Тема 1. Введение в C++

Лекция 2. Основные отличия от C

На прошлой лекции был разобран простейший пример построения классов C++ на примере вектора. Заметим, что формализм записи объявлений и определений классов не исчерпывается приведенными конструкциями. Другие возможности записи будут проиллюстрированы в отдельном примере, который мы здесь обсуждать не будем. Пример будет выложен на сайте, в нем будут некоторые комментарии, а остальное можно прояснить из справочных источников и описаний языка.

Язык C++ развивается, есть несколько принципиальных версий. Одна из наиболее "революционных— C++ 11, но нам надо до нее дорасти. Поэтому пока в рамках "классических"версий.

Перечислим основные отличия и нововведения в языке C++ по сравнению с C.

- понятие класса и все с этим связанное
  - private и public доступ к членам класса и методам
  - конструкторы и деструкторы
  - перегрузка операций
  - указатель this
  - статические члены класса
  - const методы
  - наследование и виртуальные функции [пока не рассматриваем]
- struct есть `class {public: ... };`
- объявление объектов в любом месте кода
- ссылочный тип данных — создание “синонима” имени объекта обращение по имени, но работа как по указателю
- полиморфизм функций — функции могут имет одинаковые имена, но должны иметь разные сигнатуры
- значения параметров по умолчанию `int fun(double a = 3.14, int b = 0);`
- потоковый ввод-вывод, прегрузка операторов ввода-вывода для классов
- пространства имен namespace
- строгая типизация указателей, различные способы преобразования указателей static_cast, reinterpret_cast, dynamic_cast, const_cast
- управление памятью `new delete` (“вызовы” конструктора и деструктора)
- исключения, try-catch блоки

- механизм шаблонов template

Некоторые понятия рассматривались на предыдущей лекции. Некоторые мы здесь кратко поясним, некоторые оставим то тех пор, пока они не будут реально нужны.

Статические члены класса:

```
class A                              A  a, b, c;
{                                    a.x = 1;              a.fun();
 public:                             b.x = 2;              A::fun();
   int x;                            c.x = 3;
   static int y;                     A::y = 6;
   static void fun() { y = 5; }
};                                   b.y = 7;
int A::y;

static  - устанавливает зону видимости в рамках файла
        - сохранение и первичная инициализация в блоке
        - общий элемент для всех экземпляров класса
```

const методы:

```
struct A                                              void fun(const A & r)
{                                                     {
   int x;                                                int z = r.x;
   A(int xx) { x = xx; }                                  //r.x = 3;
   int Value() const { return x; }                        z = r.Value();
   int & rValue() { return x; }                           z = r.rValue();
   const int & rValue() const { return x; }               r.rValue() = 25;
};                                                    }
```

Полиморфизм функций и параметры по умолчанию

```
                                                       fun(3);
   int fun();
   int fun(int x);
   int fun(int x, double y);                          fun(1,2);  ???
   int fun(double x, double y);
   double fun(int x);   !!!!!!!!!

   int fun(double x = 1, double y = 2);
       fun(x, y);
       fun(x);
       fun();
```

Пространства имен namespace

```
   namespace ABC                     ABC::r = 1;         using namespace ABC;
   {                                                     r = 1;
       int r;
   }

   using namespace std;
```

Управление памятью new delete (“вызовы” конструктора и деструктора)

```
void * malloc(size_t num_of_bytes);
void   free(void * ptr);

new <конструктор>                                  Vector *p = new Vector(1,2,3);
new [количество] <конструктор без параметров>      Vector *q = new [128] Vector;
delete указатель                                   delete p;
delete [] указатель                                delete [] q;

при отказе - исключение std::bad_alloc
```

Исключения, try-catch блоки.

Сигнал, “распространяющийся” вверх по последовательности вызовов функций и, возможно, несущий с собой некоторое значение (экземпляр класса).

Исключение вызывается (throw) явно в коде (своем или библиотечном) или операционной системой в процессе выполнения (run time exception).

Этот сигнал можно перехватить try-catch блоком.

```
if (возникла ошибка) throw  25;      try {                  int main()
                                       .....                {
if (другая ошибка) throw  26;        } catch (int x) {        try
                                        if (x==25) ....         {
                                        if (x==26) ....            ......
                                     }                        } catch(...) {
                                                              {
                                                             }


class MyException {....};               try {
if (....) throw new MyException();        .....
                                        } catch (MyException * p) {
                                            cout << p->Code() << std::endl;
                                            cout <<  p->Message();
                                            delete p;
                                            exit(-1);
                                        }
```

Механизм шаблонов template

```
template<class T>                    int x, ax;       ax = abs(x);
T abs(T x)                           short y, ay;     ay = abs(y);
{                                    double z, az;    az = abs(z);
   return (x>=0) ? x : -x;           char * s, *as;  // as = abs(s);

}                                                     az = abs<double>(-3);
```

Точно также можно параметризовать описания и определения классов — об этом совсем скоро ...