\sin^2(3\pi - 2\alpha)\cos^2(5\pi + 2\alpha)$, $z_2 = \big(1 - \sin(5\pi/2 - 8\alpha)\big)/4$.
2. $z_1 = \cos\alpha + \sin\alpha + \cos 3\alpha + \sin 3\alpha$, $z_2 = 2\sqrt{2}\cos\alpha \cdot \sin(\pi/4 + 2\alpha)$.
3. $z_1 = \dfrac{\sin 2\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha + 1 - 2\sin^2 2\alpha}$, $z_2 = 2\sin\alpha$.
4. $z_1 = \dfrac{\sin 2\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha - \cos 3\alpha + \cos 5\alpha}$, $z_2 = \operatorname{tg} 3\alpha$.
5. $z_1 = 1 - \dfrac{1}{4}\sin^2 2\alpha + \cos 2\alpha$, $z_2 = \cos^2\alpha + \cos^4\alpha$.
6. $z_1 = \cos\alpha + \cos 2\alpha + \cos 6\alpha + \cos 7\alpha$, $z_2 = 4\cos\dfrac{\alpha}{2} \cdot \cos\dfrac{5}{2}\alpha \cdot \cos 4\alpha$.
7. $z_1 = \cos^2\left(\dfrac{3\pi}{8} - \dfrac{\alpha}{4}\right) - \cos^2\left(\dfrac{11\pi}{8} + \dfrac{\alpha}{4}\right)$, $z_2 = \dfrac{\sqrt{2}}{2}\sin\dfrac{\alpha}{2}$.
8. $z_1 = \cos^4 x + \sin^2 y + \dfrac{1}{4}\sin^2 2x - 1$, $z_2 = \sin(y + x) \cdot \sin(y - x)$.
9. $z_1 = (\cos\alpha - \cos\beta)^2 - (\sin\alpha - \sin\beta)^2$, $z_2 = -4\sin^2\dfrac{\alpha - \beta}{2} \cdot \cos(\alpha + \beta)$.
10. $z_1 = \dfrac{\sin(\pi/2 + 3\alpha)}{1 - \sin(3\alpha - \pi)}$, $z_2 = \operatorname{ctg}\left(\dfrac{5\pi}{4} + \dfrac{3\alpha}{2}\right)$.
11. $z_1 = \dfrac{1 - 2\sin^2\alpha}{1 + \sin 2\alpha}$, $z_2 = \dfrac{1 - \operatorname{tg}\alpha}{1 + \operatorname{tg}\alpha}$.
12. $z_1 = \dfrac{\sin 4\alpha}{1 + \cos 4\alpha} \cdot \dfrac{\cos 2\alpha}{1 + \cos 2\alpha}$, $z_2 = \operatorname{ctg}(3\pi/2 - \alpha)$.
13. $z_1 = \dfrac{\sin\alpha + \cos(2\beta - \alpha)}{\cos\alpha - \sin(2\beta - \alpha)}$, $z_2 = \dfrac{1 + \sin 2\beta}{\cos 2\beta}$.
14. $z_1 = \dfrac{(m-1)\sqrt{m} - (n-1)\sqrt{n}}{\sqrt{m^3 n} + nm + m^2 - m}$, $z_2 = \dfrac{\sqrt{m} - \sqrt{n}}{m}$.
15. $z_1 = \dfrac{x^2 + 2x - 3 + (x+1)\sqrt{x^2 - 9}}{x^2 + 2x - 3 + (x-1)\sqrt{x^2 - 9}}$, $z_2 = \sqrt{\dfrac{x+3}{x-3}}$.