Программирование линейных алгоритмов

Лекция №4 по курсу «ОАИП»

Линейный - алгоритм, в котором все указанные действия выполняются один раз в том порядке, в котором они записаны.

Ввод в консоли

Для ввода данных в консоли может использоваться функция **scanf()**. Эта функция определена в заголовочном файле **stdio.h**

int scanf(форматная строка, список аргументов)

scanf() возвращает число, равное значений, которые были действительно присвоены переменным. В это количество не входят значения, которые были считаны, но их значения не были ничему присвоены вследствие использования модификатора * для подавления присваивания. Если до присвоения значения первого поля произошла ошибка, возвращается EOF.

Форматная строка содержит спецификации преобразования, которые определяют вводимые данные.

Общий вид спецификаций преобразования:

% * ширина_поля модификатор спецификатор

Знак процента % и спецификатор обязательны, остальные элементы – нет.

Символ * позволяет пропустить при вводе водимые символы для типа, указанного через спецификатор.

Ширина_поля представляет целое положительное число, которое позволяет определить, какое количество байтов будет учитываться при вводе.

Функция scanf должна считать текст из консоли, преобразовать его в данные нужного типа и разместить их **в соответствующие ячейки памяти.** *Поэтому аргументы функции scanf должны быть указателями на соответствующие переменные.*

Код	Значение
%c	Считать один символ
%d	Считать десятичное число целого типа
%i	Считать десятичное число целого типа
%e	Считать число с плавающей запятой
%f	Считать число с плавающей запятой
%g	Считать число с плавающей запятой
%o	Считать восьмеричное число
%s	Считать строку
%x	Считать шестнадцатиричное число
%p	Считать указатель
%n	Принимает целое значение, равное количеству считанных до текущего момента символов
%u	Считывает беззнаковое целое
%[]	Просматривает набор символов
%%	Считывает символ %

Модификаторы позволяют конкретизировать тип данных. В частности, есть следующие модификаторы:

h: для ввода значений типа **short int** (%hd)

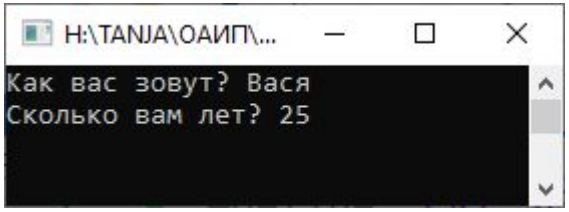
l: для ввода значений типа **long int** (%ld) или **double** (%lf, %le)

L: для ввода значений типа **long double** (%Lf, %Le)

В качестве аргументов в функцию **scanf()** передаются **адреса переменных**, которые нужно вводить.

Для получения адреса переменной перед ее именем ставится знак амперсанда **&**.

```
int age;
char name[10];
printf("Как вас зовут? ");
scanf("%10s", &name);
printf("Сколько вам лет? ");
scanf("%d", &age);
```



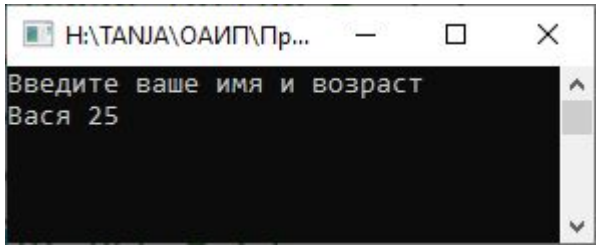
Функция scanf( ) является функцией незащищенного ввода, т.к. появилась она в ранних версиях языка Си. Поэтому, чтобы разрешить работу данной функции в современных компиляторах, необходимо в начало программы добавить строчку

```
#define _CRT_SECURE_NO_WARNINGS
```

Можно сразу вводить несколько значений. В этом случае в качестве разделителя используется пробел.

```
int age;
char name[10];

printf("Введите ваше имя и возраст\n");
scanf("%s %d", &name, &age);
```



Если необходимо ввести значение строковой переменной, то перед ее именем ставить символ & не обязательно.

