Темы урока

Ветвления

- Ветвление – это команда алгоритма, в которой делается выбор, выполнять или не выполнять какую-нибудь группу команд в зависимости от условий.
- На алгоритмической схеме ветвление изображают в виде ромба, имеющего один вход и два выхода. Внутри ромба пишется утверждение. В зависимости от истинности утверждения выполняется та или иная ветка кода.

Условный оператор if...else

Объясняем по слайдам.

- Отметить, что после блоков if...else, алгоритм снова становится линейным.
- Отметить, что блока “else” может не быть вовсе, если он не нужен.
- Отметить, что может быть множественное повторение `if...else` `if...else` `if..else`. Отобразить, как это выглядит на алгоритмической схеме.

Самостоятельная работа

Задание

Написать программу с блоками ветвления:

- Программа запрашивает у пользователя количество лет договора аренды в диапазоне от 1 до 30.
- Если введено значение за этими пределами, программа выводит сообщение *"Вы ввели неверное значение!"* и завершается.
- Если значение находится в диапазоне [1..30], на экран должна вывестись по-русски грамматически корректная фраза о длительности заключённого договора

Примеры работы

```
> Введите длительность договора аренды в годах: 21 /это ввод пользователя/
> Договор аренды оформлен на период длительностью 21 год

> Введите длительность договора аренды в годах: 5 /это ввод пользователя/
> Договор аренды оформлен на период длительностью 5 лет

> Введите длительность договора аренды в годах: 35
> Вы ввели неверное значение!
```

Подсказка

- `1, 21, 31    > год`
- `2–4, 22–24   > года`
- `5–20, 25–30  > лет`

Решение

Код: **L05_SW_C01_string_years**:

```
static void Main(string[] args)
{
	Console.Write("Введите длительность договора аренды в годах: ");
	string inputString = Console.ReadLine();

	int numberOfYears = int.Parse(inputString);

	if (numberOfYears < 0 || numberOfYears > 30)
	{
		Console.WriteLine("Вы ввели неверное значение!");
		return;
	}

	string yearsString = string.Empty;

	if (numberOfYears == 1
		|| numberOfYears == 21)
	{
		yearsString = "год";
	}
	else if ((numberOfYears > 1 && numberOfYears < 5)
		|| (numberOfYears > 21 && numberOfYears < 25))
	{
		yearsString = "года";
	}
	else
	{
		yearsString = "лет";
	}

	Console.WriteLine(
		"Договор аренды оформлен на период длительностью "
		+ $"{numberOfYears} {yearsString}");
}
```

Важно отметить

В решении стоит обратить внимание, что всё, что в условном блоке имеет смысл оставлять только действительно меняющиеся от условия вещи. Например, мы оставили только формирование переменной окончания фразы в условных блоков, а вывод фразы уже организовали за пределами условия, так как он происходит в любом случае. И даже начало фразы в любом случае — одинаковое.

Тернарный условный оператор ?:

Терна́рная условная операция (от латинского ternarius — “тройной”).

- Часто выступает заменой однострочному `if...else`
- Блок `else` обязателен, в отличие от `if...else`

Самостоятельная работа

Задание

Написать программу с оператором ?:

- Программа запрашивает число от 0 до 100
- Затем в зависимости от введенного числа будет выводиться:
  - если число меньше 50: “Введенное число меньше 50”
  - в если число больше либо равно 50: “Введенное число больше либо равно 50”

Решение

Код **L05_SW_C02_ternary_operator**:

```
static void Main(string[] args)
{
	Console.Write("Введите число от 1 до 100: ");
	int number = int.Parse(Console.ReadLine());

	Console.WriteLine(number < 50
		? "Введенное число меньше 50"
		: "Введенное число больше либо равно 50");
}
```

Конструкция switch

Конструкция switch отличается от `if...else`: она позволяет сравнить выражение с набором возможных значений.

Подчеркнуть:

- Не может проверять на диапазон! Только конкретное значение!

Самостоятельная работа

Задание

- Модифицировать программу задания на условие if...else (про длительность договора от 1 до 30 лет) используя для определения правильной формы числительного не `if...else`, а `switch`.
- Внешне программа не должна отличаться от предыдущей реализации.

Решение

Код **L05_SW_C03_switch_instead_if_else**:

```
static void Main(string[] args)
{
	Console.Write("Введите длительность договора аренды в годах: ");
	string inputString = Console.ReadLine();

	int numberOfYears = int.Parse(inputString);

	if (numberOfYears < 0 || numberOfYears > 30)
	{
		Console.WriteLine("Вы ввели неверное значение!");
		return;
	}

	string yearsString = string.Empty;

	switch (numberOfYears)
	{
		case 1:
		case 21:
			yearsString = "год";
			break;
		case 2:
		case 3:
		case 4:
		case 22:
		case 23:
		case 24:
			yearsString = "года";
			break;
		default:
			yearsString = "лет";
			break;
	}

	Console.WriteLine(
		"Договор аренды оформлен на период длительностью "
		+ $"{numberOfYears} {yearsString}");
}
```

Исключения

- Исключения позволяют обозначить, что во время выполнения программы произошла ошибка.
- Объекты исключений, описывающие ошибку, создаются и затем вызываются с помощью ключевого слова **throw**.
- Программисты должны вызывать исключения в том случае, если прогнозируется неверное поведение программы.
- Объекты исключений наследуются от базового класса **System.Exception**.
- Исключения не рекомендуется использовать для изменения потока программы в рамках обычного выполнения. Их следует использовать только для сообщения о состояниях ошибки и их обработки.

