Лекция 5

ЦИКЛИЧЕСКИЕ ВЫЧИСЛИТЕЛЬНЫЕ ПРОЦЕССЫ В VISUAL C++ 2010. ОПЕРАТОР ЦИКЛА FOR.

Цель работы: изучение вопросов разработки циклических программ с использованием оператора цикла **for**.

Вопросы лекции:

1. Циклические вычислительные процессы.
2. Оператор цикла **for** в среде Visual C++ 2010.
3. Операции инкремента и декремента.
4. Разработка программ с использованием оператора цикла **for**.
5. Вложенные циклы **for** в Visual C++ 2010.

1. Циклические вычислительные процессы

В предыдущей лекции были рассмотрены циклические вычислительные процессы в Visual C++ 2010 и реализующие их операторы **while** и **do...while**. В данной лекции будет изучена разработка циклических программ с использованием оператора цикла **for**. Кратко напомним основные понятия циклических вычислительных процессов.

Циклическим называется вычислительный процесс, содержащий многократные вычисления по одним и тем же математическим зависимостям.

Цикл выполняет оператор или группу операторов до тех пор, пока истинно (или ложно) определенное условие относительно некоторой переменной, называемой **параметром цикла**.

Многократно повторяющиеся части такого процесса составляют **тело цикла**.

Алгоритм циклических структур должен содержать (рис. 7.1):

1. **Подготовку к циклу** – присваивание начального значения параметру цикла.
2. **Проверку условия** выполнения тела цикла.
3. **Тело цикла** – действия, которые выполняются в циклической программе для разных значений параметра цикла.
4. **Изменение (модификация) значений параметра цикла**.

На рис. 1 изображена блок-схема алгоритма циклического вычислительного процесса, где помимо характеристик операционных блоков в качестве примера приведены реальные операторы.

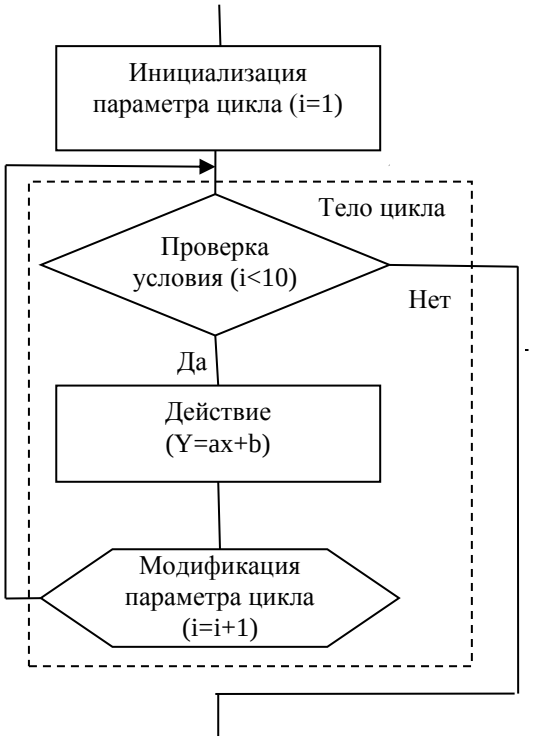


Рис. 1. Блок-схема алгоритма циклического вычислительного процесса

В среде Visual C++ 2010 циклические вычислительные процессы реализуются с помощью операторов **while**, **do...while** и **for**. Выше были исследованы операторы **while** и **do...while**. В настоящей

работе будут исследованы программы с использованием оператора **for**.

2. Оператор цикла for

Оператор цикла **for** используется, когда количество повторений тела цикла **заранее известно**. Форма записи оператора цикла **for** следующая:

for ([выражение инициализации]; [выражение проверки (условие)]; [выражение модификации])
оператор внутри цикла;

Квадратные скобки показывают, что данная секция в операторе может быть опущена.

На практике это выглядит, например, следующим образом:

```
for(i=1; i<=n; i=i+1)
Y =A*i;
```

где **i** – параметр цикла.

Анализ данной записи показывает, что оператор **for** объединяет в себе три операционных блока из блок-схемы циклического вычислительного процесса (рис. 7.1):

- блок инициализации, т. е. присвоения параметру цикла начального значения (**i=1**);
- блок проверки условия (**i<=n**);
- блок модификации параметра цикла (**i=i+1**).

Это свойство оператора цикла **for** позволяет существенно упростить вычислительные процессы и программные коды при решении различных задач в Visual C++ 2010.

