Информатика (семестр 4)

Семинар №1

Что делаем в этом семестре?

- Разбираемся с базовыми принципами ООП, базовым синтаксисом языка C++
- Учимся работать с библиотеками по работе с графикой / обработкой изображений / звуком
- Вы определяетесь с темой проекта и приступаете к реализации
- Изучаем более продвинутые темы по C++
- Обсуждаем принципы дизайны программных систем

Как оцениваем результат работы?

- Несколько упражнений во время семестра
- Командный проект
- Беседа по C++

Парадигмы программирования

- Структурное программирование
- Объектно-ориентированное программирование
- Функциональное программирование

Парадигма – способ программирования, не зависящий от конкретного языка. Парадигма определяет, какие структуры использовать и когда их использовать.

Структурное программирование

- Предложил Э.В. Дейкстра в 1968: использование **goto** вредно для структуры программы. Любую программу с **goto** можно переписать с использованием только **if/then/else** и **do/while/until**

Примеры языков: C, Pascal, Basic, Fortran

Объектно-ориентированное программирование

- Предложили О. Даль и К. Нюгер в 1968 году: заметили, что в языке ALGOL можно переместить кадр стека вызова ф-и в динамическую память (кучу) -> локальные переменные, объявленные в этой ф-и, могут сохраниться после выхода из неё. Изобрели полиморфизм через использование указателей на функции.

Примеры языков: C++, Python, Java, JavaScript, C#

Функциональное программирование

- Базируется на теории λ-исчислений, изобретенных в 1936 году А. Чёрч. На основе этого в 1958 году Дж. Маккарти изобрел LISP. Основополагающее понятие – неизменяемость <=> невозможно изменить значения символов / нет оператора присваивания.

P.S. В жизни ф-ные языки обладают некоторые средствами изменить значение переменной, но они ограничены.

Примеры языков: Haskell, Scala, LISP

Мультипарадигменные языки: Python, JavaScript … (очень большой список)

Функциональное программирование: пример

- C

```c
int i;
for (i = 0; i < 25; ++i)
    printf("%d ", i * i);
```

- Clojure (производный от Lisp)

```clojure
(println (take 25 (map (fn [x] (* x x)) (range))))
```

- Хм, а что даёт нам неизменяемость? Q: Откуда растут корни состояний гонки (race condition), взаимоблокировок (deadlocks) ? A: Из параллельного обновления и изменяемости переменных!

Обзоры ЯП

- ЯП в первую очередь зависит от области применения: Web, Mobile, Backend, Desktop
- За некоторыми задачами более или менее жестко закреплены ЯП: https://itproger.com/news/kakie-yaziki-programmirovaniya-dlya-chego-nuzhni
- Рейтинги языков

Язык C++

- Компилируемый = быстрый -> ОС, игры, высокопроизводительные вычисления, поиск, прикладные приложения, приложения для встраиваемых систем и т.д.
- Достаточно живой и востребованный язык
- Синтаксис похож на Си, который уже знаем

Компилятор? Операционная система?

- Большой разницы нет
- Когда будем работать с разными библиотеками буду заранее выдавать заготовки, чтобы не тратить семинарское время на скачивание, подключение библиотек и т.п.
- Это заготовки будут под Windows + Visual Studio

Установка VS2019

- Скачиваем Community версию (её вполне достаточно)

https://visualstudio.microsoft.com/

- Убедитесь, что устанавливаете именно английскую версию, т.к. ошибки компиляции и доп. информацию на русском сложно «гуглить»
- Выберете «выборочный тип установки» и проверьте, что на Visual C++ выставлены галочки

