Темы урока

План урока

Этот урок получается **очень(!)** насыщенным (~30!!! слайдов вместо обычных ~15).
Однако, он во многих местах пересекается с ранее пройденными темами, так что какие-то слайды должны просто пролистываться (при отсутствии вопросов, разумеется), как бы подводя итог первым четырём урокам.

- Первая половина урока
  - Домашка
    - Быстренько слушаем вопросы по домашней работе третьего урока.
    - Показываем решение со своего экрана *(я делал до включения записи, чтобы это было не “переписать” потом (код L03_HW2_solution).*
  - Новый материал
  - Далее всё по плану нужно постараться успеть подобраться вплотную к перечислениям и сделать перерыв.
- Вторая половина урока
  - Рассказываем про перечисления.
  - Потом резко переключаемся на побитовые операторы
  - Затем рассказываем про битовые флаги на базе двух предыдущих тем
  - Даём задачу для самостоятельного решения “про любимые цвета”, кто успеет доделать — молодцы, кто не успеет — забирает её на дом.

Арифметические операции

По коду **L04_C01_arithmetic_operators**:

+	Сложение
−	Вычитание
*	Умножение
/	Деление
%	Остаток (от деления)

Самостоятельная работа

- Вывести на экран результат всех пяти действий с числами 100 и 17
- Вывести на экран результат всех пяти действий с числами 48.13 и 2.5

Унарные операции (с одним операндом)

- Инкремент (автоматический, отложенный) по коду **L04_C02_increment**
- Декремент (автоматический, отложенный) по коду **L04_C03_decrement**
- Отрицание ! по коду **L04_C04_logical_negation**

Операторы отношения (сравнения)

- == (по коду **L04_C05_equals_not_equals**)
- != (по коду тот же проект ⮭)
- > (по коду **L04_C06_more_less**)
- < (по коду тот же проект ⮭)
- >= (по коду тот же проект ⮭)
- <= (по коду тот же проект ⮭)

Самостоятельная работа

Определить 2 переменные: a равное 18; b равное отложенному инкременту от переменной a. Вывести на экран результат сравнения по всем 6 пунктам выше.

Приведение (преобразование) числовых типов

Неявное

По презентации и по коду **L04_C07_type_inplicit_casting**

Явное

По презентации и по коду **L04_C08_type_explicit_casting**

Округление числовых данных

Convert.ToInt32

Одним из способов получения целого числа из дробного можно с помощью `Convert.ToInt32`.

При этом важно учитывать такую особенность округления: *когда число находится посредине между двумя другими числами, оно округляется до ближайшего четного числа.*

Такой способ округления является стандартным способом округления, использующимся в финансовых и статистических операциях. Он соответствует стандарту IEEE-754, раздел 4. При использовании нескольких операций округления, он уменьшает погрешность округления, которое может возникнуть из-за постоянного округления в одном и том же направлении. При больших объёмах, величина накопившихся погрешностей может быть значительной.

По коду: **L04_C09_convert_to_int**.

Math.Round

`Math.Round` округляет значение до ближайшего целого или указанного количества десятичных знаков по указанному стандарту. Т.е. мы можем задавать явно способ округления — “школьный” или “банковский”.

Другие способы получить целое из дробного

`Math.Floor` просто отсекает дробную часть.

`Math.Ceiling` округляет до ближайшего большего целого.

По коду: **L04_C10_ceiling_floor**.

Математика в Math и MathF

Можно также показать что ещё интересного есть в классе Math

- `DivRem`: Выполняет операцию деления с остатком и возвращает как результат деления, так и остаток
- `Abs`: Возвращает абсолютное значение для аргумента
- `Sign`: Возвращает число 1, если аргумент положительный, и -1, если отрицательный. Если он равен 0, то возвращает 0
- `Sqrt`: Возвращает квадратный корень аргумента
- `Cbrt`: Возвращает кубический корень аргумента
- `Min`: Возвращает минимальный из двух аргументов
- `Max`: Возвращает максимальный из двух аргументов

Рассказать, что класс `System.Math` предназначен для работы с аргументами типа `double` (`System.Double`).

Для математики с числами типа `float` (`System.Single`) существует отдельный класс: `System.MathF`.

Полезные заметки

- “Учимся округлять” https://aakinshin.net/ru/posts/cheatsheet-rounding.
- Англ. Одинаковы ли Math.IEEERemainder(x,y) и x%y? https://stackoverflow.com/questions/1971645/is-math-ieeeremainderx-y-equivalent-to-xy

Самостоятельная работа

Дана правильная пирамида:
Запросить параметры a (сторона), h (высота), n (количество сторон)
Рассчитать $S_{полн}$, $S_{бок}$ и V

Для проверки на слайде приведены значения при заданных a, h и n.
Решение: **L04_SW_C01_regular_pyramid**.

Приведение любого значения к строке: ToString

Чтобы получить строку от объекта можно воспользоваться методом `ToString()`.
Иногда он вызывается автоматически, например, при вызове `Console.Write()` и `Console.WriteLine()`

Стоит сказать, что этот метод есть у самого базового типа `object`, поэтому его можно вызвать от любого объекта, однако результат для разных типов объектов окажется разным (это возможно благодаря тому, что в каждом типе он может быть переопределён, однако, об этом мы будем говорить тогда при освещении темы наследования в ООП).

Для примера можно показать следующий код:

```
Console.WriteLine(9);
Console.WriteLine(DateTime.Now);
Console.WriteLine(new ConsoleKeyInfo());
```

У всех будет неявно вызываться метод `ToString()`, в случае числа и даты/времени мы увидим соответствующее число и дату/время в формате “по умолчанию”, а вот в третьем случае мы увидим просто имя класса. Это дефолтное поведение метода `ToString()` определённое базовом классе `object`.

