C/C++: Лекция 1

Воробьев Д.В

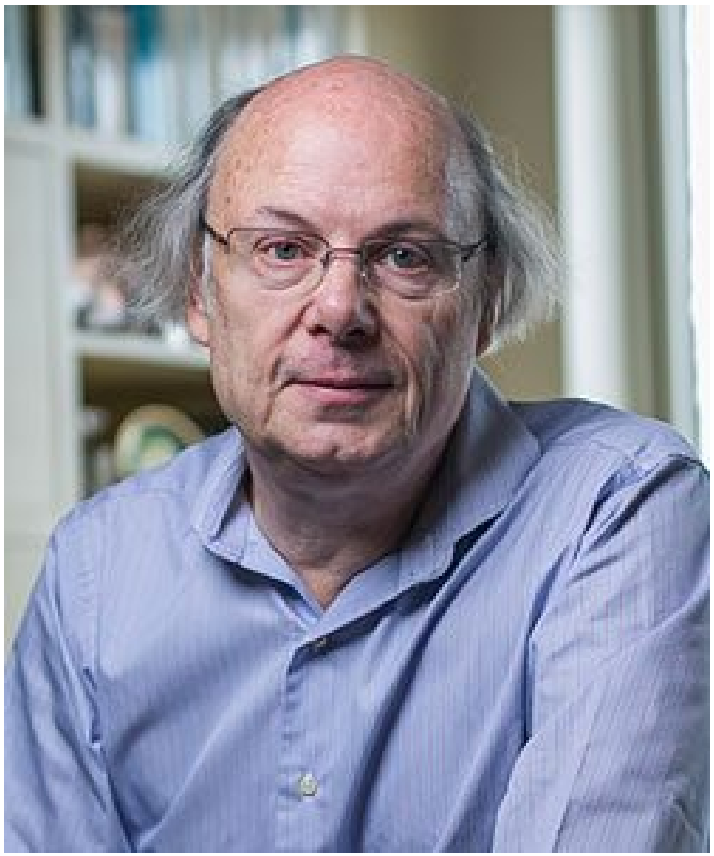
04.09.2020

Знакомство

Язык C был разработан Деннисом Ритчи в 1972 году в компании Bell Labs.

Знакомство

7-ю годами позже Бьерн Страуструп (на тот момент разработчик в Bell Labs) начал работу над расширением ”C with classes”



Знакомство

Основными требованиями для разрабатываемого языка являлось:

1. создание высокого уровня абстракции
2. сохранение возможности доступа к аппаратному обеспечению

Знакомство

Основные даты:

1. 1998: стандарт C+98 (778 страниц)
2. 2003: стандарт C+03 (786 страниц)
3. 2007: Technical Report 1
   Добавление внесенных предложений началось с C+11
4. 2011: стандарт C+11 (1350 страницы)
5. 2014: стандарт C+14 (1380 страниц)
6. 2017: стандарт C+17 (1580 страниц) Модуль 2
7. 2020: стандарт C+20 (1780 страниц)

Знакомство

За 30 лет состав коммитета увеличился с 46 до 252 человек

Сейчас стандарт выходит каждые 3 года

Основные конференции: CppCon, C++ Russia

Структура программы

```
// пространство имен
using namespace std;

// точка входа в программу
int main() {
    return 0;
}
```

Область видимости

Определение

Potential Scope - часть программы от точки **объявления** до конца блока (первый символ **}**)

Определение

Actual Scope - Potential Scope без учета вложенных блоков с объявлениями, использующими тоже имя

Замечание

Здесь рассмотрен **Block Scope** более формально
см. стандарт / cppreference

Область видимости

```
int main()
{
    int x = 0;
    {
        // внешний x перекрывается
        int x = 1;
    }
}
```

Ошибки

Ошибки делятся на 2 класса:

Compilation error

Ошибки, которые не позволяют произвести компиляцию программы (получить исполняемый файл). В нашем понимании это будут виды ошибок до построения IR-Tree.

Runtime error

В нашем понимании это ошибки, которые возникают на момент исполнения исполняемого файла. Более детально в теме "Исключения"

Compilation error

Разбиваются на 3 класса:

Лексические

Связаны с употреблением символов не из **алфавита** языка.
Детектируются на 1-ом этапе - лексическом анализе.

Синтаксические

Связаны с употреблением **некорректных конструкций**.
Детектируются на 2-ом этапе - проверки программы на принадлежность языку КС-грамматики.

