***Лабораторная работа 6.***

Оглавление

[1. Обработка исключений 2](#_Toc318767113)

[2. Элемент управления список 4](#_Toc318767114)

[3. Использование списка для вывода одномерного массива в Windows-приложениях 6](#_Toc318767115)

[4. Задания. Алгоритмы обработки одномерных массивов 9](#_Toc318767116)

# Обработка исключений

Язык С#, как и многие другие объектно-ориентированные языки, реагирует на ошибки и ненормальные ситуации с помощью механизма обработки исключений. Исключение - это объект, генерирующий информацию о "необычном программном происшествии". При этом важно проводить различие между ошибкой в программе, ошибочной ситуацией и исключительной ситуацией.

Ошибка в программе допускается программистом при ее разработке. Например, вместо операции сравнения (==) используется операция присваивания (=). Программист должен исправить подобные ошибки до передачи кода программы заказчику. Использование механизма обработки исключений не является защитой от ошибок в программе.

Ошибочная ситуация вызвана действиями пользователя. Например, пользователь вместо числа ввел строку. Такая ошибка способна вызывать исключение. Программист должен предвидеть ошибочные ситуации и предотвращать их с помощью операторов, проверяющих допустимость поступающих данных.

Даже если программист исправил все свои ошибки в программе, предвидел все ошибочные ситуации, он все равно может столкнуться с непредсказуемыми и неотвратимыми проблемами - исключительными ситуациями. Например, нехваткой доступной памяти или попыткой открыть несуществующий файл. Исключительные ситуации программист предвидеть не может, но он может отреагировать на них так, что они не приведут к краху программы.

Для обработки ошибочных и исключительных ситуаций в С# используется специальная подсистема обработки исключений. Преимущество данной подсистемы состоит в автоматизации создания большей части кода по обработке исключений. Раньше этот код приходилось вводить в программу "вручную". Кроме этого обработчик исключений способен распознавать и выдавать информацию о таких стандартных исключениях, как деление на нуль или попадание вне диапазона определения индекса.

***Оператор try***

В С# исключения представляются классами. Все классы исключений порождены от встроенного класса исключений Exception, который определен в пространстве имен System.

Управление обработкой исключений основывается на использовании оператора try. Синтаксис оператора:

try // контролируемый блок

{

…

}

catch //один или несколько блоков обработки исключений

{

…

}

finally //блок завершения

{

…

}

Программные инструкции, которые нужно проконтролировать на предмет исключений, помещаются в блок try. Если исключение возникает в этом блоке, оно дает знать о себе выбросом определенного рода информации. Выброшенная информация может быть перехвачена и обработана соответствующим образом с помощью блока catch. Любой код, который должен быть обязательно выполнен при выходе из блока try, помещается в блок finally. Рассмотрим пример, демонстрирующий, как отследить и перехватить исключение.

static void Main()

{

int x = int.Parse(Console.ReadLine());

int y =1 / x;

Console.WriteLine(y);

}

Перечислим, какие исключительные ситуации могут возникнуть:

1. пользователь может ввести нечисловое значение
2. если ввести значение 0, то произойдет деление на 0.

Теперь попробуем обработать эти ситуации. Для этого изменим код следующим образом.

static void Main()

{

try

{

int x = int.Parse(Console.ReadLine());

int y =1 / x;

Console.WriteLine("y={0}", y);

Console.WriteLine("блок try выполнился успешно");

}

catch // \*

{

Console.WriteLine("возникла какая-то ошибка");

}

Console.WriteLine("конец программы");

}

Рассмотрим, как обрабатываются исключения в данном примере. Когда возникает исключение, выполнение программы останавливается и управление передается блоку catch. Этот блок никогда не возвращает управление в то место программы, где возникло исключение. Поэтому команды из блока try, расположенные ниже строки, в которой возникло исключение, никогда не будут выполнены. Блок catch обрабатывает исключение, и выполнение программы продолжается с оператора, следующего за этим блоком.

