Pointers

SESSION 9

Pointers

A special variable that contains the “address” of the memory location of another variable

Declaring a pointer variable in C

```
int *p
```

Introduction to Pointers

When we declare a variable some memory is allocated for it. The memory location can be referenced through the identifier “i”. Thus, we have two properties for any variable : its address and its data value. The address of the variable can be accessed through the referencing operator “&”. “&i” gives the memory location where the data value for “i” is stored.

A pointer variable is one that stores an address. We can declare pointers as follows int* p; .This means that p stores the address of a variable of type int.

Introduction to Pointers

Q: Why is it important to declare the type of the variable that a pointer points to? Aren't all addresses of the same length?

A: It's true that all addresses are of the same length, however when we perform an operation of the type "p++" where "p" is a pointer variable, for this operation to make sense the compiler needs to know the data type of the variable "p" points to. If "p" is a character pointer then "p++" will increment "p" by one byte (typically), if "p" were an integer pointer its value on "p++" would be incremented by 2 bytes (typically).

Introduction to Pointers

Summary of what was learnt so far:

- Pointer is a data type that stores addresses, it is declared as follows:

  int* a;

  char* p; etc.

- The value stored in a pointer p can be accessed through the dereferencing operator "*".
- The address of a memory location of a variable can be accessed through the reference operator "&".
- Pointer arithmetic: only addition and subtraction are allowed.

Pointers and Arrays

The concept of array is very similar to the concept of pointer. The identifier of an array actually a pointer that holds the address of the first element of the array.

Therefore if you have two declarations as follows:

- “int a[10];” “int* p;” then the assignment “p = a;” is perfectly valid
- Also “*(a+4)” and “a[4]” are equivalent as are “*(p+4)” and “p[4]” .
- The only difference between the two is that we can change the value of “p” to any integer variable address whereas “a” will always point to the integer array of length 10 defined.

Character Pointers, Arrays and Strings

What is a String?

- A string is a character array that is '\0' terminated.
- E.g. "Hello"

What is a Character array?

- It is an array of characters, not necessarily '\0' terminated
- E.g. char test[4] = {'a', 'b', 'c', 'd'}; <this char array is not zero terminated>

What is a character pointer?

- It is a pointer to the address of a character variable.
- E.g. char* a; <this pointer is not initialized>

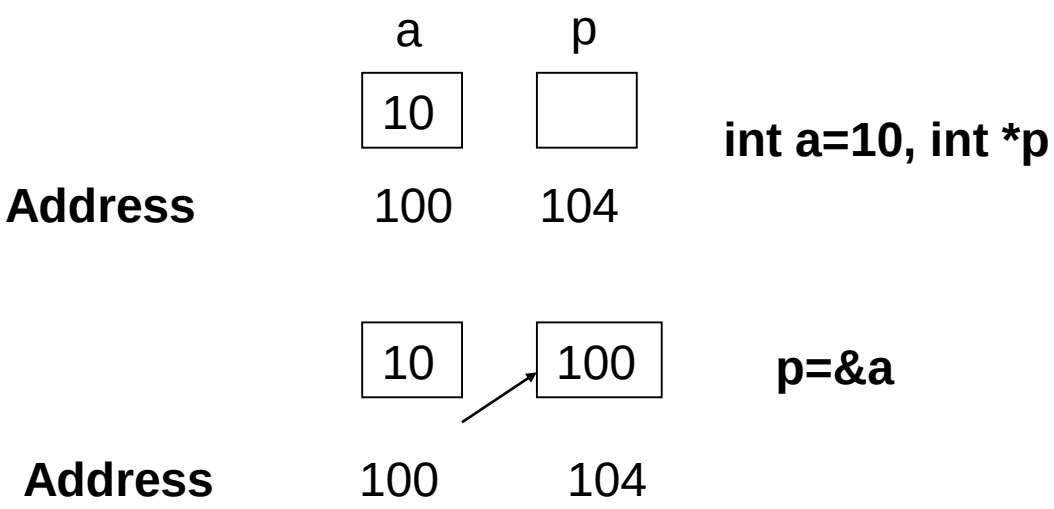
Examples

char* a = “Hello”;

- a -> gives address of ‘H’
- *a -> gives ‘H’
- a[0] -> gives ‘H’
- a++ -> gives address of ‘e’
- *a++ -> gives ‘e’
- a = &b; where b is another char variable is perfectly LEGAL. However “char a[100];” “a =&b;” where b is another char variable is ILLEGAL.

