Циклы в C++

С другой стороны, мы не можем игнорировать эффективность

- Джон Бентли

Цикл типа «n-раз» (оператор for)

Оператор for содержит три параметра.

- Первый называется инициализацией,
- второй — условием повторения,
- третий — итерацией.

```
for (инициализация; условие; итерация)
    {
        //тело цикла, т. е. действия повторяемые циклично
    }
```

Цикл типа «n-раз» (оператор for)

Перед первым шагом цикла счётчику присваивается начальное значение (выполняется инициализация). Это происходит лишь однажды.

Представленная программа выводит на экран числа от 1 до 100:

```
for (int i = 1; i <= 100; i++)
    {
        cout << i << " ";
    }
```

Цикл типа «n-раз» (оператор for)

После завершения каждого шага цикла и перед началом следующего (и, значит, перед проверкой условия повторения) выполняется итерация.

```
for (int i = 1; i <= 100; i++)
    {
        cout << i << " ";
    }
```

Цикл типа «n-раз» (оператор for)

Представленная программа выводит на экран числа от 10 до −10:

```
for (int s = 10; s > -11; s--)
    {
        cout << s << " ";
    }
```

Представленная программа выводит на экран нечётные числа от 1 до 33:

```
for (int i = 1; i <= 33; i = i + 2)
    {
        cout << i << " ";
    }
```

Цикл типа «n-раз» (оператор for)

Представленная программа вычислит сумму элемнтов фрагмента последовательности 2, 4, 6, 8,… 98, 100. Итак:

```
int sum = 0; // Сюда будем накапливать результат
for (int j = 2; j <= 100; j=j+2) {
  sum = sum + j;
}
cout << sum;
```

Цикл типа «n-раз» (оператор for)

Представленная программа будет возводить число из переменной a в натуральную степень из переменной n:

```
double a = 2;
int n = 10;
double res = 1; // Сюда будем накапливать результат
for (int i = 1; i <= n; i++) {
  res = res * a;
}
cout << res;
```

Цикл типа «n-раз» (оператор for)

Представленная программа выведет на экран 10 первых элементов последовательности 2n+2, где n=1, 2, 3…:

```
for (int i = 1; i < 11; i++)
  {
     cout << 2*i + 2 << " "
  }
```

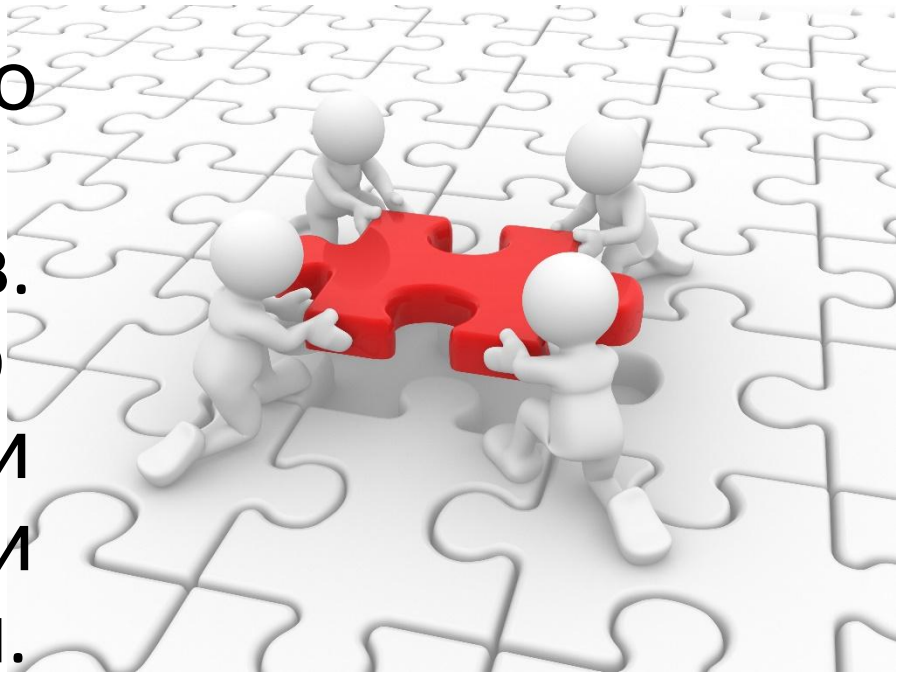
Цикл типа «n-раз» (оператор for)

Представленная программа выведет на экран 10 первых элементов последовательности $2a_{n-1}+3$, где $a_1=3$:

```
int a = 3;
for (i=1; i<=10;i++)
    {
        cout << a << " ";
        a = 2*a + 3;
    }
```

Цикл типа «n-раз» (оператор for)

В одном цикле можно задавать сразу несколько счётчиков. При этом несколько выражений в итерации и в инициализации разделяются запятыми.



Условие повторения можно задавать только одно, но оно может быть выражением, содержащим сразу несколько счётчиков.

Цикл типа «n-раз» (оператор for)

Представленная программа выведет на экран 10 первых элементов последовательности $2a_{n-1}-2$, где $a_1=3$:

```
for (int a=3, i=1; i<=10; a=2*a-2, i++)
    {
        cout << a << " ";
    }
```

Цикл типа «n-раз» (оператор for)

Представленная программа выведет на экран такую последовательность «0 -1 -4 -9 -16 -25»:

```
for (int a=0, b=0; a-b<=10; a++, b--)
    {
        cout << a*b << " ";
    }
```

Досрочное завершение цикла (оператор break)

Как цикл типа «пока» так и цикл типа «n-раз» можно завершить досрочно, если внутри тела цикла вызвать оператор **break**.

