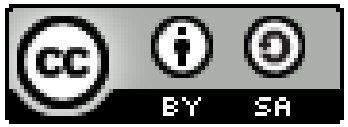
Исключительные ситуации

Создание и обработка



May 24, 2016

При работе программ происходят ошибки. При возникновении ошибок надо предпринимать определённые действия.
Есть два ключевых места в программе:

- место, где мы можем обнаружить ошибку;
- место, где мы можем её обработать.

Чаще всего эти места не совпадают. Следовательно, требуется механизм передачи информации от того места, где произошла ошибки, в то место, где её можно обработать.

Обработка ошибок в языке C

При возникновении ошибки можно:

- ничего не делать;
- аварийно завершить программу;
- потребовать взаимодействия с пользователем;
- вернуть код ошибки.

Самый общий способ – вернуть код ошибки – не всегда можно применить в программе на языке C++:

- есть вероятность того, что ошибку неявно проигнорируют;
- существуют функции, которые не могут возвращать коды ошибок (конструкторы, деструкторы, функции-операторы);
- бывают ситуации, когда не возможности получить этот код.

Исключения
Метод C++ для обработки ошибок

```
try { // – в этом блоке будем перехватывать ошибки
  if( /* здесь ошибка */ )
    throw some_error(args);
}
catch( xerror e )
{
   /* обработка ошибки типа xerror */
}
catch( some_error t ) {
   /* обработка ошибки типа some error */
}
catch( ... ) {
   /* обработка всех ошибок */
}
```

Действия при возникновении ошибки

1. Создаётся объект, сигнализирующий об исключительной ситуации.
2. Прерывается работа текущего блока.
3. Все локальные объекты этого блока уничтожаются.
4. Пункты 2 и 3 повторяются, пока не дойдёт до конка блока **try**. “Раскрутка стека”.
5. Ищется обработчик, тип которого подходит к типу ошибки. Если найден – вызывается его код. Ошибка считается обработанной. Выполнение программы идёт дальше. Если не не найден – переход на пункт 2.
6. Если разрушенный блок оказался функцией “main” – исключение считается неожиданным. Вызываются функции unexpected, terminate, abort.

Поиск подходящего обработчика

```
try {
  throw E;
}
catch( C c )
{
}
```

Обработчик ошибки с типом “C” считается подходящим к ошибке с типом “E”, если:

1. типы C и E совпадают;
2. E есть разновидность C: **class** E : **public** C {};;
3. C и E – указатели, и выполняется (1) или (2) для того на что они указывают;
4. C и E – ссылки, и выполняется (1) или (2) для того на что они ссылаются.

Повторная генерация исключения

Если обработчик приходит к выводу, что он не может обработать донное исключение, он может его сгенерировать повторно:

```
void f()
{
  try{
    /* здесь могут произойти исключения */
  }
  catch( my_error &e ) {
    if( /* могу обработать */ ) {
      // обрабатываю
      return;
    } else {
      throw;
    }
  }
}
```

Правила использования исключений

- При доставке объекта исключения в обработчик происходит его копирование (конструктор копирования). Следовательно, объект исключения должен позволять такое копирование, не создавать при этом новых исключений, и по возможности, делать это быстро.
- Если исключение происходит во время обработки другого исключения, сразу вызывается unexpected. Поэтому нельзя допускать, что бы исключение покидало деструктор.
- Обрабатывать исключения лучше по ссылке – так можно избежать “срезки” объекта исключения.
- При генерации исключений в конструкторе следует избегать появления не полностью сконструированных объектов. Метод “выделение ресурса есть инициализация”.
- Вызов исключения много дороже вызова функций. Использовать только для действительно исключительных ситуаций.

Пример обработки нехватки памяти

```
int* create_pool( int size )
{
  int *p;
  try {
    p = new int[ size ];
  }
  catch( std::bad_alloc ) {
    cerr << "Fail to alloc memory for "
         << size << " of ints\n";
    return 0;
  }
  return p;
}
```

Спецификации функций, связанные с исключениями

C++11: Если после списка аргументов функции используется слово “ **noexcept** , то такая функция “обещает” не создавать исключительных ситуаций.

```
void f( int x ) noexcept;
void f( int x ) noexcept
{
  return x + 1;
}
class A {
  public:
  ~A(); //  деструктор по умолчанию noexcept
  void g() const noexcept;
};
```

Если при вызове такой функции всё же произойдёт исключение, то оно считается *неожиданным*.
Старый C++: спецификации исключений: throw(...)