Курс лекций "Программирование"
Основы программирования на языках C и C++

Лекция 2. Конструкции и операции языков C и C++

Глухих Михаил Игоревич, к.т.н., доц.

mailto: glukhikh@mail.ru

Основные элементы языка – терминология

- Операции (operations) и знаки операций (operators)
- Идентификаторы, имена (identifiers)
- Операторы, команды, инструкции (statements)
- Конструкция – набор операторов, скомбинированный по определенным правилам
  - составной оператор
  - оператор ветвления, оператор цикла
  - …

Что является операцией в C/C++?

- Выражение типа #x1, x1#, x1#x2
  - x1, x2 – аргументы операций (они могут быть переменными или другими выражениями)
  - # – знаки операций (один или два символа)
- Специфичную форму записи имеют скобочная операция (x1) или операция индексации x1[x2]
- Примеры
  - #x1 или x1# – унарная операция (один аргумент)
  - x1#x2 – бинарная операция (два аргумента)
  - в языках C и C++ имеется также тернарная операция (три аргумента) с формой записи x1 ? x2 : x3

Результаты операций

- У любой операции есть результат!
  - Результат операции x+y равен сумме x и y
  - Результат операции x=y равен y
- Результат операции может быть использован как аргумент других операций
  - Пример: x=b+(c=d)
- Часть операций (например, присваивания) помимо формирования результата меняют свой первый аргумент
  - Если операция лишь **использует** аргумент, он может быть любой переменной или выражением
  - Если операция **изменяет** аргумент, он должен быть переменной либо другим способом описывать содержимое определенной ячейки памяти (**L-value** – значение, имеющее **адрес**)

Виды операций

- **Арифметические**
- **Присваивания/модификации**
- **Инкремента/декремента**
- **Сравнения**
- **Логические**
- **Выбора**
- **Скобочная**
- **Преобразования типа**
- **Ввода-вывода**
- *Побитовые логические*
- *Сдвига*
- *Индексации*
- *Адресации/разадресации*
- *Обращения к полю структуры*

Арифметические операции

- Целые или вещественные аргументы
  - a+b // сложение
  - a-b // вычитание
  - a*b // умножение
  - a/b // деление
  - -a // смена знака
- Целые аргументы
  - a%b // взятие остатка

Операции присваивания/модификации

- Совместимые аргументы
  - a=b (записать в a значение b)
- Целые или вещественные аргументы
  - a += b (добавить к a значение b, результат равен a+b, сокращенная форма записи a=a+b)
  - a -= b (вычесть из a значение b)
  - a *= b (домножить a на значение b)
  - a /= b (поделить a на значение b)
- Целые аргументы
  - a %= b (сокращенная форма записи a=a%b)
- Побитовые логические операции и операции сдвига также могут быть в форме модификации

Операции инкремента/декремента

- У всех – целый или вещественный аргумент
- a++ (постфиксный инкремент, a увеличивается на 1, результат равен **старому** значению a)
- a-- (постфиксный декремент, a уменьшается на 1, результат равен **старому** значению a)
- ++a (префиксный инкремент, a увеличивается на 1, результат равен **новому** значению a)
- --a (префиксный декремент, a уменьшается на 1, результат равен **новому** значению a)

Пример использования инкремента/декремента

```
#include <iostream>
using namespace std;
int main(void)
{
   int a=1, b=3;
   a--;
   ++b;
   cout<<"a="<<a<<" b="<<b<<endl;
   a=b++;
   cout<<"a="<<a<<" b="<<b<<endl;
   b=--a;
   cout<<"a="<<a<<"b="<<b<<endl;
   return 0;
}
```

Пример использования инкремента/декремента

```
#include <iostream>
using namespace std;
int main(void)
{
   int a=1, b=3;
   a--; // a=0
   ++b; // b=4
   cout<<"a="<<a<<" b="<<b<<endl; // a=0 b=4
   a=b++; // a=4 (старое b) b=5
   cout<<"a="<<a<<" b="<<b<<endl; // a=4 b=5
   b=--a; // a=3 b=3 (новое a)
   cout<<"a="<<a<<"b="<<b<<endl; // a=3 b=3
   return 0;
}
```

