Практическая работа №3

Типы данных. Консольный ввод-вывод данных

Цель работы: Усвоить систему типов данных C#, которая используется для создания переменных. Познакомиться с внутренним представлением данных в памяти компьютера.

Норма времени: 2 часа.

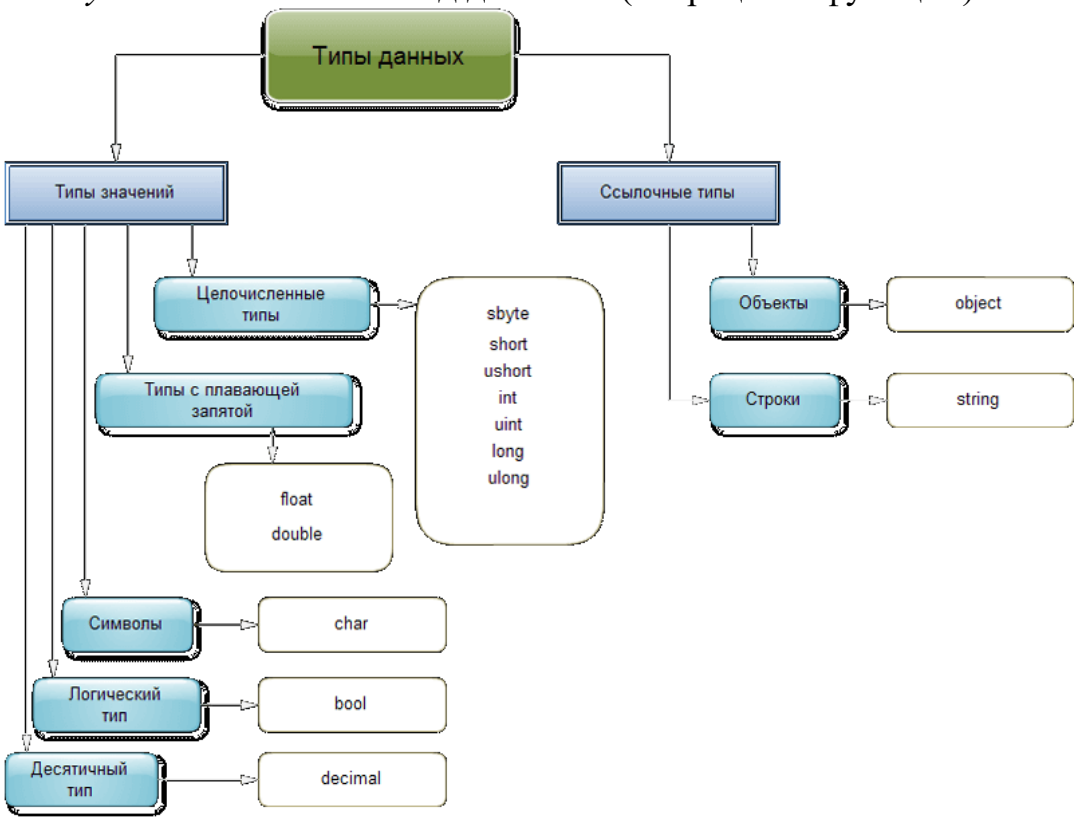
Оборудование: персональный компьютер, система программирования Microsoft Visual Studio 2019.

Ход работы

Данные, с которыми работает программа, хранятся в оперативной памяти. Естественно, что компилятору необходимо точно знать, сколько места они занимают, как именно закодированы и какие действия с ними можно выполнять. Все это задается при описании данных с помощью типа.

Тип данных однозначно определяет:

- *внутреннее представление* данных, а, следовательно, и *множество их возможных значений*;
- *допустимые действия* над данными (операции и функции).



Встроенные типы

Встроенные типы не требуют предварительного определения. Для каждого типа существует ключевое слово, которое используется при описании переменных, констант и т. д. Встроенные типы C# приведены в таблице 2. Они однозначно соответствуют стандартным классам библиотеки .NET, определенным в пространстве имен **System**.

