УДК 004.438

ОБРАБОТКА ИСКЛЮЧИТЕЛЬНЫХ СИТУАЦИЙ В РАЗЛИЧНЫХ ЯЗЫКАХ ПРОГРАММИРОВАНИЯ

Ладанова Екатерина Олеговна
студент
Тарасов Александр Владимирович
студент
Якушкин Рамиль Рашитович
студент

Мордовский государственный университет им. Н.П. Огарева, Саранск

author@apriori-journal.ru

Аннотация. В данной статье рассмотрены особенности обработки исключительных ситуаций в различных языках программирования. Проведен анализ обработки исключений на компилируемых языках (C++, C#, Java).

Ключевые слова: исключение, программирование, языки программирования.

EXCEPTION HANDLING IN DIFFERENT PROGRAMMING LANGUAGES

Ladanova Ekaterina Olegovna
student
Tarasov Alexander Vladimirovich
student
Yakushkin Ramil Rashitovich
student

Ogarev Mordovian State University, Saransk

Abstract. This article describes the features of exception handling in a variety of programming languages. Analysis of exception handling in the compiled languages (C ++, C #, Java).

Key words: exclusion, programming, programming languages.

Обработка исключительных ситуаций – механизм языков программирования, предназначенный для описания реакции программы на ошибки времени выполнения и другие возможные проблемы (исключения), которые могут возникнуть при выполнении программы и приводят к невозможности (бессмысленности) дальнейшей отработки программой её базового алгоритма.

Когда возникает исключительная ситуация, управление передаётся некоторому заранее определённому обработчику. Обработка ошибок передаётся на более высокий уровень и обеспечивается возможность нелокального выхода, т.е. передачи управления на некоторую «удалённую», возможно заранее неизвестную, точку программы через произвольное число вызовов функций. Основным недостатком исключений является невысокая скорость работы.

Большинство современных языков программирования, такие как Ada, C++, D, Delphi, Objective-C, Java, Eiffel, OCaml, Ruby, Python, Common Lisp, SML, PHP и все языки платформы .NET и др. имеют встроенную поддержку обработки исключений. В этих языках, при возникновении исключительной ситуации происходит раскрутка стека вызовов до первого обработчика исключений подходящего типа, и управление передаётся обработчику.

Особенности обработки исключений в C++, C# и Java. Родоначальником C++, C#, Java и множества других языков является язык программирования C. Язык C – это универсальный язык программирования, для которого характерны экономичность выражения, современный набор операторов и типов данных. Он является языком достаточно высокого уровня. Язык C близок к аппаратной части компьютера, т.к. создавался специально для системного программирования при создании операционных систем и программного обеспечения [1]. Он обладает всеми необходимыми для этого свойствами: программы, написанные на нем, очень эффективны, не требуют специальной среды поддержки

времени выполнения [2-3]. Поэтому нередко обучение студентов-программистов начинается с изучения языка С. В вузах проходят практические занятия, на которых студенты, с помощью преподавателя, предполагая, что он является носителем знаний [4], закрепляют свои навыки в изучении учебного материала. Занятия предусматривают выполнение различных практических заданий и лабораторных работ на компьютерной технике, которые помогают развивать способности программирования и намного лучше освоить изученный материал [5]. В курсе языков программирования обязательно рассматривается обработка исключительных ситуаций в программах. Рассмотрим, как происходит отлавливание ошибок в С-подобных языках программирования.

