НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»
Кафедра информатики и процессов управления (№17)

Лекции по информатике и программированию

Лекция 1

Основы программирования на языке C++.

Содержание

3. Основы программирования на C++

3.1. Пример программы
3.2. Оператор `using`
3.3. Встроенное функции (`inline`-функции)
3.4. Ссылки
3.5. Передача в функцию параметров по ссылке
3.6. Параметры функции по умолчанию

1. Язык программирования C++

1.1. Стандарты языка C++

C++98	**ISO/IEC 14882:1998 – имел недостатки**
C++ (C++03)	**ISO/IEC 14882:2003**
C++11	**ISO/IEC 14882:2011**
C++14	**ISO/IEC 14882:2014**
C++17	**ISO/IEC 14882:2017**
C++20	**ISO/IEC 14882:2020**
C++23	**ISO/IEC 14882:2023 – ожидаемый**

1.2. Литература по языку C++

- *Страуструп Б.* **Язык программирования C++.** 3-е издание.
  The C++ Programming Language. 3rd edition.
  4th edition. C++11
- *Страуструп Б.* **Дизайн и эволюция C++.**
  The Design and Evolution of C++.

Bjarne Stroustrup
(Бьярне Строуструп)
(b.1950)

Учебники для начинающих (самоучители):

- *Прата С.* **Язык программирования C++ (C++11). Лекции и упражнения.** (6-е издание.)
  C++ Primer Plus, 6th edition (Developer's Library)
- *Дейтел Х., Дейтел П.* **Как программировать на C++.**
  C++ How to Program. 5th edition.

Справочники:

- *Шилдт Г.* **Полный справочник по C++.** (4-е издание.)
  C++: The Complete Reference.

1.3. Основные возможности языка C++

Язык Си++ предложен **Бьёрном Страуструпом** в начале 1980-х годов как ответ на потребность в промышленном объектно-ориентированном языке программирования.

Си++ представляет собой расширение языка Си и обладает с ним **обратной совместимостью** (почти): программы на чистом Си (стандарт ANSI C, 1989 г.) корректны с точки зрения Си++ (в большинстве случаев, но не всегда).

Парадигма ***объектно-ориентированного программирования*** (ООП):
инкапсуляция, наследование, полиморфизм.

- Описание ***абстрактных типов данных*** (АТД).
- Средства ***инкапсуляции***.
- Обработка ***исключительных ситуаций***.
- ***Наследование***.
- Статический и динамический ***полиморфизм***.
- Обобщенное программирование (***шаблоны***).

Си++ – язык **произвольного** уровня.

1.4. Беглый обзор ООП и АТД.

Объект – «черный ящик», внутреннее устройство которого не доступно извне. Все, что можно – это послать сообщение и получить ответ.

Объекты, внутреннее устройство которых одинаково, образуют ***классы***.

Класс – описание внутреннего устройства объекта. Количество объектов класса может быть произвольным (даже нулевым).

Наследование – создание нового класса на основе имеющегося класса, но с отличиями (как правило в сторону усложнения). Цепочка наследований образует ***иерархию*** классов.

Недоступность деталей реализации объекта за пределами его описания называется ***инкапсуляцией***. Она позволяет снизить сложность программы по принципу «разделяй и властвуй», когда части программы реализуются независимо друг от друга и разрабатываются без учета деталей реализации других частей.

ООП не нужно смешивать с парадигмой АТД.

Абстрактным называется такой тип данных. Для которого неизвестна его внутренняя организация, а известен лишь набор базовых операций.

Например, тип `FILE` из стандартной библиотеки языка Си.

В отличие от объектов в ООП, при работе с АТД нет внутреннего состояния и нет возможности обмениваться сообщениями. АТД можно использовать в императивной парадигме программирования.

2. Некоторые отличия языков С и С++

2.1. Ключевые слова С++

В Си++ добавлены новые ключевые слова.

Полный набор ключевых слов зависит от стандарта языка и производителя компилятора.

Пример: корректный код на Си, но ошибочный на Си++:

```
int try;
```

2.2. Метки структур

В Си++ нет отдельного пространства имен для меток (тегов) структур.

Например: описание метки структуры в Си:

```
struct mystruct {
    int a, b;
};
```

в программе на Си++ является описанием типа данных с именем `mystruct`. В Си++ можно далее описать переменную типа `mystruct`:

```
mystruct s1;
```

а в чистом Си придется использовать метку структуры:

```
struct mystruct s1;
```

Но описания на Си вида:

```
typedef struct mystruct {
    int a, b;
} mystruct;
```

в языке Си++ вызывают ошибку конфликта имен.

2.3. Преобразование типов адресных выражений

В Си++ ошибкой является неявное преобразование выражений адресных типов.

Например, присваивание выражения типа `int*` переменной типа `double*`.

В Си++ ошибкой является корректное для Си неявное преобразование выражения типа `void*` в выражение другого адресного типа.