В схемах алгоритмов оператор цикла **for** отражается символом **модификация** (рис. 2):

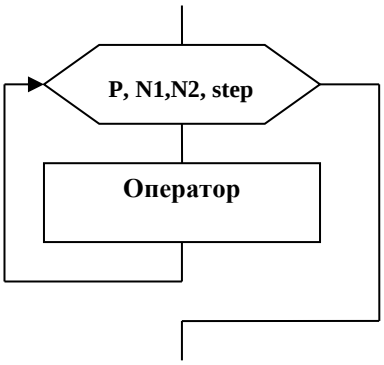


Рис. 2. Блок-схема оператора цикла **for**

На схеме алгоритма приведены следующие обозначения:

- **P** – параметр цикла;
- **N1, N2** – границы изменения параметра цикла;
- **step** – шаг изменения параметра цикла (если шаг не указан, то он равняется 1) .

Параметр цикла **P**, границы изменения **N1**, **N2** и шаг **step** должны иметь один и тот же тип.

Возможности оператора цикла **for** очень велики. Например, вместо любого из трех выражений в записи общей формы можно записать два и более выражения, разделенных запятыми:

```
for(i=1, j=1, z=1; i<=n; i=i+1, j=j+1, z=z+1)
Y =A*i*j*z;
```

Особенностью оператора цикла **for** является то, что параметр цикла **P**, границы изменения **N1**, **N2** и шаг **step** могут отсутствовать в записи оператора (знаки “**;**” должны присутствовать).

Например, если пропущено условие, то цикл будет выполняться бесконечно. Приведем три примера бесконечных циклов:

```
for (i=0; ; i++) cout<<” Бесконечный цикл ” << endl;
for (i=1; 1; i++) cout<<” Бесконечный цикл ” << endl;
for (; ; ) cout<<” Бесконечный цикл ” << endl;
```

3. Операции инкремента и декремента

В языке С++ принято операцию приращения цикла **i=i+1** записывать как **i++**, например:

```
for(i=1; i<=n; i++)
Y =A*i;
```

Такая запись называется операцией инкремента. Если **i=i-1**, то записывают **i--** и называют операцией декремента.

Эти операции свойственны только языку С++. Унарные (одиночные, одноместные) операции инкремента (декремента) увеличивают (уменьшают) свой операнд (обязательно переменную) на единицу. Они изменяют значение самой переменной, то есть являются скрытой формой операции присваивания. Иногда эти операции применяют как самостоятельный оператор:

```
i++; i--;
```

что эквивалентно записи

```
i = i +1; i = i - 1;
```

Но эти операции могут использоваться и в выражениях, например,

```
sum = sum +x *++i ;
```

Инкремент и декремент реализуются в двух формах: префиксной и постфиксной. Особенности выполнения этих форм следующие.

При префиксной форме **++i** (**--i**) - переменная **i** сначала увеличивается (уменьшается) на единицу , а затем ее значение используется в выражении.

При постфиксной форме **i++** (**i--**) – значение переменной **i** сначала используется в выражении и только после вычисления выражения переменная увеличивается (уменьшается) на единицу.

4. Разработка программ с использованием оператора цикла for

Рассмотрим простейший пример оператора цикла **for.** Необходимо исследовать программу для печати в столбец чисел от 1 до заданного числа **n**=15 с шагом 1. Использовать оператор **for**. Блок-схема алгоритма выполнения такого задания приведена на рис. 3.

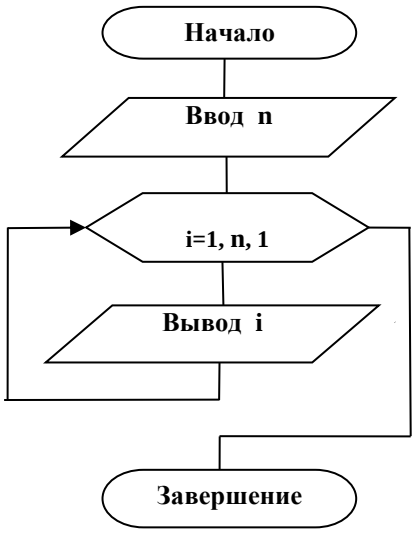


Рис. 3. Блок-схема алгоритма для печати в столбец **n** чисел

Ниже приведен программный код, реализующий данный алгоритм:

```
int _tmain(int argc, _TCHAR* argv[])
{
  int i, n ;
  cout<<"Vvedite n "<<endl;
```

```
  cin>>n;                          //Ввод значения переменной  n
        for ( i=1; i< n; i++ )     //Оператор цикла  for
        cout << i <<endl;          //Тело цикла – вывод параметра цикла i
  getch();
  return 0;
}
```