В нашем случае при вводе нечислового значения или 0 будет выведено сообщение "возникла ошибка", а затем сообщение "конец программы".

Обработчик исключений позволяет не только отловить ошибку, но и вывести полную информацию о ней. Для демонстрации сказанного заменим блок catch следующим фрагментом.

catch (Exception error)

{

Console.WriteLine("Возникла ошибка {0}", error);

}

Теперь, если возникнет исключительная ситуация, "выброшенная" информация будет записана в идентификатор error. Данную информацию можно просмотреть с помощью метода WriteLine. Такое сообщение очень полное и будет полезно только разработчику на этапе отладки проекта.

Для пользователя на этапе эксплуатации приложения достаточно более краткой информации о типе ошибке. С этой целью в С# выделены стандартные классы исключений, такие как DivideByZeroException, FormatException. Внесем изменения в программу.

static void Main()

{

try

{

int x = int.Parse(Console.ReadLine()); // 1 ситуация

int y =1 / x; // 2 ситуация

Console.WriteLine("y={0}", y);

Console.WriteLine("блок try выполнился успешно");

}

catch(FormatException) // обработка 1 ситуации

{

Console.WriteLine("Ошибка: введено нечисловое значение!");

}

catch (DivideByZeroException) // обработка 2 ситуации

{

Console.WriteLine("Ошибка: деление на 0!");

}

Console.WriteLine("конец программы");

}

В данном примере обрабатывается каждая ситуация в отдельности, при этом пользователю сообщается лишь минимальная информация об ошибке. В следующей таблице содержится описание наиболее часто используемых обработчиков стандартных исключений.

| **Имя** | **Описание** |
| --- | --- |
| ArithmeticException | Ошибка в арифметических операциях или преобразованиях |
| ArrayTypeMismatchException | Попытка сохранения в массиве элемента несовместимого типа |
| DivideByZeroException | Попытка деления на ноль |
| FormatException | Попытка передать в метод аргумент неверного формата |
| IndexOutOfRangeException | Индекс массива выходит за границу диапазона |
| InvalidCastException | Ошибка преобразования типа |
| OutOfMemoryException | Недостаточно памяти для нового объекта |
| OverflowException | Переполнение при выполнении арифметических операций |
| StackOverflowException | Переполнение стека |

# Элемент управления список

Приложения Windows позволяют построить дружелюбный интерфейс пользователя, облегчающий работу по вводу и выводу массивов. Для работы с одномерными массивами в Windows-приложениях часто используются списки. К членам семейства списков относятся CheckedListBox (список с флажками), ListBox (список) и ComboBox (комбинированный список).

**Список с флажками.** Элемент управления CheckedListBox (список с флажками) позволяет помещать обычные флажки внутри поля с полосами прокрутки. Кроме того, в этом элементе управления предусмотрена возможность использования нескольких столбцов. Для этого достаточно установить значение true для свойства MultiColumn.

CheckedListBox наследует большинство своих возможностей от типа ListBox. To же самое справедливо и в отношении класса ComboBox. Наиболее важные свойства System.Windows.Forms.ListBox представлены в следующей таблице.

|  |  |
| --- | --- |
| **Свойства класса ListBox** | |
| **Свойство** | **Назначение** |
| ScrollAlwaysVisible | Определяет, будет ли полоса прокрутки выводиться всегда |
| SelectedIndex | Индекс выделенного в настоящий момент элемента в списке (если такой имеется). Если ни один элемент не выделен, то возвращается значение -1 |
| SelectedIndices | Набор индексов выделенных в настоящий момент элементов в списке. Если не выделен ни один элемент, то возвращается пустой набор |
| SelectedItem | Значение выделенного в настоящий момент элемента. Если ни один из элементов не выделен, то возвращается null |
| SelectedItems | Возвращает коллекцию значений выделенных элементов (для списков, в которых допускается выбор нескольких значений) |
| SelectionMode | Определяет число элементов, которые возможно выбрать в списке одновременно. Для этого свойства используются значения из перечисления SelectionMode |
| Sorted | Определяет, будут ли элементы в списке упорядочены (по алфавиту) или нет |
| TopIndex | Возвращает индекс первого видимого элемента в списке |

Помимо свойств в классе ListBox определены также многочисленные методы.

