

Основы программирования

Лабораторная работа 3

Тема: Циклы и случайные числа

Цель лабораторной работы

Целями данной работы являются изучение операторов цикла и обучение написанию программ с циклической последовательностью действий с использованием генерации случайных чисел.

Краткие теоретические сведения

Операторы цикла

Оператор `while`

Данный оператор реализует **цикл с предусловием**. Общий вид:

```
while (условие) действие;
```

Действие цикла выполняется повторно до тех пор, пока значение *условие* истинно. Условие считается *истинным*, если значение выражения не равно 0 (как и в *условных операторах*).

Если условие сразу оказывается *ложным*, то действие не выполняется ни разу.

Пример цикла:

```
int i = 0;
while(i<10){
    printf("%d ", i);
    i++;
}
```

```
0 1 2 3 4 5 6 7 8 9
```

Если нужно повторно выполнять несколько действий, то можно заключить их в фигурные скобки:

```
while (условие)
{
    действие1;
    действие2;
    ...
}
```

К примеру:

```
int a = 0; // чтобы условие по началу точно было истинным
while(a>=0){
    scanf_s("%d", &a);
    printf("Input: %d\n", a);
}
```

```
10
Input: 10
1
Input: 1
-5
Input: -5
```

Оператор do-while

Оператор реализует **цикл с постусловием**. Если требуется выполнить *тело* цикла, как минимум один раз до проверки *условия продолжения*, используют оператор **do-while**:

```
do
    действие;
while (условие);
```

Данный цикл по своему механизму работы практически совпадает с циклом *while*. Он также выполняется пока значение условия *истинно*. Если нужно повторно выполнять несколько действий, то их также можно заключить в фигурные скобки:

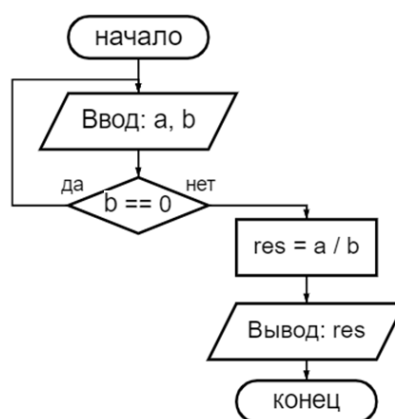
```
do{
    действие1;
    действие2;
    ...
} while(условие);
```

Перепишем пример со считыванием и выводом числа на экран с циклом *do-while*:

```
int a;
do {
    scanf_s("%d", &a);
    printf("Input: %d\n", a);
} while(a>=0);
```

Цикл *do-while* также можно использовать для проверки введенных пользователем значений. Если пользователь вводит неправильное значение, то программа, проверив это, может повторно запросить ввод.

К примеру, пользователю необходимо ввести *два числа*, и программа должна вычислить результат *деления одного на другое*. Однако **делить на 0 нельзя**. Поэтому с помощью цикла с постусловием можно проверить вводимые пользователем значения после считывания. Алгоритм программы будет таким:



Код программы:

```
double a, b;
do {
    cout << "Введите 2 числа: ";
    cin >> a >> b;
} while(b == 0);

double res = a / b;
cout << "Результат: " << res;
```

Оператор for

Помимо простых операторов циклов существует и специальный оператор цикла с известным числом повторений – цикл **for**. Общая форма оператора:

```
for (инициализация; условие; приращение) действие;
```

В скобках данного цикла есть три блока, которые обеспечивают его работу:

- **Инициализация** – здесь задаются начальные значения **перед** началом работы цикла, к примеру, здесь объявляются и инициализируются счетчики цикла.
- **Условие** – условие продолжения цикла.
- **Приращение** – этот блок отрабатывает **после** каждой *итерации* цикла, к примеру, изменяет значение счетчика цикла.

Пример цикла:

```
int summa=0;
for(int i = 1; i <= 100; i++){
    if (i % 2 == 0){
        summa = summa + i;
        printf("%d ", summa);
    }
}
```

```
2 6 12 20 30
```

Цикл *for* имеет множество других вариантов использования.