Откуда брать материалы по C++

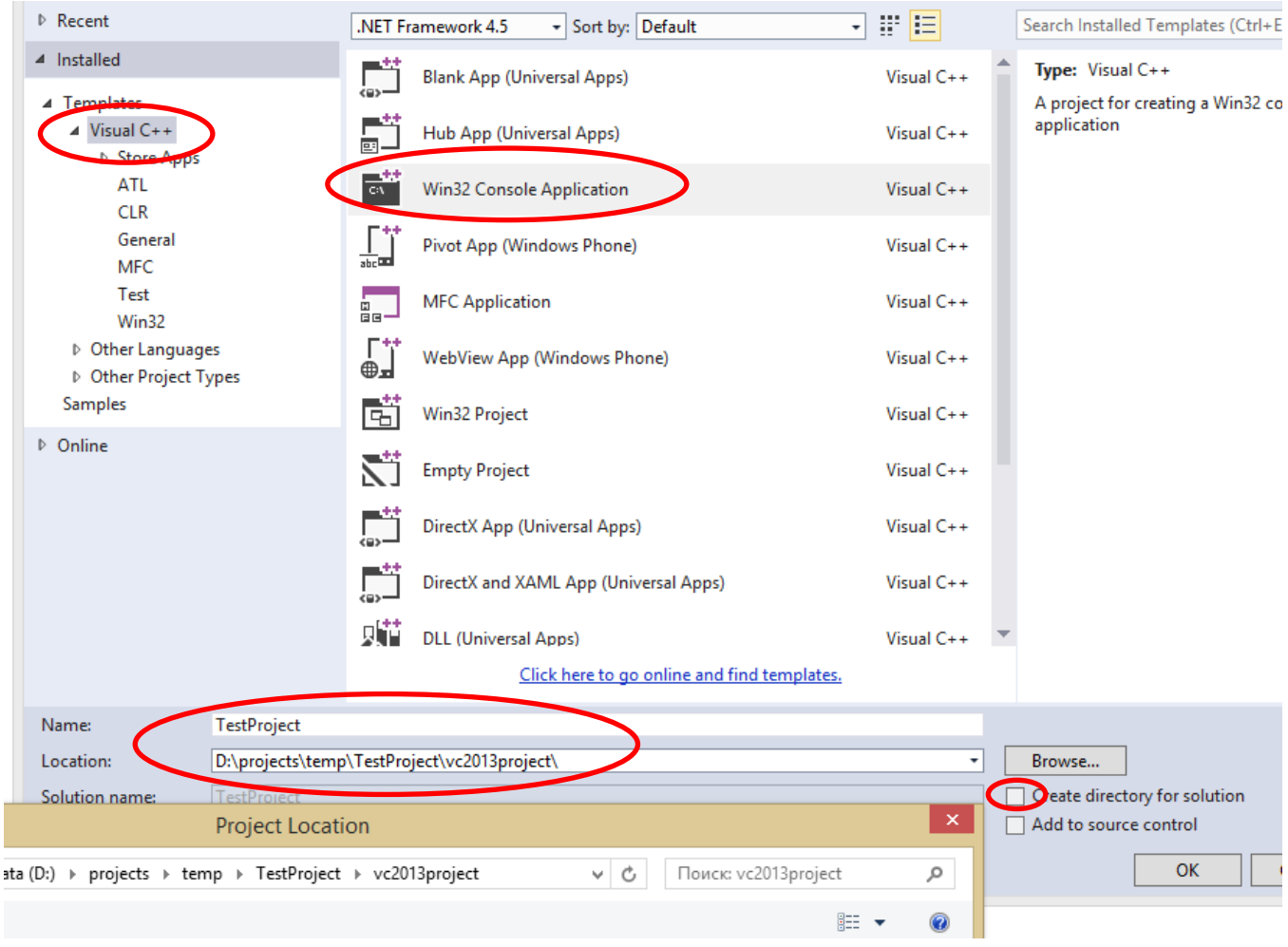
- Не знаю одну хорошую книгу на все случаи жизни
- Отдельные темы хорошо «гуглятся»: https://stackoverflow.com/ и др.
- Онлайн-курсы на https://stepik.org/
  - Введение в программирование (C++)
  - Программирование на языке C++
  - Программирование на языке C++ (продолжение)

Что же такое объектно-ориентированное программирование?

- Нет единственно верного и полного определения
  - «комбинация данных и функций» (сегодня)
  - «способ моделирования реального мира» (следующие семинары)
  - инкапсуляция + полиморфизм + наследование (следующие семинары)

