Информатика. Семинар №1

https://dl.dropboxusercontent.com/u/96739039/sem4/infa_s01.pdf

Язык C++

Компилируемый (быстрый) -> ОС, прикладные приложения, приложения для встраиваемых систем, высокопроизводительные вычисления, игры

P.S. про различные языки программирования https://habrahabr.ru/company/yandex/blog/272759/

Устанавливаем VS2015

- Скачиваем Community версию https://www.visualstudio.com/downloads/
- Выберете лучше английскую версию, т.к. впоследствии нагуглить что-то на русском не получится
- Выберете «выборочный тип установки» и проверьте, что на Visual C++ выставлены галочки

Откуда брать материалы по C++

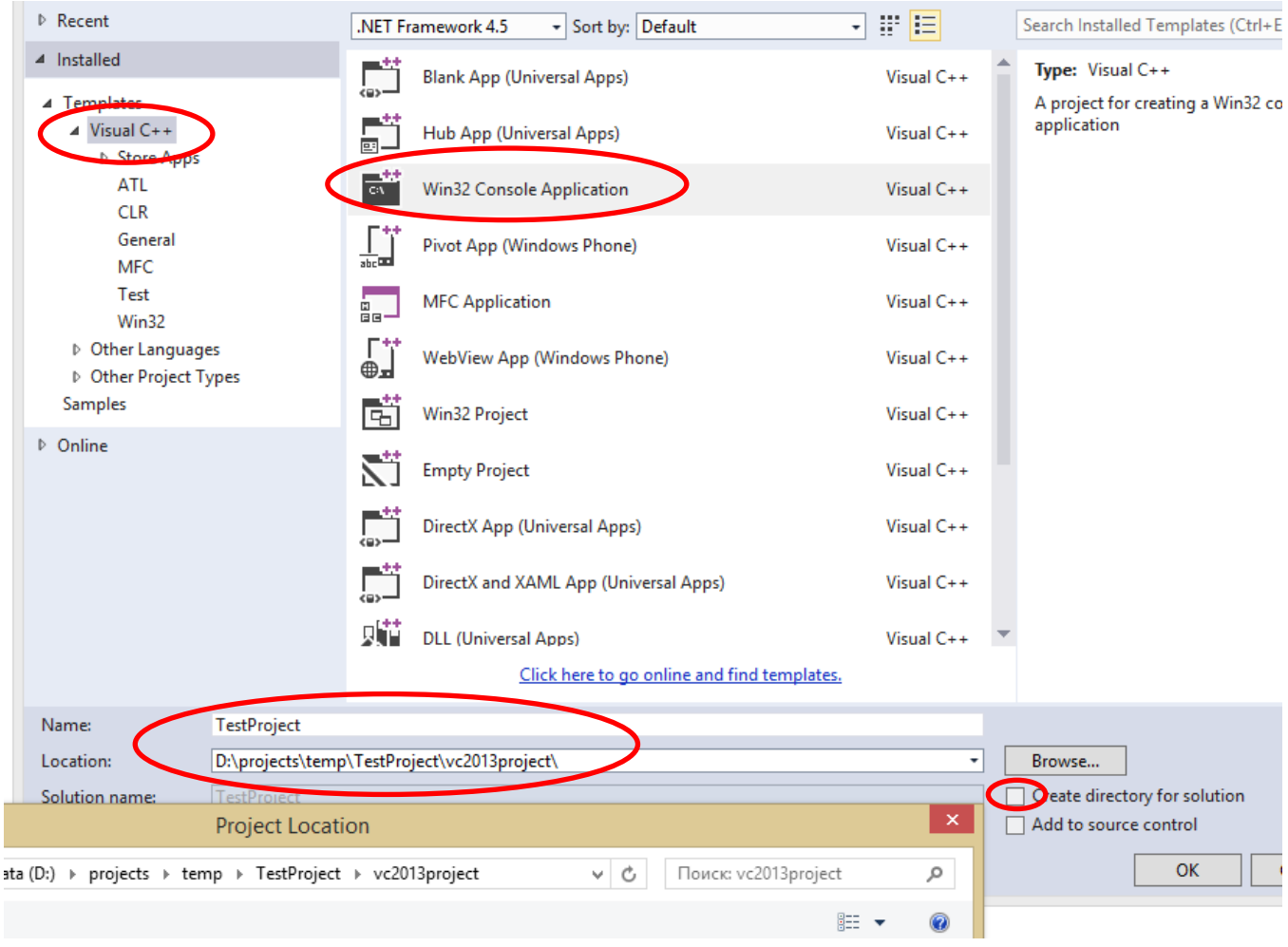
- Какой-то одной хорошей книги не знаю, но как вариант можно пользоваться https://www.dropbox.com/s/nk1j58oqkrks70k/Pratt_Cpp.djvu?dl=0