Assigning an Address to a pointer

```
int a = 10
int *p
p = &a
```



Pointers (Contd.)

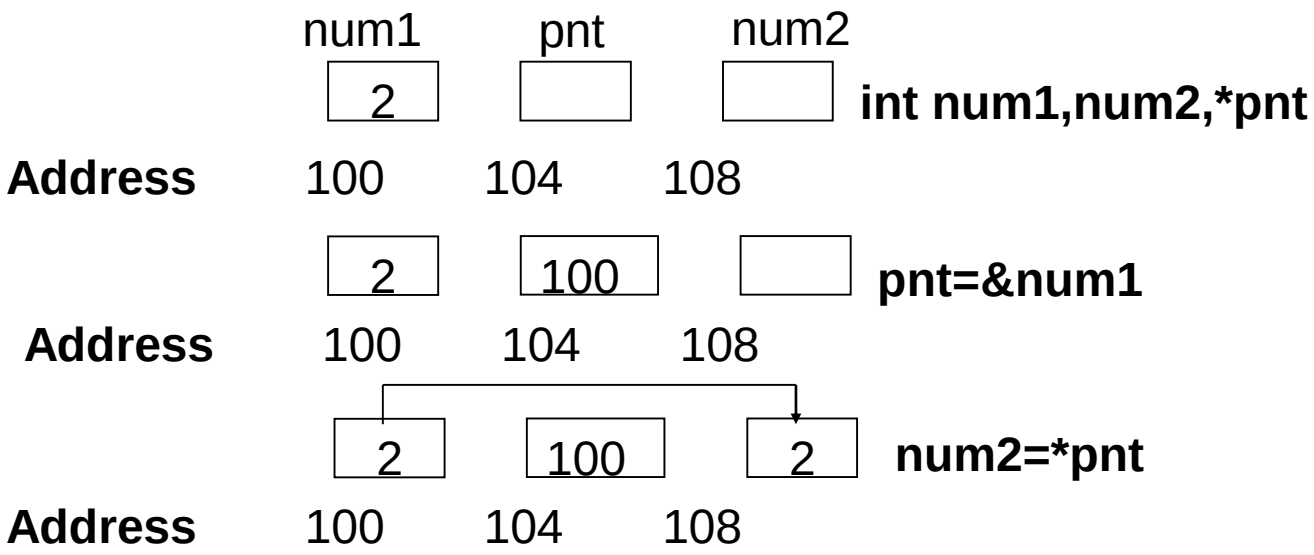
```
char c = 's', *cp
cp = &c
```

c is a variable of type character

cp is a pointer that points to c

Retrieving Values from a Pointer

```
int num1=2,num2,*pnt
pnt=&num1
num2=*pnt
```



Pointers in C#

POINTER

A **pointer** is a programming language object, whose value refers to (or "points to") another value stored elsewhere in the computer memory using its memory address.

