Операторы условия, цикла, функции

Функция

- Повторно используемая часть кода
- Функция имеет имя
- Функция имеет ни одного, один или несколько аргументов – формальных параметров
- Функция (ее прототип) должен быть заранее создан
- В прототипе используются формальные параметры функции
- Функция (обычно) вызывается по ее имени
- При вызове в скобках указываются «настоящие» аргументы – фактические параметры

Функция

- Фактические параметры рассчитываются, их значения присваиваются формальным параметрам
- Далее в функции используются эти фактические значения
- Внутри функции могут быть свои – локальные переменные
- Они «перекрывают» внешние переменные
- Функции могут (а обычно и возвращают) давать результат
- За выдачу результата отвечает ключевое слов return

Пример

- Я хочу, чтобы у меня была функция, печатающая строки разными цветами
- Таким образом, мне нужна функция PrintColor (например)
- Представим себе ее логику:
- Надо задать цвет символов консоли
- Надо напечатать строку

Пример

- Но, что если я хочу ПОТОМ напечатать что-то НЕ цветное
- Хорошо бы запоминать старый цвет
- Заменить его на желаемый
- Напечатать нужное
- Восстановить старый цвет

Пример

- Итак, есть имя функции – PrintColor
- У нее два аргумента:
  - Строка для печати
  - Цвет строки
- Функция:
  1. Запоминает старый цвет символов
  2. Задает новый
  3. Печатает строку
  4. Восстанавливает старый цвет

Текст функции

```
void PrintColor(string value, ConsoleColor color)
        {
            ConsoleColor old = Console.ForegroundColor;
            Console.ForegroundColor = color;
            Console.WriteLine(value);
            Console.ForegroundColor = old;
        }
```

Пояснения

- Функции не нужно возвращать результат (поэтому она возвращает условное «пустое» - void значение)
- Старый цвет сохраняется в переменной old

Что улучшить

- Возвращать цвет печати на момент вызова функции (вернем старый цвет – old)
- Предположим, я считаю нужным в основном печатать цветные строки зеленым
- Тогда мне следует задать значение аргумента цвета «по умолчанию»

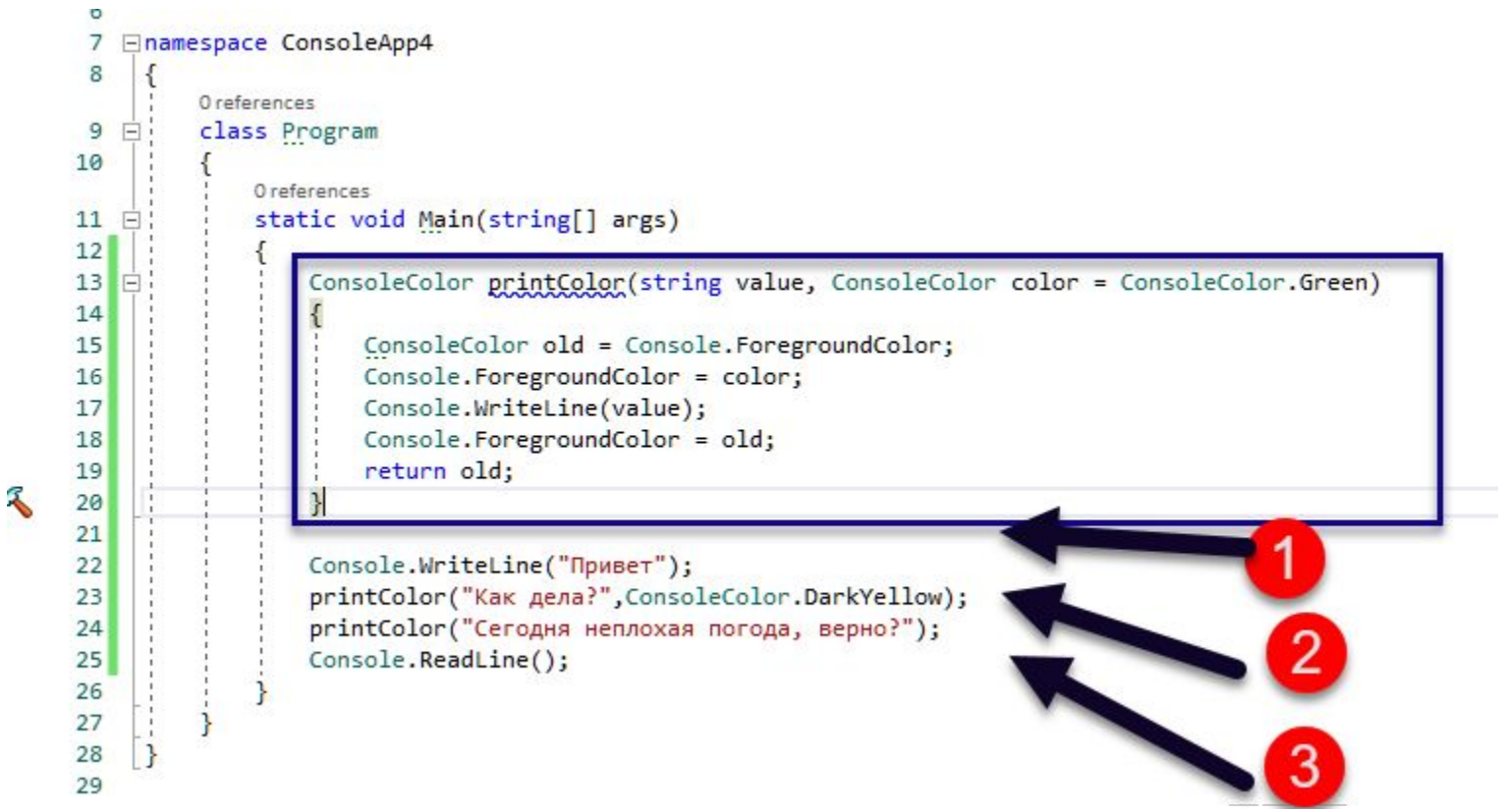
Улучшенная версия

```
ConsoleColor printColor(string value, ConsoleColor color = ConsoleColor.Green)
        {
            ConsoleColor old = Console.ForegroundColor;
            Console.ForegroundColor = color;
            Console.WriteLine(value);
            Console.ForegroundColor = old;
            return old;
        }
```