Операции сравнения

- Обычно используются для формирования условий
  - **a == b** (сравнение на равенство)
  - a != b (сравнение на неравенство)
  - a > b (больше)
  - a >= b (больше или равно)
  - a <= b (меньше или равно)
  - a < b (меньше)
- Целые или вещественные аргументы
- Целый или логический результат
  - 1 (true), если условие верно
  - 0 (false) в обратном случае

Связь целых и логических значений

- В языке С логических значений нет, вместо них используются целые
- В языке С++ везде, где допустимы логические значения, допустимы и целые
- В языках С/С++
  - целое значение, не равное 0, соответствует истинному условию
  - целое значение, равное 0, соответствует ложному условию

Логические операции

- Обычно используются для формирования условий
  - !a (отрицание, истинно, если a ложно)
  - a || b (логическое или, истинно, если a истинно, или b истинно, или они оба истинны)
  - a && b (логическое и, истинно, если a истинно и b истинно)
- Целые, вещественные или логические аргументы и результат
  - не 0 используется в значении истина
  - 0 используется в значении ложь

Принцип ленивых вычислений

- c = a || b
  - вначале считаем **a**
  - если **a** истина, то **c** истина (**b** не считаем)
  - иначе считаем **b**, c=b
- c = a && b
  - вначале считаем **a**
  - если **a** ложь, то **c** ложь (**b** не считаем)
  - иначе считаем **b**, **c**=**b**
- Имеет значение, если
  - расчет **b** изменяет какие-нибудь переменные или может вести к ошибкам
  - расчет **b** требует много времени

Операция выбора

- x ? y : z
- Тернарная операция
- Простейший способ проверки условия
- Аргумент x целый или логический
- Аргументы y и z целые или вещественные
- Результат равен y, если x истинно (не равно 0)
- Результат равен z, если x ложно (равно 0)

Примеры использования операции выбора

```
x >= 0 ? x : -x; // Что это?
a >= b ? a : b; // А это?
a >= b ? b : a; // А это?
```

Пример – максимум из двух

```
#include <iostream>
using namespace std;
// Рассчитываем max(a,b)
int main(void)
{
   double a, b;
   cout<<"Calculating max(a,b). "<<
      "Enter a,b: "<<endl;
   cin>>a>>b;
   double max = a > b ? a : b;
   cout<<"max(a,b)="<<max<<endl;
   return 0;
}
```

Скобочная операция

- (x), где x – переменная или выражение
- Операции в скобках выполняются раньше операций вне скобок

Операции преобразования типа

```
double x=3.5;
// Снижение точности
float y=(float)x;
// Преобразование к меньшему целому
// (например, 2.6→2, -3.1→-4)
int n=(int)y;
// Округление по стандартным правилам
int m=(int)(y+0.5);
// Добавление нулевой вещ. части
// (обычно пишут просто double z=n)
double z=(double)n;
```

Операции вывода

- Левый аргумент – поток (сущность, куда можно выводить или откуда можно вводить)
  - Поток вывода на консоль cout
  - Поток ввода с клавиатуры cin
  - Потоки ввода/вывода в файл
- Правый аргумент – переменная или выражение, для которых поддерживается ввод или вывод
- Результат – поток, что позволяет команды типа cout<<x<<y<<z
- cout<<x операция вывода в поток
- cin>>x операция ввода из потока (x должно быть L-value)

Порядок выполнения операций (приоритеты)

- (), постфиксные ++, --; слева направо
- !, унарный -, префиксные ++, --; справа налево
- *, /, %; слева направо
- бинарные +, -; слева направо
- <<, >>; слева направо
- <, <=, >=, >; слева направо
- ==, !=; слева направо
- && слева направо
- || слева направо
- x ? y : z справа налево
- =, +=, -=, *=, /=, %=; справа налево
- Помнить порядок необязательно!
  Когда не уверены – используйте скобки!