Таблица 2. Встроенные типы C#

Название	Ключевое слово	Тип .NET	Диапазон значений	Описание	Размер, битов
Логический тип	bool	Boolean	true, false		
Целые типы	sbyte	SByte	От –128 до 127	Со знаком	8
	byte	Byte	От 0 до 255	Без знака	8
	short	Int16	От –32768 до 32767	Со знаком	16
	ushort	UInt16	От 0 до 65535	Без знака	16
	int	Int32	От –2 109 до 2 109	Со знаком	32
	uint	UInt32	От 0 до 4 109	Без знака	32
	long	Int64	От –9 1018 до 9 1018	Со знаком	64
	ulong	UInt64	От 0 до 18 1018	Без знака	64
Символьный тип	char	Char	От U+0000 до U+ffff	Unicode символ	16
Вещественные	float	Single	От 1.5 10-45 до 3.4 1038	7 цифр	32
	double	Double	От 5.0 10-324 до 1.7 10308	15–16 цифр	64
Финансовый тип	decimal	Decimal	От 1.0 10-28 до 7.9 1028	28–29 цифр	128
Строковый тип	string	String	Длина ограничена объемом доступной памяти	Строка из Unicode символов	
Тип object	object	Object	Можно хранить все, что угодно	Всеобщий предок	

Целые типы, а также символьный, вещественные и финансовый типы можно объединить под названием *арифметических типов*.

Пример: Определим несколько переменных разных типов и выведем их значения на консоль:

```
using System;

namespace HelloApp
{   class Program
    {       static void Main(string[] args)
        {
            string name = "Tom";
            int age = 33;
            bool isEmployed = false;
            double weight = 78.65;

            Console.WriteLine($"Имя: {name}");
            Console.WriteLine($"Возраст: {age}");
            Console.WriteLine($"Вес: {weight}");
            Console.WriteLine($"Работает: {isEmployed}");
        }
    }
}
```

Типы литералов

Литералы (константы) тоже имеют тип. Если значение целого литерала находится внутри диапазона допустимых значений типа int, литерал рассматривается как int, иначе он относится к наименьшему из типов uint, long или

ulong, в диапазон значений которого он входит. Вещественные литералы по умолчанию относятся к типу double.

Консольный вывод

Для вывода информации на консоль мы уже использовали встроенный метод **Console.WriteLine**. То есть, если мы хотим вывести некоторую информацию на консоль, то нам надо передать ее в метод Console.WriteLine:

```
using System;

namespace HelloApp
{
    class Program
    {
        static void Main(string[] args)
        {
            string hello = "Привет мир";
            Console.WriteLine(hello);
            Console.WriteLine("Добро пожаловать в C#!");
            Console.WriteLine("Пока мир...");
            Console.WriteLine(24.5);

            Console.ReadKey();
        }
    }
}
```

Консольный вывод:

```
Привет мир!
Добро пожаловать в C#!
Пока мир...
24,5
```

Нередко возникает необходимость вывести на консоль в одной строке значения сразу нескольких переменных. В этом случае мы можем использовать прием, который называется интерполяцией:

```
using System;

namespace HelloApp
{
    class Program
    {
        static void Main(string[] args)
        {
            string name = "Tom";
            int age = 34;
            double height = 1.7;
            Console.WriteLine($"Имя: {name}  Возраст: {age}  Рост: {height}м");

            Console.ReadKey();
        }
    }
}
```

Для встраивания отдельных значений в выводимую на консоль строку используются фигурные скобки, в которые заключается встраиваемое значение. Это можем значение переменной ({name}) или более сложное выражение (например, операция сложения {4 + 7}). А перед всей строкой ставится знак доллара $.

