Data type in C

Value type variables can be assigned a value directly. They are inherited from the System.ValueType class.

The variables in C # are divided into the following types:

1. Value type (Value type)
2. Reference type (Reference type)
3. Pointer type (Pointer type)

Value type in C #

Value type variables can be assigned a value directly. They are inherited from the **System.ValueType** class.

Direct value type contains data. Some examples are **int, char** , and **float** , respectively keeping integers, letters, and real numbers. When you declare an int, the system allocates memory to store that value.

The following table lists the available value types in C # 2010:

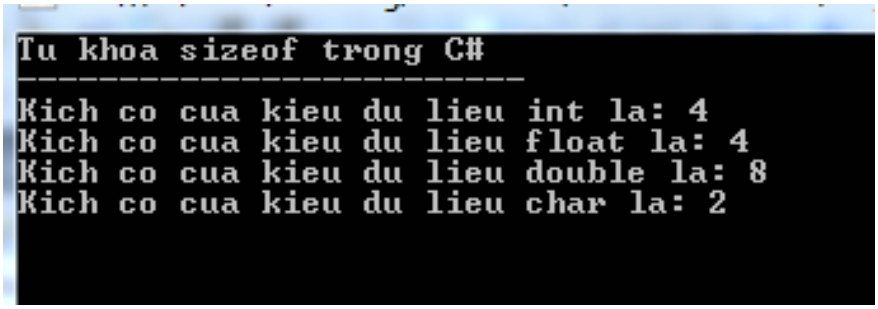
Type Representation Value Range Default value boolean value True or False False byte unsigned integer type (8 bits) 0 to 255 0 char Unicode character type (16 bit) U +0000 to U + ffff " decimal Decimal type (128 bits)) (-7.9 x 10^{28} to 7.9 x 10^{28}) / $10^{0 \text{ to } 28}$ 0.0M double Type double (64 bit) (+/-) 5.0 x 10^{-324} to (+/-) 1.7 x 10^{308} 0.0D float Float type (32 bits) -3.4 x 10^{38} to + 3.4 x 10^{38} 0.0F int Integer type (32 bits) -2,147,483,648 to 2,147,483,647 0 long Signing type integer (64 bits) -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 0L sbyte Type signed integer (8 -128 to 127 bit 0 short Signed integer type (16 bits) -32,768 to 32,767 0 uint unsigned integer (32 bits) 0 to 4,294,967,295 0 ulong unsigned integer (64 bit) 0 to 18,446,744,073,709,551,615 0 ushort unsigned integer (16 bit) 0 to 65,535 0

Sizeof keyword in C #

To get the exact size of a type or variable on a particular platform, you can use the sizeof method. The expression **sizeof** (type) displays the size of the object or type with the byte value. The example below is to get the size of **int** type on any computer:

```
using System ; namespace QTMCsharp { class TestCsharp { static void Main ( stri
```

Compiling and running the above C # program will produce the following results:



Reference type in C #

The reference type contains no actual data stored in a variable, but they contain a reference to the variables.

In other words, they refer to a memory location. Using multiple variables, the reference type can refer to a memory location. If the data in the memory location is changed by one of the variables, the other variable automatically reflects this change in value. Examples of reference types **available** in C # are: **object, dynamic** , and **string** .

Object type in C #

Object type is the basic base class for all data types in C # Common Type System (CTS). Object is an alias for the System.Object class. Object types can be assigned values ??of any other type, value type, reference type, self-defined type (user-defined). However, before assigning values, it needs a type conversion.

When a value type is converted to an object type, it is called **boxing** and vice versa, when an object type is converted to a value type, it is called **unboxing** .

```
object  obj ;  obj  = 100 ; // day la vi du boxing
```

Dynamic type in C #

You can store any type of value in the dynamic data type variable. Checking these variable types takes place at run time.

The syntax for declaring a dynamic type in C # is:

```
dynamic > =  gi a _tr i;
```

For example:

```
dynamic  d  = 20 ;
```

Dynamic types are similar to object types, except that checking for object type variables takes place at compile time, while checking for dynamic type variables takes place at run time.

String type in C #

The **string** type in C # allows you to assign any string value to a variable. The string type is an alias for the **System.String** class. It inherits from object type. Values ??for a string type can be assigned using string constants in two forms: quoted and @quoted.

For example:

```
String  str  = "Hoc C# co day du tai QTM" ;
```

And a @quoted string constant looks like this:

```
@ "QTM Nhom" ;
```

The self-defined types (user-defined) in C # are: Class, Interface, or Delegate. We will discuss these types in later chapters.

Cursor type in C #

Pointer type variables store memory addresses of other types. Pointers in C # are capable of pointers in C or C ++.

The syntax for declaring a pointer type in C # is:

```
type *  identifier ;
```

For example:

```
char *  cptr ; int *  iptr ;
```

Follow tutorialspoint

Old post: Basic C # syntax

Next article: Converting data types in C #