Математические функции

- Описаны в подключаемом файле math.h
  - **#include** <math.h>
- Основные функции
  - abs(x) – модуль
  - sqrt(x) - квадратный корень
  - exp(x) – экспонента e^x
  - log(x), log10(x) – натуральный и десятичный логарифмы
  - pow(x, y) – возведение в степень
  - sin(x), cos(x), tan(x) – синус, косинус, тангенс, **аргумент в радианах (!)**
  - asin(x), acos(x), atan(x) – арксинус, арккосинус, арктангенс, **результат в радианах (!)**
  - M_PI = 3.1415926536...; M_E = 2.7182818284…

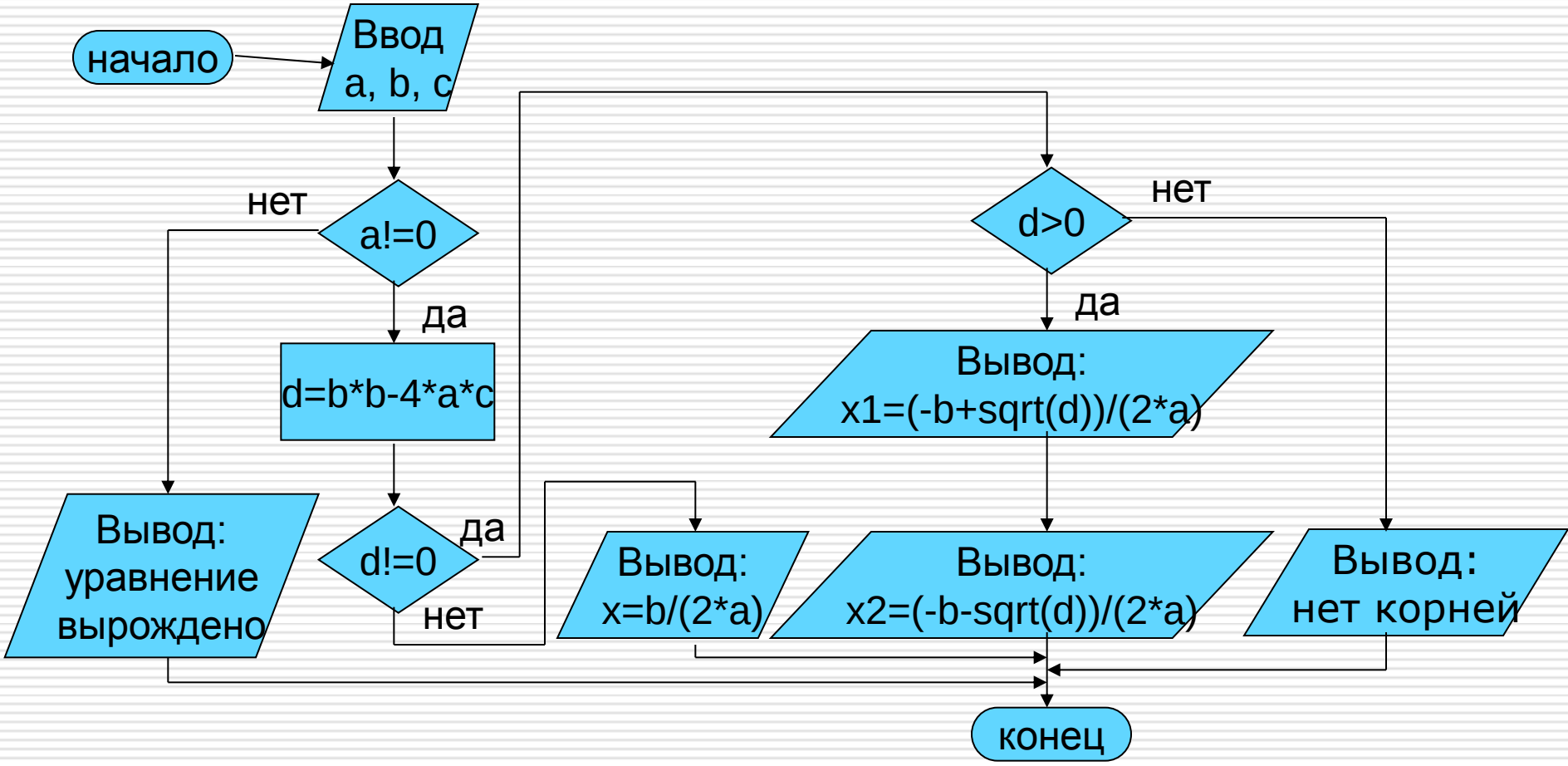
Математические константы

- Требуется доп. настройка
  - **#define** _USE_MATH_DEFINES
  - **#include** <math.h>
- M_PI = 3.1415926536...
- M_E = 2.7182818284…

Пример – корни квадратного уравнения

```
#include <iostream>
#include <math.h>
using namespace std;
// Вычисляем корни ax^2+bx+c=0
int main(void)
{
   double a, b, c;
   cout<<"Decision of ax^2+bx+c=0, enter a, b, c: ";
   cin>>a>>b>>c;
   double x1, x2, d;
   d=sqrt(b*b-4*a*c); // Корень из определителя, где ошибка?
   x1=(-b+d)/2*a; // Больший корень уравнения, где ошибка?
   x2=(-b-d)/2*a; // Меньший корень уравнения, где ошибка?
   cout<<"x1="<<x1<<" x2="<<x2<<endl;
   return 0;
}
```

Полноценный алгоритм



Форматирование вывода

```
#include <iostream>
using namespace std;
int main(void)
{
    double x=1.234567e-5;
    double y=1.234567e-4;
    double z=5.467e+6;
    double w=5.467e+5;
    cout<<"x="<<x<<endl; // x=1.234567e-005