При выводе на консоль вместо помещенных в фигурные скобки выражений будут выводиться их значения:

```
Имя: Tom  Возраст: 34  Рост: 1,7м
```

Есть другой способ вывода на консоль сразу нескольких значений:

```
using System;

namespace HelloApp
{
    class Program
    {
        static void Main(string[] args)
        {
            string name = "Tom";
            int age = 34;
            double height = 1.7;
            Console.WriteLine("Имя: {0}  Возраст: {2}  Рост: {1}м", name, height, age);
             Console.ReadKey();
        }
    }
}
```

Этот способ подразумевает, что первый параметр в методе Console.WriteLine представляет выводимую строку ("Имя: {0} Возраст: {2} Рост: {1}м"). Все последующие параметры представляют значения, которые могут быть встроенны в эту строку (name, height, age). При этом важен порядок подобных параметров. Например, в данном случае вначале идет name, потом height и потом age. Поэтому у name будет представлять параметр с номером 0 (нумерация начинается с нуля), height имеет номер 1, а age - номер 2. Поэтому в строке "Имя: {0} Возраст: {2} Рост: {1}м" на место плейсхолдеров {0}, {2}, {1} будут вставляться значения соответствующих параметров.

Кроме Console.WriteLine() можно также использовать метод **Console.Write()**, он работает точно так же за тем исключением, что не осуществляет переход на следующую строку.

Консольный ввод

Кроме вывода информации на консоль мы можем получать информацию с консоли. Для этого предназначен метод **Console.ReadLine()**. Он позволяет получить введенную строку.

```
using System;

namespace HelloApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Введите свое имя: ");
            string name = Console.ReadLine();
            Console.WriteLine($"Привет {name}");

            Console.ReadKey();
        }
    }
}
```

В данном случае все, что вводит пользователь, с помощью метода Console.ReadLine передается в переменную name.

Пример работы программы:

```
Введите свое имя: Том
Привет Том
```

Таким образом мы можем вводить информацию через консоль. Однако минусом этого метода является то, что Console.ReadLine считывает информацию

именно в виде строки. Поэтому мы можем по умолчанию присвоить ее только переменной типа string. Как нам быть, если, допустим, мы хотим ввести возраст в переменную типа int или другую информацию в переменные типа double или decimal? По умолчанию платформа .NET предоставляет ряд методов, которые позволяют преобразовать различные значения к типам int, double и т.д. Некоторые из этих методов:

- **Convert.ToInt32()** (преобразует к типу int)
- **Convert.ToDouble()** (преобразует к типу double)
- **Convert.ToDecimal()** (преобразует к типу decimal)

Пример ввода значений:

```
using System;

namespace HelloApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Введите имя: ");
            string name = Console.ReadLine();
             Console.Write("Введите возраст: ");
            int age = Convert.ToInt32(Console.ReadLine());
             Console.Write("Введите рост: ");
            double height = Convert.ToDouble(Console.ReadLine());
             Console.Write("Введите размер зарплаты: ");
            decimal salary = Convert.ToDecimal(Console.ReadLine());
             Console.WriteLine($"Имя: {name}  Возраст: {age}  Рост: {height}м
                                                        Зарплата: {salary}$");
             Console.ReadKey();
        }
    }
}
```

При вводе важно учитывать текущую операционную систему. В одних культурах разделителем между целой и дробной частью является точка (США, Великобритания...), в других - запятая (Россия, Германия...). Например, если текущая ОС - русскоязычная, значит, надо вводить дробные числа с разделителем запятой. Если локализация англоязычная, значит, разделителем целой и дробной части при вводе будет точка.

Пример работы программы:

```
Введите имя: Том
Введите возраст: 25
Введите рост: 1,75
Введите размер зарплаты: 300,67
Имя: Том  Возраст: 25  Рост: 1,75м  Зарплата: 300,67$
```

Контрольные вопросы:

1. Дать определение алфавита и лексемы.
2. Что такое идентификаторы?
3. Перечислите типы данных, типы литералов.
4. Что такое переменные, литералы?
5. Какие методы ввода, вывода данных вы знаете?