2.4. Нулевой указатель

В чистом Си для обозначения нулевого указателя используется макрос `NULL`, а в Си++ – целочисленный ноль.

Однако макрос `NULL` в Си++ использовать можно (например, для совместимости с кодом на Си), но это не принято.

2.5. Логический тип данных

В Си++ введен логический тип данных `bool` со значениями `true` и `false`.

Целочисленной переменной можно присвоить значение типа `bool` тогда `false` примет значение ноль, а `true` – единица.

2.6. Символьные константы

Символьные константы (например, `'a'` или `'7'`) в Си++ считаются константами типа `char`, а не `int` как в Си.

2.7. Описание переменных

Описание переменных (и типов данных) в Си++ является **оператором**. Это позволяет использовать его в произвольном месте программы, даже в заголовке цикла., например:

```
for (int i=0; i<10; ++i)
    ...
```

3. Основы программирования на C++

3.1. Пример программы

Сложение двух целых чисел:

```
#include <iostream>   // заголовочный файл потоков ввода-вывода

int main() {

    int a;

    std::cout << "Input a:\n";  // вывод (на экран)
    std::cin >> a;              // ввод (с клавиатуры)

    int b, sum;      // объявления переменных

    std::cout << "Input b:\n";
    std::cin >> b;

    sum = a + b;
    std::cout << "a + b = " << sum << std::endl; // конкатенация, сцепление

    return 0;
}
```

`<<` – операция передачи в поток,
`>>` – операция извлечения из потока,
`std::cout` и `std::cin` – ***объекты*** стандартного потока,
`std::endl` – ***манипулятор*** потока (**end** of **l**ine), выводит символ новой строки и сбрасывает буфер ввода.

3.2. Оператор `using`

Та же программа:

```
#include <iostream>   // заголовочный файл потоков ввода-вывода
using std::cout;
using std::cin;
using std::endl;
int main() {
    ...
    cout << "Input a:\n";  // вывод (на экран)
    cin >> a;              // ввод (с клавиатуры)
    ...
    cout << "a + b = " << sum << endl; // конкатенация, сцепление
    ...
}
```

Разрешение использования любых функций стандартной библиотеки:

```
#include <iostream>   // заголовочный файл потоков ввода-вывода
using namespace std;
int main() {
    ...
    cout << "Input a:\n";  // вывод (на экран)
    cin >> a;              // ввод (с клавиатуры)
    ...
    cout << "a + b = " << sum << endl; // конкатенация, сцепление
    ...
}
```

3.3. Встроенное функции (`inline`-функции)

Пример программы со встроенной функцией:

```
#include <iostream>
using namespace std;
inline double cube(const double a) {
    return a*a*a;
}
int main() {
    double side;
    for (int k=1; k<4; k++) {
        cout << "Input side of cube:\n";
        cin >> side;
        cout << "Volume of cube with side " << side
             << " is " << cube(side) << endl;
    }
    return 0;
}
```

Квалификатор `inline` рекомендует компилятору генерировать в месте вызова функции копию ее кода (если это возможно), чтобы не производить вызова функции, который увеличивает время исполнения программы.

Переменная `k` объявлена в заголовке цикла `for` и не доступна за пределами тела данного цикла.

3.4. Ссылки

Ссылки – псевдонимы других переменных.

Пример:

```
int a = 1;       // объявление переменной a
int &aRef = a;   // объявление ссылки как псевдонима a
...
++aRef;          // приращение переменной a через псевдоним
```

Ссылка всегда должна быть инициализирована при объявлении:

```
int a = 1;
int &aRef;    // ошибка!
```

Исключение – случай, когда ссылка является параметром функции.

3.5. Передача в функцию параметров по ссылке

Ссылка инициализируется в момент вызова функции (значением фактического параметра (аргумента), передаваемого в функцию при вызове).

```
#include <iostream>
using std::cout;
using std::endl;
int  squareByVal(int);
void squareByRef(int &);
int main() {
    int x=3, y=4;
    cout << "x^2=" << squareByVal(x) << endl;
    squareByRef(y);
    cout << "y^2=" << y << endl;
    return 0;
}
int squareByVal(int a) {  // передача по значению
    return  a *= a;       // переменная x не изменится

}
void squareByRef(int &aRef) {  // передача по ссылке
    aRef *= aRef;              // переменная y изменится
}
```

3.6. Параметры функции по умолчанию

При вызове функции можно не передавать в нее фактические параметры (по значению), а инициализировать формальные параметры значениями, указанными в прототипе функции по умолчанию.

```
#include <iostream>
using std::cout;
using std::endl;
int volume(int length=1, int width=1, int heigth=1);
int main() {
    cout << "Volume =" << volume(10) << endl;       // =10*1*1=10
    cout << "Volume =" << volume(10,5) << endl;     // =10*5*1=50
    cout << "Volume =" << volume(10,5,2) << endl;   // =10*5*2=100
    return 0;
}
int volume(int length, int width, int heigth) {
    return  length*width*heigth;
}
```