    cout<<"y="<<y<<endl; // y=0.0001234567
    cout<<"z="<<z<<endl; // z=5.467e+006
    cout<<"w="<<w<<endl; // w=546700
    return 0;
}
```

Выбор формата вывода

- По умолчанию, формат выбирается исходя из величины числа
  - обычный 3124.2642
  - инженерный 3.12426e+03
- Что, если мы хотим выводить только в обычном формате или только в инженерном?

```
// Дальнейший вывод в обычном формате
cout<<fixed;
// Дальнейший вывод в инженерном формате
cout<<scientific;
```

Выбор ширины и точности

- Для некоторых манипуляторов необходим **#include** <iomanip>
- По умолчанию, вещественные числа выводятся с 6 знаками после точки

```
cout<<setprecision(3); // 3 знака
```

- По умолчанию, числа выводятся «как есть», без дополнительных символов и пробелов

```
cout<<setw(10)<<x<<endl; // 10 позиций
cout<<right; // Выровнять вправо (по умолчанию)
cout<<left; // Выровнять влево
cout<<setfill('0'); // Чем заполнять пустые места
// По умолчанию пробелами
```

Пример – вывод времени

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(void)
{
   int hh=9, mm=5;
   // Вывод в формате Time: 09:05
   // Обратите внимание на одинарные кавычки
   // в манипуляторе setfill
   cout<<"Time: "<<setfill('0')<<
      setw(2)<<hh<<":"<<
      setw(2)<<mm<<endl;
   return 0;
}
```

Другие манипуляторы

- Вывод логических (**bool**) переменных

```
cout<<boolalpha; // Вывод как true/false
cout<<noboolalpha; // Вывод как 1/0
```

- Показывать ли знаки

