САОД

А. Задорожный

2017

Содержание

- История и место C++
- Hellow, World!
- Процесс построения программы
- Консольный ввод-вывод
- Препроцессор C++
- Функции и передача параметров
- Указатели и массивы
- Z-строки

История и место C++

- C -> C++, Objective C.
- ~C -> Java, C#, PHP, Perl, Java script, …
- Native код (*выполняется непосредственно под операционной системой*)
- Переносимость, особые требования к производительности, кросс-компиляторы

Hello, World!

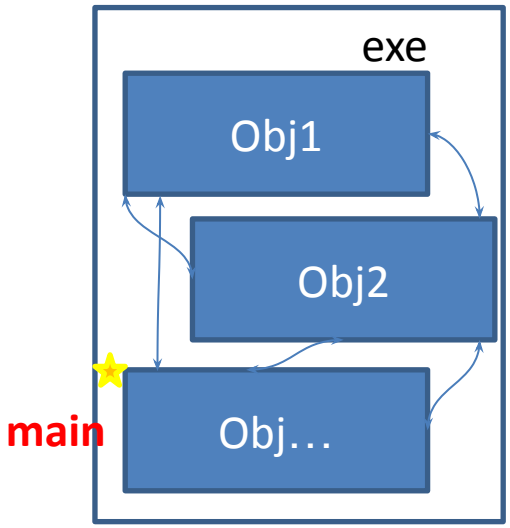
```
#include <iostream> //заголовок (*.h, *.hpp)

using namespace std;

int main(int argc, char* argv[])
{
    cout << ”Hello, World!” << endl;
    return 0;
}
```

Построение и выполнение программы

- Препроцессирование
- Компиляция
- Сборка
- Загрузка и выполнение



Препроцессирование

Предварительная обработка текстовых файлов исходного кода и подготовка текстов программы для последующей компиляции.

- на входе исходники C++,
- на выходе тексты C++, готовые е компиляции.

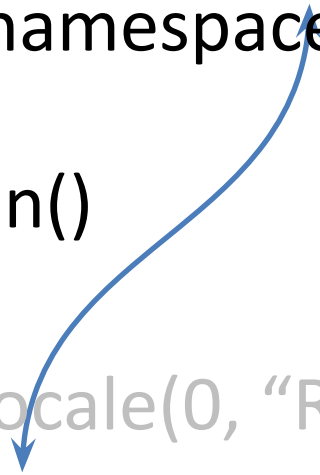
Инструкции #include приводят к тому, что тексты, передаваемые на компиляцию имеют большой объем, а их обработка требует много времени.

Каждая среда разработки решает эту проблему по своему. Visual Studio создает прекомпилированные заголовочные файлы (stdafx.h). #include “stdafx.h” – должно быть первой строкой в каждом *.cpp.

Hello, World! - 2

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

int main()
{
    setlocale(0, “Russian”);
    cout << ”Hello, World!” << endl;
    return 0;
}
```



Ввод данных и потоки

```
int main()
{
    cout << ”? ”;
    double x;
    cin >> x;

    cout << “x^2 = ” << x*x << endl;
    return 0;
}
```

```
>? 2
>x^2 = 4
```

Препроцессор

#include <iostream>/“iostream”

#define MY_KEY
#undefine MY_KEY

#define max(a,b) ((a)>(b)?(a):(b))
Если нужно переносить строки, то в конце ставится обратная косая черта – ‘\’.

#ifdef MY_KEY
    Текст, если раньше было определено MY_KEY
#else
    Текст, если не было определено MY_KEY
#endif

Контрольные вопросы

1. Из каких шагов состоит построение программы на C++?
2. Каковы действия ОС для выполнения программы?
3. Можно ли создать C-программу без единой функции?

4. Как в C++ вывести переменную типа double на консоль с использованием стандартной библиотеки шаблонов?
5. Что означает имя *endl* в стандартной библиотеке C++?
6. Как выполнить чтение целого числа с консоли в переменную y?

7. Назовите инструкции препроцессора, рассмотренные в лекции.
8. Как в тексте программы обнаружить инструкции препроцессора?
9. Что определяет заголовочный файл *iostream*?

Встроенные типы данных

- int … uint …, double … и char - числовые
- bool, byte …

```
char ch = ‘a’;
bool f = true;//false; В качестве bool может выступать значение любого типа!
```

- указатели и массивы - производные типы.

Операции над числами

```
int x = 6, y = 5;
cout << x << " + " << y << "= " << x + y << endl;
cout << x << " - " << y << "= " << x - y << endl;
cout << x << " * " << y << "= " << x * y << endl;
cout << x << " / " << y << "= " << x / y << endl;
cout << x << " % " << y << "= " << x % y << endl;

cout << x << " | " << y << "= " << (x | y) << endl;
cout << x << " & " << y << "= " << (x & y) << endl;
cout << x << " ^ " << y << "= " << (x ^ y) << endl;

cout << x << " >> " << y << "= " << (x >> y) << endl;
cout << x << " << " << y << "= " << (x << y) << endl;

cout << "++x = " << ++x << endl;
cout << "~"<< x <<" = " << ~x << endl;

return 0;
```

Имеется так же полный набор операций **x <операция>= …**

Операции и выражения

- Выражение (expression) – конструкция, которая может быть вычислена и примет определенное значение.
- Выражение можно использовать как величины в другом выражении.
- В C++ (как и в C-подобных) многие операции неожиданно являются выражениями. Например, операция присваивания x = y. Поэтому можно x = y = z = 3;

Из-за такого соглашения можно if(x = 3) … что часто приводит к ошибкам!

- Аналогично операции << для cout и >> для cin.

Операции над числами (битовые маски)

#define MAIN 1

#define STAFF 2

#define PERMANENT 4

- MAIN | STAFF

- MAIN | PERMANENT

Функции

