Введение в C++

Лекция #1

Пустовалова О.Г.
доцент. каф. мат.мод.
ИММиКН ЮФУ

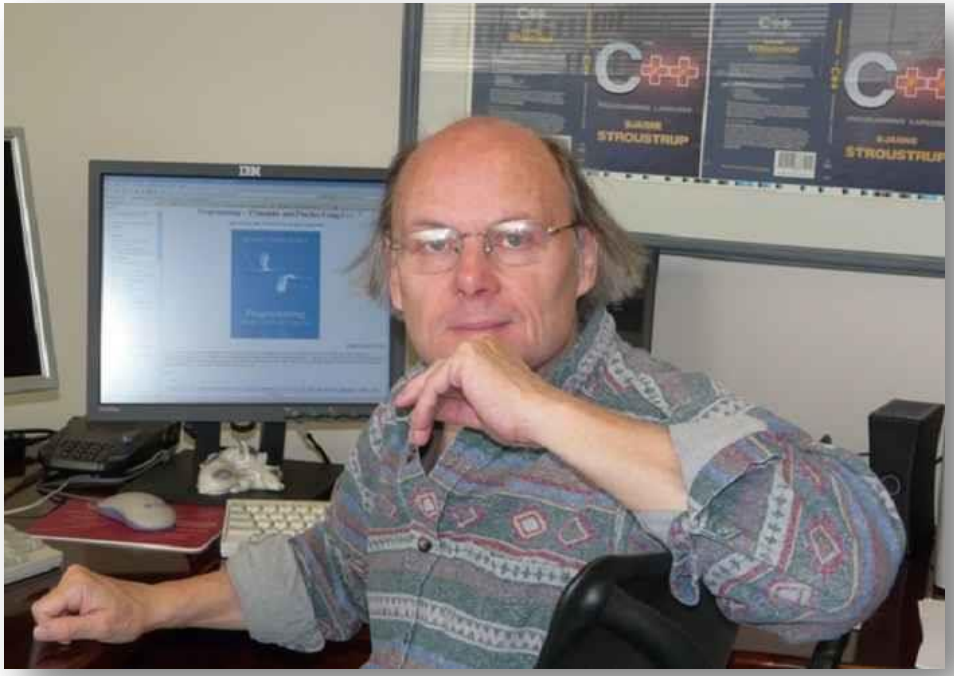
Содержание

О языке программирования C++

- C++ компилируемый язык программирования общего назначения, сочетает свойства как **высокоуровневых**, так и **низкоуровневых** языков программирования.
- В сравнении с его предшественником, языком программирования Си, наибольшее внимание уделено поддержке **объектно-ориентированного** и **обобщённого** программирования.
- Название «язык программирования C++» происходит от языка программирования C, в котором унарный оператор **++** обозначает **инкремент** переменной.

История возникновения языка программирования C++

Язык программирования C++ был создан в начале 1980-х годов, его создатель сотрудник фирмы Bell Laboratories — **Бьёрн Страуструп**.



Бьёрн Страуструп

- Книга Страуструпа «Язык программирования C++» — одна из самых широко читаемых книг из своей области, которая была переведена на 19 языков.
- Следующая книга, «Дизайн и эволюция C++», открыла много нового в описании языков программирования: новые идеи, идеалы, проблемы.
- В дополнение к своим пяти книгам, Страуструп опубликовал более сотни академических и других популярных статей.
- Бьёрн Страуструп был избран членом Национальной Академии Инженерии в США («National Academy of Engineering») в 2004 году за «создание языка C++».

Бьёрн Страуструп (дат. Bjarne Stroustrup, ˈ род. 30 декабря 1950 (Орхус, Дания), автор языка программирования C++.

Какие программы написаны на C++

- http://www.stroustrup.com/applications.html - список программ, написанных на C++

C++ Applications

Modified August 17, 2014

Here is a list of systems, applications, and libraries that are completely or mostly written in C++. Naturally, this is not intended to be a complete list. In fact, I couldn't list a 1000th of all major C++ programs if I tried, and this list holds maybe 1000th of the ones I have heard of. It is a list of systems, applications, and libraries that a reader might have some familiarity with, that might give a novice an idea what is being done with C++, or that I simply thought "cool".