Семантические

Связаны с употреблением конструкций не в соответствии с их смысловым значением.

Compilation error

```
// Лексическая
int main() {
    // кириллица
    int икс
    return 0;
}
```

```
// Синтаксическая
int main() {
    // отсутствует ;
    int x
    return 0;
}
```

```
// Семантическая
int main() {
    // различные типы
    int x = "x";
    return 0;
}
```

Runtime error

Segmentation fault

Ошибка, возникшая при поптыке обращения к памяти, для которой явно не предоставлен доступ.

Stack overflow

Ошибка, возникающая при переполнении стековой памяти. Как следствие Segmentation fault.

Runtime error

```
// Stack overflow
int main() {
    int x = 0;
    main();
    x++;
    return 0;
}
```

```
// Segmentation fault
int main() {
    int x[10];
    x[20000] = 10;
    return 0;
}
```

Идентификаторы

Определение

Идентификатор - последовательность произвольной длины, состоящая из: 0-9, _ , a-z, A-Z и не начинающаяся с 0-9.

```
int main() {
    // корректный идентификатор
    int num_cars;

    // не корректный идентификатор
    int 100500num_cars;

    return 0;
}
```

Переменные

Определение

Переменная = идентификатор + память.

```
int main() {
    int num_cars = 0;
    return 0;
}
```

Фундаментальные типы

Type	Size in bits	Format	Value range	
			Approximate	Exact
character	8	signed		-128 to 127
		unsigned		0 to 255
	16	unsigned		0 to 65535
	32	unsigned		0 to 1114111 (0x10ffff)
integer	16	signed	$\pm\ 3.27 \cdot 10^4$	-32768 to 32767
		unsigned	0 to $6.55 \cdot 10^4$	0 to 65535
	32	signed	$\pm\ 2.14 \cdot 10^9$	-2,147,483,648 to 2,147,483,647
		unsigned	0 to $4.29 \cdot 10^9$	0 to 4,294,967,295
	64	signed	$\pm\ 9.22 \cdot 10^{18}$	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
		unsigned	0 to $1.84 \cdot 10^{19}$	0 to 18,446,744,073,709,551,615
floating point	32	IEEE-754	• min subnormal: $\pm\ 1.401,298,4 \cdot 10^{-45}$ • min normal: $\pm\ 1.175,494,3 \cdot 10^{-38}$ • max: $\pm\ 3.402,823,4 \cdot 10^{38}$	• min subnormal: ±0x1p-149 • min normal: ±0x1p-126 • max: ±0x1.fffffep+127
	64	IEEE-754	• min subnormal: $\pm\ 4.940,656,458,412 \cdot 10^{-324}$ • min normal: $\pm\ 2.225,073,858,507,201,4 \cdot 10^{-308}$ • max: $\pm\ 1.797,693,134,862,315,7 \cdot 10^{308}$	• min subnormal: ±0x1p-1074 • min normal: ±0x1p-1022 • max: ±0x1.fffffffffffffp+1023

Модели типов и спецификаторы

Type specifier	Equivalent type	Width in bits by data model				
		C++ standard	LP32	ILP32	LLP64	LP64
`short`	`short int`	at least 16	16	16	16	16
`short int`						
`signed short`						
`signed short int`						
`unsigned short`	`unsigned short int`					
`unsigned short int`						
`int`	`int`	at least 16	16	32	32	32
`signed`						
`signed int`						
`unsigned`	`unsigned int`					
`unsigned int`						
`long`	`long int`	at least 32	32	32	32	64
`long int`						
`signed long`						
`signed long int`						
`unsigned long`	`unsigned long int`					
`unsigned long int`						
`long long`	`long long int` (C++11)	at least 64	64	64	64	64
`long long int`						
`signed long long`						
`signed long long int`						
`unsigned long long`	`unsigned long long int` (C++11)					
`unsigned long long int`						

Представление отрицательных

Положим n число разрядов. Для удобства возьмем $n = 4$.
Способы:

- Прямой код
- Смещенный код
- Обратный код
- Дополненный код

Прямой код

Кладем

Старший разряд отвечает за знак. Вводим 2 нуля.