**Комбинированные списки.** Как и списки (объекты ListBox ), комбинированные списки (объекты ComboBox ) позволяют пользователю производить выбор из списка заранее определенных элементов. Однако у комбинированных списков есть одно существенное отличие от обычных: пользователь может не только выбрать готовое значение из списка, но и ввести свое собственное. Класс ComboBox наследует большинство своих возможностей от класса ListBox (который, в свою очередь, является производным от Control ), однако в нем предусмотрены и собственные важные свойства, представленные в следующей таблице:

| **Свойства класса ComboBox** | |
| --- | --- |
| **Свойство** | **Назначение** |
| DroppedDown | "Раскрывающийся вниз": определяет, будет ли список ниспадающим |
| MaxDropDownItems | Определяет максимальное количество элементов, которое будет показано в нижней части ниспадающего списка. Допустимые значения - от 1 до 100 |
| MaxLength | Определяет максимальную длину текста, который пользователь может ввести в ComboBox |
| SelectedIndex | Определяет индекс выделенного элемента ComboBox. Если ни один элемент не выделен, возвращается значение -1 |
| SelectedItem | Возвращает ссылку на объект выделенного элемента ComboBox |
| SelectedText | Возвращает выделенный текст в поле редактирования ComboBox |
| SelectionLength | Определяет длину (в символах) выделенного текста в поле редактирования ComboBox |
| Style | Позволяет получить или установить стиль ComboBox. Для этого свойства используются значения из перечисления ComboBoxStyle |
| Text | Позволяет получить доступ к тексту в поле редактирования. При работе с ComboBox это унаследованное свойство используется чаще всех остальных |

Стиль для ComboBox можно настроить при помощи свойства Style, для которого используются значения из перечисления ComboBoxStyle.

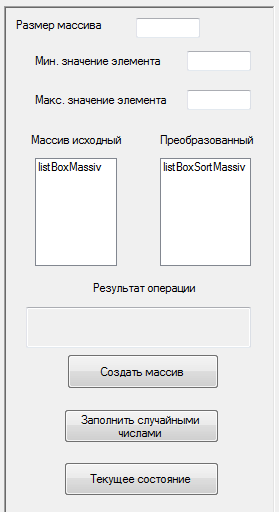
|  |  |
| --- | --- |
| **Значения перечисления ComboBoxStyle** | |
| **Значение** | **Описание** |
| DropDown | Пользователь может вводить значения в поле редактирования. Для отображения списка пользователь должен нажать на кнопку со стрелкой, направленной вниз ( Arrow Button ) |
| DropDownList | Пользователь не может вводить значения в поле редактирования. Для отображения списка пользователь должен нажать на кнопку со стрелкой, направленной вниз ( Arrow Button ) |
| Simple | Пользователь может вводить значения в поле редактирования. Список значений виден всегда |

# Использование списка для вывода одномерного массива в Windows-приложениях

В Windows-приложениях при работе с массивами проходят через те же этапы, что рассматривались для консольных приложений (см. пример класса Arrs, рассмотренный на лекции по теме «Массивы»), к этим этапам можно отнести следующие:

* Объявление массива
* Создание массива
* Инициализация (заполнение) массива
* Отображение исходного массива
* Обработка массива
* Отображение измененного массива

На следующем рисунке приведен пример элементов соответствующего интерфейса Windows-приложения для работы с одномерным массивом. Здесь рассматривается одномерный динамический массив, для отображения его элементов до преобразования и после используются два списка типа ListBox.



Посмотрим, как это все организовано программно. Предположим, что у нас создана форма(класс) FormList

public partial class FormList : Form

{

. . .