Онлайн-курсы по C++

- https://stepik.org/course/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D0%BD%D0%B0-%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5-C++-7/syllabus

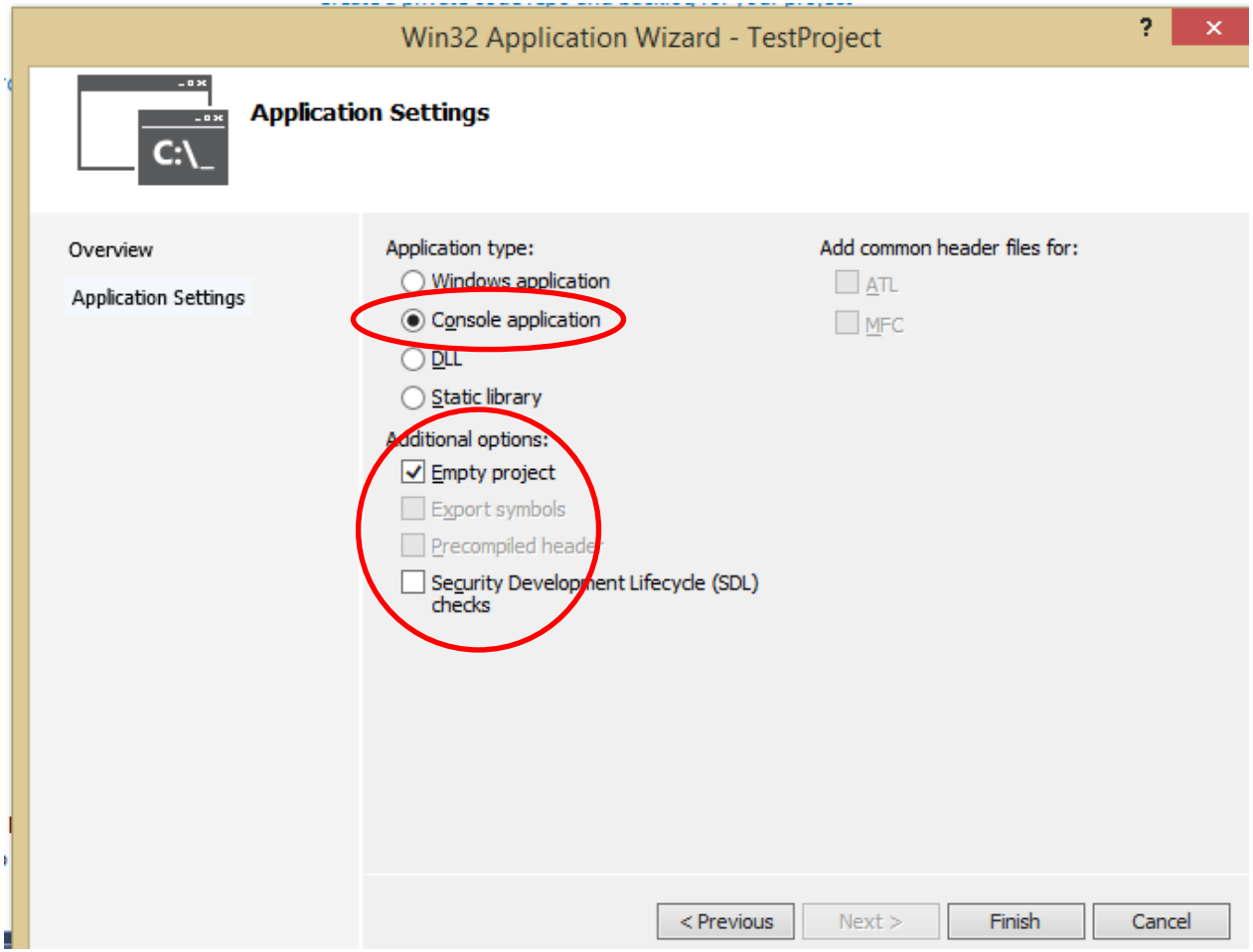
Как создать проект в VS?

- При создании проекта обращаем внимание на:



Как создать проект в VS?

