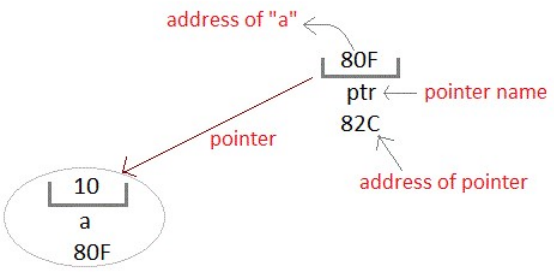
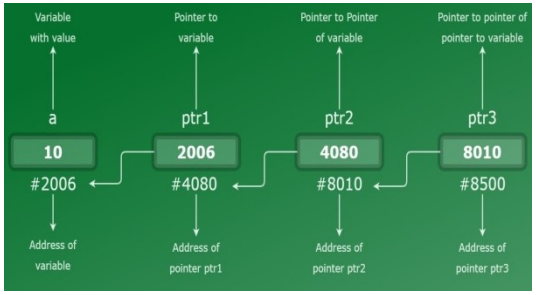
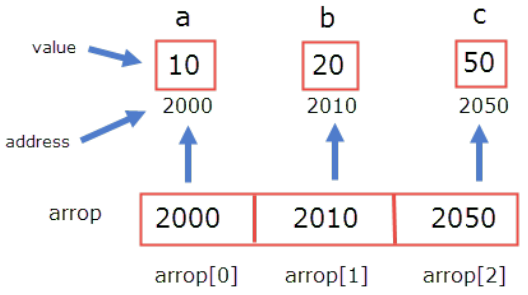
AMIRAJ

COLLEGE OF ENGINEERING & TECHNOLOGY

CHAPTER - 7
POINTERS







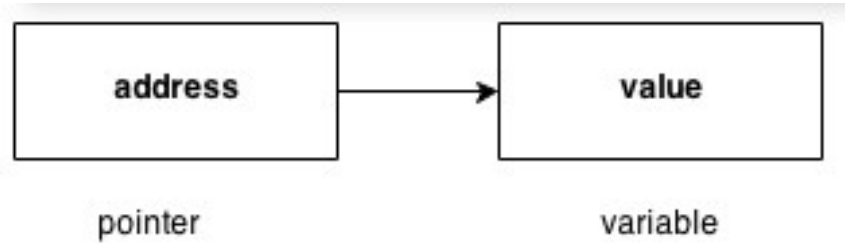
Subject : PPS Code : 3110003	Prepared By: Asst. Prof. Rupali Patel (CSE Department, ACET)	AMIRAJ COLLEGE OF ENGINEERING & TECHNOLOGY

Pointers

The pointer in C language is a variable which stores the address of another variable.

This variable can be of type int, char, array, function, or any other pointer. The size of the pointer depends on the architecture.

However, in 32-bit architecture the size of a pointer is 2 byte.



AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Pointers (cont..)

Consider the following example to define a pointer which stores the address of an integer.

```
int n = 10;
int * p = &n;
```

Declaring a pointer

The pointer in c language can be declared using * (asterisk symbol). It is also known as indirection pointer used to dereference a pointer.

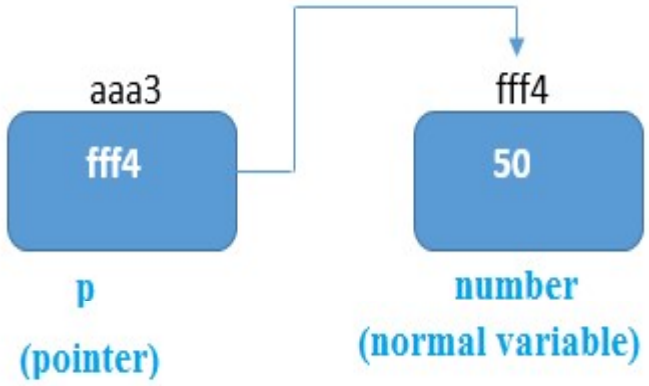
```
int *a;          //pointer to int
char *c;         //pointer to char
```

Pointer Example

```
#include<stdio.h>
void main()
{

int number=50;
int *p;
p=&number;
printf("Address of p variable is %x \n",p);
 printf("Value of p variable is %d \n",*p);

}
```



AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Advantages of pointer

1. Pointer **reduces the code** and **improves the performance**, it is used to retrieving strings, trees, etc. and used with arrays, structures, and functions.

2. We can **return multiple values from a function** using the pointer.

3. It makes you able to **access any memory location** in the computer's memory.

Address Of (&) Operator

The address of operator '&' returns the address of a variable. But, we need to use %u to display the address of a variable.

```
#include<stdio.h>
void main()
{
int number=50;
printf("value of number is %d, address of number is %u",number, &number);
}
```

NULL Pointers

It is always a good practice to assign a NULL value to a pointer variable in case you do not have an exact address to be assigned. This is done at the time of variable declaration. A pointer that is assigned NULL is called a **null** pointer.

```
#include <stdio.h>
Void main ()
{
        int  *ptr = NULL;
        printf("The value of ptr is : %x\n", ptr  );
 }
```

Arithmetic Pointer

A pointer in c is an address, which is a numeric value.

