C++

compscicenter.ru

Башарин Егор

eaniconer@gmail.com

ОРГАНИЗАЦИОННЫЕ ВОПРОСЫ

АКТИВНОСТИ

- Лекции
- Домашние работы
- Семинары (опционально для CSC)
  - Дорешки
  - Контрольные работы
- Экзамен (опционально для CSC)
  - теория
  - задача

ИСТОРИЯ

`C++` *начал разрабатываться с начала* `1980`*-х сотрудником* `Bell Labs` *Бьярне Страуструпом*

"С с классами" и транслятор `cfront`

ИСТОРИЯ

К `1983` было реализовано большое количество возможностей, поэтому язык был переименован в `C++`

Имя языка связано с оператором постфиксного инкремента

СТАНДАРТИЗАЦИЯ

Classic

`C++98`

`C++03`

СТАНДАРТИЗАЦИЯ

Modern

- `C++11`
- `C++14`
- `C++17`
- `C++20`

ФИЛОСОФИЯ ЯЗЫКА

- Совместимость с `C`
- Мультипарагдигмальность
- Zero-overhead principle
- Избегать платформозависимых особенностей
- Дать программисту свободу выбора

ХАРАКТЕРИСТИКИ ЯЗЫКА

- сложность
- мультипарадигмальность
- эффективность
- низкоуровневость
- статическая типизация
- компилируемость

СЛОЖНОСТЬ

~1500 страниц стандарта

Больше ответственность из-за свободы выбора

Много дополнительной работы: работа с памятью и обработкой низкоуровневых ошибок

ПАРАДИГМЫ

Процедурное программирование

ООП

Обобщенное программирование

Функциональное программирование

Метапрограммирование

ЭФФЕКТИВНОСТЬ

Zero-overhead principle

Самостоятельный выбор абстракций

Эффективная реализация критически важных по производительности участков кода

Использование в компьютерной графике

НИЗКОУРОВНЕВОСТЬ

Можно работать напрямую с ресурсами компьютера

Отсутствует автоматическое управление памятью

СТАТИЧЕСКАЯ ТИПИЗАЦИЯ

Каждая сущность в программе имеет тип, который определен на момент компиляции

Это позволяет вычислить размер памяти, которую будет занимать переменная, и определить какая функция будет вызываться

КОМПИЛИРУЕМОСТЬ

Компиляция - преобразование текста программы в машинный код

- для каждой платформы отдельно
- позволяет отловить некоторые ошибки
- нет накладных расходов при выполнении программы
- при изменении программы нужно компилировать снова

ЭТАПЫ КОМПИЛЯЦИИ

1. Preprocessor: Source Code Files → Translation Units
2. Compilation: Translation Unit → Object Files
3. Linker: (Object Files, Libraries) → Executable | Library

ПРОГРАММА

Программа - последовательность инструкций

Точка входа:

```cpp
// main.cpp
int main() {
  return 0;
}
```

ОСНОВНЫЕ КОНСТРУКЦИИ

```cpp
// main.cpp
#include <iostream>

bool isGood(int n) { return n == 42; }

int main() {
    int n = 32;
    for (int i = 0; i < n; ++i) {
        if (isGood(i)) {
            std::cout << i << " is good!" << std::endl;
        }
    }
    return 0;
}
```

САМЫЙ ВАЖНЫЙ САЙТ

https://en.cppreference.com/

РАБОТА С ФАЙЛАМИ

```cpp
// main.cpp
#include <fstream>

int main() {
    ifstream in ("in.txt");
    ofstream out("out.txt");

    double value = 0.0;
    in >> value;
    out << value;

    return 0;
}
```

МНОГОФАЙЛОВАЯ ПРОГРАММА

```cpp
// main.cpp
#include <iostream>
#include "factorial.hpp"

int main() {
    std::cout << factorial(10);
    return 0;
}
```

ЗАГОЛОВОЧНЫЙ ФАЙЛ

```cpp
// factorial.hpp

int factorial(int n) {
    // your code here...
}
```

```cpp
// main.cpp
#include <iostream>
#include "factorial.hpp"

int main() {
    std::cout << factorial(10);
    return 0;
}
```

ПРОБЛЕМА 1

```cpp
// main.cpp
#include <iostream>
#include "factorial.hpp"
#include "factorial.hpp" // двойное включение

int main() {
    std::cout << factorial(10);
    return 0;
}
```

РЕШЕНИЕ ПРОБЛЕМЫ 1

```cpp
// factorial.hpp
#pragma once

int factorial(int n) {
    // your code here...
}
```

ПРОБЛЕМА 2

Изменение фукнции `factorial`
приводит к перекомпиляции
`main.cpp`

РЕШЕНИЕ ПРОБЛЕМЫ 2

```cpp
// factorial.hpp
#pragma once

int factorial(int n);
```

```cpp
// factorial.cpp
#include "factorial.hpp"

int factorial(int n) {
   // your code here...
}
```

ВОПРОСЫ