FACULTY OF ENGINEERING AND TECHNOLOGY
UNIVERSITY OF LUCKNOW
LUCKNOW

Computer System and Programming in ‘C’

CS-101/201

Er. Zeeshan Ali Siddiqui
Assistant Professor
Deptt. of C.S.E.

POINTERS

Overview

- **Pointer**:
  - A pointer *points* to a memory location
  - It stores the memory *address*
  - It also can gives *value* stored at that address

```
int xyz_var=25;
```

xyz_var	←Variable_name
25	←Value
1000	←Address

- **Pointer variable**:
  - A pointer variable is a variable which stores the address of *another* variable.
  - *Data type* of the pointer variable *must be the same* as that of variable whose address it stores.

Syntax and Example

- **Syntax**:

```
data_type *pointer_var;
```

- **Example:**

```
int *ptr_var;
```

Here,

- ptr_var is the name of pointer variable
- the '*' indicates that we need a pointer variable, that is to set aside required bytes to store an address in memory.
- The int says that this pointer variable is intended to store the address of an integer variable.
- This type of pointers is said to "*point to*" an integer.

Visualization

```
int xyz_var=25;
```

xyz_var	←Variable_name
25	←Value
1000	←Address

```
int *ptr_var= &xyz_var;
```

ptr_var	←Pointer_variable_name
1000	←Value
2000	←Address

Sample Program-1

```c
#include<stdio.h>
int main()
{
    int a =2020,*ptr;
    ptr=&a;
    //It will show the value that ptr contains that is the adress of variable a
    printf("ptr=%d\t ptr=%p\t ptr=%u\t ptr=%x\n",ptr,ptr,ptr,ptr);

    /* when we use the '*' as below way we are referring to the value of that
    which ptr is pointing to, not the value of the pointer itself. */
    printf("value=%d",*ptr); // *-> value at address operator (dereferencing operator)
    return 0;
}
```

- Output

D:\DEVC\pointertest2.exe

```
ptr=2293316      ptr=000000000022FE44    ptr=2293316     ptr=22fe44
value=2020
--------------------------------
Process exited after 0.005863 seconds with return value 0
Press any key to continue . . .
```

Sample Program 2- Give the output!

```c
#include<stdio.h>
int main()
{
    int *int_ptr;
    char *char_ptr;
    float *float_ptr;

    printf("Size of int_ptr=%d\n",sizeof(int_ptr));
    printf("Size of char_ptr=%d\n",sizeof(char_ptr));
    printf("Size of float_ptr=%d\n",sizeof(float_ptr));
    return 0;
}
```

Arithmetic and Expressions 1/2

Rules for pointer operations:

- A pointer variable can be assigned the address of another variable.

```
int a =2020;
int *ptr;
ptr=&a;
```

- A pointer variable can be assigned the address of another pointer variable.

```
int *ptr;
int **q;
q=&ptr;
```

- A pointer variable can be initialized with null value.

```
int a =2020;
int *ptr=NULL;
ptr=&a;
```

Arithmetic and Expressions 2/2

Rules for pointer operations:

- Prefix or postfix increment and decrement operators can be *applied* on a pointer variable.
- An *integer* value can be added or subtracted from a pointer variable.
- A pointer variable can be *compared* with another pointer variable of the same type using relational operator.
- In an expression multiplication and division operation on pointers are *not* allowed.
- A pointer variable *cannot be added* to another pointer variable.

Sample Program-3

```
//swap values of two variables using pointers
int main()
{
    int a=5,b=6,temp,*x,*y;
    x=&a;
    y=&b;
    printf("Before\na=%d\nb=%d",a,b);
    temp=*x;
    *x=*y;
    *y=temp;
    printf("\nAfter\na=%d\nb=%d",a,b);
    return 0;
}
```

- Output

D:\DEVC\pointertest3.exe

```
Before
a=5
b=6
After
a=6
b=5
--------------------------------
Process exited after 0.00483 seconds with return value 0
Press any key to continue . . .
```

