ООП
Лекция 4

29 сентября, 2012 г.,
Алексей Лебедев

Обработка ошибок в стиле Си

```
int f()
{
   ...
   if (error)
      return 1;
   ...
}
...
int status = f();
if (status != 0)
   return 1
// и т. д.
```

C++: исключения

```
void f() { ... throw E(); ... }
void g() { ... f(); ... }
void h() { ... f(); ... }
int main()
{
   try {
      h();
   }
   catch (E &) {
      ...
   }
}
```

Проблема: ресурсы

```
void f()
{
   A *a = new A;
   g(); // исключение
   delete a;
}
void g()
{
   FILE *f = fopen(...);
   g(); // исключение
   fclose(f);
}
```

Уродливое решение: try/catch/finally

```
void f()
{
   A *a = new A;
   try {
      g(); // исключение
   }
   catch(...) {
      delete a;
      throw;
   }
   delete a;
}
```

Красивое (правильное) решение: деструкторы

```
class resource_owner {
public:
   resource_owner(); // захват ресурса
   ~resource_owner(); // освобождение ресурса
};

void f()
{
   resource_owner r(alloc_resource());
   g(); // исключение
   // тут вызовется деструктор!
}
```

Выделение ресурса есть инициализация

1. Освобождение памяти.
2. Закрытие файлов, сокетов и пр.
3. Снятие блокировки.

и т. д. и т. п.

Любым ресурсом должен владеть объект, деструктор которого освободит этот ресурс.

Деструкторы: не решают всех проблем

```
template <class T>
class vector {
   T *m;
public:
   vector(long n) {
      m = malloc(sizeof(T) * n);
      for (long i = 0; i != n; ++i)
         new (m + i) T(); // исключение в T()
      // ?
   }
};
```