Объектно-ориентированное программирование

Лекция 1.
Язык C++: новые возможности

Феськов Сергей Владимирович
каб. 3-01М,
serguei.feskov@gmail.com

Литература (теория)

1) **Лафоре Р.** *Объектно-ориентированное программирование в C++.* – СПб.: Питер, 2013. – 928 с.

2) **Б. Страуструп.** *Дизайн и эволюция C++ = The Design and Evolution of C++.* – СПб.: Питер, 2007. – 445 с.

3) **С. Прата.** *Язык программирования C++ (C++11). Лекции и упражнения = C++ Primer Plus, Developer's Library*. – М.: Вильямс, 2012. – 1248 с.

Литература (практика)

1) **Хохлова С.С., Юданов В.В., Феськов С.В.** *Объектно-ориентированное программирование на языке C++. Лабораторный практикум*. – Изд. ВолГУ, 2017. – 107 с.

2) **Павловская Т.А., Щупак Ю.А.** C/C++. *Объектно–ориентированное программирование: Практикум*. – СПб.: Питер, 2004. – 265 с.

Создание языка C++

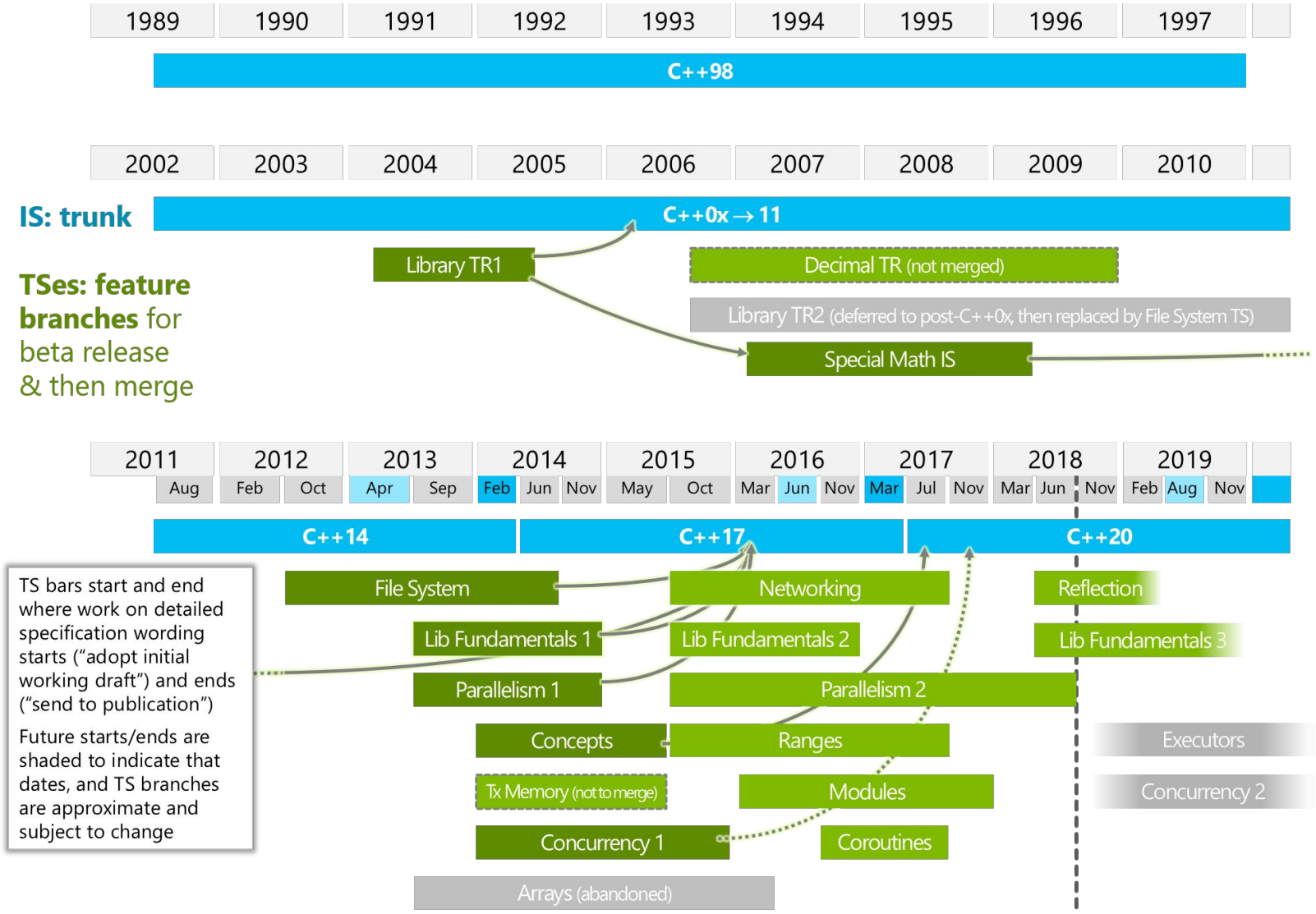
Бьярн Страуструп
(Bjarne Stroustrup), 1980

Основой для C++ стал язык C,
прообразом - Simula-67

Цели создания C++

- поддержка пользовательских типов данных (через классы);
- разработка инструментов для ООП;
- улучшение конструкций языка C.