Обращаем внимание на обведенные «галочки»

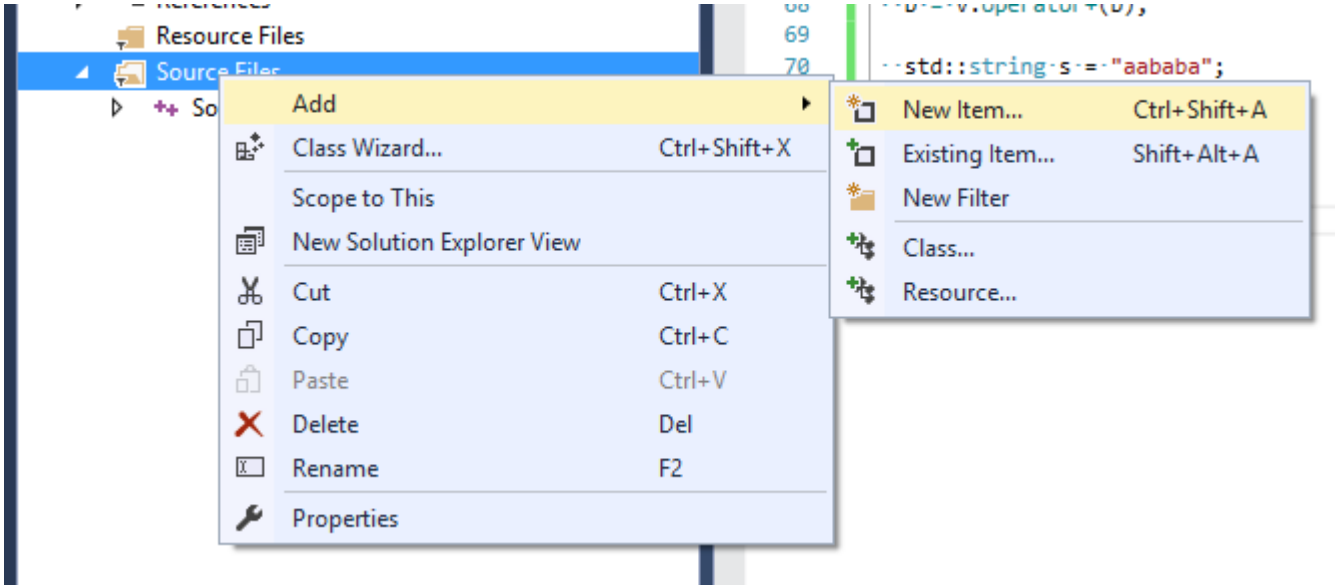


Как создать проект в VS?

F7 – компиляция

F5 – запуск программы

F10, F11 – отладка (step over, step into)



Первая программа

```
#include <iostream>
int main()
{
        int x;
        std::cin >> x;
        std::cout << “Hello, world!” << x << “\n”;
        return 0;
}
```

Пространство имён (namespace)

```
namespace MyLib {
        namespace Math {
                struct Point {
                        float x, y;
                };
        };
        namespace Physics {
                struct Point {
                        Math::Point position;
                        float mass;
                };
        };
};
```

Пространство имён (namespace)

```
using MyLib::Math::Point;

int main()
{
        Point                       mathPoint;
        MyLib::Physics::Point physPoint;
}
```

Структура Vector2

```
#include <math.h>
struct Vector2
{
	float Len() // метод – ф-я внутри структуры
	{
		return sqrt(x * x + y * y);
	}
	float x, y;
};
```

Указатель vs Ссылка

```
void Swap(int* x, int* y)
{
        int temp = *x;
        *x = *y;
        *y = temp;
}
void Swap(int& x, int& y)
{
        int temp = x;
        x = y;
        y = temp;
}
```

```
int x = 5;
int& y = x;

int* z;
z = &x;

int& t; // error, т.к. ссылка не является самостоятельным объектом
```

const

1. Известный на момент компиляции программы:

```
const int N = 5;
int a[N];
```

2. Запрет модификации переменной:

```
void f(const Vector2 v)
{
        v.x = 1; // error
}
```

Перегрузка операторов

Vector2 Add(const Vector2& a, const Vector2& b);

Список возможных операторов + приоритеты
http://ru.cppreference.com/w/cpp/language/operator_precedence

Нельзя перегрузить:

- Оператор выбора члена класса ".".
- Оператор разыменования указателя на член класса ".*"
- В C++ отсутствует оператор возведения в степень (как в Fortran) "**".
- Запрещено определять свои операторы (возможны проблемы с определением приоритетов).
- Нельзя изменять приоритеты операторов

Перегрузка операторов

https://habrahabr.ru/post/132014/

```
struct Vector2
{
	Vector2 operator+(const Vector2& other) const
	{
		Vector2 result;
		result.x = x + other.x;
		result.y = y + other.y;
		return result;
	}
	float x, y;
};
```

P.S. Лишний раз вызывается конструктор копирования, поэтому лучше вызвать конструктор return Vector2(x + other.x, y + other.y);