Sample Program-4

```c
//Program to print all odd numbers from m to n (m<=n)
int main()
{
    int m,n,*pm=&m,*pn=&n;
    printf("Please give starting and ending values\n");
    scanf("%d%d",pm,pn);
    while(*pm<=*pn)
    {
        if(*pm%2!=0)
        printf("%d is odd\n",*pm);
        (*pm)++;
    }
    return 0;
}
```

- Output

D:\DEVC\pointertest4.exe

```
Please give starting and ending values
1
5
1 is odd
3 is odd
5 is odd

--------------------------------
Process exited after 3.622 seconds with return value 0
Press any key to continue . . .
```

Scale Factor

- An integer value can be *added* to or subtracted from a pointer variable.
- In reality, it is not the integer value which is added/ subtracted, but rather the *scale factor* times the value.
- Example: for 64 bit compiler scale factor:

Data Type	Scale Factor
char	1
int	4
float	4

```
int a =2020;
int *ptr=NULL;
ptr=&a;
ptr++;
```

Here, value of ptr will be incremented by 4.

Points to Remember

***ptr++**
-> fetch the value and use, then increase the pointer by scale factor

++*ptr
-> increment the value and use

(*ptr)++
-> fetch the value and use, then increase the value by 1

++(*ptr)
-> increment the value and use

Errors!

- Pointer variables must always point to a data item of the same type only.

```
float f=25.6;
int *p=&f;      ← Error
```

- Assigning an absolute address to a pointer variable is prohibited.

```
int *p;
p=2000;         ← Error
```

- If ptr1 and ptr2 are two pointers, then we cannot perform following operations:
- Add: `ptr1 =ptr1+ptr;` ← Error
- Multiply: `ptr1 =ptr1*ptr2;` ← Error
- Divide: `ptr1 =ptr2/3;` ← Error

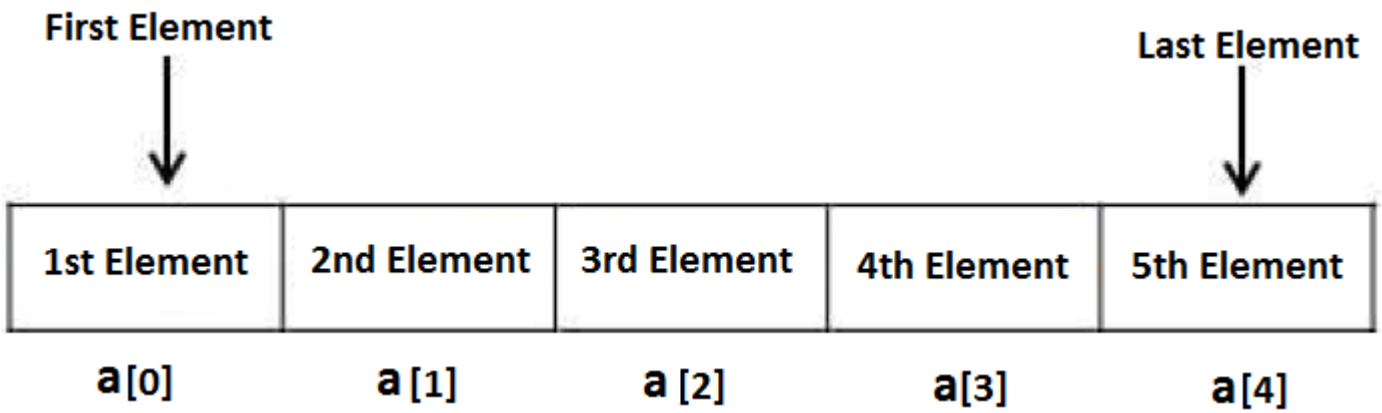
POINTERS AND ARRAYS

Pointers and Arrays

- When an array is declared
  - The compiler allocates a *base address* and sufficient amount of storage to contain all the elements of the array in contiguous memory locations.
  - The base address is the location of the *first element* (index 0) of the array.
  - The compiler also defines the array name as a *constant pointer* to the first element.
  - An array name is an *address*, or a pointer value.

