C# - UNSAFE CODES

C# allows using pointer variables in a function of code block when it is marked by the **unsafe** modifier. The **unsafe code** or the unmanaged code is a code block that uses a **pointer** variable.

> ***Note*** *– To execute the programs mentioned in this chapter at* *codingground**, please set compilation option in Project >> Compile Options >> Compilation Command to*
>
> `mcs *.cs -out:main.exe -unsafe"`

Pointers

A **pointer** is a variable whose value is the address of another variable i.e., the direct address of the memory location. similar to any variable or constant, you must declare a pointer before you can use it to store any variable address.

The general form of a pointer declaration is –

```
type *var-name;
```

Following are valid pointer declarations –

```
int    *ip;    /* pointer to an integer */
double *dp;    /* pointer to a double */
float  *fp;    /* pointer to a float */
char   *ch     /* pointer to a character */
```

The following example illustrates use of pointers in C#, using the unsafe modifier –

```
using System;

namespace UnsafeCodeApplication {
   class Program {
      static unsafe void Main(string[] args) {
         int var = 20;
         int* p = &var;

         Console.WriteLine("Data is: {0} ",  var);
         Console.WriteLine("Address is: {0}",  (int)p);
         Console.ReadKey();
      }
   }
}
```

When the above code wass compiled and executed, it produces the following result –

```
Data is: 20
Address is: 99215364
```

Instead of declaring an entire method as unsafe, you can also declare a part of the code as unsafe. The example in the following section shows this.

Retrieving the Data Value Using a Pointer

You can retrieve the data stored at the located referenced by the pointer variable, using the **ToString** method. The following example demonstrates this –

```
using System;

namespace UnsafeCodeApplication {
   class Program {
      public static void Main() {
         unsafe {
            int var = 20;
            int* p = &var;
            
            Console.WriteLine("Data is: {0} " , var);
            Console.WriteLine("Data is: {0} " , p->ToString());
            Console.WriteLine("Address is: {0} " , (int)p);
         }
         Console.ReadKey();
      }
   }
}
```

When the above code was compiled and executed, it produces the following result –

```
Data is: 20
Data is: 20
Address is: 77128984
```

Passing Pointers as Parameters to Methods

You can pass a pointer variable to a method as parameter. The following example illustrates this –

```
using System;

namespace UnsafeCodeApplication {
   class TestPointer {
      public unsafe void swap(int* p, int *q) {
         int temp = *p;
         *p = *q;
         *q = temp;
      }
      public unsafe static void Main() {
         TestPointer p = new TestPointer();
         int var1 = 10;
         int var2 = 20;
         int* x = &var1;
         int* y = &var2;
         
         Console.WriteLine("Before Swap: var1:{0}, var2: {1}", var1, var2);
         p.swap(x, y);

         Console.WriteLine("After Swap: var1:{0}, var2: {1}", var1, var2);
         Console.ReadKey();
      }
   }
}
```

When the above code is compiled and executed, it produces the following result –

```
Before Swap: var1: 10, var2: 20
After Swap: var1: 20, var2: 10
```

Accessing Array Elements Using a Pointer

In C#, an array name and a pointer to a data type same as the array data, are not the same variable type. For example, int *p and int[] p, are not same type. You can increment the pointer variable p because it is not fixed in memory but an array address is fixed in memory, and you can't increment that.

Therefore, if you need to access an array data using a pointer variable, as we traditionally do in C, or C++ (please check: C Pointers), you need to fix the pointer using the **fixed** keyword.

The following example demonstrates this –

```
using System;

namespace UnsafeCodeApplication {
   class TestPointer {
      public unsafe static void Main() {
         int[]  list = {10, 100, 200};
         fixed(int *ptr = list)

         /* let us have array address in pointer */
         for ( int i = 0; i < 3; i++) {
            Console.WriteLine("Address of list[{0}]={1}",i,(int)(ptr + i));
            Console.WriteLine("Value of list[{0}]={1}", i, *(ptr + i));
         }

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

When the above code was compiled and executed, it produces the following result –

```
Address of list[0] = 31627168
Value of list[0] = 10
Address of list[1] = 31627172
Value of list[1] = 100
Address of list[2] = 31627176
Value of list[2] = 200
```

Compiling Unsafe Code

For compiling unsafe code, you have to specify the **/unsafe** command-line switch with command-line compiler.

For example, to compile a program named prog1.cs containing unsafe code, from command line, give the command –

```
csc /unsafe prog1.cs
```

If you are using Visual Studio IDE then you need to enable use of unsafe code in the project properties.

To do this –

- Open **project properties** by double clicking the properties node in the Solution Explorer.
- Click on the **Build** tab.
- Select the option "**Allow unsafe code**".