Приведение строки к конкретному типу: Parse/TryParse

Методы `Parse` и `TryParse` часто используется, когда входные данные поступают через консоль — `Console.ReadLine()`, читаются из файла или приходят от веб-сервиса:

- Метод `Parse`
  Вызовет ошибку (можно приучать студентов аккуратно к слову “исключение”) в случае, если не удалось сконвертировать строчку к указанному типу.
- Метод `TryParse`
  Позволяет безопасно “попробовать” сконвертировать строчку к указанному типу. В случае неудачи исключения (ошибки) не будет. Метод вернёт `false`.

Самостоятельная работа

- Запросить у пользователя 2 числа
- Вывести `Math.Floor` и `Math.Ceiling` для их:
  - суммы
  - разницы
  - результата умножения
  - и результата деления

Перечисления (enum)

- Тип перечисления предоставляет способ определения набора именованных целочисленных констант, который можно назначить переменной.
- По умолчанию базовый тип - int, однако он может быть переопределён.
- Показать, как переопределять числовые значения элементов перечисления.
- Показать, как приводить к типу перечисления число и наоборот.

По коду **L04_C13_enum**.

Самостоятельная работа

- Создать перечисление для времён года, где сезоны будут иметь значения:
  - Winter: 3
  - Spring: 6
  - Summer: 9
  - Autumn: 12

Побитовые операторы

Такие операторы проводят операции непосредственно на битах числа:

| ИЛИ (OR)

& И (AND)

~ Инверсия / отрицание (NOT)

^ Исключающее ИЛИ (XOR)

- принимает значение “истина”, если всего один из аргументов имеет значение “истина”

<< Сдвиг влево (left-shift)

- Сдвигает биты влево на определенное количество разрядов
- Биты, расположенные слева, удаляются, справа появляются нули

>> Сдвиг вправо (right-shift)

- Сдвигает биты вправо на определенное количество разрядов
- Биты, расположенные справа, удаляются, слева появляются нули

Пример побитовых операций

Показать пример в проекте **L04_C14_bin_operators**.

Битовые флаги

Тип перечисления можно использовать для определения битовых флагов, благодаря чему экземпляр типа перечисления может хранить любую комбинацию значений, определенных в списке перечислителя.
Для этого нужно добавлять перед определением перечисления атрибут [Flags]

```
[Flags]
enum Days: byte
{
      None = 0x0,
      Sunday = 0x1,
      Monday = 0x1 << 1,
      Tuesday = 0x1 << 2,
      Wednesday = 0x1 << 3,
      Thursday = 0x1 << 4,
      Friday = 0x1 << 5,
      Saturday = 0x1 << 6
}

Days weekendDays = Days.Saturday | Days.Sunday;
```

Битовые операторы

С флагами работают используя битовые операторы

- AND & И
- OR | ИЛИ
- XOR ^ Исключающее ИЛИ

XOR — операция, которая принимает значение “истина” только если всего один из аргументов имеет значение “истина”. Полезная статья про XOR: https://habr.com/ru/post/183462.

Итого, что нужно запомнить для практики

Установить / добавить бит можно через OR

```
nonWorkingDays = nonWorkingDays | Days.Friday;
```

Удалить бит можно через XOR

```
nonWorkingDays = nonWorkingDays ^ Days.Sunday;
```

Проверить, установлен ли бит можно AND

```
isThursdayWorking = (nonWorkingDays & Days.Thursday) != Days.Thursday;
```

Пример работы перечислением типа “битовая маска”

Показать пример в проекте **L04_C15_enum_flags**.

Самостоятельная работа

Написать программу для добавления цветов заданной палитры в “избранное”.
Список допустимых цветов в палитре:

- Black
- Blue
- Cyan
- Grey
- Green
- Magenta
- Red
- White
- Yellow

Программа выводит список цветов с их порядковыми номерами и просит пользователя в цикле выбрать 4 цвета для добавления их в палитру “Избранное”.

Выбор производится путём введения порядковых номеров этих цветов.

После завершения ввода программа выводит список любимых цветов, а также отдельно список нелюбимых цветов.

Решение: **L04_SW_C02_favorite_colors**.

Домашнее задание

- Написать консольное приложение, которое будет спрашивать, “Какой объем сока (в литрах) требуется упаковать?”.
- Затем оно будет рассчитывать и выдавать в качестве ответа наименьшее необходимое количество контейнеров каждого типа.
- В нашей модели будет 3 типа контейнеров: 1 литр, 5 литров, 20 литров.
- Типы контейнеров должны быть определены в перечислении (представленным битовыми флагами).
- Кроме количества контейнеров необходимо посчитать значение переменной типа Int32, в битах которой будет лежать признак наличия контейнера того или иного этого типа (0001 - 1л, 0010 - 5л, 0100 - 20л) .
- При выводе, если бит, отвечающий за наличие хотя бы одного контейнера данного типа, равен 0, строку с данными по этому контейнеру не выводить.

Пример работы программы

```
> Какой объем сока (в литрах) требуется упаковать?
> 76.4 /это ввод пользователя/
> Вам потребуются следующие контейнеры:
> 20 л: 3 шт.
>  5 л: 3 шт.
>  1 л: 2 шт.
```

Пример работы программы (где количество 5-ти литровых контейнеров равно 0)

```
> Какой объем сока (в литрах) требуется упаковать?
> 61.4 /это ввод пользователя/
> Вам потребуются следующие контейнеры:
> 20 л: 3 шт.
>  1 л: 2 шт.
```