Семестр 2. Лекция 3. Исключения.

Евгений Линский

10 Марта 2017

Stack unwinding - III

Во время stack unwinding вызываются деструкторы!

```
f() {
   MyArray a(100);
   if (...) throw MyException(...);
} // a. MyArray()

g() {
   Matrix m(10, 30);
   f();
} // m. Matrix()

main() {
   try {
      g();
   }
   catch(...) { }
}
```

Утечка памяти

```
f() {
  int *buffer = new int [n];
  if( ... ) throw MyExcpetion(...);
  delete [] buffer;
}
```

```
g() {
  Person *p = new Person("Jenya", 36, true);
  divide(c, e); // could throw exception
  delete p;
}
```

При возникновении исключения поток управления до *delete* не дойдет.

Идиома RAII

- Идиома в данном контексте — “так все делают =)”
- RAII — Resource Acquisition Is Initialization (“Взятие Ресурса Должно Происходить через Инициализацию” или как-то так)
- Взятие ресурса нужно “инкапсулировать” в класс, чтобы в случае исключения вызвался деструктор и освободил ресурс.

```
f() {
  MyArray buffer(n);
  if( ... ) throw MyExcpetion(...);
}
```

```
g() {
  auto_ptr p(new Person("Jenya", 36, true)); // or another smart ptr
  divide(c, e); // could throw exception
}
```

Исключения в конструкторе - I

Произошло исключение в divide, объект “недостроен”. Деструкторы у недостроенных объектов не вызываются.

```
class PhoneBookItem {
    PhoneBookItem(const char *audio, const char *pic) {
        af = fopen(audio, "r");
        pf = fopen(pic, "r");
        divide(c, e); // could throw excpetion
        f();
    }

    ~PhoneBookItem() {
        fclose(af);
        fclose(pf);
    }

};
```

Исключения в конструкторе - II

Надо предусмотреть такую ситуацию.

```
class PhoneBookItem {
    PhoneBookItem(const char *audio, const char *pic) {
        try {
            af = fopen(audio, "r");
            pf = fopen(pic, "r");
            divide(c, e); // could throw excpetion
            f();
        }
        catch(MyException& e) {
            fclose(af);
            fclose(pf);
        }
    }

    ...
};
```

Исключения в деструкторе - I

```
class PersonDatabase {
  ~PersonDatabase() {
    // throws exception if server is unavailable.
    networkLogger.log("Database is closed.");
    ...
  }
};

f() {
  PersonDatabase db;
  if(...) throws MyException("Error: disk is full.")
}
```

- Исключение от networkLogger может “подменить” исключение о том, что “места на диске больше нет”, и мы не узнаем истинную причину ошибки.
- Поэтому исключения в деструкторах бросать запрещено!
- Если это происходит, то программа аварийно завершается.

Исключения в деструкторе - II

Надо предусмотреть такую ситуацию.

```
class PersonDatabase {
  ~PersonDatabase() {
    try {
      // throws exception if server is unavailable.
      networkLogger.log("Database is closed.");
    }
    catch(...) { }//catch everything
  }
};
```

Гарантии при работе с исключениями

Гарантии:

- обязательства функции (метода) с точки зрения работы с исключениями
- документация для программиста, работающего с функцией (методом)

Виды гарантий:

- no throw guarantee — не бросает исключений вообще
- basic guarantee — в случае возникновения исключения ресурсы не утекают
- strong guarantee — переменные принимают те же значения, что были до возникновения ошибки

no throw

```
void strlen(const char *s) {
  int count = 0;
  while(*s != 0) {
    s++; count++;
  }
  return count;
}
```

```
void f() {
  try {
    strlen(...);
    divide(a, b);
  }
  catch(...) { //catch everything

  }
}
```

basic guarantee

- Если произойдет исключение, то измененные элементы MyArray свои значения не восстановят

```
class PersonDatabase {
  MyVector<Person> array;
  void process() {
    auto_ptr<Person> p(new Person(...));
    for(int i = 0; i < array.length; i++) {
      int a = divide( srand() / srand() ); // could
throw exception
      array[i]->setAge(a);
      std::cout << p;
    }
  }
};
```

- Функции, в которых мы в этой лекциях применяли RAII, обеспечивают как минимум basic guarante.

strong guarantee

Идиома: copy-and-swap

```
class PersonDatabase {
  MyVector<Person> array;
  void process() {
    auto_ptr<Person> p(new Person(...));
    MyVector<Person> copy(array);

    for(int i = 0; i < array.length; i++) {
      int a = divide( srand() / srand() ); // could
throw exception
      copy[i]->setAge(a);
      std::cout << p;
    }

    array = copy;

  }
};
```