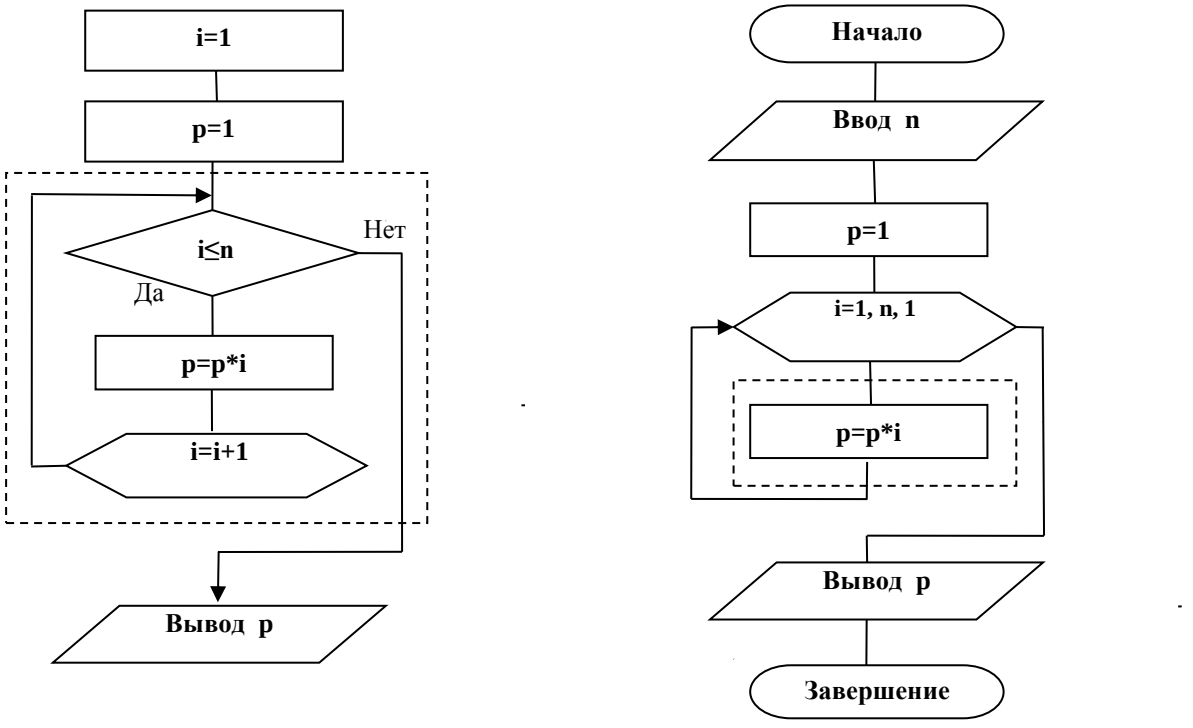
В результате выполнения программы будет напечатан столбец положительных чисел от 1 до 15. Анализ результатов показывает, что работа оператора цикла **for** происходит в полном соответствии со схемой вычислительного процесса на рис. 1. В данном примере параметром цикла **for** является переменная **i** (она же – счетчик). В начале цикла счетчик (переменная **i**) инициализируется значением 1. Потом выполняется тело цикла (**cout << i <<endl;**) и проверяется, не достиг ли счетчик значения **n**=15. После каждого выполнения тела цикла счетчик (**i**) увеличивается на единицу. Как только **i** станет равным 15, тело цикла пропускается и управление передается следующему оператору программы.

Пример 1. Исследуем программу вычисления факториала целого числа **n** с использованием оператора **for**. **Факториал** – это произведение целых чисел от 1 до **n**. Необходимо проанализировать блок-схемы вычислительного процесса и алгоритма решения задания при помощи оператора цикла **for**, ввести код программы, построить решение и произвести вычисления для n=10.