При этом произойдёт моментальный выход из цикла, не будет закончен даже текущий шаг (т. е. если после break присутствовали какие-то ещё операторы, то они не выполнятся).

Досрочное завершение цикла (оператор break)

В результате работы следующего примера на экран будут выведены только числа «1 2 3 4 Конец»:

```
for (int a=1; a<=10; a++) {
  if(a == 5) {
    break;
  }
  cout << a << " ";
}
cout << "Конец";
```

Досрочное завершение цикла (оператор break)

С помощью оператор break можно прервать заведомо бесконечный цикл. Пример (на экран выведется «100 50 25 12 6 3 1 0 » и после этого цикл остановится):

```
int s = 100;
while (true) {
  cout << s << " ";
  s = s / 2;
  if(s == 0) {
    break;
  }
}
```

Досрочное завершение цикла (оператор break)

Оператор break имеет смысл вызывать только при наступлении какого-то условия, иначе цикл будет завершен досрочно на первом же своём шаге.

```
int a;
for (a=25; a>0; a--)
 {
   break;
   cout << a << " ";
 }
cout << "a=" << a;
```

Задачи

- Создайте программу, выводящую на экран все четырёхзначные числа последовательности 1000 1003 1006 1009 1012 1015 ….
- Создайте программу, выводящую на экран первые 55 элементов последовательности 1 3 5 7 9 11 13 15 17 ….
- Создайте программу, выводящую на экран все неотрицательные элементы последовательности 90 85 80 75 70 65 60 ….
- Создайте программу, выводящую на экран первые 20 элементов последовательности 2 4 8 16 32 64 128 ….

Задачи

- Выведите на экран все члены последовательности $2a_{n-1}-1$, где $a_1=2$, которые меньше 10000.
- Выведите на экран все двузначные члены последовательности $2a_{n-1}+200$, где $a_1=-166$.
- Создайте программу, вычисляющую факториал натурального числа n, которое пользователь введёт с клавиатуры.

Задачи

- Выведите на экран все положительные делители натурального числа, введённого пользователем с клавиатуры.

Проверьте, является ли введённое пользователем с клавиатуры натуральное число — простым. Постарайтесь не выполнять лишних действий (например, после того, как вы нашли хотя бы один нетривиальный делитель уже ясно, что число составное и проверку продолжать не нужно). Также учтите, что наименьший делитель натурального числа n, если он вообще имеется, обязательно располагается в отрезке $[2; \sqrt{n}]$.

Задачи

- Для введённого пользователем с клавиатуры натурального числа посчитайте сумму всех его цифр (заранее не известно сколько цифр будет в числе).

- Пользователь вводит с клавиатуры последовательность ненулевых целых чисел. Программа должна вывести на экран максимальный и минимальный элементы последовательности сразу после того, как пользователь введёт 0 (т. е. заранее длина последовательности неизвестна).

Задачи

- Пользователь вводит с клавиатуры арифметический пример в таком формате «2+3.5» или «3.14*8», программа должна вычислить и вывести правильный ответ на экран. В примере должны быть допустимы операции сложения, умножения, вычитания, деления (с остатком). После вывода ответа программа должна спросить пользователя, требуется ли решить другой пример? Если пользователь введёт «y» программа должна запуститься повторно, иначе — завершиться.

Цикл типа «пока» (оператор while)

Оператор **while** повторяет указанные действия до тех пор, пока его параметр имеет истинное значение.

Например, такой цикл выполнится 4 раза, а на экран будет выведено «1 2 3 4 »:

```
int i = 1;
while (i < 5)
  {
    i++;
    cout << i << " ";
  }
```

Цикл типа «пока» (оператор while)

Такой цикл не выполнится ни разу и на экран ничего не выведется:

```
int i = 1;
  while (i < 0)
     {
          i++;
          cout << i << " ";
     }
```

Цикл типа «пока» (оператор while)

Такой цикл будет выполняться бесконечно, а на экран выведется «1 2 3 4 5 6 7 …»:

```
int i = 1;
while (true)
  {
    i++;
    cout << i << " ";
  }
```

Цикл типа «пока» (оператор do…while)

Бывает цикл типа «пока» с постпроверкой условия. Для его записи используется конструкция из операторов do…while.

Такой цикл выполнится 4 раза, а на экран будет выведено «2 3 4 5 »:

```
int i = 1;
do {
   i++;
   cout << i << " ";
} while (i < 5);
```

Цикл типа «пока» (оператор do…while)

Такой цикл выполнится 1 раз, а на экран будет выведено «2 »:

```
int i = 1;
do {
    i++;
    cout << i << " ";
} while (i < 0);
```

Задачи

- В американской армии считается несчастливым число 13, а в японской — 4. Перед международными учениями штаб российской армии решил исключить номера боевой техники, содержащие числа 4 или 13 (например, 40123, 13313, 12345 или 13040), чтобы не смущать иностранных коллег. Если в распоряжении армии имеется 100 тыс. единиц боевой техники и каждая боевая машина имеет номер от 00001 до 99999, то сколько всего номеров придётся исключить?