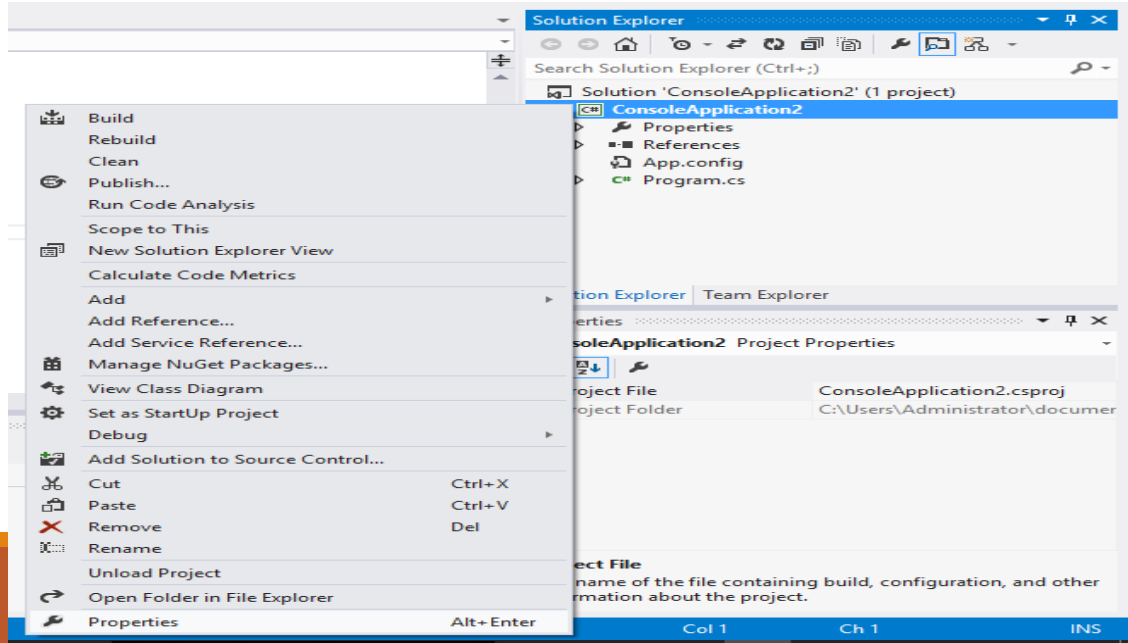
A **pointer** is a variable whose value is the address of another variable i.e., the direct address of the memory location. similar to any variable or constant, you must declare a pointer before you can use it to store any variable address.

Pointer types are not tracked by the default garbage collection mechanism.

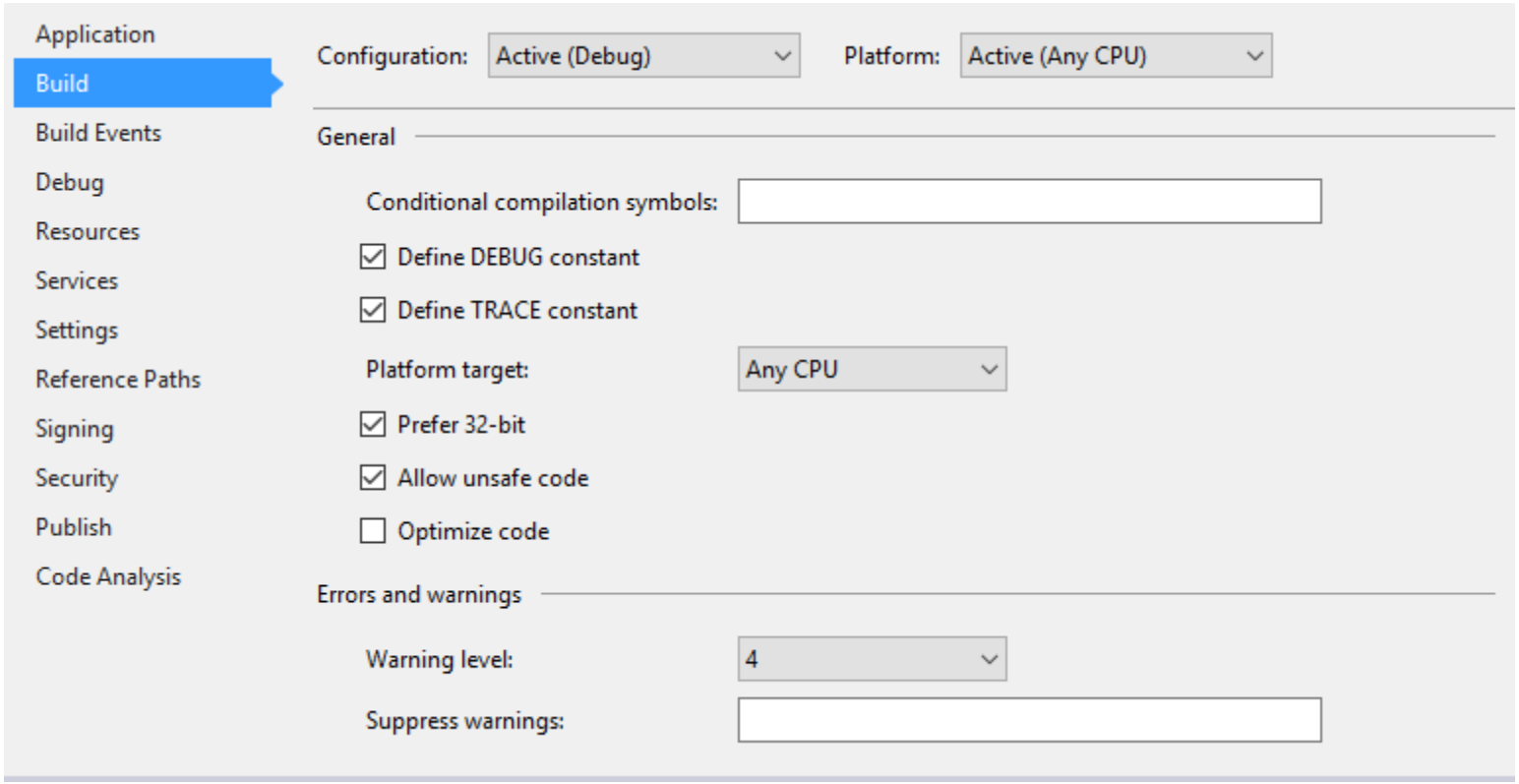
Unsafe Codes

The C# statements can be executed either as in a safe or in an unsafe context. The statements marked as unsafe by using the keyword unsafe runs out side the control of Garbage Collector. Remember that in C# any code involving pointers requires an unsafe context.

Remember to enable unsafe code in the **Project Designer**; choose **Project**, **Properties** on the menu bar, and then select **Allow unsafe code** in the **Build** tab.



Allow unsafe code



representation

```
type* identifier;
```

Example	Description
`int* p`	`p` is a pointer to an integer.
`int*[] p`	`p` is a single-dimensional array of pointers to integers.
`char* p`	`p` is a pointer to a char.
`void* p`	`p` is a pointer to an unknown type.

Operator/Statement	Use
*	Performs pointer indirection.

The pointer indirection operator * can be used to access the contents at the location pointed to by the pointer variable.

&	Obtains the address of a variable.

```
class Program
{
    public static unsafe void Method()
    {
        int x = 10;
        int y = 20;
        int* ptr1 = &x;
        int* ptr2 = &y;
        Console.WriteLine((int)ptr1);//address of x
        Console.WriteLine((int)ptr2);//address of y
        Console.WriteLine(*ptr1);//value of x
        Console.WriteLine(*ptr2);//value of y
    }
    static void Main(string[] args)
    {
     Program.Method();
     }
     }
```

```
5239160
5239156
10
20
Press any key to continue . . .
```

ON RUNNING THE CODE AGAIN

```
12317384
12317380
10
20
Press any key to continue . . .
```

```
public static unsafe void Swap(int* a, int* b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}

static unsafe void Main(string[] args)
{

int num1 = 5;
int num2 = 7;
int* x = &num1;
int* y = &num2;
Console.WriteLine("Values before swap num1=" +*x + " num2=" +*y);
Program.Swap(x,y);
Console.WriteLine("Values after swap num1=" + *x + " num2=" + *y);

}
}
```