Практическая работа №1.
Знакомство с базовыми типами и конструкциями в C#

Задание: в консольном проекте реализуйте вывод информации по типам.

Начальное меню может выглядеть следующим образом:

```
Информация по типам:

    1 – Общая информация по типам
    2 – Выбрать из списка
    3 – Ввести имя типа
    4 – Параметры консоли
    0 - Выход из программы
```

Работа с консолью в C# реализуется с помощью статических методов класса Console. Основные операции с консолью:

- вывод строки на консоль:Write и WriteLine;
- ожидание ввода пользователя: ReadKey и ReadLine;
- очистка консоли: Clear.
- изменение цвета фона и текста: BackgroundColor, ForegroundColor.

Вывод строки с форматированием, заданным в коде программы без специальных символов, осуществляется с помощью префикса “@”:

```
Console.WriteLine(
@”Выберете необходимое действие:
     1 – Задача №1
     2 – Задача №2
     3 – Выход”);
```

Чтение одного символа с консоли реализуется с помощью метода `Console.ReadKey()`. Метод возвращает объект типа `ConsoleKeyInfo`, свойство `KeyChar` которого возвращает введенный символ.

Для выбора действия в зависимости от нажатой кнопки используйте конструкцию `switch`. В качестве параметра могут быть и символы, и строки.

```
while(true)
{
    switch(char.ToLower(Console.ReadKey(true).KeyChar))
    {
        case ‘h’ : Help(); break;
        case ‘a’ : ShowTypeInfo(); break;
```

```
            case 'e' : return;
            default: break;
        }
}
```

Для каждого пункта меню необходимо создать отдельный метод. Если методы определяются в том же классе, что и метод Main, то они должны быть помечены как статические:

```
class Program {
    public static void ChangeConsoleView() { .. }
    public static void ShowAllTypeInfo() { .. }
    public static Type SelectType() { .. }
    public static void ShowInfo(Type t) { .. }
    public static void Main() { .. }
}
```

Методы одного и того же класса могут быть расположены в разных файлах. В этом случае объявление класса во всех файлах должно включать модификатор **partial** (частично-определенный класс).

Для работы с типами, которые определены в нашей сборке и подключенных сборках необходимо использовать элементы пространства имен `System.Reflection` (технология «Отражения»). Основной объект для работы с информацией о типах – Type.

Выбор типа из меню может быть следующим:

```
Выберете тип из списка:

    1 – uint
    2 – int
    3 – long
    4 – float
    5 – double
    6 – char
    7 - string
    8 – MyClass
    9 – MyStruct
    0 – Выход в главное меню
```

В зависимости от выбора пользователя инициализируем экземпляр класса `Type`:

```
Type t = typeof(int)
```

Вывод информации о типе должен быть следующим:

```
Информация по типу: System.Int32

    Значимый тип:             +
    Пространство имен:        System
    Сборка:                   mscorlib
    Общее число элементов:    19
    Число методов:            17
    Число свойств:            0
    Число полей:              2
    Список полей:             MaxValue, MinValue
    Список свойств:           -
Нажмите ‘M’ для вывода дополнительной информации по
методам:
Нажмите ‘0’ для выхода в главное меню
```

Объект Type содержит множество булевых свойств, характеризующих тип: `IsValueType, IsAbstract, IsPointer, IsByRef, IsClass` и т.д.

Короткое имя сборки (без указания версии, региональных настроек) можно получить следующим образом:

```
string assemblyName = t.Assembly.GetName().Name;
```

Информацию об отдельных элементах типа (свойства, поля, методы) можно получить с помощью методов объекта Type: `GetMethods, GetFields, GetProperties, GetMembers`. Каждый из этих методов возвращает массив элементов соответствующего типа.

Например, чтобы узнать общее число полей используем свойство массива:

```
int nFields = t.GetFields().Length;
```

Для вывода имени отдельного поля используем:

```
string sField = t.GetFields()[i].Name;
```

Сформировать список полей можно с помощью метода String.Join:

```
string[] fieldNames = ..
string sFieldNames = String.Join(“, ”, fieldNames);
```

При нажатии на клавишу «M» предполагается вывод информации по методам, сгруппированных по именам:

```
Методы типа System.Int32
```

Название	Число перегрузок	Число параметров
ToString	4	0..2
Parse	4	1..3
CompareTo	2	1

При выборе пункта *«Общая информация по типам»* в главном меню отображается следующий экран:

```
Общая информация по типам

Подключенные сборки:        17
Всего типов по всем подключенным сборкам:        26103
Ссылочные типы:                                  20601
Значимые типы:                                    4377
Типы-интерфейсы:
Тип с максимальным числом методов:
Самое длинное название метода:
Метод с наибольшим числом аргументов:

Нажмите любую клавишу, чтобы вернуться в главное меню
```

Для работы с типами используем следующие методы рефлексии:
Получаем ссылку на исполняемую сборку (наш проект):

```
Assembly myAsm = Assembly.GetExecutingAssembly();
```

Получаем множество типов, определенных в нашей сборке:

```
Type[] thisAssemblyTypes = myAsm.GetTypes();
```

Но в нашей сборке типов не так много. Поэтому необходимо сформировать список типов, определенных во всех подключенных сборках:

```
Assembly[] refAssemblies =
    AppDomain.CurrentDomain.GetAssemblies();

List<Type> types = new List<Type>();
foreach(Assembly asm in refAssemblies)
    types.AddRange(asm.GetTypes());
```

Далее, циклически обрабатывая множество типов, получаем итоговую информацию:

```
foreach(var t in types) {
    ..
    if(t.IsClass)
        nRefTypes++;
```

```
    else if(t.IsValueType)
        nValueTypes++;
    ..
}
```