Где помещать функции?

- В C# функций в чистом виде не так много.
- Как правило, у функции есть хозяин в виде объекта.
- Тогда функция называется методом класса.
  - Иногда даже специально создают класс для сбора функций в виде методов
- Можно создавать и локальные функции

Прототип функции и пример использования



Примеры

1. Вызов “старой” функции печати
2. Печатать с новым цветом
3. Печатать с цветом по умолчанию.

Где размещать функции?

- Локально, по мере надобности (см. пример выше)
- Как метод класса (см. ниже)

```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {

        Console.WriteLine("Привет");
        printColor("Как дела?",ConsoleColor.DarkYellow);
        printColor("Сегодня неплохая погода, верно?");
        Console.ReadLine();
    }

    2 references
    static ConsoleColor printColor(string value, ConsoleColor color = ConsoleColor.Green)
    {
        ConsoleColor old = Console.ForegroundColor;
        Console.ForegroundColor = color;
        Console.WriteLine(value);
        Console.ForegroundColor = old;
        return old;
    }

}
```

Тонкость

- Метод main статический, поэтому методы также должны быть статическими (static)

Лямбда выражения

- Создают локальную функцию или цепочку функций, которые даже не имеют собственного имени
- Такие функции нужны, чтобы локально сделать быструю (и обычно простую) операцию над данными
- Лямбда-выражение — это анонимная функция , с помощью которой можно создавать типы делегатов или деревьев выражений . С помощью лямбда-выражений можно писать локальные функции, которые можно передавать в качестве аргументов или возвращать в качестве значений из вызовов функций. Лямбда-выражения особенно полезны при написании выражений запросов LINQ.
- Чтобы создать лямбда-выражение, необходимо указать входные параметры (если они есть) с левой стороны лямбда-оператора =>, и поместить блок выражений или операторов с другой стороны. Например, лямбда-выражение x => x * x задает параметр с именем x и возвращает значение x

Печать квадратов и кубов массива

- double []x= { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,11,12,13,14,15,16,17,18,19,20 };
-
- foreach (double d in x)
- {
- StringBuilder stringBuilder = new StringBuilder();
- PrintColor(stringBuilder.AppendFormat("x = {0} x^2= {1} x^3= {2}", d, d * d,d * d * d).ToString());
- }

- Console.ReadLine();

Пояснения

- x – массив десятичных дробей
- foreach (double d in x) – берет из массива x поочередно его компоненты
- StringBuilder stringBuilder = new StringBuilder(); - создает строку, которая может менять содержание
- PrintColor(stringBuilder.AppendFormat("x = {0} x^2= {1} x^3= {2}", d, d * d,d * d * d).ToString()); - печатает в цвете

x = {0}	x^2= {1}	x^3= {2}
d	d*d	d*d*d

Выбор данных

- Чтобы напечатать лишь часть данных, надо создать для них запрос с подходящим условием
- Программа просмотрит данные и оставит только те, которые соответствуют запросу

Запрос

- IEnumerable<double> query =
- x.Where(number => number*number>10*number);
- Строка создает запрос данных типа double
- Where – условие запроса
- Условие должно давать логический (bool) критерий

Запрос

- • Таким образом – условие это локальная функция, которая дает или true или false
- Как задать такую функцию?
  - Использовать лямбда-выражение (оператор =>)
- На входе – переменная number
- На выходе – результат проверки $number^2 > 10\ number$
- После выполнения query содержит только отфильтрованные данные

Вывод фильтрованных данных

- foreach (double d in query)
- {
- StringBuilder stringBuilder = new StringBuilder();
- PrintColor(stringBuilder.AppendFormat("x = {0} x^2= {1} x^3= {2}", d, d * d, d * d * d).ToString());
- }

- Console.ReadLine();

Репозиторий GitHub

- https://github.com/Alex-Samarkin/ConsoleApp4/