```
cout<<showpos; // Всегда со знаком
cout<<noshowpos; // + не выводится
```

- Полный список в MSDN, ключевые слова
  - ios
  - iomanip

Вывод русских символов на консоль

- Текст программы набирается в кодировке 1251
  - Кодировка - таблица соответствия номеров (кодов) конкретным символам
- Вывод на консоль (по умолчанию) производится в кодировке 866
  - В ней эта таблица (для русских символов) другая, поэтому русские символы выводятся неправильно
- Для смены кодировки используется функция **setlocale** (метод работает начиная с Visual Studio 2005, и может не работать в поздних версиях)

Пример: здравствуй, мир

```
#include <iostream>
// Здесь описана функция setlocale
#include <locale.h>
using namespace std;
int main(void)
{
   // Вывод на консоль в 866
   cout<<"Здравствуй, мир!"<<endl;
   setlocale(LC_ALL, "Russian");
   // Далее, вывод на консоль в 1251
   cout<<"Здравствуй, мир!"<<endl;
   return 0;
}
```

Основные конструкции языков C и C++

- Составной оператор { ... }
- Ветвления
  - if ... else
  - switch ... case
- Циклы
  - while
  - do ... while
  - for
- Все перечисленные конструкции – операторы!

Конструкция if (если…)

```
if (условие)
    оператор;
```

- Оператор (**один!**) будет выполнен, если условие верно
- На месте условия может быть целое или вещественное значение
  - условие считается верным, если значение не равно 0
- На месте условия может быть логическое значение
  - условие считается верным, если значение истинно (**true**)

Вариант с составным оператором

```
if (условие)
{
    оператор1;
    оператор2;
    оператор3;
}
```

- Операторы 1, 2 и 3 будут выполнены, только если условие верно

Конструкция if/else (если… иначе…)

```
if (условие)
    оператор1;
else
    оператор2;
```

- Если условие верно, выполняется оператор 1
- Если условие неверно, выполняется оператор 2

Составной вариант

```
if (условие)
{
   команда1;
   команда2;
   команда3;
}
else
{
   команда4;
   команда5;
}
// Если условие верно, выполняются операторы 1-3
// Если условие неверно, выполняются операторы 4-5
```

Каскадный вариант if – else if – … – else

```
if (условие1)
{
   … // Если верно условие1
}
else if (условие 2)
{
   … // Если неверно условие1 и верно условие2
}
else if (условие3)
{
   … // Если неверны условия 1 и 2 и верно условие3
}
else
{
   … // Если неверны все условия
}
```

Пример – вычисление корней квадратного уравнения

```
#include <iostream>
#include <math.h>
#include <locale.h>
using namespace std;
// Вычисляем корни ax^2+bx+c=0
int main(void)
{
   double a, b, c;
   setlocale(LC_ALL, “Russian”);
   cout<<"Решаем уравнение ax^2+bx+c=0, введите a, b, c: ";
   cin>>a>>b>>c;
   if (a==0.0)
   {
      cout<<"Уравнение вырождено"<<endl;
      return 0;
   }
```

Пример – вычисление корней квадратного уравнения

```
    // …
    double x1, x2, d;
    d=b*b-4*a*c; // Определитель
    if (d<0.0)
       cout<<"Корней нет!"<<endl;
    else if (d==0.0)
       cout<<"x="<<-b/(2*a)<<endl;
    else
    {
       d=sqrt(d); // Корень из определителя
       x1=(-b+d)/(2*a); // Больший корень уравнения
       x2=(-b-d)/(2*a); // Меньший корень уравнения
       cout<<"x1="<<x1<<" x2="<<x2<<endl;
    }
    return 0;
}
```

Результаты работы

- …Введите a, b, c: 0 1 -1
- Уравнение вырождено
- …Введите a, b, c: 1 -2 1
- x=1
- …Введите a, b, c: 1 2 2
- Корней нет
- …Введите a, b, c: 2 -3 1
- x1=1 x2=0.5

Конструкция switch-case