Факториал числа **n** вычисляется по следующей формуле:

$$\mathbf{n! = 1*2*3*4* \ldots *n} = \prod_{i=1}^{n} i$$

При вычислении факториала начальному значению произведения П необходимо присвоить 1. При каждом выполнении тела цикла $П_{i-1}$ будем умножать на **i**. Примем для обозначения произведения идентификатор **p**. Блок-схемы вычислительного процесса и алгоритма решения задания при помощи оператора цикла **for** имеют следующий вид:



а) б)

Рис. 4. Блок-схемы вычисления факториала целого числа **n**:
а) вычислительного процесса; б) алгоритма с использованием оператора **for**

Учитывая блок-схемы на рис. 4 программу вычисления факториала можно записать в следующем виде:

```
int _tmain(int argc, _TCHAR* argv[])
{
        int i, n, p;
```

```
        cout<<"Vvedite N"<<endl;
        cin>>n;                          //Ввод значения переменной  n
        p =1;                            //Начальное значение 1 переменной p для произведения
        for (i=1; i<=n; i++)             //Оператор цикла  for
        p =p*i;                          //Тело цикла – произведение p и  i
        cout<<"faktorial " <<n<<"   =  "<<p<<endl;      //Печать результата
  getch();
  return 0;
}
```

После запуска программы на экране появится результат:

Vvedite celoe chislo: 5
factorial 5   = 120

Обратите внимание на необходимость присваивания начального значения (единица) переменной **p** для вычисления последующих произведений.

Пример 2. Исследуем программу для вычисления суммы **n** четных чисел, начиная с двойки. Формула для вычисления суммы **n** четных чисел, начиная с двойки, имеет следующий вид:

$$s = \sum_{i=1}^{n} (2 * i),$$

где **i** – параметр цикла (номер четного числа на данном шаге).

Блок-схема алгоритма решения такой задачи приведена на рис. 5. Обратите внимание на необходимость присваивания начального значения (ноль) переменной **p** для вычисления последующих сумм.

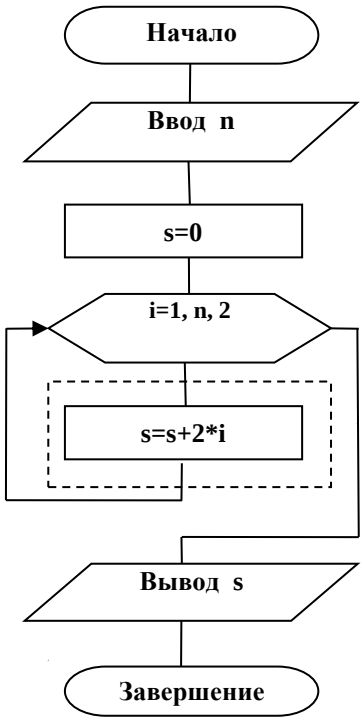


Рис. 5. Блок-схема алгоритма для вычисления суммы **n** четных чисел

Программа вычисления суммы **n** четных чисел имеет следующий вид:

```
int _tmain(int argc, _TCHAR* argv[])
{
        int i, n, s;
        cout<<"Vvedite N"<<endl;
```

```
        cin>>n;                                  //Ввод значения переменной  n
        s =0;                                    //Начальное значение 0 переменной s для
                                                 //суммирования
        for ( i=1; i<=n; i++ )                   //Оператор цикла  for
        s=s+2*i;                                 //Тело цикла – расчет суммы
        cout<<"Summa  " <<n<<"    =  "<<s<<endl; //Печать результата
        getch();
        return 0;
}
```

После запуска программы на экране появится результат:

```
Vvedite celoe chislo: 5
Summa 10   = 30
```

Обратите внимание на необходимость присваивания начального значения (ноль) переменной **s** для вычисления последующихсумм.

Пример 3. Исследуем программу вычисления функции $\mathbf{Y=\sum_{k=1}^{6}\left(\cos^3 kx+\sin\frac{0{,}5x}{\sqrt[3]{k^2+x^2+1}}\right)}$ для **x**=1,32. Блок-схема алгоритма для вычисления данной функции аналогична блок-схеме алгоритма для вычисления суммы **n** четных чисел на рис. 5. Необходимо ввести код программы, построить решение и произвести вычисления.

Для решения задачи используется оператор цикла **for**, т. к. известно количество повторений тела цикла (k=10). Введем такие идентификаторы: **x, Y, k** и **S**. Формула для расчета функции **Y** на каждом шаге вычислений на языке С++ имеет следующий вид:

Y=pow(cos(k*x),3)+sin(0.5*x/pow((k*k+x*x+1),1/3)).

