C Programming

Class- BCA IInd Semester

Dr. Dharm Raj Singh

Assistant Professor, (HOD)

Department of Computer Application

Jagatpur P. G. College, Varanasi

Affiliated to Mahatma Gandhi Kashi Vidhyapith Varanasi

Email- dharmrajsingh67@yahoo.com

Outline

unit 2 : Pointer

- Introduction
- Declaration of Pointer Variable
- Accessing the Address of a Variable
- Initialization of Pointer Variable
- Accessing a Variable through its Pointer
- Pointer Increments and Scale Factor
- Pointer and Arrays

Introduction:-

- Pointers are a fundamental part of C. A pointer is a derived data type in‘C’. It is built from any one of the primary data type available in ‘C’programming language. A pointer is a variable whose value is the address of another variable, i.e., direct address of the memory locationor Pointer is a variable that holds the Address of another variable. Like any variable or constant, you must declare a pointer before using it to store any variable address.

Address operator &

- The actual location of a variable in the memory is depending on system. How we know the address of a variable?This can be done with the help of **operator(&) in C.** The **& operator (ampersand)** is also known as "**Address of**" Operator.The**&**operator can be used only with a simple variable or an array element. Some of the following:-

```
1.#include<stdio.h>
2.#include<conio.h>
3.void main()
4.{
5.int a;
6.printf("Input any no.:\n");
7.scanf("%d",&a);
8.printf("Your data is %d.\n",a);
9.printf("Variable 'a' is stored in memory at %u\n",a);
10.getch();
11. }
```

Declaring Pointer Variable:-

- Pointer declaration is similar to other type of variable except asterisk (*) character before pointer variable name. it is also called dereference operator.
- **“Value at Address”(*) Operator**
- **The * Operator is also known as Value at address operator.**
- **Here is the syntax to declare a pointer:-*data_type *variable_name;***

```
#include<stdio.h>
int main()
{
int*p;
intv=10;
p=&v;/* Assigning the address of variable v to the pointer p*/
printf("Value of variable v is: %d",v);
printf("\n Value of variable v is: %d",*p);
printf("\n Address of variable v is:%p",&v);
printf("\n Address of variable v is: %p", p);
printf("\nAddress of pointer p is: %p",&p);
}
```

INITIALIZATION OF POINTER VARIABLE:-

- The process of assigning the address of a variable to a pointervariable is known as **initialization.** Once a pointer variable has been declare we can use the assignment operator to initialize the variable. EXAMPLE:
- **int quantity;**
- **int *p;**
- **p= &quantity;**

```
#include<stdio.h>
int main()
{
int a;
int *ptr;
a = 10;
ptr = &a;
printf("Value of ptr:%u", ptr);
return (0);
}
```

Accessing the Address of a Variable through its Pointer:-

Once a pointer has been assigned the address of variable, the question is how to access a value of a variable through its pointer? This is done by using another unary operator *(asterisk),Consider the following statements.

int ds, *p, n;

ds = 133;

p = &ds;

n = *p;

Here we can seen that dsand **n** declare as integer and **p** pointer variable that point to an integer. The second line assigns **133 to ds** and third line assign the address of **ds to variable p.**The fourth line contain the indirection operator * . When the operator * is placed before the pointer variable in an expression, the pointer returns the values of variable of which the pointer value is the address. In this case, ***p** returns the value of variable**ds.** Thus the value of **n**would be **133.** The two statements

p =&ds;

n = *p;

are equivalent to

n = *&ds;

which is turn is equivalent to

n = ds;

Pointer Arithmetic

- **Pointer Arithmetic:-**The size of data type which the pointer variable points to is the number of bytes accessed in memory. The size of the pointer variable is dependent on the data type of the variable pointed by the pointer. Some arithmetic operations can be performed with pointers. C language supports four arithmetic operators which can be performed with pointers .They are

  addition (+)
  subtraction (-)
  Pointer increment(++)
  Pointer decrement(- -)

- **Pointer increment and decrement:-**Integer, float, char, double data type pointers can be incremented and decremented. For all these data types both prefix and post fix increment or decrement is allowed. Integer pointers are incremented or decremented in the multiples of two. Similarly character by one, float by four and double pointers by eight etc.**Note** that pointer arithmetic cannot be performed on void pointers, since they have no data type associated with them.

```
#include <stdio.h>
main()
{
int *p1,x;
float *f1,f;
char *c1,c;
p1=&x;
f1=&f;
c1=&c;
printf("Memory address before increment:\n int=%p\n,float=%p\n,
    char=%p\n", p1,f1,c1);
p1++;
f1++;
c1++;
printf("Memory address after increment:\n int=%p\n,
    float=%p\n,char=%p\n",p1,f1,c1);
}
```