О рейтинге C++

- https://www.tiobe.com/tiobe-index// - рейтинг языков программирования

Jan 2018	Jan 2017	Change	Programming Language	Ratings	Change
1	1		Java	14.215%	-3.06%
2	2		C	11.037%	+1.69%
3	3		C++	5.603%	-0.70%
4	5	^	Python	4.678%	+1.21%
5	4	v	C#	3.754%	-0.29%
6	7	^	JavaScript	3.465%	+0.62%
7	6	v	Visual Basic .NET	3.261%	+0.30%
8	16	^^	R	2.549%	+0.76%
9	10	^	PHP	2.532%	-0.03%
10	8	v	Perl	2.419%	-0.33%
11	12	^	Ruby	2.406%	-0.14%

Основные этапы развития C++

В **1979-80** годах Бьерн Страуструп разработал расширение к языку Си - **"Си с классами"**.

В **1983** язык был переименован в **C++.**

В **1985** году была выпущена первая коммерческая версия языка C++, а также первое издание книги "Языка программирования C++", которая представляла первое описание этого языка при отсутствии официального стандарта.

В **1989** была выпущена новая версия языка **C++ 2.0**, которая включала ряд новых возможностей.
Первый стандарт получил название ISO/IEC 14882:1998 или сокращенно **C++98**

Основные этапы развития C++

В **2003** была издана новая версия стандарта **C++03**.

После этого язык развивался относительно медленно вплоть до 2011 года.

В **2011** году был издан новый стандарт **C++11**, который содержал множество добавлений и обогащал язык C++ большим числом новых функциональных возможностей.

В **2014** году было выпущено небольшое добавление к стандарту, известное также как C++14.

2017

Достоинства C++

- Высокая совместимость с языком Си : код на Си может быть с минимальными переделками скомпилирован компилятором C++

- Вычислительная производительность.

- Один из базовых принципов C++ — «*не платишь за то, что не используешь*» — никакое языковое средство не должно приводить к снижению производительности программ, не использующих его.

Достоинства C++

- Поддержка различных стилей программирования:
  - традиционное структурное, объектно-ориентированное
  - обобщённое программирование,
  - функциональное программирование,
  - порождающее метапрограммирование.

Функциональные языки
- Scheme
- Common Lisp
- Erlang
- Elixir
- APL
- Scala
- Miranda
- Haskell

Языки, включающие **обобщенное программирование**
- C++
- Java
- .NET
- D
- Object Pascal
- Nim.

Недостатки C++

- Сложность и избыточность, из-за которых C++ трудно изучать. Для новичков C++ имеет высокий порог вхождения.

- Ограниченные возможности функционального программирования по сравнению с языками чисто-функциональными.

- Метапрограммирование на основе шаблонов C++ сложно и при этом ограничено в возможностях. Менее распространённые языки Lisp/Scheme, Nemerle имеют более мощные и одновременно более простые для восприятия подсистемы метапрограммирования.

Привет, Мир!

```
// подключение библиотеки
#include <iostream>
// использование пространства имен
using namespace std;
// main() - главная программа (функция)
int main()
{   // подключение русского языка
setlocale(0, "");
// вывод на экран
cout << "Привет, Мир!" << endl;
return 0;
}
```

О пространстве имен

```
#include <iostream>

int main(){

   setlocale(0, "");
    std::cout << "Привет, Мир!" << std::endl;
return 0;
}
```

Базовые типы данных C++

`int` – целочисленный тип данных

`float` – тип данных с плавающей запятой

`double` – тип данных с плавающей запятой двойной точности

`char` – символьный тип данных

`bool` – логический тип данных

`unsigned` `int`

`short` `int`

`long` `int (double)`

Базовые типы данных C++

```
#include <iostream>
using namespace std;

int main()
{
cout << "int = " << sizeof(int) << endl;   //4
cout << "float = " << sizeof(float) << endl;  //4
cout << "double = " << sizeof(double) << endl;//8
cout << "char = " << sizeof(char) << endl;    //1
cout << "bool = " << sizeof(bool) << endl;    //1
return 0;
}
```