Генерация собственного исключения

- `throw new Exception()`
- Следует использовать только для обозначения ошибки, не использовать для ветвлений!

Самостоятельная работа

Задание

Модифицировать программу из задачи на блок switch таким образом, чтобы в случае, если число выходит за диапазон от 1 до 30, генерировалось новое исключение с текстом “Введенное значение выходит за допустимые пределы от 1 до 30”.

Решение

Код **L05_SW_C04_exception_for_range**:

```
static void Main(string[] args)
{
	Console.Write("Введите длительность договора аренды в годах: ");
	string inputString = Console.ReadLine();

	int numberOfYears = int.Parse(inputString);

	if (numberOfYears < 1 || numberOfYears > 30)
	{
		throw new Exception(
			"Введенное значение выходит за допустимые пределы от 1 до 30!");

		Console.WriteLine("Вы ввели неверное значение!");
		return;
	}

	...
}
```

Обработка исключений

- Функции обработки исключений помогают справиться с непредвиденными или исключительными проблемами, которые возникают при выполнении программы.
- Обработка исключений использует ключевые слова `try`, `catch` и `finally` для действий, которые могут оказаться неудачными.
- Существует три возможные комбинации этих блоков
    - `try...catch`
    - `try...finally`
    - `try...catch...finally`
- Рассмотрим каждый блок детально:
    - В блок **`try`** помещаются потенциально опасные команды (которые могут генерировать исключения). Как только в какой-либо строке внутри блока try происходит исключение, дальнейшее выполнение команд внутри этого блока прерывается и начинают выполняться команды блока catch, если он есть, если его нет, то finally (один из блоков – catch или finally обязательно должен быть определен)
    - В блоке **`catch`** помещаются команды, которые должны быть вызваны, если в блоке `try` сгенерировалось исключение. Если в блоке `try` ошибок не произошло, выполнение команд блока catch будет пропущено.
      В блоке `catch` обычно выполняется логирование ошибок. Если не написать проброс исключения наружу с помощью команды **`throw`**, исключение будет подавлено и программа будет вести себя так, как будто ошибки и не было.
    - В блоке **`finally`** помещаются команды, которые нужно запустить независимо от того, произошло исключение или нет после завершения выполнения блоков `try` или `catch` (если блок `catch` присутствует). Блок finally нужен, чтобы выполнить завершающий код по освобождению ресурсов выделенных внутри блока `try` (например, если в try открыли файл на запись, закрывать его надо в блоке `finally` – независимо от того – “упало” наше приложение или нет, ресурс файл надо “отпустить”, иначе он останется заблокированным до перезагрузки ОС).

Пример

- Рассмотреть как можно использовать обработку исключений для улучшения работы программы на примере простого вычислителя деления двух целых чисел.
- Указать, что пустой `catch` — это очень плохой и даже **опасный(!)** стиль использования обработки исключений, так как можно скрыть проблемы, о которых не подумал заранее.
- Лучше всего использовать точечные типы исключений в `catch` блоке.
- Рассказать про генерацию исходного исключения в блоке **catch** при помощи **throw** без параметров.

Самостоятельная работа

Задача

В предыдущей задаче обернуть в блок try...catch код, который запрашивает строку у пользователя и пытается представить его в виде числа с помощью функции Parse().

В блоке catch выводить на экран сообщение “Введенная строка не распознаётся как число указанного типа!” и

- Сначала вариант 1: Корректно завершать работу приложения.
- Затем вариант 2: Аварийно завершать работу приложения (пробросив наружу изначальное исключение с помощью команды throw).

Решение

Код **L05_SW_C05_exception_demo**:

```
static void Main(string[] args)
{
	Console.Write("Введите длительность договора аренды в годах: ");
	string inputString = Console.ReadLine();

	try
	{
		int numberOfYears = int.Parse(inputString);
	}
	catch
	{
		Console.WriteLine(
			"Введенная строка не распознаётся как число указанного типа!");
		throw;
	}

	...
}
```

Домашнее задание

- Написать консольное приложение, которое спросит у пользователя тип фигуры (1 - круг, 2 - равносторонний треугольник, 3 - прямоугольник), затем спросит параметры фигуры:
    - для круга - диаметр
    - для треугольника - длину стороны
    - для прямоугольника - ширину и высоту
- В качестве результата программа должна вывести площадь поверхности и длину периметра соответствующей фигуры.
- Тип фигур должен быть объявлен в виде перечисления.
- Необходимо обработать все предсказуемые исключения.
- Пример работы программы (при корректном вводе):

```
> Введите тип фигуры (1 круг, 2 равносторонний треугольник, 3
прямоугольник):
> 3 /это ввод пользователя, соответствующий выбору прямоугольника/
> Введите длину прямоугольника:
> 12.1 /ввод пользователем ширины/
> Введите высоту прямоугольника:
> 9.4 /ввод пользователя высоты/
> Площадь поверхности: 113.74
> Длина периметра: 43
```

- Пример работы программы (при неверном вводе):

```
> Введите тип фигуры (1 круг, 2 равносторонний треугольник, 3
прямоугольник):
> 3 /это ввод пользователя, соответствующий выбору прямоугольника/
> Введите длину прямоугольника:
> Abcd /ввод пользователем ширины/
> Ошибка! Введено нечисловое значение!
```

- **ОБРАТИТЬ ВНИМАНИЕ**, что для формул потребуется
    - вычисление квадратного корня с помощью функции `Math.Sqrt()`
    - округление значений для вывода с помощью функции `Math.Round()`