История развития языка C++



Новые возможности языка C++

- пространство имен (namespace) – окружение для логической группы уникальных идентификаторов (имен);
- перегрузка функций – использование одного и того же имени для функций с различными типами входных данных;
- возможность неявной передачи аргументов в функцию по адресу (понятие ссылки);
- комментарии в языке C++: `/* ... */` или `// ...`
- объявления можно помещать в любом месте программы

C++: новые операции

- динамическое выделение памяти – операция **`new`**
- освобождение памяти – операция **`delete`**
- получение информации о типе переменной на этапе выполнения программы – операция **`typeid`**
- обработка исключительных ситуаций – операции **`try/throw/catch`**

Язык С. Простейшая программа.

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

void main(void)
{
  printf("Standard C style");
  getch();
}
```

Язык C++. Простейшая программа.

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

int main()
{
  cout << "C++ style!" << endl;
  cin.get();
  return 0;
}
```

Библиотека потокового ввода/вывода iostream

Подключение заголовочных файлов

```
#include <iostream>
#include <iomanip>
```

Объекты потокового ввода/вывода

`cin` - входной поток (с клавиатуры)

`cout` - выходной поток (на монитор)

Операции над объектами **cin** и **cout**

cin >> - извлечь данные из потока

cout << - поместить данные в поток

cin.get() - получить код символа

Примеры использования:

```
int a, b, c;
float x, y;
char text[20];
cin >> a;
cout << b << x;
cout << c << text << y;
cout << "Вывод простого текста";
```

Форматирование потокового В/В

`width(int x)`	Минимальное число знаков поля вывода (до следующего поля)
`fill(char x)`	Устанавливает символ-заполнитель. По умолчанию - пробел.
`precision(int x)`	Устанавливает число значащих знаков для чисел с плавающей точкой.

Пример:

```
cout.width(10);
cout << "ten" << "four";
cout << "four" << endl;
```

endl – манипулятор потока (END Line)

Пространства имен

Определение: *Пространством имен* в языке C++ называется абстрактное хранилище (или окружение) для группы уникальных идентификаторов (имен).

Необходимость в использовании пространств имен возникает при разработке больших программ, когда возможна **коллизия (конфликт) имен**.

Пространства имен

Объявление нового пространства

```
namespace my_names
{
    int x;
};
```

Доступ к переменной (вне блока my_names)

```
my_names::x = 5;
```

:: – операция расширения видимости

Стандартное пространство имен - std

```
#include <iostream>
#include <iomanip>
int main()
{
    std::cout << "text" << std::endl;
    std::cin.get();
    return 0;
 }
```

Директива using

```
#include <iostream>
#include <iomanip>
using std::cout;
using std::cin;
using std::endl;

int main()
{
    cout << "text" << endl;
    cin.get();
}
```

Директива using namespace

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

int main()
{
 cout << "text" << endl;
 cin.get();
}
```

Текстовые строки (string)

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

int main()
{
    string s1, s2;          // Пустые строки
    string s3 = "We";
    s2 = "are";             // Присваивание
    s1 = s3 + " " + s2;     // Слияние строк
    s1 += " from VolSU";
    cout << s1 + "!" << endl;
}
```

Динамическая память

- выделение памяти под одиночную переменную

```
указатель = new тип;
```

- освобождение памяти

```
delete указатель;
```

- пример

```
float *p = new float;
*p = 5.1;
cout << *p << endl;
delete p;
```

Динамическая память (массивы)

- выделение памяти под одномерный массив

```
указатель = new тип [размер];
```

- освобождение памяти

```
delete [] указатель;
```

- пример

```
int *a, n;
cin >> n;
a = new int[n];
for(int i = 0; i < n; i++)
 a[i] = rand();
delete []a;
```

Преимущества new/delete

1) Операция **`new`** автоматически выделяет необходимое количество памяти.

2) Операция **`new`** автоматически возвращает указатель на заданный тип данных.

3) Операции **`new`** и **`delete`** можно перегружать.

4) При выделении памяти с помощью **`new`** автоматически вызывается конструктор объекта, а при освобождении памяти с помощью **`delete`** – вызывается деструктор объекта.