Тогда программа для расчета функции **Y** примет вид:

```
int _tmain(int argc, _TCHAR* argv[])
{
int i, k=10;
double x=1.32, Y, S;
Y=0;                                     //Начальное значение  переменной Y для
                                         //суммирования
cout<<"x= "<<x<<endl;                    //Вывод x
        for (i=1; i<k; i++)              //Оператор цикла  for
        {                                //Начало тела цикла
        S=pow(cos(k*x),3)+sin(0.5*x/pow((k*k+x*x+1),1/3));  //Расчет функции на
                                                           //каждом шаге
        Y=Y+S;
        cout<<"k= "<<i<<"    Y = "<<Y<<endl;   //Вывод  Y на каждом шаге
        }                                //Конец тела цикла
getch();
return 0;
}
```

Так как тело цикла содержит более одного оператора, то эти операторы охвачены фигурными скобками.

После запуска программы на экране появится результат:

```
x= 1.32
k= 1     Y = 1.1365
k= 2     Y = 2.27299
k= 3     Y = 3.40949
k= 4     Y = 4.54599
k= 5     Y = 5.68249
k= 6     Y = 6.81898
k= 7     Y = 7.95548
k= 8     Y = 9.09198
k= 9     Y = 10.2285
```

Рис. 6. Результаты вычислений для **Примера 3**

Обратите внимание на то, что результат расчета функции **Y** выводится на каждом шаге вычислений. Это сделано с целью контроля при отладке программы. После устранения возможных ошибок можно выводить только итоговый результат.

Пример 4. Исследовать программу вычисления значения функции $\mathbf{Y=\sum_{n=0}^{4} n\left(\sum_{k=1}^{5}\sin^3(knx-a)\right)}$ для **x**=1,4 и **a**=2,3 (использовать вложенный цикл **for**). Необходимо ввести код программы, построить решение и произвести вычисления.

Определим исходные данные, которые понадобятся для решения задачи:

a – константа, инициализируем ее, как **a** = 2,3 и определим ее тип как **double** (действительное двойной точности);

x – константа, инициализируем ее, как **x**=1,4, тип **double**;

Y – переменная, тип **double**;

s – промежуточная переменная, обозначим ее как **S**, тип **double**.

Формула для расчета промежуточной суммы (внутри цикла по **k**) $\mathbf{s=\sin^3(knx-a)}$ на k-м шаге вычислений на языке C++ имеет следующий вид:

s=pow(sin(k*n*x-a),3) .

Блок-схема алгоритма решения данной задачи приведена на рис. 7.