- $0000_2 = +0_{10}$
- $1000_2 = -0_{10}$

- $0101_2 = 5_{10}$
- $0101_2 = -5_{10}$

Проблема

+ в 10-й системе, не соответсвуют + в 2-ой

$$5_{10} + (-2)_{10} = 3_{10}$$
$$0101_2 + 1010_2 = 1111_2$$
$$3_{10} \neq -7_{10}$$

Смещенный код

Кладем

- $1000_2 = 0_{10}$
- отрицательное число = число $< 1000_2$
- положительное число = число $> 1000_2$

- $1011_2 = 3_{10}$
- $0011_2 = -3_{10}$

Проблема

Нельзя получить положительное целое возведением разрядов в степень 2. В прямом коде такая возможность была.

$$3_{10} = \text{по опр.} = 1011_2 = \text{не состыковочки} = 2^3 + 2^1 + 2^0 = 11_{10}$$

Обратный код

Кладем

Отрицательноые числа будут инверсией положительных.
Вводим 2 нуля.

- $0000_2 = +0_{10}$
- $1111_2 = -0_{10}$

- $0101_2 = 5_{10}$
- $1010_2 = -5_{10}$

Проблема

\+ в 10-й системе, не соответсвуют + в 2-ой

$$5_{10} + (-1)_{10} = 4_{10}$$
$$0101_2 + 1110_2 = 0011_2$$
$$4_{10} \neq 3_{10}$$

Дополненный код

Кладем

Отрицательноые числа будут инверсией положительных плюс 1.

- $01100_2 = 12_{10}$
- $10100_2 = -12_{10}$

Бонус

Факт $x + \neg x = 2^n$ позволяет упростить проектирование сумматоров. Вычитание можно производить на том же аппаратном обеспечении.

Неявные преобразования типов

Для **expression** есть **promotion** и есть **conversion**

Promotion

Преобразование **expression** к более "общему" типу

Conversion

Преобразование **expression** к более "частному" типу

Неявные преобразования типов

- Integral promotion
  - char $\rightarrow$ int
  - unsigned char $\rightarrow$ unsigned int
  - wchar_t $\rightarrow$ к первому из списка [int, unsigned int, long, unsigned int]
  - bool $\rightarrow$ int
- Float-point promotion
  - float $\rightarrow$ double

Неявные преобразования типов

- Integral conversion - берем по модулю 2^n, где n - число битов destination типа
- Float-point conversion - отбрасываем вещественную часть

Неявные преобразования типов

```cpp
//  Integral conversion
int main()
{
    unsigned int x = 128000;
    unsigned short int y = x;
    // 62464
    std::cout << y;
    return 0;
}
```

```cpp
//  Float-point conversion
int main()
{
    double x = 12.8;
    int y = x;
    // 12
    std::cout << y;
    return 0;
}
```

Константы

```
int main() {
    // указатель на константные данные
    const int* ptr1;

    // константный указатель
    int* const ptr2 = new int(1);

    // константый указатель на константные данные
    const int* const ptr3 = new int(1);

    return 0;
}
```

One definition rule

Единица трансляции

Исходный файл, в котором выполнена подстановка директив

One definition rule

В каждой из единиц трансляции может быть не более одного определения (definition) для переменной, функции, класса.

Замечание

Один и тот же класс можно определить (одновременно) в разных единицах трансляции

One definition rule: Нельзя

```cpp
// file1.cpp
#include <iostream>

struct A {};
struct A {};

int main() {
    return 0;
}
```

```cpp
// file1.cpp
#include <iostream>

int x = 0;
int x = 0;

int main() {
return 0;
}
```

```cpp
// file1.cpp
#include <iostream>

void f() {};
void f() {};

int main() {
return 0;
}
```

One definition rule: Нельзя

```cpp
// file1.cpp
#include <iostream>

int x = 0;
```

```cpp
// file2.cpp
#include <iostream>

int x = 0;
int main()
{
    return 0;
}
```

One definition rule: Нельзя

```cpp
// file1.cpp
#include <iostream>

void f() {};
```

```cpp
 //  file2.cpp
#include <iostream>

void f() {};

int main()
{
    return 0;
}
```

