**Science College for Women/ Second Class/System programming/ Lec3**

**Lecture: Asraa A.H.**

**Introduction to Pointers**

Some C programming tasks are performed more easily with pointers and other tasks such as dynamic memory allocation cannot be performed without using pointers. So it becomes necessary to learn pointers to become a perfect C programmer.

A Pointer in C language is a variable which holds the address of another variable of same data type. Pointers are used to access memory and manipulate the address.

Before we start understanding what pointers are and what they can do, let's start by understanding what does "Address of a memory location" means.

### Address in C

Whenever a variable is defined in C language a memory location is assigned for it in which its value will be stored. We can easily check this memory address using the & symbol.

If var is the name of the variable, then &var will give it's address. Let's write a small program to see memory address of any variable that we define in our program.

#include<stdio.h>

void main()

{ int var = 7;

printf("Value of the variable var is: %d\n", var);

printf("Memory address of the variable var is: %x\n", &var);

getch();

}

**Output**

Value of the variable var is: 7

Memory address of the variable var is: bcc7a00

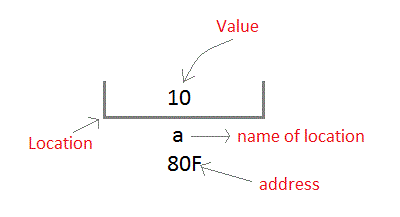
You must have also seen in the function scanf(), we mention &var to take user input for any variable var.

scanf("%d", &var);

This is used to store the user inputted value to the address of the variable var.

### Concept of Pointers

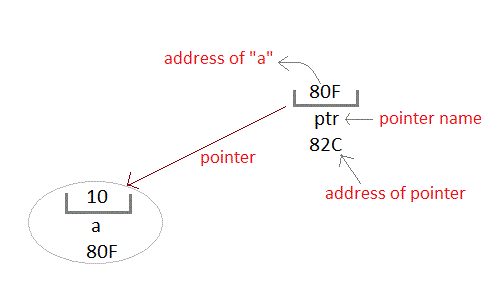
Whenever a **variable** is declared in a program system allocates a location i.e an address to that variable in the memory to hold the assigned value. This location has its own address number which we just saw above. Let us assume that system has allocated memory location 80F for a variable a. int a = 10;



We can access the value 10 either by using the variable name a or by using its address 80F.

The question is how we can access a variable using it's address? Since the memory addresses are also just numbers they can also be assigned to some other variable. The variables which are used to hold memory addresses are called **Pointer variables**.

A **pointer** variable is therefore nothing but a variable which holds an address of some other variable. And the value of a **pointer variable** gets stored in another memory location.



### Declaration of Pointer variable

General syntax of pointer declaration is::

datatype \*pointer\_name;

Data type of a pointer must be same as the data type of the variable to which the pointer variable is pointing.

Here are a few examples:

int \*ip // pointer to integer variable

float \*fp; // pointer to float variable

double \*dp; // pointer to double variable

char \*cp; // pointer to char variable

### Initialization of Pointer variable

**Pointer Initialization** is the process of assigning address of a variable to a **pointer** variable. Pointer variable can only contain address of a variable of the same data type. In C language **address operator** & is used to determine the address of a variable. The & (immediately preceding a variable name) returns the address of the variable associated with it.

#include<stdio.h>

void main()

{ int a = 10;

int \*ptr; //pointer declaration

ptr = &a; //pointer initialization

}

Pointer variable always point to variables of same datatype. Let's have an example to showcase this:

#include<stdio.h>

void main()

{ float a;

int \*ptr;

ptr = &a; // ERROR, type mismatch

}

If you are not sure about which variable's address to assign to a pointer variable while declaration, it is recommended to assign a NULL value to your pointer variable. A pointer which is assigned a NULL value is called a **NULL pointer**.

#include <stdio.h>

void main()

{ int \*ptr = NULL;

}

### Using the pointer or Dereferencing of Pointer

Once a pointer has been assigned the address of a variable to access the value of the variable pointer is **dereferenced** using the **indirection operator** or **dereferencing operator** \*.

#include <stdio.h>

Void main()

{ int a, \*p; // declaring the variable and pointer

a = 10;

p = &a; // initializing the pointer

printf("%d", \*p); //this will print the value of 'a'

printf("%u", &a); //this will print the address of 'a'

printf("%u", p); //this will also print the address of 'a'

printf("%u", &p); //this will print the address of 'p'

getch();

}

%u int unsigned decimal

%x (%X) int unsigned hex value

#### Points to remember while using pointers:

1. While declaring/initializing the pointer variable, \* indicates that the variable is a pointer.
2. The address of any variable is given by preceding the variable name with &.
3. The declaration int \*a doesn't mean that a is going to contain an integer value. It means that a is going to contain the address of a variable storing integer value.
4. To access the value of a certain address stored by a pointer variable, \* is used. Here, the \* can be read as **'value at'**.

### Time for an Example

#include <stdio.h>

void main()

{ int i = 10; // normal integer variable storing value 10

int \*a; // since '\*' is used, hence its a pointer variable

/\*

'&' returns the address of the variable 'i'

which is stored in the pointer variable 'a'

\*/

a = &i;

/\*

below, address of variable 'i', which is stored

by a pointer variable 'a' is displayed

\*/

printf("Address of variable i is %u\n", a);

/\*

below, '\*a' is read as 'value at a' which is 10

\*/

printf("Value at the address, which is stored by pointer variable a is %d\n", \*a);

getch();

}