Объектно-ориентированное программирование

Исключения. Операции приведения типов. Шаблоны

Гришмановский Павел Валерьевич,
кафедра автоматики и компьютерных систем, Политехнический институт, СурГУ

Исключения

Исключительная ситуация – особая ситуация, возникающая во время выполнения программы, после которой дальнейшее выполнение программы (функции, метода, фрагмента кода) не имеет смысла (приводит к неадекватному поведению программы или невозможности продолжения)

- синхронные – возникают при выполнении определенных участков кода (структурная обработка)
- асинхронные – не зависят от состояния программы (неструктурная обработка)

Термин «исключение» означает:

- исключительная ситуация, событие
- объект (значение), обозначающий исключительную ситуацию

При возникновении исключения:

- выполнение основного потока прерывается
- управление передается обработчику исключения
- если его нет, то функция завершается и исключение выбрасывается выше и т.д.

Генерация и обработка исключений

Генерация (выброс) исключений:
- оператор **throw**

Структурная обработка исключений:
- защищаемый блок **try**
- обработчик (ловушка) исключения **catch**

Тип объекта исключения может быть любым

Ловушки учитывают наследование – вначале ловушки для исключений-потомков, затем – для предков

Если исключение выброшено из конструктора, то создание объекта прерывается, деструктор не вызывается, но память освобождается

```
throw <выражение>;

try
{
  // здесь ожидается исключение
}
catch (<тип> <имя>)
{
  // обработка исключения указанного типа
}
catch (<класс> & <имя>)
{
  // обработка исключения – объекта класса
}
catch (...)
{
  // обработка исключения любого типа
}
```

Исключения

Спецификация функций и методов:

- `throw (<список типов исключений>)`

Стандартные классы исключений:

- `exception`
- `bad_exception`
- `bad_alloc`
- `bad_cast`
- и др.

Операции приведения типов

Традиционное приведение типа

- операция приведения типа, в т.ч. перегруженная («в стиле С»)
  (<*тип*>**)** <*выражение*>
- конструктор преобразования («в стиле С++»)
  <*тип*>**(**<*выражение*>**)**

Операции приведения типов в С++:

- статическое приведение типов – аналог традиционного
  static_cast < <*тип*> > (<*выражение*>**)**
- изменение константности (снятие) указателя или ссылки
  const_cast < <*тип*> > (<*выражение*>**)**
- приведение несовместимых типов без проверки типов и двоичного представления
  reinterpret_cast < <*тип*> > (<*выражение*>**)**
- приведение типа указателя или ссылки с учетом наследования
  dynamic_cast < <*тип*> > (<*выражение*>**)**

Шаблоны функций

Шаблоны функций – параметризованные функции

```
template < typename T >
<т.в.з.> <имя_функции>([<сп.форм.парам.>])
{
    <тело_функции>
}
```

T – параметр шаблона, обозначающий обобщенный тип:

- используется в реализации функции вместо конкретного типа
- может быть несколько параметров (указываются через запятую)
- используются ключевые слова **typename** T или **class** T

Инстанцирование шаблона – использование с конкретным подставляемым типом

```
                    <имя_функции> ([<сп.факт.парам.>])
                    <имя_функции> < <тип> > ([<сп.факт.парам.>])
template <т.в.з.> <имя_функции> < <тип> > ([<сп.форм.парам.>])
```

Шаблоны классов

Шаблоны классов – параметризованные классы

```
template < typename T >
class <имя_класса>
{
    <описания_элементов>
};
```

Реализация метода вне класса – почти как шаблон функции

```
template < typename T >
<т.в.з.> <имя_класса> < T >::<имя_метода>([<сп.форм.парам.>])
{
    <тело_метода>
}
```

Инстанцирование шаблона класса – использование с конкретным подставляемым типом

```
                <имя_класса> < <тип> >
template class <имя_класса> < <тип> >;
```