Использование нескольких счетчиков цикла

Например, используем два счетчика одновременно:

```
for(int i=0, j=0; i + j <= 10; i++)
{
    cin >> j;
    cout << i << ": sum = " << i+j << endl;
}
```

```
2
0: sum = 2
3
1: sum = 4
8
2: sum = 10
```

Операторы инициализации счетчиков разделены **запятой**, то есть они выполняются последовательно.

Пропуск разделов цикла

Так как каждый из разделов цикла является *необязательным*, то их можно пропускать.

К примеру, данный цикл выполняется до тех пор, пока пользователь не введет число 100:

```
int i;
for(i = 1; i != 100; ) scanf_s("%d ", &i);
```

Таким образом, можно легко сделать *бесконечный* цикл, не задав условного выражения:

```
for( ; ; ) printf("бесконечный цикл\n");
```

```
бесконечный цикл
бесконечный цикл
бесконечный цикл
бесконечный цикл
```

Бесконечный цикл можно организовать и с помощью циклов *while* и *do-while*. Нужно просто сделать, чтобы их условие продолжения всегда было истинным:

```
while( true ) printf("бесконечный цикл\n");
```

Вложенные циклы

Очень часто в программе требуется организовать выполнение двух или более вложенных циклов. К примеру, имеем:

```
for(int i=0; i < 5; i++){
    for(int j=0; j < 5; j++)
        printf("%3d", j + i*5);
    printf("\n");
}
```

```
 0  1  2  3  4
 5  6  7  8  9
10 11 12 13 14
15 16 17 18 19
20 21 22 23 24
```

Операторы безусловного перехода

Оператор break

Используя оператор **break**, можно вызвать немедленное **завершение** цикла, пропуская *условие продолжения* и другой код в *теле* цикла. Когда приложение встречает оператор *break* внутри цикла, оно сразу прекращает выполнение цикла и переходит к оператору, следующему за циклом. Пример:

```
int i = 0, sum = 0, value;
while(i < 10){
    i++;
    cin >> value;
    if(value == 0) break; //цикл завершится, если value равно 0
    sum += value;
}
cout << sum;
```

Оператор continue

Иногда требуется, чтобы прервать текущую *итерацию* цикла и перейти сразу к следующей. В циклах это делается с помощью оператора **continue**. Пример:

```
for(int i=0; i <= 10; i++){
    printf("%d ", i);
    if( i%2 ) continue;
    printf("(%d) ", i*i);
}
```

```
0 (0) 1 2 (4) 3 4 (16) 5 6 (36) 7 8 (64) 9 10 (100)
```

Генератор случайных чисел

В языке C++ можно генерировать случайное число (точнее псевдослучайное). Для этого используется функция **rand()**. Она находится в библиотеке *stdlib.h* (*cstdlib*). На место вызова *rand()* будет подставлено случайное число в диапазоне от 0 до 32 767.

```
int r = rand();
```

[0, 32 767]

Также потребуется функция **srand()**, чтобы установить начальную точку для генерации случайных чисел. Эту команду следует прописывать где-нибудь в начале программы. К примеру, в начале функции *main()*.

Но ее необходимо использовать в комбинации с функцией **time()** (библиотека *time.h*). Если вызвать эту функцию с аргументом NULL, то она вернет текущее время в секундах, что позволит использовать разные числа для инициализации *srand()* при повтором запуске программы.

Пример:

```
srand(time(NULL));  
int r = rand();
```

Управление диапазоном генерации случайного числа

Можно регулировать диапазон случайных чисел с помощью простых арифметических операций, т.к. мы знаем изначальный диапазон:

начало_отсчета + rand() % диапазон

К примеру:

```
int a = 10 + rand() % 40;
```

```
int a = 10 + rand() % 40;
```

Число [0, 39]

[0 + 10, 39 + 10]
[10, 49]

В переменной *a* окажется случайное число от 10 до 49.

Задание

Разработать программу для каждой задачи из **общего** и **индивидуального** задания.

Для решения задач необходимо использовать операторы **цикла** и **ветвления**, а также функции для генерации случайных чисел.

Также можно выполнить **дополнительное** задание, чтобы получить больше баллов.

Методические указания

Выполните **общие** и **индивидуальные** задания.

При выполнении **общих** заданий не требуется отчет, но все еще нужна защита с объяснением кода программы.

