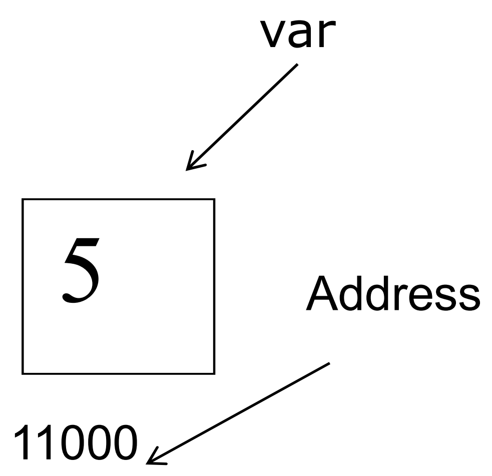
Data Types in C# can be divided in two catagories

- ❑ Value Types
- ❑ Reference Types

Value Types

```
int var;
var=5;
```

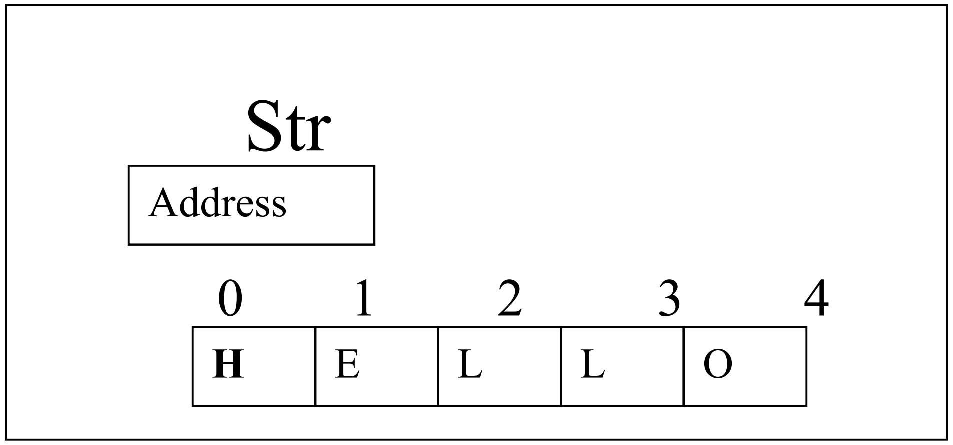
Variable declaration and initialization



Memory Allocation in Value Type

Reference Types

```
string Str="HELLO";
```



Examples of Value Types

- Integer
- Character
- Float
- Structure

Examples of Reference Types

- Classes
- Delegates
- Interfaces

Function Call Types

- ❑ Call By Value
- ❑ Call By Reference

In Call by value:

The calling function passes the values to the called function and those values are copied to the input parameter(s), represented by data types, of the function.

In Call by Reference:

The calling function does not pass the values to the called function. Instead it passes the reference of variables or objects defined in its scope or class scope.

In call by reference the input parameters of the called function hold reference(s) passed by calling function.

In C# Call By Reference is made by using two keywords:

- ❑ **ref** The arguments passed by calling function must be initialized.
- ❑ **out** The out parameter must be assigned to before control leaves the called method or function.

Example

```
public class Book
{
      public static void Main()
      {
            int price=200;
            int P2;
            Book.Display(ref price);
            Book.SetPrice(out P2);
      }
      public static void Display(ref int price)
      {
               Console.WriteLine(price.ToString());
      }
      public static void SetPrice(out int price)
      {
                price=300;
      }
}
```

Static Variables and Static Functions

- Each object has its own set of member variables.
- To retain the value of a variable throughout the program, you can declare the variable as a static variable.
- To manipulate and use the values of static variables, you can define a function as static function.

Static Variables

- The keyword '`static`' means that only one instance of a given variable exists for a class.
- Static variables are used to define constants because their values can be retrieved by invoking the class without creating an instance of it.
- Static variables can be initialized outside the member function or class definition.
- Unlike other member variables, only one copy of the static variable exists in the memory for all the objects of that class.

```
Public class B
{
    public int count1;
    public static int count2;

}
```

Non static data member

Static data member

Static Functions

- Static functions exist even before the object is created.
- Static functions can access only static variables.

<u>Example</u>

```
public class myClass
{
    public static void StatFunction()
    {
          Console.WriteLine("I am static function.");
    }

}
Public class B
{
  public static void Main()
  {
     myClass.StatFunction();
  }
}
```