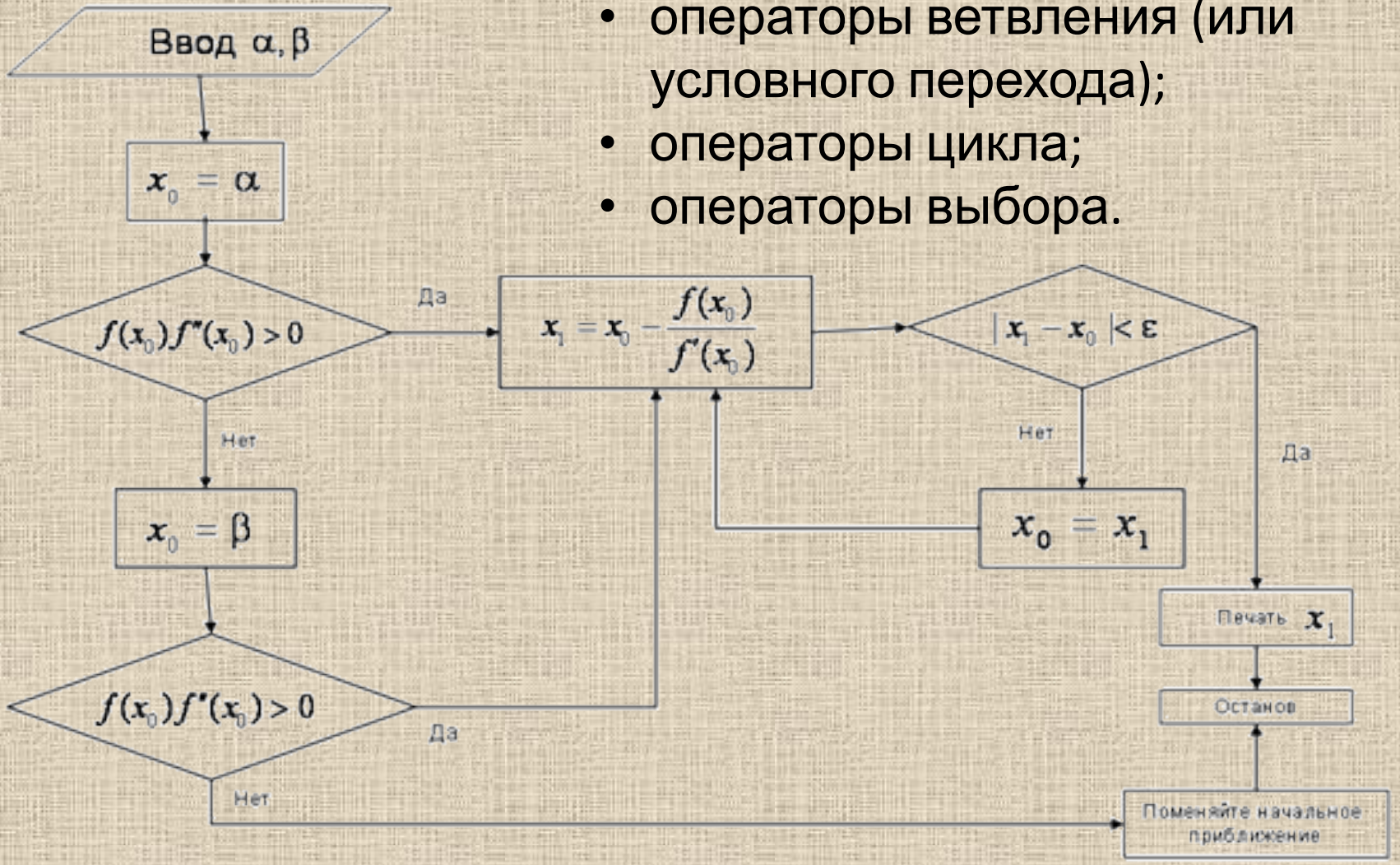
Операторы управления

Три типа управляющих инструкций:

- операторы ветвления (или условного перехода);
- операторы цикла;
- операторы выбора.



Управление ходом программы с помощью команд ветвления

Оператор условного перехода реализуется с помощью инструкции if.

```
if (a > b)
  {
    ...
  }
else
  {
    ...
  }
```

Оператор else не обязателен: если он отсутствует, C++ считает, что он существует, но является пустым.

Если в текущей ветви оператора if имеется только одна инструкция, то скобки использовать не обязательно.

Но безопаснее скобки использовать всегда.