```
switch (выражение) // значение которого проверяем
{
   case константа1: // в случае выражение==константа1
      оператор11;
      …
      оператор1K;
      break; // прервать выполнение конструкции
   // если break не указать, выполняется след. оператор
   case константаM: // в случае выражение==константаM
      операторM1;
      …
      операторMN;
      break; // прервать выполнение конструкции
   default: // если ни одно условие неверно
      операторd1;
      …
      операторdL;
}
```

Возможно указание нескольких констант

```
switch (выражение)
{
    case константа1: // если выражение==константа1
    case константа2: // или выражение==константа2
        операторы12;
        break;
    case константа3: // если выражение==константа3
    case константа4: // или выражение==константа4
    case константа5: // или выражение==константа5
        операторы345;
        break;
    default: // если ни одно условие неверно
        операторы67;
}
```

Применение конструкции switch-case

- Несколько разнотипных условий проверить **нельзя**
- Условия, связанные с вещественными выражениями, проверить **нельзя**
- Сравнивать значение выражений с другими переменными или выражениями **нельзя**
- Поэтому – **только** проверка нескольких жестко заданных вариантов для целочисленного выражения (или выражения, имеющего перечислимый тип, например символа)

Пример – формирование названия оценки

```
#include <iostream>
#include <locale.h>
using namespace std;
// Формирование названия оценки
// по ее численному значению:
// 5 - отлично, 4 - хорошо,
// 3 - удовлетворительно,
// 2 - неудовлетворительно
int main(void)
{
   setlocale(LC_ALL, "Russian");
   int mark; // оценка
   cout<<"Введите оценку: ";
   cin>>mark;
```

Пример – формирование названия оценки

```
    switch(mark)
    {
        case 5:
            cout<<"Оценка отлично"<<endl;
            break;
        case 4:
            cout<<"Оценка хорошо"<<endl;
            break;
            // ...
        default:
            cout<<"Оценки "<<mark<<" не бывает"<<endl;
    }
    return 0;
}
```

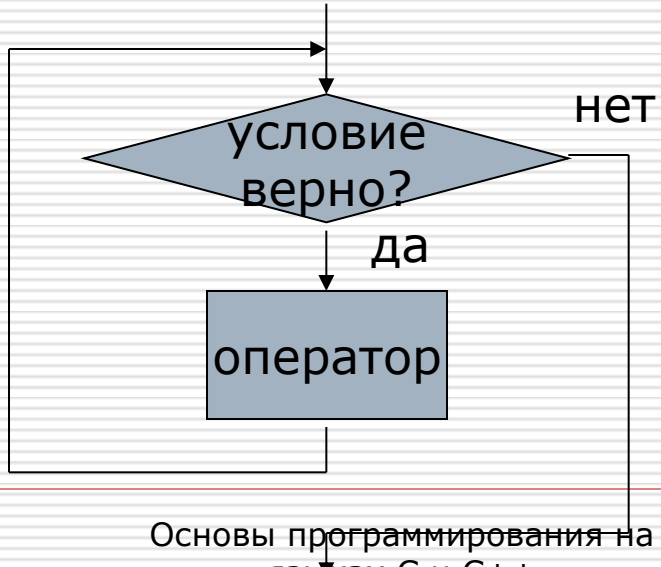
Циклы, основные понятия

- Цикл – конструкция, позволяющая выполнить одну и ту же часть программы последовательно несколько раз
- Тело цикла – многократно повторяемая часть цикла
- Итерация цикла – однократное выполнение тела цикла
- Условие продолжения цикла – условие, при истинности которого тело цикла выполняется еще раз (происходит еще одна итерация)
- Условие окончание цикла – условие, при истинности которого выполнение цикла прекращается

Конструкция while (пока, цикл с предусловием)

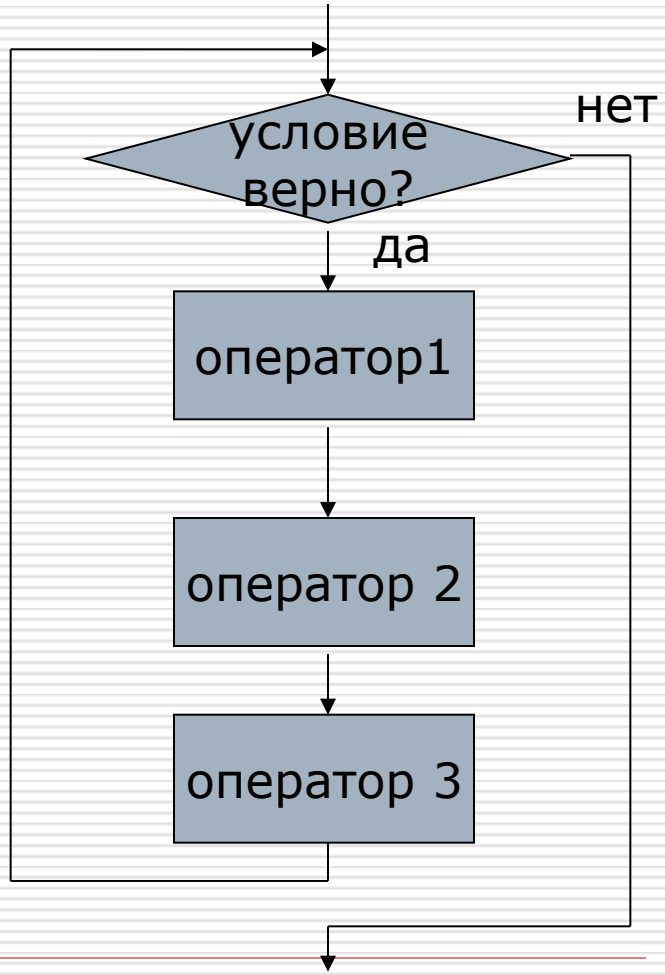
```
while (условие)
    оператор;
```

- Оператор (**один**) будет выполняться, пока условие будет истинно



Составной вариант