Математические функции

Описание математических функций находятся в библиотеках **<stdlib.h>, <math.h> и <cmath>**

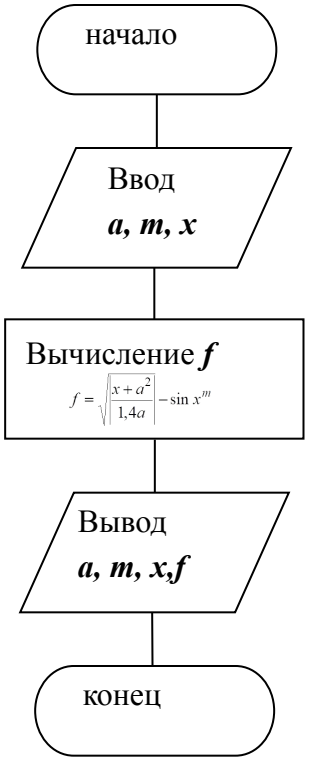
Имя	Описание
abs	Возвращает абсолютную величину целого числа
acos	арккосинус
asin	арксинус
atan	арктангенс
atan2	арктангенс с двумя параметрами
ceil	округление до ближайшего большего целого числа
cos	косинус
exp	вычисление экспоненты
fabs	абсолютная величина (числа с плавающей точкой)
floor	округление до ближайшего меньшего целого числа

Имя	Описание
fmod	вычисление остатка от деления нацело для чисел с плавающей точкой
frexp	разбивает число с плавающей точкой на мантиссу и показатель степени.
ldexp	умножение числа с плавающей точкой на целую степень двух
log	натуральный логарифм
log10	логарифм по основанию 10
modf(*x*,*p*)	извлекает целую и дробную части (с учетом знака) из числа с плавающей точкой
pow(*x*,*y*)	результат возведения *x* в степень *y*, x^y
sin	синус
sinh	гиперболический синус
cosh	гиперболический косинус
sqrt	квадратный корень
tan	тангенс
tanh	гиперболический тангенс

По умолчанию значения функций и их аргументы имеют тип **double**. Вещественные литералы также по умолчанию double.

Пример 1. Разработать консольное приложение для вычисления значения функции

$$f = \sqrt{\left|\frac{x + a^2}{1{,}4a}\right|} - \sin x^m$$

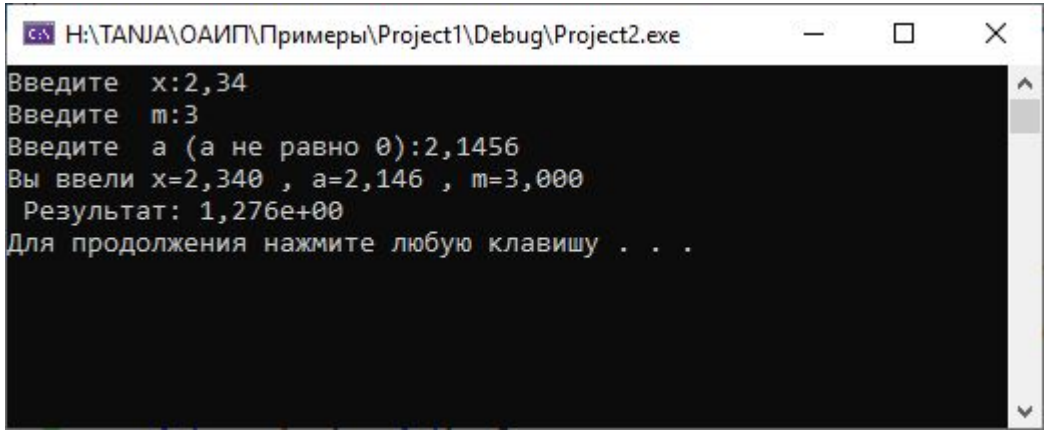

начало
Ввод
a, m, x
Вычисление f
$f = \sqrt{\left|\frac{x+a^2}{1,4a}\right|} - \sin x^m$
Вывод
a, m, x,f
конец

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <math.h>  // подключаем математические функции
#include <locale.h> // подключаем библиотеку локализации
void main(void) {
    setlocale(LC_ALL, "Russian");// устанавливаем русскую локаль
    double x,a,m,f;   // декларируем необходимые переменные