P.P.S. Бинарные операторы можно перегружать вне структуры

```
Vector2 operator+(const Vector2& a, const Vector2& b)
{
	Vector2 res;
	res.x = a.x + b.x;
	res.y = a.y + b.y;
	return res;
}
```

Перегрузка операторов

Vector2 a, b, c;

c = a.operator+(b); // можем работать как с обычным методом структуры

c = a + b;

Перегрузка функций/методов

```
void Print(const Vector2&);
void Print(Vector2*, int size);
```

P.S. Разницы только в типе возвращаемого значения недостаточно:

```
int f(int x);
float f(int x); // error
```

std::string, std::vector

```
#include<vector>
#include<string>

int main()
{
	std::vector<int> a;
	a.resize(10);
	for (size_t i = 0; i < a.size(); ++i)
		a[i] = i;

	std::string s = “abacaba”;
}
```

Упражнение 1

Дано N целых неотрицательных чисел не превышающих 10^50.

Необходимо упорядочить их в порядке неубывания с использованием ф-и std::sort.

```
#include <algorithm>
std::vector<float> a;
…
std::sort(a.begin(), a.end());
```

P.S. Нужно написать ф-ю compare для строк:
bool compare(const std::string& lhs, const std::string& rhs);
и передать её 3м параметром в ф-ю sort

Контейнеры STL

- std::vector< float >
- std::set< int >, std::map< std::string, int >

(в основе КЧ-дерево, поэтому для элементов должен быть определен оператор <)

Итераторы контейнеров

```
std::map<int, int> m;
```

1) `for (std::map<int, int>::iterator it = m.begin(); it != m.end(); ++it) …`

2)
```
for (auto it = m.begin(); it != m.end(); ++it)
{
        std::cout << it->first << “ “ << it->second;
}
```

3) `for (auto it : m) …`

P.S. auto работает только в версиях C++ начиная с 11
Компилировать под linux в g++ нужно с флагом –std=c++11
В VS2015 1)-3) будут работать по умолчанию

Упражнение 2

Найти N наиболее часто употребляемых слов в тексте.

http://www.cplusplus.com/reference/map/map/operator[]/

http://www.cplusplus.com/reference/map/map/find/

P.S. Для считывания текста можно воспользоваться перенаправлением ввода/вывода, либо см. заготовку

Заготовка для упражнения 2

```
#include <map>
#include <algorithm>
#include <fstream>
#include <string>

std::string prepare(const std::string& s)
{
  // должны удалить знак препинания с конца слова + перевести в нижний регистр
  // http://stackoverflow.com/questions/313970/how-to-convert-stdstring-to-lower-case
  /*

    std::string result;
    std::transform(s.begin(), s.end(), result.begin(), ::tolower);

    if (в конце строки знак препинания)
      result.pop_back();

    return result;
  */
}
```

Заготовка для упражнения 2

```
int main()
{
  std::ifstream file("file.txt");

  if (file.is_open())
  {
    std::string word;
    while (!file.eof())
    {
      file >> word;
      word = prepare(word);
      // заполняем map'у <слово, сколько раз встретилось>
    }

    /*
      создаем вектор из структур
      struct Statistics
      {
        int count;
        std::string word;
      };
      std::vector<Statistics> s;

      упорядочиваем s по убыванию count, используя std::sort и написав свой компаратор для Statistics
    */

    file.close();
  }
}
```

конструктор / деструктор

Ф-и, вызываемые при создании/удалении объекта.

```
struct Complex
{
        explicit Complex(float x);
        Complex(float x = 0, float y = 0); // конструктор (параметры по умолчанию только в конце списка)
        ~Complex(); // деструктор
};
```

Порядок вызова конструкторов/деструкторов

```
int main()
{
	Vector2 a, b;
}
```

В какой момент уничтожаются? В каком порядке?

конструктор копирования (КК) vs оператор присваивания (ОП)

1. Vector2 a; // Vector2()
2. Vector2 b = a; // Vector2(const Vector2& other) (КК)
3. Vector2 c(b); // Vector2(const Vector2& other) (КК)
4. Vector2 d; // Vector2()

d = c; // Vector2& operator=(const Vector2& other); (ОП)

Спецификаторы доступа

- public vs private
- struct vs class

P.S. На самом деле спецификаторов доступа 3: public, private, protected. Пока не проходили наследование, о нём нет смысла говорить.

Разбиением кода на отдельные файлы

- Каждая структура/класс – отдельный файл
- Для использования этого класса – директива #include:

#include “Vector2.h” – в локальной папке проекта

#include <Vector2.h> - в специально указанных директориях

Отделение объявления от реализации

- компиляция + линковка
- директива pragma once

- время вызова + время работы функции
- директива inline