Типы данных

Как и во многих языках программирования, в C# есть своя система типов данных, которая используется для создания переменных. Тип данных определяет внутреннее представление данных, множество значений, которые может принимать объект, а также допустимые действия, которые можно применять над объектом.

В языке C# есть следующие базовые типы данных:

- **bool**: хранит значение `true` или `false` (логические литералы). Представлен системным типом `System.Boolean`

```
bool alive = true;
bool isDead = false;
```

- **byte**: хранит целое число от `0` до `255` и занимает 1 байт. Представлен системным типом `System.Byte`

```
byte bit1 = 1;
byte bit2 = 102;
```

- **sbyte**: хранит целое число от `-128` до `127` и занимает 1 байт. Представлен системным типом `System.SByte`

```
sbyte bit1 = -101;
sbyte bit2 = 102;
```

- **short**: хранит целое число от `-32768` до `32767` и занимает 2 байта. Представлен системным типом `System.Int16`

```
short n1 = 1;
short n2 = 102;
```

- **ushort**: хранит целое число от `0` до `65535` и занимает 2 байта. Представлен системным типом `System.UInt16`

```
ushort n1 = 1;
ushort n2 = 102;
```

- **int**: хранит целое число от `-2147483648` до `2147483647` и занимает 4 байта. Представлен системным типом `System.Int32`. Все целочисленные литералы по умолчанию представляют значения типа int:

```
int a = 10;
int b = 0b101;  // бинарная форма b =5
int c = 0xFF;   // шестнадцатеричная форма c = 255
```

- **uint**: хранит целое число от `0` до `4294967295` и занимает 4 байта. Представлен системным типом `System.UInt32`

```
uint a = 10;
uint b = 0b101;
uint c = 0xFF;
```

- **long**: хранит целое число от `-9 223 372 036 854 775 808` до `9 223 372 036 854 775 807` и занимает 8 байт. Представлен системным типом `System.Int64`

```
long a = -10;
long b = 0b101;
long c = 0xFF;
```

- **ulong**: хранит целое число от `0` до `18 446 744 073 709 551 615` и занимает 8 байт. Представлен системным типом `System.UInt64`

```
ulong a = 10;
ulong b = 0b101;
ulong c = 0xFF;
```

- **float**: хранит число с плавающей точкой от $-3.4*10^{38}$ до $3.4*10^{38}$ и занимает 4 байта. Представлен системным типом `System.Single`
- **double**: хранит число с плавающей точкой от $\pm5.0*10^{-324}$ до $\pm1.7*10^{308}$ и занимает 8 байта. Представлен системным типом `System.Double`
- **decimal**: хранит десятичное дробное число. Если употребляется без десятичной запятой, имеет значение от $\pm1.0*10^{-28}$ до $\pm7.9228*10^{28}$, может хранить 28 знаков после запятой и занимает 16 байт. Представлен системным типом `System.Decimal`
- **char**: хранит одиночный символ в кодировке Unicode и занимает 2 байта. Представлен системным типом `System.Char`. Этому типу соответствуют символьные литералы:

```
char a = 'A';
char b = '\x5A';
char c = '\u0420';
```

- **string**: хранит набор символов Unicode. Представлен системным типом `System.String`. Этому типу соответствуют строковые литералы.

```
string hello = "Hello";
string word = "world";
```

- **object**: может хранить значение любого типа данных и занимает 4 байта на 32-разрядной платформе и 8 байт на 64-разрядной платформе. Представлен системным типом `System.Object`, который является базовым для всех других типов и классов .NET.

```
object a = 22;
object b = 3.14;
object c = "hello code";
```

Например, определим несколько переменных разных типов и выведем их значения на консоль:

```
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}");
```

Для вывода данных на консоль здесь применяется интерполяция: перед строкой ставится знак $ и после этого мы можем вводить в строку в фигурных скобках значения переменных. Консольный вывод программы:

```
Имя: Tom
Возраст: 33
Вес: 78,65
Работает: False
```

Использование суффиксов

При присвоении значений надо иметь в виду следующую тонкость: все вещественные литералы (дробные числа) рассматриваются как значения типа **double**. И чтобы указать, что дробное число представляет тип **float** или тип **decimal**, необходимо к литералу добавлять суффикс: F/f - для float и M/m - для decimal.

```
float a = 3.14F;
float b = 30.6f;

decimal c = 1005.8M;
decimal d = 334.8m;
```

Подобным образом все целочисленные литералы рассматриваются как значения типа **int**. Чтобы явным образом указать, что целочисленный литерал представляет значение типа uint, надо использовать суффикс **U/u**, для типа **long** - суффикс **L/l**, а для типа **ulong** - суффикс **UL/ul**:

```
uint a = 10U;
long b = 20L;
ulong c = 30UL;
```

Использование системных типов

Выше при перечислении всех базовых типов данных для каждого упоминался системный тип. Потому что название встроенного типа по сути представляет собой сокращенное обозначение системного типа. Например, следующие переменные будут эквивалентны по типу:

```
int a = 4;
System.Int32 b = 4;
```

Неявная типизация

Ранее мы явным образом указывали тип переменных, например, `int x;`. И компилятор при запуске уже знал, что x хранит целочисленное значение.

Однако мы можем использовать и модель неявной типизации:

```
var hello = "Hell to World";
var c = 20;
```

Для неявной типизации вместо названия типа данных используется ключевое слово `var`. Затем уже при компиляции компилятор сам выводит тип данных исходя из присвоенного значения. Так как по умолчанию все целочисленные значения рассматриваются как значения типа `int`, то поэтому в итоге переменная с будет иметь тип `int`. Аналогично переменной `hello` присваивается строка, поэтому эта переменная будет иметь тип `string`

Эти переменные подобны обычным, однако они имеют некоторые ограничения.

Во-первых, мы не можем сначала объявить неявно типизируемую переменную, а затем инициализировать:

```
// этот код работает
int a;
a = 20;

// этот код не работает
var c;
c= 20;
```

Во-вторых, мы не можем указать в качестве значения неявно типизируемой переменной `null`:

```
// этот код не работает
var c=null;
```

Так как значение null, то компилятор не сможет вывести тип данных.

Вопросы:

1. Какие из нижеперечисленных НЕ являются встроенными типами языка C#?

- uint
- sbyte
- real
- int128
- object
- float64

2. Какой тип данных языка C# будет представлять следующая переменная?

```
bool enabled = true;
```

3. Какой тип данных языка C# будет представлять следующая переменная?

```
var weight = 84.45f;
```

4. Сколько байт занимает значение типа **uint**?

5. Какие из следующих вариантов представляют корректное определение переменных:

```
A. string person = "Tom";
B. var person = "Tom";
C. var person;
D. string person;
```