В языке С++ реализован специальный механизм исключительных ситуаций. Исключение возникает при выполнении оператора throw. В качестве аргумента throw задаётся любое значение. Это может быть значение одного из встроенных типов данных или объект любого класса, определённого в программе. При возникновении исключительной ситуации выполнение текущей функции или метода немедленно прекращается, созданные к этому моменту автоматические переменные уничтожаются, и управление передаётся в точку, откуда была вызвана текущая функция или метод. В точке возврата создаётся та же самая исключительная ситуация, прекращается выполнение текущей функции или метода, уничтожаются автоматические переменные, и управление передаётся в точку, откуда была вызвана эта функция или метод. Происходит своего рода откат всех вызовов до тех пор, пока не завершится функция main и, соответственно, вся программа. В С++ есть возможность определить для исключений иерархию классов – по классу на каждый тип исключительной ситуации. Чтобы облегчить обработку ошибок и сделать запись о них более наглядной, описания методов и функций можно дополнить информацией, какого типа исключительные ситуации они могут создавать:

```
class Database
{
public :
Open(const char*serverName) throw ConnectDbException;
 };
```

Такое описание говорит о том, что метод Open класса Database может создать исключительную ситуацию типа ConnectDbException. Соответственно, при использовании этого метода желательно предусмотреть обработку возможной исключительной ситуации. Если исключительная ситуация возникла в конструкторе объекта, считается, что объект сформирован не полностью, и деструктор для него вызван не будет. В общем случае, пример иллюстрирует также то, что подключение к базам данных представляет собой одно из слабых мест в работе программы. В силу самых разных причин клиент может не получить доступ к базе данных. При возникновении исключительной ситуации при соединении с БД MS Access обычно возникает исключение типа OleDbException [6-7]. Поэтому при создании таких приложений следует обязательно включать обработку исключений[8].

Язык C# наследовал схему исключений языка C++, внеся в неё свои коррективы [2]. Рассмотрим схему подробнее и начнём с синтаксиса конструкции try-catch-finally:

```
try {…}
catch (T1 e1)
{…}
 …
```

```
catch(Tk ek)
{…}
finally
{…}
```

Всюду в тексте модуля, где синтаксически допускается использование блока, этот блок можно сделать охраняемым, добавив ключевое слово try. Вслед за try-блоком могут следовать catch-блоки, называемые блоками-обработчиками исключительных ситуаций, их может быть несколько, они могут и отсутствовать. Завершает эту последовательность finally-блок – блок финализации, который также может отсутствовать. Вся эта конструкция может быть вложенной – в состав try-блока может входить конструкция try-catch-finally. В рассматриваемой нами модели исключения являются объектами, класс которых представляет собой наследника класса Exception. Этот класс и многочисленные его наследники являются частью библиотеки FCL, хотя и разбросаны по разным пространствам имён. Каждый класс задаёт определённый тип исключения в соответствии с классификацией, принятой в Framework.Net. При выполнении оператора throw создается объект te, класс TE которого характеризует текущее исключение, а поля содержат информацию о возникшей исключительной ситуации. Выполнение оператора throw приводит к тому, что нормальный процесс вычислений на этом прекращается. Если это происходит в охраняемом try-блоке, то начинается этап «захвата» исключения одним из обработчиков исключений. Класс T, указанный в заголовке catch-блока, должен принадлежать классам исключений. Блок catch с формальным аргументом e класса T потенциально способен захватить текущее исключение te класса TE, если и только если объект te совместим по присваиванию с объектом e. Другими словами, потенциальная способность захвата означает допустимость присва-

ивания e = te, что возможно, когда класс TE является потомком класса T. Обработчик, класс T которого является классом Exception, является универсальным обработчиком, потенциально он способен захватить любое исключение, поскольку все они являются его потомками. Потенциальных захватчиков может быть много, исключение захватывает лишь один – тот из них, кто стоит первым в списке проверки. Когда же будет исчерпаны списки вложенных блоков, а потенциальный захватчик не будет найден, то произойдет подъем по стеку вызовов.