    //ввод переменных с клавиатуры
    printf("Введите  x:");
    scanf("%lf",&x);
    printf("Введите  m:");
    scanf("%lf",&m);
```

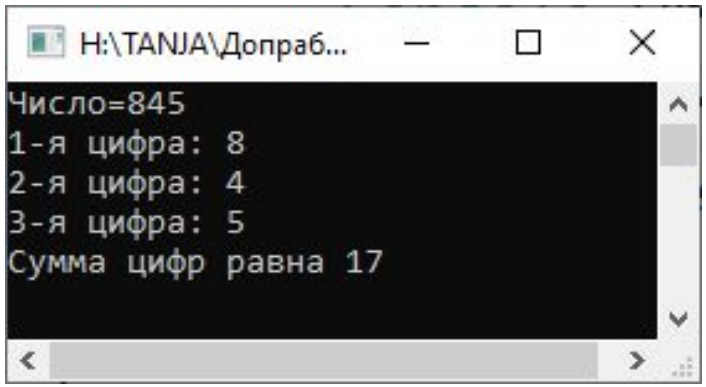
```
   printf("Введите  a (a не равно 0):");
    scanf("%lf", &a);
    //рассчитываем результат
    f=sqrt(fabs(x+a*a)/(1.4*a))-sin(pow(x,m));

    //вывод результата на экран
  printf("Вы ввели x=%.3f ,a=%.3f , m=%.3f\n Результат:%5.3e\n",
x,a,m,f);
    system("pause");
}
```



Пример 2. Написать программу, которая позволяет ввести целое трехзначное число, выводит его на экран, находит сумму цифр введенного числа и выводит эту сумму.

Пример вывода:



```
int number;
    //ввод переменных с клавиатуры
    printf("Введите  целое трехзначное число: ");
    scanf("%d", &number);
    printf("Число=%d\n",number);

    int digit3 = number % 10;
    number = number / 10;
    int digit2 = number % 10;
    int digit1 = number / 10;

    printf("1-я цифра: %d\n", digit1);
    printf("2-я цифра: %d\n", digit2);
    printf("3-я цифра: %d\n", digit3);
    printf("Сумма цифр равна\n", digit1+digit2+digit3);
```

Генерирование псевдослучайных чисел

Задача генерации **случайных** чисел на классическом процессоре не может быть решена, так как работа компьютера детерминирована по определению. Но можно сгенерировать очень длинные наборы чисел такие, что их распределение будет иметь те же свойства, что и наборы истинно случайных чисел.

Сначала необходимо инициализировать генератор случайных чисел (ГСЧ, или RNG - random number generator), задать начальное значение, на основе которого в дальнейшем будет происходить генерация. Для этого используем функцию

void srand(unsigned *seed*);

Функция srand() устанавливает исходное число для последовательности, генерируемой функцией **rand()**.

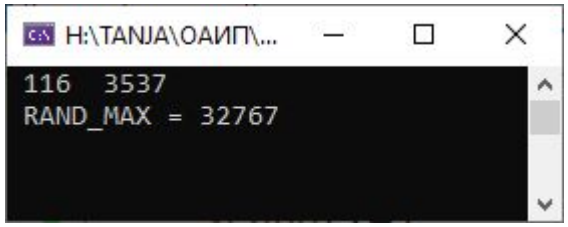
Важно, что для одного и того же начального значения генератор будет возвращать одни и те же числа.

Функция

int rand(void)

генерирует последовательность псевдослучайных чисел. При каждом обращении к функции возвращается целое в интервале между нулем и значением **RAND_MAX**, которое в любой реализации должно быть не меньше числа 32 767.

```
srand(24);
int num1 = rand();
int num2 = rand();
printf(" %d  %d  \n", num1, num2);
printf(" RAND_MAX = %d\n", RAND_MAX);
```



Каждый раз при новом запуске результат будет тем же.

Для того, чтобы при следующем запуске получить новый набор чисел, нужно инициализировать генератор *каждый раз разными значениями*.

Для этого можно использовать системное время:

```
srand(time(NULL));
```

Для работы с системной датой и временем нужно подключить заголовочный файл **time.h.**

Для определения текущего календарного времени используется функция

time_t time(NULL);

Данная функция возвращает время в секундах начиная с 1 января 1970 г.

Получение случайного числа из диапазона от нуля до единицы:

```
double double_number = rand() / (double)RAND_MAX;
```

Для получения числа в отрезке от нуля до N нужно умножить N на случайное число от нуля до единицы:

```
double double_number =n* rand() / (double)RAND_MAX;
```

```
int int_number = n * rand() / RAND_MAX;
```

Для получения случайного числа в заданном диапазоне:

```
int int_number = rand() % (max - min + 1) + min;
```

```
double double_number =min+(max-min)*(rand() / (double)RAND_MAX);
```