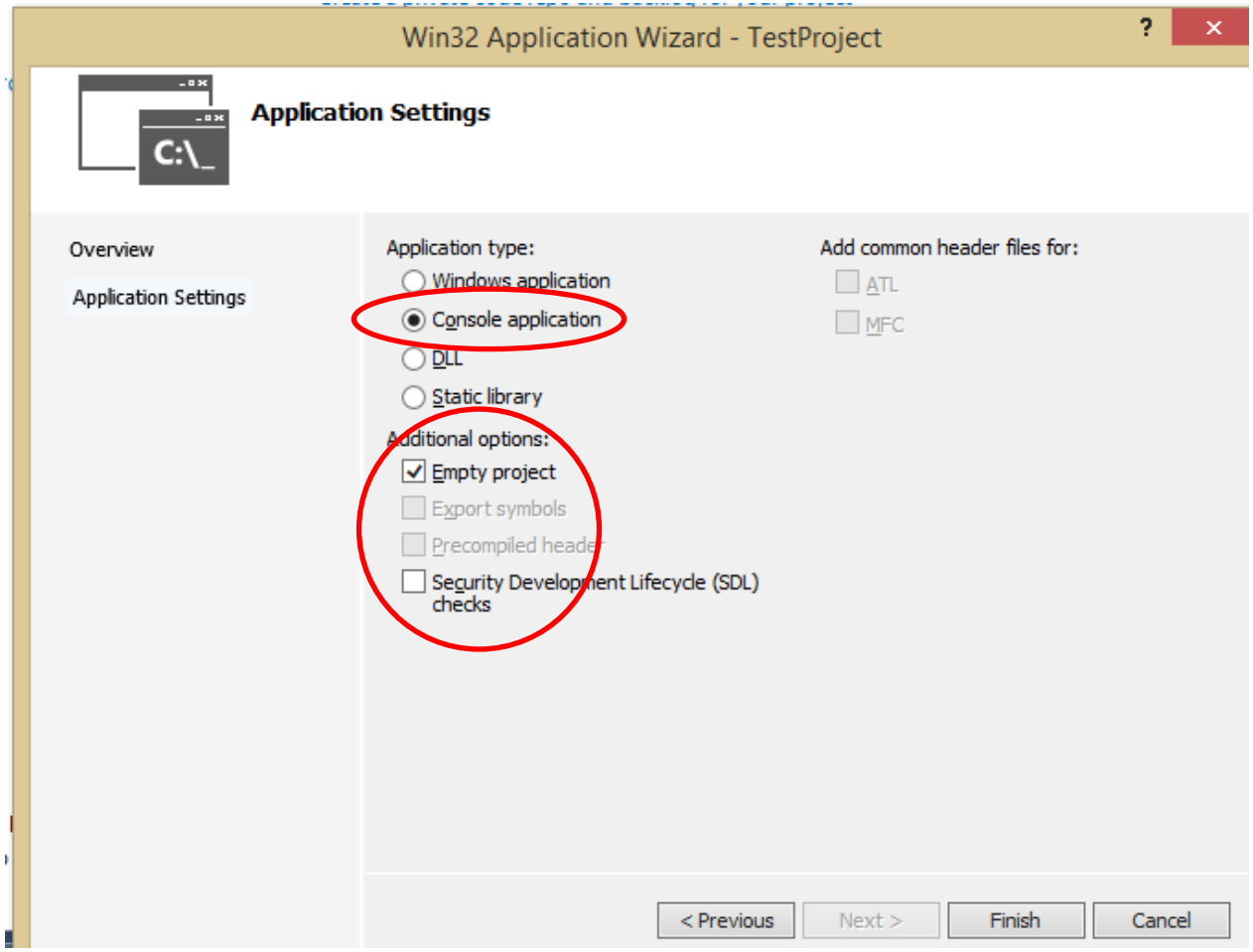
Как создать проект в VS?

- При создании проекта обращаем внимание на:



Как создать проект в VS?

Обращаем внимание на обведенные «галочки»

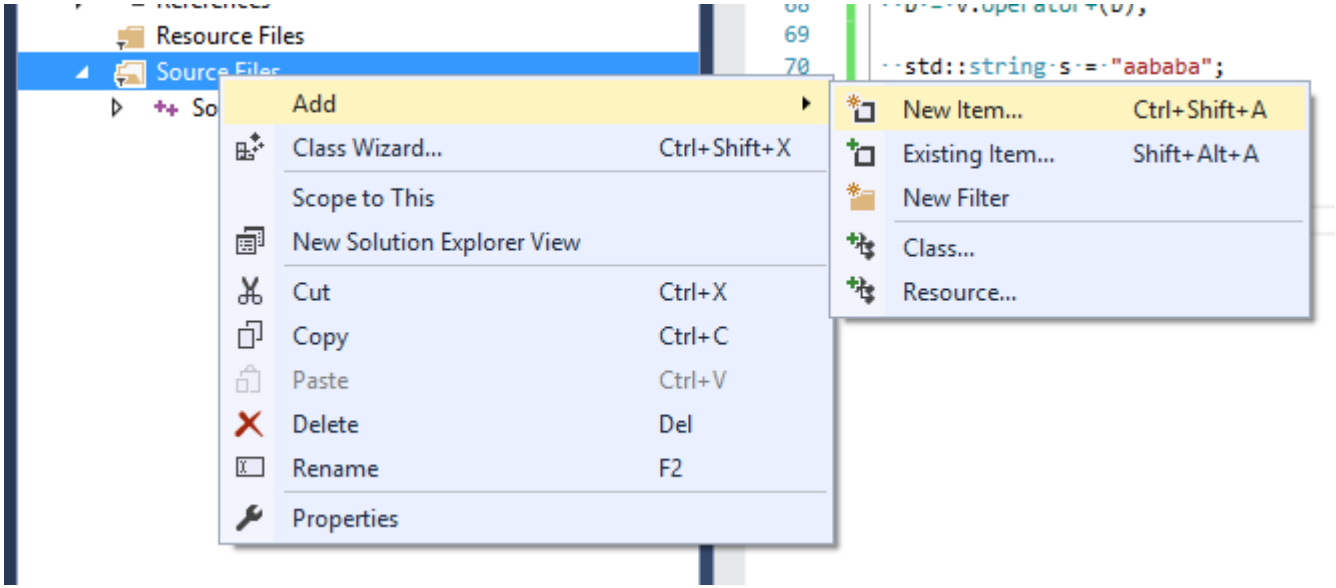


Как создать проект в VS?

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