Pointers and Arrays

- When an array is declared: int a[5], *p;



- The expression a[i] is equivalent to *(a+i)

- p=a+2; is equivalent to
  p=&a[2]; ← **p pointing to the third element of the array**

- When an array is declared the compiler allocates a sufficient amount of contiguous space in memory. Suppose the system assigns 1000 as the base address of a. a[0], a[1], ...,a[4] are allocated 1000, 1004, ..., 1016.

Example

- Let we have an array: `int a[3]={1,2,3};`
- Let the Base address that is the address of first element of this array is 1000 and each integer is of 4 bytes.

Element	Value	Address
a[0]	1	1000
a[1]	2	1004
a[2]	3	1008

- Let we have a pointer: `int *p;`

```
p=a;
or
p=&a[0];
```

← **Both a and &a[0] have the value 1000**

- Now,

p	&a[0]	1000
p+1	&a[1]	1004
p+2	&a[2]	1008

← ***(p+i) gives the value of a[i]**

Sample Program-5

```c
#include<stdio.h>
//print the values of an array using pointers
int main()
{
    int milestone[]={1857, 1947, 1950, 1957, 1995, 2000};
    int *ptr1= &milestone[0], *ptr2=&milestone[5];
    while(ptr1<=ptr2)
    {
        printf("%d\t",*ptr1++);
    }

    return 0;
}
```

D:\DEVC\pointrearray.exe

```
1857    1947    1950    1957    1995    2000
--------------------------------
Process exited after 0.002848 seconds with return value 0
Press any key to continue . . .
```

Array of Pointers

- An array of pointers can be defined as: `int *p[3];`
- Above statement declares an array of 3 pointers where each of the pointer points to an integer variable.
- Sample program with output:

```c
#include<stdio.h>
int main()
{
    int a=1,b=2,c=3,*p[3];
    p[0]=&a;
    p[1]=&b;
    p[2]=&c;
    printf("p[0]=%d\tp[1]=%d\tp[2]=%d",*p[0],*p[1],*p[2]);
    return 0;
}
```

D:\DEVC\arrayquiz.exe

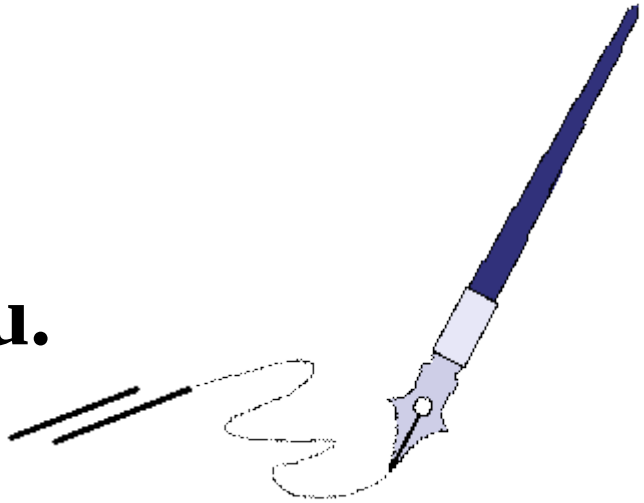
```
p[0]=1  p[1]=2  p[2]=3
--------------------------------
Process exited after 0.02537 seconds with return value 0
Press any key to continue . . .
```

Exercise

- Define pointer.
- Define NULL pointer.
- Define wild pointer.
- Define arrays of pointers.
- What is the difference between pointer and a pointer variable?
- Give the difference between * and & in C pointer.
- Data type of the pointer variable must be the same as that of variable whose address it stores.
- WAP to print a character. Also prints its ASCII value using pointers.
- Give the output of below program:

```c
int main()
{
    int a[]={1,2,3,4,5};
    printf("data=%d",++*a);
    return 0;
}
```

Thank You.



BTQ

BTQ: Brain Teaser Question

Today is Mr. X's birthday.

A year Ago on his birthday, he had five candles and he lit all except the last one.

Today he is going to light all the candles.

How old is Mr. X today?

☺

**Note: He is not turning five today.*