Пример работы оператора if:

```
include <iostream>
using namespace std;
int main ();
{
    int a;
    cout << “Введите a: “;
    cin >> a;

    int b;
    cout << “Введите b: “;
    cin >> b;

    if (a > b)
       {cout << “a больше b” << endl;}
    else
       {cout << “a не больше b” << endl;}
    system (“PAUSE”); return 0;
}
```

Управление ходом программы с помощью операторов цикла

Самый просто цикл можно организовать с помощью оператора while:

```
while (условие)
  {
     // Этот код выполняется повторно,
     // пока условие остается истинными
  }
```

```
include <iostream>
using namespace std;
int main ();
{
    // Ввод счетчика цикла
    int loopCount;
    cout << “Введите loopCount : “;
    cin >> loopCount;

    // Теперь в цикле выводим значения
    while (loopCount > 0)
    {
       loopCount = loopCount – 1;
       cout << “Осталось выполнить” << loopCount << “
       циклов\n”;
    }
    system (“PAUSE”);
    return 0;
```

Программа “Счетчик цикла”

Если loopCount = 5

```
Осталось выполнить 4 циклов
Осталось выполнить 3 циклов
Осталось выполнить 2 циклов
Осталось выполнить 1 циклов
Осталось выполнить 0 циклов
```

```
//Цикл с предусловием
do
{
  // Тело цикла
}
while (условие);
```

Компактная запись операторов инкремента и декремента

```
while (i > 0)
{
 i = i - 1;
 cout << “Осталось выполнить ” << i << “ циклов\n”;
}
```

```
while (i > 0)
{
 i = i --;
 cout << “Осталось выполнить ” << i << “ циклов\n”;
}
```

```
while (i -- > 0)
{
 cout << “Осталось выполнить ” << i << “ циклов\n”;
}
```

Цикл for

```
for (инициализация; условие;
увеличение)
{
   // тело цикла
}
```

Цикл for можно заменить эквивалентным ему циклом while:

```
инициализация;
while (условие)
   {
      // тело цикла
   }
увеличение;
}
```

Все три параметра цикла for являются необязательными. *Если опущено условие, C++ будет выполнять цикл for вечно.*

```
int main ();
{
    // Ввод счетчика цикла
    int loopCount;
    cout << “Введите loopCount: “;
    cin >> loopCount;

    // В цикле выводим значения
    while (loopCount > 0)
    {
       loopCount = loopCount – –;
       cout << “Осталось выполнить”
       << loopCount << “ циклов\n”;
    }
    system (“PAUSE”);
    return 0;
}
```

```
int main ();
{
    // Ввод счетчика цикла
    int loopCount;
    cout << “Введите loopCount : “;
    cin >> loopCount;

    // Работает loopCount раз
    for ( ; loopCount > 0; )
    {
       loopCount = loopCount – –;
       cout << “Осталось выполнить
       << loopCount << “ циклов\n”;
    }
    system (“PAUSE”);
    return 0;
}
```

Чтобы избежать нисходящего цикла и потери значений переменной loopCount, можно добавить специальную переменную – счетчик цикла.

```
int main ();
{
  // Ввод количества циклов
  int loopCount;
  cout << “Введите loopCount: “;
  cin >> loopCount;

  // Цикл до достижения значения loopCount
  for (int i = 1; i <= loopCount; i++)
  {
     cout << “Выполнено ” << i << “ циклов /n”;
  }
  system (“PAUSE”);
  return 0;
}
```

```
int main ();
{
  // Ввод количества циклов
  int loopCount;
  cout << “Введите loopCount: “;
  cin >> loopCount;
  // Цикл до достижения значения loopCount

  for (int i = 1;        i <= loopCount;     i++)
       инициализация;   условие;  увеличение

  {
     cout << “Выполнено ” << i << “ циклов /n”;
  }
  system (“PAUSE”);
  return 0;
}
```

Бесконечный цикл

```
while (loopCount > 0)
{
   cout << “Осталось выполнить ”
                        << loopCount << “ циклов\n”;
}
```

Специальные операторы управления циклом

В C++ определены 2 спец.команды – break и continue.

В случаях, если условие работы цикла нарушается не в начале или в конце, а в середине цикла, можно использовать команду break, чтобы выйти из цикла.

```
while (условие)
{
   if (другое условие)
   {
      break; // выход из цикла
   }
} // когда программа встретит break,
  // управление будет передано этой строке
```

Программа, которая суммирует введенные пользователем числа (цикл прерывается, когда пользователь вводит отрицательное число):

```
...
int main ()
{
    // Введите счетчик цикла
    int Sum = 0;
    cout << “Программа суммирует числа, введенные пользователем\n”;
    cout << “Выполнение цикла заканчивается после того, как пользователь “
         << “введет отрицательное число\n”;
    // Бесконечный цикл
    for (;;)
    {
        // Ввод следующего числа
        int x = 0;
        cout << “Введите следующее число: “;
        cin >> x;
        // если оно отрицательно
        if (x < 0)
        {
            // тогда выйти из цикла
            break;
        }
        // иначе добавляем число к общей сумме
        Sum = Sum + x;
    }
    // После выхода из цикла выводим результат суммирования
    cout << “\nОбщая сумма равна “ << Sum << “\n”;
...
```

Результат

```
Программа суммирует числа, введенные пользователем
Выполнение цикла заканчивается после того, как пользователь введет отрицательное число
Введите следующее число: 7
Введите следующее число: 8
Введите следующее число: 9
Введите следующее число: -1

Общая сумма равна 24
Press any key to continue...
```

Написать программу, которая каждый раз (бесконечный цикл) запрашивает у пользователя делимое и делитель и получает частное. Выход из цикла, если пользователь вводит нулевое значение делителя.