One definition rule: Можно

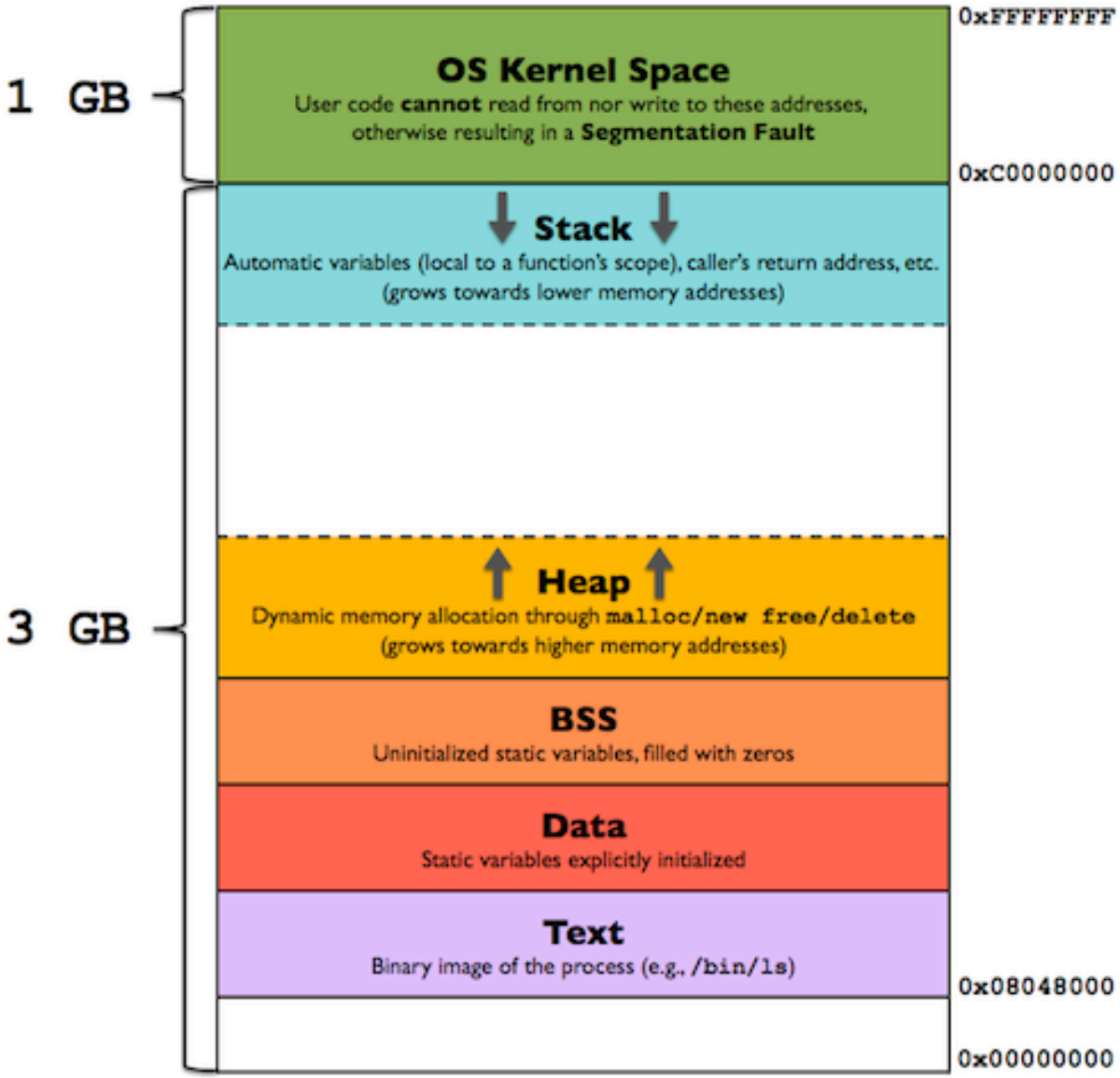
```cpp
// file1.cpp
#include <iostream>

struct A {
    A() {std::cout << "file1";}
};
```

```cpp
//  file2.cpp
#include <iostream>

struct A {
    A() {std::cout << "file2";}
};
int main()
{
    A a;
    return 0;
}
```

Секции


1 GB
3 GB
0xFFFFFFFF
OS Kernel Space
User code cannot read from nor write to these addresses, otherwise resulting in a Segmentation Fault
0xC0000000
Stack
Automatic variables (local to a function's scope), caller's return address, etc.
(grows towards lower memory addresses)
Heap
Dynamic memory allocation through malloc/new free/delete
(grows towards higher memory addresses)
BSS
Uninitialized static variables, filled with zeros
Data
Static variables explicitly initialized
Text
Binary image of the process (e.g., /bin/ls)
0x08048000
0x00000000