```
while (условие)
{
    оператор1;
    оператор2;
    оператор3;
}
```



Важная особенность

- Первый раз условие проверяется **до** выполнения команд цикла
- Если условие сразу оказалось неверно, команды цикла **не выполняются**

Теория алгоритмов доказывает…

- … что, используя последовательное выполнение команд в совокупности с конструкцией ветвления (if-else) и циклом с предусловием (while) **можно реализовать алгоритм решения любой разрешимой задачи**

Немного о вычислимости – машина Тьюринга

- Бесконечная лента с ячейками
- В ячейках лежат числа
- Указатель на одну из ячеек
- Конечное количество состояний
- Команды вида: если мы находимся в заданном состоянии И значение текущей ячейки равно заданному...
  - Изменить состояние
  - И/или изменить значение ячейки на другое
  - И/или сместить указатель влево
  - И/или сместить указатель вправо

Понятия вычислимости и Тьюринг-полноты

- Функция (математическая) вычислима или «формально вычислима», если её можно вычислить с помощью машины Тьюринга
- Тезис Чёрча – Тьюринга: любая «интуитивно вычислимая» функция является также и «формально вычислимой»
- Язык программирования называется Тьюринг-полным, если на нём можно запрограммировать любую вычислимую функцию

Что достаточно для Тьюринг-полноты

- sequence + if + goto
- sequence + if + while
- sequence + if + infinite loops + break
- sequence + if + recursion
- …

Пример решения задачи

- Необходимо составить расписание движения автобуса при следующих условиях:
  - автобус ходит из пункта A в пункт B и обратно, не имея промежуточных остановок
  - на маршруте один автобус
  - время первого отправления (из пункта A) 7:00
  - маршрут должен завершить работу до 19:00, при этом автобус должен оказаться в пункте A
  - известно время пути из A в B в минутах; путь в обратную сторону занимает столько же времени
  - известно время отдыха на конечных остановках, оно одинаковое для пунктов A и B

Вопросы при решении

- Как хранить текущее время, чтобы не возиться с отдельным сложением часов и минут?
  - удобнее всего хранить текущее время в минутах с момента 0:00, а при выводе на экран переводить его в часы и минуты
- В каком случае автобус уедет из пункта А и успеет вернуться обратно до 19:00?
  - если currTime+2*inWayTime+restTime<19*60,
  - где currTime – текущее время в минутах,
  - inWayTime – время в пути от А до В в минутах,
  - restTime – время отдыха в минутах

Текст программы - начало