}

Форма включает следующие поля и методы. Начнем с полей формы FormList, показанной на рисунке.

//fields

int size, minValue, maxValue;

int[] arr;

Random rnd = new Random();

Полями этого класса является одномерный массив, его размер, границы диапазона случайных целых чисел, объект класса Random, используемый для генерирования случайных чисел.

Обработчик события Click командной кнопки, отвечающей за создание массива, имеет вид:

private void buttonCreateMassiv\_Click(object sender, EventArgs e)

{

try

{

size = Convert.ToInt32(textBoxSize.Text);

arr = new int[size];

sortArr = new int[size];

textBoxResult.Text = ". . . ";

}

catch (Exception err)

{

//textBoxResult.Text = ". . . ";

textBoxResult.Text = err.ToString();

}

}

Первым делом принимается размер массива, введенный пользователем. Преобразование к типу int введенного значения помещено в охраняемый блок, поэтому ошибки некорректного ввода будут перехвачены с выдачей соответствующего сообщения в поле textBoxResult. Если же массив успешно создан, в это поле также будет выдано сообщение об успешном выполнении операции.

Следующий метод демонстрирует считывание с полей формы textBoxMinValue и textBoxMaxValue границ диапазона случайных целых чисел minValue и maxValue. Метод throw (new Exception()) генерирует исключение в случае ошибочной ситуации для границ диапазона. Далее массив заполняется случайными целыми числами из заданного диапазона.

private void buttonFillRnd\_Click(object sender, EventArgs e)

{

try

{

minValue = Convert.ToInt32(textBoxMinValue.Text);

maxValue = Convert.ToInt32(textBoxMaxValue.Text);

if (minValue > maxValue) throw (new Exception());

//Заполнение массива случайными числами

for (int i = 0; i < size; i++)

arr[i] = rnd.Next(minValue, maxValue);

textBoxResult.Text = ". . . ";

}

catch (Exception)

{

textBoxResult.Text = ". . . ";

}

}

Следующий метод демонстрирует отображение (вывод) созданного исходного массива в виде линейного списка listBoxMassiv типа System.Windows.Forms.ListBox. Семейство listBoxMassiv.Items содержит коллекцию элементов этого списка. Сначала выполняется очистка списка – удаление всех его элементов с помощью метода listBoxMassiv.Items.Clear(), затем в цикле в список добавляются последовательно все элементы массива arr[i], для чего используется метод listBoxMassiv.Items.Add(...)

private void buttonCurrentState\_Click(object sender, EventArgs e)

{

listBoxMassiv.Items.Clear();

for (int i = 0; i < size; i++)

listBoxMassiv.Items.Add(arr[i]);

textBoxResult.Text = ". . . ";

}

Далее на данную форму можно добавить управляющую кнопку, запускающую процесс преобразования массива и еще одну кнопку, запускающую процедуру отображения преобразованного массива во втором списке listBoxSortMassive.