Встретив команду continue, программа немедленно возвращается к началу цикла:

```
while
{
   // Ввод значения
   cout << “Введите значение”;
   cin >> x;

   // Если число отрицательное
   if (x <0)
   {
      // Вывести сообщение об ошибке
      cout << “Не допускается ввод отрицательных ”
           << “чисел\n”;
      // Возврат в началу цикла
      continue;
   }
   // Введено неотрицательное число
}
```

Создать новый проект и протестировать программу, представленную на предыдущем слайде.

Вложенные команды управления

Вложенный цикл - это цикл, размещённый внутри другого цикла. На первом проходе внешний цикл вызывает внутренний, который исполняется до своего завершения, после чего управление передается в тело внешнего цикла. На втором проходе внешний цикл опять вызывает внутренний. Как внешний, так и внутренний цикл может быть прерван командой break.

Программа, демонстрирующая работу вложенных циклов:

[*] main.cpp

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
    // Внешний цикл работает с последовательностями чисел
    int Sum;
    do
    {
        // Ввод очередной последовательности чисел
        Sum = 0;
        cout << "\nVvedite posledovatelnost\n";

        // Бесконечный цикл
        for (;;)
        {
            // Введение очередного числа
            int x = 0;
            cout << "Vvedite ocherednoe chislo: ";
            cin >> x;
            // Если оно отрицательное
            if (x < 0)
            {
                // Выходим из цикла
                break;
            }
            // Иначе добавляем число к общей сумме
            Sum = Sum + x;
        }
        // Вывод результата вычислений
        cout << "\nObshaa summa ravna " << Sum << "\n";

        // Если накопленная общая сумма чисел последовательности не равна нулю,
        // начинаем работать со следующей последовательностью
    } while (Sum != 0);
    cout << "Program has happy end\n";
    system ("pause");
    return 0;
}
```

Программа, демонстрирующая работу вложенных циклов:

```
Uvedite posledovatelnost
Uvedite ocherednoe chislo: 1
Uvedite ocherednoe chislo: 2
Uvedite ocherednoe chislo: 3
Uvedite ocherednoe chislo: 0
Uvedite ocherednoe chislo: 0
Uvedite ocherednoe chislo: 4
Uvedite ocherednoe chislo: -3

Obshaa summa ravna 10

Uvedite posledovatelnost
Uvedite ocherednoe chislo: 2
Uvedite ocherednoe chislo: -2

Obshaa summa ravna 2

Uvedite posledovatelnost
Uvedite ocherednoe chislo: -5

Obshaa summa ravna 0
Program has happy end
Для продолжения нажмите любую клавишу . . .
```

Инструкция выбора

Инструкция выбора эффективна, если есть необходимость выбора при ограниченном количестве возможных вариантов. Инструкция выбора похожа на усложненную инструкцию if, которая вместо проверки одного условия анализирует множество разных возможностей.

```
switch (выражение)
{
    case a:
        // Переход сюда, если выражение == a;
        break;
    case b
        // Переход сюда, если выражение == b;
        break;
    default:
        // Если ни одно условие не выполнено, то переход
        сюда
}
```

Значением выражения должно быть целое число (int, long или char); a, b и т.д. должны быть константами, указанными после служебного слова case. Если константа соответствует значению выражения, то программа передает управление этой ветви. Если ни один вариант не подходит, то выполняется условие default.

Пример (фрагмент программы):

```
cout << “Введите 5, 10 или 15: ”;
cin >> choice;
switch (choice)
{
    case 5:
        // Обработка случая “5”;
        break;
    case 10:
        // Обработка случая “10”;
        break;
    case 15:
        // Обработка случая “15”;
        break;
    default:
        cout << “Вы ввели не 5, не 10 и не 15\n”
}
```

Для выхода из инструкции switch необходимо использовать команды break, иначе управление будет переходить от одного случая к следующему, даже если совпадение значений уже было найдено.

Написать программу, которая увеличивает значение введенной пользователем переменной в 10 раз, если это число 200; уменьшает значение на единицу, если это число 300; выводит на экран это число, если оно равно 0; превращает число -200 в положительное; выводит сообщение о некорректном вводе числа.