```
int main (int argc, char * argv[])
{
    return 0;
}
```

- Сначала объявление, а потом использование
- Прототип, заголовочные файлы и рекурсия
- Функции C++ и статические методы в C# (.Net)

Передача параметров

```
void Swap (int a, int b)
{
        int c = a;
        a = b;
        b = c;
}

int x = 2, y = 3;
Swap(x, y);              // Не сработает

void Swap (int &a, int &b) // Параметры – ссылки
```

Ссылки – алиасы переменных
Синтаксически их использование не отличается от обычных переменных

```
int x = 3, &y = x;
```

Теперь x и y – синонимы.

Пример распределения кода по файлам

Test.h ----------

```
void Swap (int &a, int &b); // Прототип функции
```

Test.cpp ----------

```
void Swap (int &a, int &b)
{
        int c = a;
        a = b;
        b = c;
}
```

Для применения функции в CPP-файле:

```
#include “test.h” // В Unix имя файла чувствительно к регистру. В Windows - нет
…
Swap(x, y);
…
```

Контрольные вопросы

1. Назовите 4 основных типа данных, определенных в языке C++.
2. Дайте определение понятию “выражение” (expression) в языке программирования.
3. **Чему равны переменные x и y после выполнения оператора x = 2 * (y +=3), если до его выполнения x равнялся 0, а y – единице?**
4. **Будут ли численно равны выражения: 9/5 и 9.0/5.0?**
5. **Чему численно будет равно выражение 2 | N, если перед этим стоит инструкция #define N 5? Чему будет равно N & 2?**
6. Какие из следующих условных выражений эквивалентны:
   a. (x) c. (!x)
   b. (x == 0) d. (x != 0)
7. Возникнет ли синтаксическая ошибка, если в условном операторе написать (x = 0)?
8. Можно ли в C++ написать x = y = 3? Почему?
9. Можно ли вызвать void foo(int & a) следующим образом: foo(x+y); Почему?
10. Как в C# обошлись без функций?

Указатели

```
int x = 5;
int *p = &x;    // указатель указывает на память переменной x (его значение - адрес x)

cout << *p << endl; //5 - разыменование
(*p)++;             // увеличили значение
cout << *p << endl; //6
cout << x << endl;  //6 изменилась x
```

Операции над указателями

*p – разименование

p++ - переход к следующему элементу того типа, на который указывает указатель (p--)

p+i - сложение (вычитание) с целым – получаем указатель на i элементов правее (левее).

p – t - вычитание двух указателей одного типа - целое число, количество элементов между указателями.

p==t, p< t, p<=t, p>t, p>=t, p!=t

0 – указатель имеющий значение 0.

Указатели и массивы

```
int m[5] = {1,2,3,4,5};
int *p = m;    //int *p = &x; для массива амперсанд не нужен!
```

- Имя массива – указатель на первый элемент!

```
for (int *p = m; p < m + 5; p++) //p++  для указателя передвигает на след.
      *p = *p * 2;

for (int i = 0; i< 5; i++)  // p + i сдвигает на i-тый элемент, а не байт
      cout << *(m + i) << endl;

for (int i = 0; i< 5; i++)
      cout << m[i] << endl;   // p[i] тождественно *(p + i).
```

Правила интерпретации и объявления типов в C++

Объявление типов в C (C++) бывает сложным

int *(*p[])(int, int *)

В этом объявлении найти имя переменной - p, затем движемся вправо до окончания объявления или закрывающей скобки: (*p[]) – “p – это массив”, затем движемся влево до окончания объявлений или открывающей скобки - “p – это массив указателей”. Переходим ко внешнему объявлению и снова движемся вправо: (int, int *) – “p – это массив указателей на функцию с двумя аргументами”, и наконец переходим влево – “, которая возвращает указатель на целое”.

Окончательно имеем:

p – это массив указателей на функции с двумя аргументами, которые возвращают указатель на целое

Массивы 2

- Многомерных массивов нет

```
int v[5][3]; // см. правило объявления типов
```

- Размер массива узнать нельзя.

- Если массив объявлен в текущем блоке, то общее количество элементов можно вычислить с помощью:

```
sizeof(v)/sizeof(int);
```

Массивы символов и z-строки

- Длина z-строки

```
char s[32] = “Hi, Kristy!”, *p;
for (p = s; *p; p++);
int n = p – s;
```

- Копирование z-строки

```
char s[32] = “You are a crazy driver!”, t[32];
for (char *p=s, *d=t; *d++ = *p++; );
```

Массивы символов и z-строки

- Сравнение z-строк

char s[32] = **“You are a crazy driver!”,** t[32]= **“Hi, Kristy!”**, *p, *d;

for (p=s, d=t; (*d == *p) && *p; d++, p++);

cout << *p-*d << endl;

Контрольные вопросы

1. Проинтерпретируйте объявление второго параметра main.
   int main(**int** argc, **char*** argv[])
2. Сколько операций умножения в операторе *p = *p * 2?
3. Сколько операций разыменования в операторе *p = *p * 2?
4. Перепишите оператор *p = *p * 2, используя только 2 звездочки. А одну?
5. Какие 3 различных смысла в C++ имеет символ &?
6. Какие 3 различных смысла в C++ имеет символ *?
7. Можно ли в C сделать так, что бы первый элемент массива имел индекс 2? А 0?
8. Если массив w проинициализирован значениями 1, 2, 3. Чему равно значение выражения *w?
9. Чему равна величина p + i, где p – указатель на массив целых, а i – целое число?
10. Чему равна величина p – d, где p и d – указатели на величины типа double?
11. **Поставьте эксперимент и выясните, чему равны элементы массива w для каждого из определений: int w[5]; и int w[5] = {}.**