Therefore, you can perform arithmetic operations on a pointer just as you can on a numeric value.

There are four arithmetic operators that can be used on pointers: ++, --, +, and –.

Incrementing a Pointer

We prefer using a pointer in our program instead of an array because the variable pointer can be incremented, unlike the array name which cannot be incremented because it is a constant pointer.

The next program increments the variable pointer to access each succeeding element of the array

Incrementing a Pointer (cont..)

```
#include <stdio.h>
int main ()
{
        int  var[] = {10, 100, 200};
        int  i, *ptr;
        ptr = var;
        for ( i = 0; i < 3; i++)
        {
                printf("Address of var[%d] = %x\n", i, ptr );
                printf("Value of var[%d] = %d\n", i, *ptr );
                ptr++;
        }
}
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Pointer Comparisons

Pointers may be compared by using relational operators, such as ==, <, and >.

If p1 and p2 point to variables that are related to each other, such as elements of the same array, then p1 and p2 can be meaningfully compared.

Pointer Comparisons (cont..)

```
#include <stdio.h>
void main ()
{
        int  var[] = {10, 100, 200};
        int  i, *ptr;
        ptr = var;
        i = 0;
        while ( ptr <= &var[2] )
        {
                printf("Address of var[%d] = %x\n", i, ptr );
                printf("Value of var[%d] = %d\n", i, *ptr );
                ptr++;
                i++;
        }
  }
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Array of pointers

There may be a situation when we want to maintain an array, which can store pointers to an int or char or any other data type available.

Following is the declaration of an array of pointers to an integer.

Syntax: int *ptr[3];

Array of pointers (cont..)

```
#include <stdio.h>
Void main ()
{
        int  var[3] = {10, 100, 200};
        int i, *ptr[3];
        for ( i = 0; i < 3; i++)
        {
                ptr[i] = &var[i];
        }
        for ( i = 0; i < 3; i++)
        {
                printf("Value of var[%d] = %d\n", i, *ptr[i] );
         }
}
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Array of pointers to character

```
#include <stdio.h>
void main ()
{
        char *names[ ] = { "Zara ", "Hina ", "Nuha ","Sara " };
        int i = 0;
        for ( i = 0; i < 4; i++)
        {
                printf("Value of names[%d] = %s\n", i, *names[i] );
        }
}
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Passing pointers to functions

```
#include <stdio.h>
void salaryhike(int  *var, int  b)
{
   *var = *var+b;
}
int main()
{
        int salary=0, bonus=0;
        printf("Enter the employee current salary:");
        scanf("%d", &salary);
        printf("Enter bonus:");
        scanf("%d", &bonus);
        salaryhike(&salary,bonus);
        printf("Final salary: %d", salary);
        return 0;
}
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Passing array pointer to function

```
#include <stdio.h>
double getAverage(int *arr, int size);
void main ()
{
  int balance[5] = {1000, 2,3, 17,
50};
  double avg;
  avg = getAverage( balance, 5 );

printf("Average value is: %f\n", avg
);
 }
```

```
double getAverage(int *arr, int size)
{
        int  i, sum = 0;
        double avg;
        for (i = 0; i < size; ++i)
        {
                sum += arr[i];

         }
        avg = (double)sum / size;
        return avg;
}
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Return pointer from functions in C

C also allows to return a pointer from a function.

To do so, you would have to declare a function returning a pointer as in the following example.

```
int * myFunction()
{
   .......
   .......
 }
```

Return pointer from functions in C (cont..)

```
#include<stdio.h>
int  *return_pointer(int *, int);
int main()
{
   int i, *ptr;
   int arr[5] = {11, 22, 33, 44,
55};
   i = 4;
    printf("Address of arr = %u\n",
arr);
   ptr = return_pointer(arr, i);

 printf("\nAfter incrementing arr
by 4 \n\n");
printf("Address of ptr = %u\n\n" ,
ptr);
printf("Value at %u is =%d\n",
ptr, *ptr);
 return 0;
}

int *return_pointer(int *p, int n)
{
        p = p + n;
        return p;
}
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Pointer to Pointer

A pointer to a pointer is a form of multiple indirection, or a chain of pointers. Normally, a pointer contains the address of a variable.

When we define a pointer to a pointer, the first pointer contains the address of the second pointer, which points to the location that contains the actual value as shown below.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Pointer to Pointer (cont..)

A variable that is a pointer to a pointer must be declared as such.

This is done by placing an additional asterisk in front of its name. For example, the following declaration declares a pointer to a pointer of type int.

int **var;

When a target value is indirectly pointed to by a pointer to a pointer, accessing that value requires that the asterisk operator be applied twice, as is shown next in the example.

Pointer to Pointer (cont..)

```
#include <stdio.h>
int main ()
{
        int  var;
        int  *ptr;
        int  **pptr;
        var = 3000;
        ptr = &var;
        pptr = &ptr;
        printf("Value of var = %d\n", var );
        printf("Value available at *ptr = %d\n", *ptr );
        printf("Value available at **pptr = %d\n", **pptr);
        return 0;
}
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY