Введение в C++

Актуальность C++

Структура программы на C++

Ввод/вывод

Базовые типы в C++

История языка C/C++

- Язык программирования C++ создан на базе языка Си.
- Язык программирования C разработан в 1973 Кеном Томпсоном и Деннисом Ритчи (Bell Labs)
- Язык создан для использования в ОС Unix
- Получил широкое распространение и оказал существенное влияние на многие новые языки (Си-подобные)
- В настоящее время язык C остаётся одним из самых распространенных
- Особенности языка C:
  - Эффективность
  - Простота (процедурный стиль без «лишних» абстракций)

Появление C++

- Бьёрн Страуструп (Bell) с начала 80х разрабатывал расширение языка C для работы с абстракциями ООП (классы)
- C with classes
- Первый компилятор выполнял трансляцию в исходный код C
- К 1983 в язык добавлено много новых возможностей (виртуальные функции, перегрузка операторов, ссылки, константы и пр.)
- Язык переименован в “C++”
- «Язык является расширением C и не пытается устранять проблемы путём удаления элементов C»

Стандарты языка C++

- Первый стандарт языка C++ в 1998 г.

  ISO/IEC 14882:1998 “Standard for the C++ Programming Language”

- C++ 11

  вывод типов (auto); лямбда выражения; средства многопоточного программирования

- C++ 14
- C++ 17

  параллельные версии алгоритмов в STL

- Язык C также расширен с учетом некоторых новшеств из C++ (, стандарт ANSI C, 1999 г.). Такую версию языка называют **C99**

C++ мультипарадигмальный

- Язык C++ допускает программирование в разных парадигмах и их совмещение:
  - **процедурное программирование**
    (код в стиле C)
  - **объектно-ориентированное программирование**
    (классы, наследование, полиморфизм и др.)
  - **обобщенное программирование**
    (шаблоны функций и классов, библиотека STL)
  - **функциональное программирование**
    (лямбда-выражения, функторы)
  - **генеративное программирование**
    (метапрограммирование)

C++ эффективный

- Язык C++ является одним из самых быстродействующих языков («иногда может быть быстрым как C»)
- Эффективность достигается за счёт того, что:
  - C/C++ является **компилируемым** языком.

  Компиляция осуществляется непосредственно в машинные инструкции для данной вычислительной системы

  Разница с
  - Программы на C/C++ выполняются **без «виртуальной машины»**, **без сборки мусора** (без автоматического управления памятью)
  - **Уровень абстракции выбирает сам программист**. Можно писать высокоуровневый ООП-код или ФП-код, но с дополнительными накладными расходами; можно отдельные фрагменты кодировать без «тяжелый» абстракций.

C++ современный

- Язык C/C++ продолжает быть одним из самых популярных и востребованных языков

  «По индексу TIOBE язык 2019 года – C»

- Язык C/C++ применяется:
  - как **системный язык** для работы с оборудованием (драйверы ОС)
  - **как язык высокопроизводительных вычислений**

    Множество библиотек (в т.ч. для других языков) написано на C/C++ :

    - для математических, инженерно-технических расчетов (MKL, IPP)
    - для машинного обучения (tensorflow, mxnet, cntk, tiny-dnn, DAAL и др.)

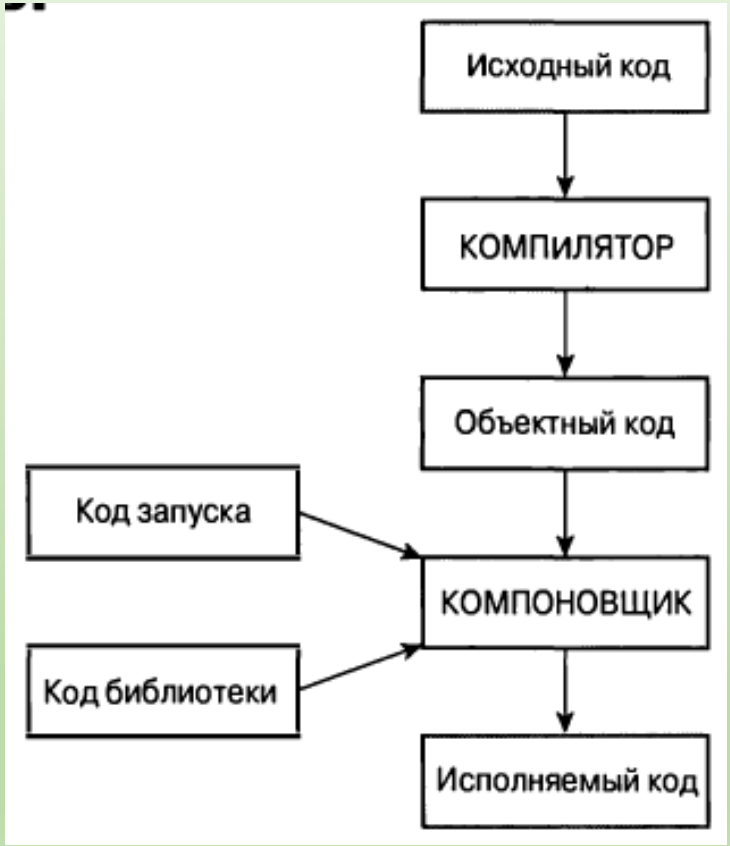
    Язык C/C++ предоставляет доступ к GPU для эффективных вычислений
- Многие современные языки программирования создавались на базе C++: Java, C#, Rust

Компиляция программы

- Исходный код: файлы *.cpp, *.h, *.cc, *.cxx
- **Компилятор** выполняет преобразование исходного кода на C++ в машинные инструкции, оптимизируя их для конкретной вычислительной платформы (тип процессора, операционная система)
- Компилятор порождает «объектный код»: *.o

Скомпилированный в машинные инструкции код без внешних библиотек

- **Компоновщик** связывает «наш» объектный код с объектным кодом библиотек и порождает исполняемый код



Директивы препроцессора

- Язык C/C++ поддерживает *препроцессорную обработку*
- Препроцессор – это программа, которая выполняет обработку файла исходного кода перед началом компиляции
- Директивы препроцессора начинаются с символа #

```
#include <Windows.h>
#define CORE_NUMBER 4
#define IntPtr int*
#define DEBUG
```

Hello world

```
// включаем файл iostream
#include <iostream>
// подключаем пространство имен std (нужно для cout, endl)
using namespace std;
// точка входа в приложение
int main()
{
	// Вывод на консоль в стиле C
	printf("Привет мир!\n");
	// Вывод на консоль в стиле C++
	cout << "Hello, world" << endl;
	return 0;
}
```

#include

- Директива #include – заставляет препроцессор добавить (включить) содержимое указанного файла в программу
- Традиционно «включаются» объявления типов, функций, констант
- Можно включать библиотечные файлы

  #include <stdio.h>
- Можно включать «свои» файлы:

  #include «myHelperFunctions.h»
- Необходимость подключения «заголовочных» файлов связана особенностью C/C++:

  В коде выполнения можно использовать только то, что уже **ранее объявлено** (выше в файле)

Разделение на объявление и определение

- В C/C++ часто *отделяют* **объявление** от **определения** для любой сущности (переменная, тип, функция)

- **Объявление** (declaration) вводит имена типов, функций, не раскрывая информацию о хранении и особенностях реализации

- **Определение** полностью *определяет* сущность; в случае функций - предоставляет реализацию (тело функции).

- Любой отдельный файл *.cpp программы будет успешно скомпилирован, если все имена типов, функций *заранее объявлены* (но не обязательно определены) = *известны компилятору*

```
int f1(int);   // объявление функции f1
int f2(int y);   // объявление функции f2 = "прототип функции"

int main()    // определение функции main (включает и объявление)
{
      int answer = f1(2);
      return 0;
}
int f1(int a) // определение функции f1
{
      return 10 * f2(a);
}
int  f2(int x) // определение функции f2
{
      return x * x;
}
```

Объявления и определения

- Определение всегда предполагает и объявление
- Допускается многократные объявления одной сущности (типа, функции), но только одно определение
- В случае встроенных переменных обычное «объявление» является и определением, так как предоставляет всю информацию о сущности, необходимую для выделения памяти:

```
int a;              // объявление и определение
```

- Для объявления переменной без определения необходимо использовать:

```
extern int a;          // объявление без определения
```

Имена и расширения заголовочных файлов

- Традиционно (со времен языка Си) заголовочные файлы имеют расширение *.h.

```
#include <stdio.h>
```

- В современной практике расширение *.h используется со «старыми» файлами для Си.

```
#include <math.h>
```

- Заголовочные файлы для C++ не имеют расширений.

```
#include <iostream>
```

- Преобразованные заголовочные файлы для Си называют с буквой «c»:

```
#include <cmath>
```

- Есть практика использования специальных расширений для «новых» заголовочных файлов: *.hxx, *.hpp, *.hh

Пространства имен

```
using namespace std;
```

- Пространства имен позволяют объединять типы и функции в «логическую единицу»
- Это полезно для исключения неоднозначности в именах
- Без подключения пространства имен необходимо использовать «полные имена» типов, функций:

```
std::cout << “Hello, world" << std::endl;
```

- Конструкцию `using` можно использовать и внутри функций
- Можно «подключать» конкретные типы из пространства имен:

```
int main()
{
        using std::cout;
        using std::endl;
        cout << "Hello, world" << endl;
```

Ввод/вывод

- В консольном проекте можно использовать классический вариант ввода/вывода, характерный для C:

```
char month[] = "September";
int day = 7;
printf("Day: %d month: %-20s year %5d\n", day, month, 2020);
```

- В C++ используется «потоковый» ввод\вывод посредством объектов cout и cin.
- Для работы требуется файл iostream и пространство имен std

Ввод\вывод в C++

```
cout << "Day: " << day << ", month: " << month << endl;
```

- Объекты cout, cin не требуют указания спецификаторов типа вывода/ввода. Тип определяется компилятором по переменной или константе:

```
int age;
cin >> age;
if (cin.fail())
{
   cout << "bad input";
  ..
}
```

Форматирование вывода

- Для форматирования ввода\вывода используются манипуляторы и флаги

```
#include <iomanip>
int day = 7;
cout << "Day: " << setw(20) << left << day;
```

- Флаги устанавливаются с помощью метода setf

```
cout.setf(ios::fixed | ios::showpos);
```

╧Ёштхє, ьшЁ! или кириллица в программе

- В Windows-системах консоль (окно вывода) использует однобайтовую кодировку **cp866**, а в редакторе кода (Visual Studio) **cp1251**.

```
char answer[100];
cin >> answer;
cout << "Привет, " << answer << endl;
```

- Команда, которая позволяет установить региональные настройки (в том числе кодировку)

```
setlocale(LC_ALL, "ru-RU");
```

- Но в этом случае кодировка **cp1251** подключается только для вывода
- Для установки кодировки для ввода и вывода можно использовать:

```
// #inclue <Windows.h>
SetConsoleCP(1251);
SetConsoleOutputCP(1251);
```

Встроенные типы

Конкретный размер типа зависит от компилятора и процессора, не определяется на уровне стандарта

Целочисленные типы:

- Тип `short` – не меньше 16 битов
- Тип `int` – не меньше `short`
- Тип `long` – не меньше 32 и не меньше `int`
- Тип `long long` – не меньше 64 и не меньше `long`

С каждым целочисленным типом можно использовать модификатор unsigned. В этом случае вдвое увеличивается множество положительных чисел

Целочисленные типы

```
int n_int = INT_MAX;          // инициализация переменных максимальными значениями
short n_short = SHRT_MAX;     // константы определены в файле climits
long n_long = LONG_MAX;
long long n_llong = LLONG_MAX;

cout << "int is " << sizeof (int) << " bytes." << endl;
cout << "short is " << sizeof n_short << " bytes." << endl;
cout << "long is " << sizeof n_long << " bytes." << endl;
cout << "long long is " << sizeof n_llong << " bytes." << endl;

cout << "Maximum values:" << endl;
cout << "int: " << n_int << endl;
cout << "short: " << n_short << endl;
```

Типы с плавающей точкой

Типы с плавающей точкой:

- Тип `float` - 4 байта, 7 значащих цифр.
- Тип `double` - 8 байт, 15 значащих цифр
- Тип `long double` – не меньше, чем double

Преобразование типов

```
cout.setf(ios_base::fixed, ios_base::floatfield);
float tree = 3;     // int -> float
int guess = 3.9832; // float -> int
int debt = 7.2E12;  // неопределенный результат
cout << "tree = " << tree << endl;
cout << "guess = " << guess << endl;
cout << "debt = " << debt << endl;
```

```
tree = 3.000000
guess = 3
debt = 1634811904
```

Символьные типы

- Тип `char` - 1 байт
- Тип `wchar_t` – 2 байта
- Типы `char16_t, char32_t` – “новые стандартные типы для работы с символами»