Описание и инициализация переменных

```
#include <iostream>
int main()
{
double a;                 // описание
a = 123;                  // инициализация
double b { 1.2 };         // новый стиль
double c = { 4.5 }; // новый стиль
std::cout<<a<<" "<< b << " "<< c << std::endl;
return 0;
}
```

Неявное преобразование типов

```
#include <iostream>
using namespace std;

int main()
{
int a = 1.3;       // 1
double b = 7 / 2; // 3
//int c = { 1.3 };// - ошибка компиляции
return 0;
}
```

Явное преобразование типов

```
#include <iostream>
using namespace std;

int main()
{
int a =static_cast<int>(1.3);// 1 новая форма
double b = double(7) / 2;   //  3.5 старая форма
return 0;
}
```

Ввод данных с помощью клавиатуры

```
#include <iostream>
using namespace std;

int main()
{
setlocale(0, "");
int a;
cout << "Введите целое число : ";
// ввод в переменную a
cin >> a;
double b;
cout << "Введите вещественное число : ";
// ввод в переменную b
cin >> b;
cout << "Сумма = "<<a+b<<endl;
return 0;
}
```

Арифметические операции и операции сравнения

+	сложение;
-	вычитание;
*	умножение;
/	деление;
%	**остаток от деления**

Условие	Отношение
a == b	a равно b
a != b	a не равно b
a > b	a больше b
a < b	a меньше b
a >= b	a больше или равно b
a <= b	a меньше или равно b

Логические операции

Операция	Пример применения
И &&	(5>3)&&(4<7)
ИЛИ ||	(0==0)||(3>7)
НЕ !	!(5==4)

```
#include <iostream>
using namespace std;

int main()
{
cout << (true && true) << endl; // 1
cout << (false && true)<<endl;  // 0
cout << (true && false) << endl;// 0
cout << (false && false) << endl;//0
return 0;
}
```

Поразрядные логические операции

Операция	Обозначение
поразрядное И	**&**
поразрядное исключающее ИЛИ	**^**
поразрядное включающее ИЛИ	**|**
~ – поразрядная инверсия (НЕ).	**~**

```
#include <iostream>
using namespace std;

int main()
{   // проверка на нечетность
cout << (5&1) << endl; //
    // проверка на четность
cout << (!(4&1)) << endl;//
return 0;
}
```

Операции сдвига

Операция и пример	Действие
<< - сдвиг влево **(1<<5)** // 2 в 5-й степени	сдвиг влево значения операнда на заданное количество бит. Операнд размещается слева от знака операции. Число сдвигаемых бит указывается справа от знака операции. Выдвижные биты переходят в старшие разряды, «входят» нулевые биты.
>> - сдвиг вправо **(8>>2)** // получим 2	сдвиг вправо значения операнда на заданное количество бит. Операнд размещается слева от знака операции. Количество сдвигаемых бит размещается справа от знака операции. Выдвижные биты теряются, а «входят» нулевые биты.

```
cout << (1<<5) << endl;
```

Операции инкремента и декремента

Операция	Обозначение
Префиксный инкремент	**++a**
Постфиксный инкремент	**a++**
Префиксный декремент	**--a**
Постфиксный декремент	**a--**

```
int a = 100;
cout << "a = " << a << endl;   //100
cout << "a++ " << a++ << endl; //100
cout << "++a " << ++a << endl; //102
```

Множественное присваивание c=b=a

```
#include <iostream>
using namespace std;
int main()
{
int a = 100;
int b, c;

// Справа на лево
c = b = a;
```

```
cout << a<<" "<< b<<" "<<c << endl; // 100 100 100
return 0;
}
```

Операция запятая

```
#include <iostream>
using namespace std;
int main()
{
int x,y;
x = (y = 3, y =-100*y);
cout << x; //-300
return 0;
}
```

Оператор «запятая» вызывает выполнение последовательности действий. Когда он используется с правой стороны оператора присваивания, то присваиваться будет значение последнего выражения, стоящего в разделенном запятыми списке.