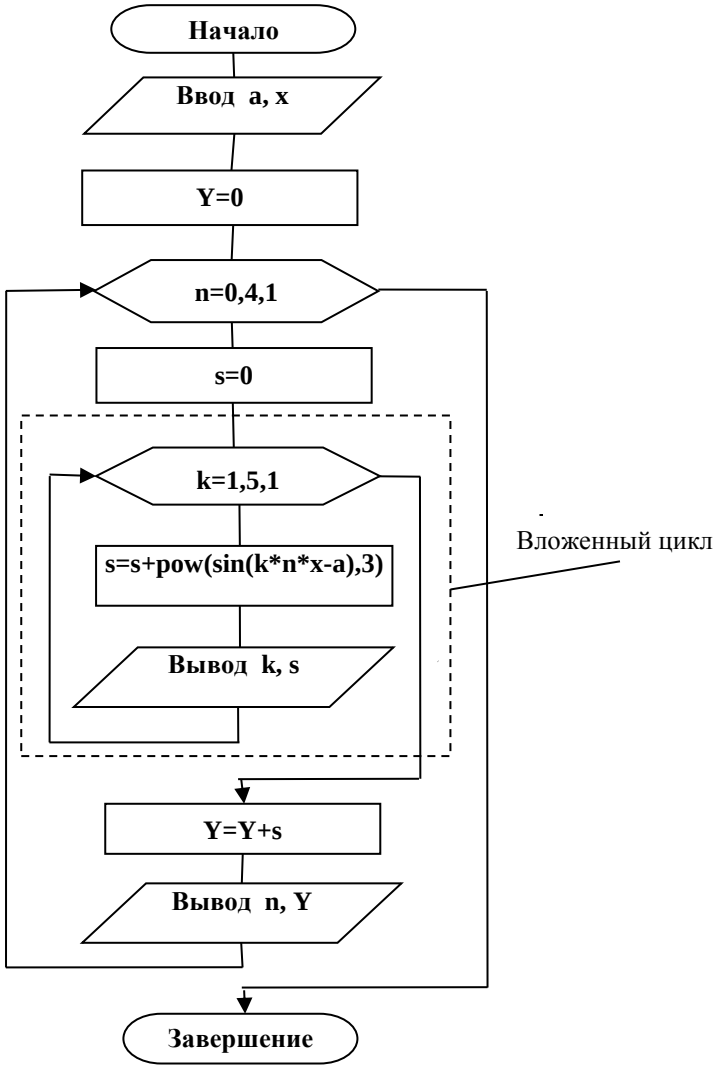


Рис. 7. Блок-схема алгоритма решения Примера 4 с использованием вложенного цикла **for**

Введите и выполните программный код, реализующий данный алгоритм. Обратите внимание на сдвиг фигурных скобок, ограничивающих сложные операторы, относительно скобок, охватывающих тело главной функции.

```
int _tmain(int argc, _TCHAR* argv[])    //Объявление главной функции _tmain
{
const double x=1.4, a=2.3;              //Инициализация констант a и x
double Y, n, k, s;                      //Объявл-е переменных и параметров цикла: y, s, n  и k
cout<<"x= "<<x<<"  a= "<<a<<endl;       //Вывод исходных данных
Y=0;                                    //Начальное значение переменной s для суммирования
        for(n=0;n<=4;n++)               //Внешний оператор for (условие цикла n ≤ 4)
        {                               //Начало составного оператора – вложенного цикла
        s=0;                            //Начальное значение переменной y для суммирования
                for(k=1;n<=5;k++)       //Вложенный оператор for (условие цикла k ≤ 5)
                {
                s=s+ pow(sin(k*n*x-a),3); //Суммирование y на k-ом шаге
                cout<<k<<"   s =  "<<s<<endl;      //Вывод на экран переменной y на k-ом шаге
                }
        Y=Y+n*s;                        //Суммирование s на n-ом шаге
        cout<<n<<"  Y =  "<<Y<<endl;    //Вывод на экран переменной s на k-ом шаге
        }                               //Конец составного оператора с вложенным циклом
getch();                                //Функция задержки окна DOS на экране
return 0;
}                                       //Конец главной функции
```

Результаты вычислений приведены на рис. 8. Промежуточные результаты вычислений выводятся на экран для дополнительного контроля правильности работы программы (тестирования программы).

```
x= 1.4  a= 2.3
  k = 1    s =  -0.414669
  k = 2    s =  -0.829338
  k = 3    s =  -1.24401
  k = 4    s =  -1.65868
  k = 5    s =  -2.07334

     n = 0  Y =  0
  k = 1    s =  -0.48065
  k = 2    s =  -0.370455
  k = 3    s =  0.476942
  k = 4    s =  0.473016
  k = 5    s =  -0.526753

     n = 1  Y =  -0.526753
  k = 1    s =  0.110195
  k = 2    s =  0.10627
  k = 3    s =  0.100225
  k = 4    s =  0.225993
  k = 5    s =  -0.21643

     n = 2  Y =  -0.959613
  k = 1    s =  0.847396
  k = 2    s =  0.841352
  k = 3    s =  0.388923
  k = 4    s =  1.20605
  k = 5    s =  1.20274

     n = 3  Y =  2.64861
  k = 1    s =  -0.0039253
  k = 2    s =  0.121842
  k = 3    s =  0.938967
  k = 4    s =  1.79397
  k = 5    s =  1.94911

     n = 4  Y =  10.445
```

Рис. 8. Результаты вычислений для Примера 4

Вопросы для самоконтроля:

1. Перечислите, что должно быть в конструкции цикла.
2. С помощью каких действий реализуются циклические вычислительные процессы в языке С++?
3. Запишите в общем виде оператор цикла **for** и опишите, как он выполняется.
4. Чем оператор цикла **for** отличается от операторов цикла **while** и **do...while**?
5. В каких случаях используется оператор цикла **for**?
6. Запишите и дайте краткую характеристику операции инкремента?
7. Сколько раз выполнится оператор цикла **int N=3; for (int i=1; i<=N; i++) N=N-1;**?
8. Сколько раз выполнится в программе оператор цикла **for (i = 0; i<1; i++) cout <<i;**?
9. Записан оператор **for (i = 2; i <10; i+=2) cout << i ;**. Что будет выведено на экран дисплея?