# Задания. Алгоритмы обработки одномерных массивов

1. Дан одномерный массив из случайных целых чисел. Требуется найти минимальный и максимальный элементы массива (и их индексы) и поменять их местами соответственно с первым и последним элементом массива. Вывести в поля формы в качестве результата значения найденных элементов, их индексы. Преобразованный массив отобразить в виде линейного списка.
2. Дан одномерный массив из случайных целых чисел. Переписать элементы массива в обратном порядке. Преобразованный массив отобразить в виде линейного списка.
3. Дан одномерный массив из случайных целых чисел. Преобразовать массив так, чтобы все неположительные элементы размещались в том же порядке в начале массива, а все положительные – в том же порядке, но - в конце массива. Преобразованный массив отобразить в виде линейного списка.
4. Дан одномерный массив из случайных целых чисел. Реализовать процедуру бинарного поиска некоторого заданного числа в массиве. В качестве результата указать индекс найденного элемента в массиве, если элемент существует и соответствующее сообщение.
5. Дан одномерный массив из случайных целых чисел. Создать два массива, в первый – переписать четные элементы исходного массива, а во второй – нечетные. Построенные массивы отобразить на форме в виде линейных списков.
6. Дан одномерный массив из случайных целых чисел. Обнулить все отрицательные элементы, а положительные уплотнить – сдвинуть к началу массива. Полученный массив (вместе с нулевыми элементами) отобразить в виде линейного списка.
7. Дан одномерный массив из случайных целых чисел. Отобразить в виде списка номера всех минимальных элементов массива.
8. Дан одномерный массив из случайных целых чисел. Заменить все максимальные элементы нулями. Полученный массив (вместе с нулевыми элементами) отобразить в виде линейного списка.
9. Дан одномерный массив из случайных целых чисел. Построить список элементов массива, не совпадающих с максимальным. Указать максимальный элемент.
10. Дан одномерный массив из случайных целых чисел. Найти номер последнего минимального элемента и номер последнего максимального.
11. Дан одномерный массив из случайных целых чисел. Подсчитать сумму элементов, расположенных между первым максимальным и последним минимальными элементами. Если максимальный элемент встречается позже минимального, то выдать сообщение об этом.
12. Дан одномерный массив из случайных целых чисел. Найти максимум из отрицательных элементов.
13. Дан одномерный массив из случайных целых чисел. Найти минимум из положительных элементов.
14. Дан одномерный массив из случайных целых чисел. Найти количество пар соседних элементов, разность между которыми равна заданному числу. Построить список из этих пар.
15. Дан одномерный массив из случайных целых чисел. Подсчитать количество элементов, значения которых больше значения предыдущего элемента. Построить список из таких элементов.
16. Дан одномерный массив из случайных целых чисел. Найти количество пар соседних элементов, в которых предыдущий элемент кратен последующему. Построить список таких пар.
17. Дан одномерный массив из случайных целых чисел. Найти количество пар соседних элементов, в которых предыдущий элемент меньше последующего. Построить список таких пар.
18. Дан одномерный массив из случайных целых чисел. Найти сумму максимального и минимального элементов массива.
19. Дан одномерный массив из случайных целых чисел. Найти произведение элементов находящихся на четных позициях в массиве.
20. Дан одномерный массив из случайных целых чисел. Найти количество и порядковые номера элементов массива вещественных чисел, удовлетворяющих условию 2,1 ≤ *xi* ≤ 7.
21. Дан одномерный массив из случайных целых чисел. Поменять местами в массиве вещественных чисел максимальный и минимальный элементы и вывести массив на экран дисплея.
22. Дан одномерный массив из случайных целых чисел. Найти среднее геометрическое элементов массива вещественных чисел.
23. Дан одномерный массив из случайных целых чисел. Найти порядковый номер максимального элемента среди четных значений массива целых чисел
24. Дан одномерный массив из случайных целых чисел. Найти количество и сумму положительных элементов в массиве вещественных чисел, удовлетворяющих условию 0<*xi*<5,7.
25. Дан одномерный массив из случайных целых чисел. Заменить отрицательные элементы массива находящиеся на нечетных позициях единицей и вывести новый массив на экран дисплея.
26. Дан одномерный массив из случайных целых чисел. Найти произведение элементов находящихся на четных позициях в массиве. Вывести элементы находящихся на четных позициях
27. Дан одномерный массив из случайных целых чисел. Вывести на экран дисплея положительные элементы массива и найти их количество.
28. Дан одномерный массив из случайных целых чисел. Вывести на экран дисплея номера элементов массива, удовлетворяющих условию 0<*xi*<3,2.
29. Дан одномерный массив из случайных целых чисел. Заменить четные элементы в массиве целых чисел нулями и вывести новый массив на экран.
30. Дан одномерный массив из случайных целых чисел. В массиве вещественных чисел найти произведение элементов, стоящих на четных позициях.