К механизму обработки исключений в Java имеют отношение 5 ключевых слов: — try, catch, throw, throws и finally. Схема работы этого механизма следующая. Вы пытаетесь try выполнить блок кода, и если при этом возникает ошибка, система возбуждает throw исключение, которое в зависимости от его типа вы можете перехватить catch или передать умалчиваемому finally обработчику. В вершине иерархии исключений стоит класс Throwable. Каждый из типов исключений является подклассом класса Throwable. Два непосредственных наследника класса Throwable делят иерархию подклассов исключений на две различные ветви. Один из них – класс Exception – используется для описания исключительных ситуации, которые должны перехватываться программным кодом пользователя. Другая ветвь дерева подклассов Throwable – класс Error, который предназначен для описания исключительных ситуаций, которые при обычных условиях не должны перехватываться в пользовательской программе. Только подклассы класса Throwable могут быть возбуждены или перехвачены. Простые типы – int, char и т.п., а также классы, не являющиеся подклассами Throwable, например, String и Object, использоваться в качестве исключений не могут. Наиболее общий путь для использования исключений – создание своих собственных подклассов класса Exception.

Следует отметить, что вложенная обработка исключений в языках C++ и C# больше нагружает систему при выполнении, чем обработка ис-

ключений на месте их возникновения, но не значительно. В языке Java обработка исключений на месте их возникновения практически не влияет на скорость работы программы, а при подъеме по стеку вызовов значительно замедляет программу. Во всех языках программирования исключения снижают производительность, особенно если их обработка передаётся выше по стеку вызова функций. Следовательно, следует использовать исключения максимально разумно – там, где нельзя их избежать и там, где это диктуется архитектурой приложения, исходя из предъявленных требований [9-10], но при этом и не игнорировать критичные исключения.

Список использованных источников

1. Александров Э.Э., Афонин В.В. Введение в программирование на языке С. Саранск: Изд-во Мордов. ун-та, 2009. 316 с.
2. Александров Э.Э., Афонин В.В. Программирование на языке С в Microsoft Visual Studio 2010 [Электронный ресурс]. Режим доступа: http://www.intuit.ru/department/pl/prcmsvs2010 (дата обращения: 19.11.2015).
3. Александров Э.Э., Афонин В.В. Программирование на языке С в Microsoft Visual Studio 2010. Саранск: Изд-во Мордов. ун-та, 2010. 424 с.
4. Ладанова Е.О. Сможет ли сознание робота полностью заменить сознание человека? // APRIORI. Серия: Естественные и технические науки. 2015. № 5. Режим доступа: http://www.apriori-journal.ru/seria2/5-2015/Ladanova.pdf (дата обращения: 19.11.2015).

5. Афонин В.В., Федосин С.А. О структурировании лабораторно-практических занятий при изучении дисциплин программирования // Образовательные технологии и общество. 2014. Т. 17. № 4. С. 497-506. [Электронный ресурс]. Режим доступа: http://ifets.ieee.org/russian/periodical/izgurn.html (дата обращения: 19.11.2015).
6. Аббакумов А.А., Акимов В.Л., Егунова А.И., Лещанкин К.А., Таланов В.М. Базы данных (MS ACCESS, MYSQL). Саранск: Изд-во Средне-волжского математического общества, 2011. 112 с.
7. Аббакумов А.А., Акимов В.Л., Егунова А.И., Лещанкин К.А., Таланов В.М. Базы данных (MS ACCESS, MYSQL). Саранск: Изд-во Средне-волжского математического общества, 2015. 66 с.
8. Егунова А.И., Таланов В.М. Защита баз данных при организации портала электронного магазина. Проблемы управления, обработки и передачи информации (АТМ-2013) сб. трудов III Междунар. научной конф.: в 2 т. / под ред. А.А. Львова, М.С. Светлова. Саратов: Издательский ом «Райт-Экспо», 2013. Т. 1.С. 136-138.
9. Афонин В.В. Основы анализа систем массового обслуживания. [Электронный ресурс]. Режим доступа: http://elibrary.ru/item.asp?id=19581660 (дата обращения: 17.11.2015).
10. Афонин В. В. Анализ управляемости нелинейных аффинных систем управления в системе Matlab // Вестник мордовского университета. 2012. № 2. С. 177-181.