Тернарный оператор

"условие" ? "выражение 1" : "выражение 2";

Если условие истинно, то выполняется **выражение 1**,

иначе (условие ложно) выполняется **выражение 2**.

```
#include <iostream>
using namespace std;
int main()
{
int x,y;
cin>>x>>y;
cout << ((x>y)?x:y)<<endl; //Большее из x и y
return 0;
}
```

Тернарный оператор

```
#include <iostream>
using namespace std;
int main()
{
int x;
cin>>x;

((x>100)?cout<<"1":
((x>0)&&(x<=100)? cout << "2": cout << "3"))<<endl;

return 0;
}
```

Условный оператор

```
cin >> num;
if (num > 0) {
        cout << "Число положительное" << endl;
}
else {
        cout << "Число не является положительным."
}
```

```
cin >> num;
if (num > 0) {
        cout << "Число положительное" << endl;
}
```

Цикл for

```
for (счетчик = значение1; счетчик < значение2; шаг цикла) {
        тело цикла;
}
```

```
// Бесконечные циклы
for (; ;);
for (int i = 1; ; ) { i++;}
for (int i = 1; ; i++ ) { cout<<i; }
```

Цикл for. Примеры

```
// Целые числа от 0 до 9
for (int i = 0;i<10 ; i++ ) { cout << i<<" "; }


// Вещественные числа
for (double i = 0.1; i<=1; i+=0.1) { cout << i<<" "; }


// Символы
for (char i = 'a'; i<='z' ; i++ ) { cout << i; }


// Коды символов
for (int i = 'a'; i <= 'z'; i++) { cout << i; }
```

Функция clock

```
#include <iostream>
#include <ctime>
using namespace std;
int main()
{
setlocale(0, "");
double s = 0.0;
unsigned int start = clock();
for (int i = 1; i < 1E+8; i++)
s += i;
unsigned int stop = clock();
cout << "Время работы программы = " <<
stop - start << endl;
return 0;
}
```

Генерация случайных чисел

```
// генерация числа в диапазоне от 1 до n
rand() % n+1;
```

```
#include <iostream>
#include <ctime>
int main() {
// инициализация функции rand значением функции time
srand(time(NULL));
std::cout << rand() % 2 ;
return 0;
}
```

Цикл while

```
while (/*условие продолжения цикла*/)
{
  /*операторы*/;
  /*управление условием*/;
}
```

```
// бесконечный цикл
int i = 0;
while (true) {
       i++;
}
```

```
int i = 0;
while (i<10) {
       i++;
}
```

Цикл do while

```
int i = 10;
do {
        i--;
}
while (i > 0);
```

```
// бесконечный цикл
int i = 10;
do {
i--;
} while (true);
```

Оператор выбора switch

```
switch (/*переменная или выражение*/)
{
        {       case /*константное выражение 1/*:
                    /* операторы */;
                break;
        }
        {
                case /*константное выражение 2/*:
                    /* операторы */;
                break;
        }
        default:
        {
            /* операторы */;
        }
}
```

Оператор выбора switch. Пример 1

```
srand(time(NULL));
int x = rand() % 3;

switch (x)
{
	case 1:{
		cout << 1;
		break;}

	case 2:{
		cout << 2;
		break;}

	default:cout << 0;
}
```

Оператор выбора switch. Пример 2

```
srand(time(NULL));
int x = rand() % 3;

switch (2*x-1)
{
	case 1:{
		cout << 1;
		break;}

	case 2:{
		cout << 2;
		break;}

	default:cout << 0;
}
```

Что произойдет, если пропустить оператор `break`?

Форматированный вывод данных

В библиотеке **iomanip** существует несколько функций, которые позволяют задавать форматированный вывод на экран

```
#include <iostream>
// библиотека манипулирования вводом/выводом
#include <iomanip>
using namespace std;
int main()
{
double a = 7.1234;
cout << setfill('~')    // заполнитель
 << setw(10)            // ширина
 << setprecision(5)     // точность
 << right               // выравнивание
 << a << endl;
return 0;
}
```

Спасибо за внимание!