```
int main(void)
{
   const int minInHour = 60; // константы: минут в часе
   const int startTime = 7*minInHour; // время начала
   const int endTime = 19*minInHour; // время окончания
   int inWayTime, restTime; // время в пути, время отдыха
   setlocale(LC_ALL, "Russian");
   cout<<"Построение расписания движения автобусов"<<endl;
   cout<<"Введите время в пути от A до B в минутах: ";
   cin>>inWayTime;
   cout<<"Введите время отдыха на остановках в минутах: ";
   cin>>restTime;
   cout<<endl<<"Расписание движения"<<endl;
   cout<<"Пункт A        Пункт B"<<setfill('0')<<endl<<endl;
   int currTime = startTime;
```

Текст программы – окончание

```
    while (currTime + 2*inWayTime + restTime < endTime)
    {
        // находимся в A
        cout<<setw(2)<<currTime / minInHour<<":";
        cout<<setw(2)<<currTime % minInHour<<"->";
        currTime += inWayTime; // приехали в B
        cout<<setw(2)<<currTime / minInHour<<":";
        cout<<setw(2)<<currTime % minInHour<<"   ";
        currTime += restTime; // отдохнули
        cout<<setw(2)<<currTime / minInHour<<":";
        cout<<setw(2)<<currTime % minInHour<<"->";
        currTime += inWayTime; // вернулись в A
        cout<<setw(2)<<currTime / minInHour<<":";
        cout<<setw(2)<<currTime % minInHour<<endl;
        currTime += restTime; // еще раз отдохнули
    }
    return 0;
}
```

Ошибки и проблемы

- Что будет, если ввести отрицательные числа?
  - Когда вы пишете программу, которая взаимодействует с пользователем, вы обязаны помнить, что **он всегда прав!** Иначе говоря, в ответ на ваш запрос он может ввести все что угодно. И если программа при этом работает некорректно, то это ваши проблемы.

Ошибки и проблемы

- Что будет, если условие окажется неверным с самого начала? Например, если ввести 500 и 300?
  - Таблица не заполнится, что некрасиво
  - Стоило бы, если условие неверно с самого начала, не выводить заголовок таблицы, а вывести соответствующее сообщение
- Внутри цикла, один и те же команды повторяются четыре раза. Нельзя ли с этим что-то сделать?
  - Можно – реализовав свою функцию вывода времени.

Функция printTime

```
const int minInHour = 60; // минут в часе
// определение функции
void printTime(int time) // заголовок
{
   cout<<setw(2)<<time/minInHour<<":";
   cout<<setw(2)<<time%minInHour;
}
```

Фрагмент программы с вызовом printTime

```
while (currTime + 2*inWayTime + restTime < endTime)
{
   printTime(currTime); // Вызов функции
   cout<<"->"; currTime += inWayTime;
   printTime(currTime); // Вызов функции
   cout<<"   "; currTime += restTime;
   printTime(currTime); // Вызов функции
   cout<<"->"; currTime += inWayTime;
   printTime(currTime); // Вызов функции
   cout<<endl; currTime += restTime;
}
```

Итоги

- Рассмотрели
  - операции C/C++
  - манипуляторы вывода
  - ветвления
  - **while**
- Далее
  - **do…while**, **for**
  - **continue**, **break**
  - …