- **Subtraction of one pointer from another:-**
- One pointer variable can be subtracted from another provided both variables point to elements of the same array. The resulting value indicates the number of bytes separating the corresponding array elements. This is illustrated in the following program.

```
#include <stdio.h>
main()
{
int x[] = {10, 20, 30, 45, 67, 56, 74} ;
int *i, *j ;
i = &x[1] ;
j = &x[5] ;
printf ( "%d %d", j - i, *j - *i ) ;
}
```

- **Pointers does not allow the flowing operation**
  (a) Addition of two pointers
  (b) Multiplication of a pointer with a constant
  (c) Division of a pointer with a constant

```
#include<stdio.h>
#include<conio.h>
 main ()
{
int  a[3], i, sum=0, *p;
printf ("enter 3 elements \n");
for (i=0; i<3; i++)
scanf ("%d", & a[i]);
p = a;
for (i = 0; i<3; i++)
{
 sum = sum+ *p;
 p++;
}
printf ("the sum is % d", sum);
getch ();
}
```

Pointers and one dimensional Arrays:-

The elements of an array are store in contiguous memory location. When an array is declared, compiler allocates sufficient amount of memory to contain all the elements of the array. Base address i.e address of the first element of the array is also allocated by the compiler.

Suppose we declare an array x,

intx[5] = { 1, 2, 3, 4, 5 };

Assuming that the base address of x is 1000 and each integer requires two bytes, the five elements will be stored as follows:

Element	x[0]	x[1]	x[2]	x[3]	x[4]
Address	1000	1002	1004	1006	1008

Here variable **x** will give the base address, which is a constant pointer pointing to the first element of the array**, x[0].** Hence **x**contains the address of **x [0]** i.e. **1000**. In short, **x** has two purpose - it is thename of the array and it acts as a pointer pointing towards the first element in the array.**x** is equal to **&x[0]** by default.

We can also declare a pointer of type int to point to the array **x**.

int *p; p = x; // or,

p = &x[0];   //both the statements are equivalent.

Now we can access every element of the array **x** using **p++** to move from one element to another.

NOTE: You cannot decrement a pointer once incremented. **p--** Won't work.

- **Pointer to Array :-**
- **A**s studied above, we can use a pointer to point to an array, and then we can use that pointer to access the array elements. Let's have an example,

```
#include <stdio.h>
intmain()
{
int i;
int a[5]={1,2,3,4,5};
int*p = a; // same as int*p = &a[0]
for(i =0; i <5; i++)
{
printf("%d\n",*p);
p++;
}
}
```

Pointer to an Array:

- We have seen that a pointer that pointed to the first element of array. We can also declare a pointer that can point to the whole array. For example int (*p)[10]; here p is point to an array of 10 integer. Note that it is necessary to enclose the pointer name inside parentheses.

```
#include <stdio.h>
int main()
{
int *p;
int x[5];
int (*pa) [5];
 p=x;
 pa=x;
printf(" p=%u,   pa=%d\n", p, pa);
 p++;
 pa++;
printf(" p=%u,   pa=%d\n", p, pa);
 }
```

Exercise

1. Write a C program to swap two numbers using pointers.
2. Write a C program to input and print array elements using pointer.
3. Write a C program to copy one array to another using pointer.
4. Write a C program to swap two arrays using pointers.
5. Write a C program to reverse an array using pointers.
6. Write a C program to search an element in array using pointers.
7. Write a C program to access two dimensional array using pointers.
8. Write a C program to add two matrix using pointers.
9. Write a C program to multiply two matrix using pointers.

References

- Kanetkar, Yashavant P. "Let UsC Fifth Edition." (2017).
- Kernighan, Brian W., and Dennis M. Ritchie. *The C programming language*. 2006.
- Ritchie, Dennis M., Brian W. Kernighan, and Michael E. Lesk. The C programming language. Englewood Cliffs: Prentice Hall, 1988.
- McGraw-Hill, Herbert Schildt Tata. "The Complete Reference C fourth Edition”." (2005).
- Griffiths, David, and Dawn Griffiths. Head First C: A Brain-Friendly Guide. " O'Reilly Media, Inc.", 2012.
- Programming in C-Balguruswamy
- Structured programming approach using C-Forouzah & Ceilberg Thomson learning Publication

Declaration

“The content is exclusively meant for academic purpose and for enhancing teaching and learning. Any other use for economic/commercial purpose is strictly prohibited. The users of the content shall not distribute, disseminate or share it with anyone else and its use is restricted to advancement of individual knowledge. The information provided in this e-content is authentic and best as per knowledge”.

Dr. Dharm Raj Singh
Assistant Professor, (HOD)
Department of Computer Application
JagatpurP. G. College, Varanasi

Thanks