

Исключения

13 июня 2018 г.

Исключительная ситуация (исключение) - ошибка, возникающая во время выполнения программы. При их возникновении во время работы программы автоматически вызывается обработчик исключений.

Управление механизмом обработки исключений держится на трех ключевых словах: `try`, `catch` и `throw`. Программные инструкции, которые вы считаете нужным проконтролировать на предмет исключений, помещаются в `try`-блок. Если там возникает исключение, оно дает о себе знать выбросом информации, благодаря `throw`. Эта информация перехватывается `catch`-блоком, который обрабатывает данное исключение.

Пример:

```
int main ()

int a,b,c;
cin>>a>>b;
try

if (b==0) throw 99;
else c=a/b;

catch (int i)

cout<<"На ноль делить нельзя!!!";

cout<<"Конец";
return 0;
```

* Если в `try`-блоке исключение не генерировалось, `catch`-блок не сработает. Программа его обойдет

* `Catch`-инструкций может быть несколько, каждая перехватывает свой тип исключения (в примере это `int`)

* `exit (EXIT-SUCCESS)` - завершение работы программы, с сообщением об успешном окончании; `exit (EXIT-FAILURE)` - завершение работы программы, с сообщением о неудачном окончании; `abort ()` - просто завершение работы программы. Чтобы пользоваться этими функциями необходимо подключить `<cstdlib>`

* `Catch`-выражение для базового класса перехватит исключение любого производного типа. Поэтому, если необходимо перехватывать исключения

как базового, так и производного типа, в catch-последовательности catch-инструкцию для производного типа необходимо поставить перед catch-инструкцией для базового типа.

Существует catch-инструкция перехватывающая исключения всех типов:

blackwhitecatch (...) Тело catch-инструкции; //... - в прямом смысле.

* Обычно перехват всех исключений помещают последним в очереди.

Существуют средства, которые позволяют ограничить тип исключений, которые может генерировать функция за пределами своего тел. Для этого нужно внести в определение функции throw-выражения:

"тип возвращаемого значения функции"(аргументы) throw (список имен типов) Тело функции Пример:

blackwhitevoid Xhandler (int test) throw (int, char, double)

* Можно повторно сгенерировать исключение в его обработчике (см. повторное генерирование исключения)

Пример по исключениям, когда ошибка ловится, но данные все равно потеряны:

- Работаем с текстом, в котором записаны числа. Мы переводим числа из типа char в int и пытаемся выполнить над ними какие-то арифметические действия, все в том же тексте.

Ошибка отлавливается, переведенные числа записываются в заранее определенные переменные, но, к примеру, не учитываются пробелы или знаки препинания (точка в конце строки) и действие все равно происходит не так, как мы хотим.

Представим, что мы работаем с некоторым классом MyClass и у нас есть переменные A и B, на классе определен оператор =. Представим такой код:

```
Try
Myclass T = A;
A=B;
B=T;
```

Если ошибка произойдет в последней строчке, то у нас теряется значение A, а в B лежит то же значение.

В таких случаях нужно заранее думать о том, что ошибка может возникнуть. И нужно заранее готовиться к возможности возникновения этой ошибки.

Что важно запомнить об исключениях:

- try-блок — так называемый блок повторных попыток. В нем надо располагать код, который может привести к ошибке и аварийному закрытию программы;
- throw генерирует исключение. То что остановит работу try-блока и приведет к выполнению кода

- catch-блока. Тип исключения должен соответствовать, типу принимаемого аргумента catch-блока;
- catch-блок — улавливающий блок, поймает то, что определил throw и выполнит свой код. Этот блок должен располагаться непосредственно под try-блоком. Никакой код не должен их разделять. если в try-блоке исключение не генерировалось, catch-блок не сработает. Программа его обойдет.