При выполнении **индивидуальных** заданий необходимо выполнить следующие этапы для каждой задачи отдельно:

- 1) словесная постановка задачи;
- 2) анализ задачи и формальная постановка задачи;
- 3) проектирование (разработка алгоритма) – лучше всего в графической форме (блок-схема), но можно и в словесной форме или псевдокод;
- 4) реализация (кодирование, отладка);
- 5) тестирование.

Результаты выполнения **индивидуальных** заданий оформить в виде отчета.

Общее задание

Пользователь с клавиатуры вводит числа. Посчитать сумму этих чисел. Завершить ввод, когда пользователь введет число 0. После завершения вывести посчитанную сумму.

```
Введите число: 5
Введите число: -1
Введите число: 7
Введите число: 0
-----
Сумма: 11
```

Варианты индивидуальных заданий

Вариант 1

Используя цикл, напишите программу, которая позволяет бесконечно вводить с консоли числа, сразу определяет, является ли число **простым**.

Простое число – это число, которое делится без остатка только на 1 и на само себя.

Вариант 2

Используя цикл, напишите программу, которая позволяет бесконечно вводить с консоли числа, сразу вычислять их факториал и выводить его на консоль.

Вариант 3

Используя цикл, напишите программу, которая позволяет бесконечно вводить с консоли числа, сразу вычислять **сумму их цифр** и выводить ее на консоль.

Вариант 4

Нарисовать равнобедренный треугольник из символов ^. Высоту выбирает пользователь. Например: высота = 5, на экране :



Вариант 5

Найдите все четырехзначные числа, сумма цифр каждого из которых равна X.

Число X – введенное пользователем число, которое может быть от 1 до 36 (больше 36 сумма цифр четырехзначного числа быть не может). Проверяйте правильность ввода.

Вариант 6

С помощью цикла найдите количество *четных* и *нечетных* (отдельно) цифр введенного натурального числа.

Вариант 7

Напишите программу, которая бесконечно в цикле считывает число K и символ C (+ - / *) и сразу выполняет соответствующую операцию над числом S (S изначально = 0) и выводит результат. К примеру: ввели «3 +» – $S=0+3=3$, на экран выводится «3». Далее ввели «2 -» – $S=3-2=1$, на экран выводится «1». И так далее.

Вариант 8

Найдите все **простые** числа, лежащие между двумя введенными числами A и B.

Вариант 9

Организуйте непрерывный ввод чисел с клавиатуры, пока пользователь не введёт 0. После ввода нуля, определить, больше было введено четных или нечетных чисел.

Вариант 10

Найдите **сумму цифр** всех чисел, лежащих между двумя введенными числами A и B.

Вариант 11

Нарисуйте квадрат (выглядит в консоли как прямоугольник) из символов ‘\’ и ‘/’. При этом квадрат поделен по диагонали и в верхней части выводится символ ‘\’, а в нижней – ‘/’. Длину стороны выбирает пользователь. Например: длина = 5:



Вариант 12

Напишите программу, которая бесконечно в цикле считывает целое число K и символ C и сразу выводит K раз символ C . Если введено отрицательное число K , то выводить символы через пробел. Если число $K = 0$, то завершить цикл.

Вариант 13

Из десяти вводимых последовательно целых чисел (без использования массива) найдите то число, сумма цифр, которого **максимальная**.

Вариант 14

Пользователь вводит число K . С помощью цикла выведите на экран символ «*» в K строк, при этом количество символов в строке должно быть равно $2K$. Упорядочивать символы необходимо по правому краю. Например: введено число 4, на экране:

```
  **
 ***
****
*****
*****
```

Дополнительное задание

Запрограммировать игру «Чет/Нечет».

У пользователя в начале игры имеется 100 очков.

В течение игры каждый раз повторяется следующая последовательность действий:

1. Игрок выбирает, сколько очков ставит на кон.
2. Программа имитирует бросок двух игральных костей (2 случайных числа от 1 до 6).
3. Пользователь пытается угадать, *четная* или *нечетная* сумма выпала на костях. Если он угадал, то его ставка удваивается. Если нет, то он теряет ставку.

Игра продолжается пока либо пользователь не заработает 500 очков (**победа**), либо не потеряет все очки (**поражение**).