C# Pointers

# Definition

1. C# pointer is a variable that holds the memory address of another type.
2. C# pointer can be declared to hold the memory address of only value types and arrays.
3. Unlike reference types, pointer types are not tracked by the GC mechanisms and hence they are not allowed (designed) to point to a reference type.
4. Pointers are not allowed to point to a managed struct type also.

# Declaring a Pointer

The general form of declaring a pointer type is as shown below,

**type \* variable\_name;**

where \* is known as the de-reference operator.

For example the following statement

**int \*x ;**

declares a pointer variable x, which can hold the address of an int type. The reference operator (&) can be used to get the memory address of a variable.

# Safe-Unsafe Codes

Code that runs under the control of the GC is called as Safe code. Otherwise unsafe.

As Pointers doesn’t come under the ambit of GC, any code that **involves pointers has to be marked unsafe.**

**unsafe** can be used as a modifier to a

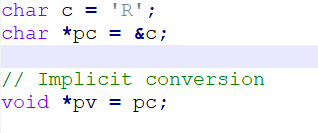
1. Constructor
2. Method
3. Property
4. A group of statements.

# Pointer Conversions

In C# pointer types do not inherit from object and no conversion exists between pointer types and objects. That means boxing and un-boxing are not supported by pointers. But C# supports conversions between the different pointer types and integral types.

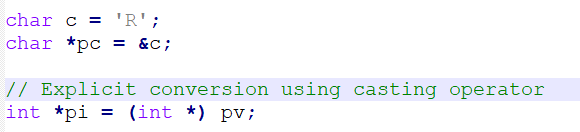
## Implicit Pointer Conversion

1. From any pointer type to void \* type.
2. From null type to any pointer type.



## Explicit Pointer Conversion

1. From any pointer type to any other pointer type.
2. From sbyte, byte, short, ushort, int, uint, long, ulong to any pointer type.
3. From any pointer type to sbyte, byte, short, ushort, int, uint, long, ulong types.



# References

1. Pointers in C#: <https://www.c-sharpcorner.com/article/pointers-in-C-Sharp/>