15
Pointer

15.1 Introduction

In C++, the programming with pointers is more powerful and it is used extensively. It saves the processing time. Pointer is a variable which holds the address of another variable. So, programming is concerned with the address, not with the direct value stored.

15.2 Objectives

After going through this lesson, you would be able to:

- use pointers in arrays
- define pointer variables in a structure and access data members through pointer
- define pointer objects in a class and access members through pointer

15.3 Pointer

A pointer is a variable that represents the location (rather than the value) of a data item such as a variable or an array element. Pointers are used frequently in C++, as they have a number of useful applications. Consider the following example:

```
# include < iostream.h >
void main ( )
{
        int A = 5;
        cout << & A;
        int *ptr;
        ptr = & A;
        cout < < *ptr;
    }
```

If variable A in the above example has a value 5, then & A holds the address of memory cell A. The variable which holds the address is called pointer variable. int *ptr means that a variable ptr is a pointer to a memory cell which can hold the int data type. *ptr is used for the value stored in a memory cell pointed to by the pointer ptr. It is called de-referencing the pointer. The output of the above program is the address of memory cell A and value 5.

void *ptr;

Here ptr can point to any data type.

15.3.1 Pointer to Array

Consider the following declaration:

int A [5];

The name of the array A itself is a pointer which holds the address of zero location (&A[0]). It is a constant in a program, its value cannot be changed. The following program prints all the values of an array A.

```
    # include < iostream.h >
    void main (  )
    {
        int [5] = { 20, 35, 25, 22, 27 };
        for (int i = 0; i < 5; i + + )
        cout << “\n” << A [ i ];
    }
        The output is       20
                            35
                            25
                            22
                            27
```

The above program can be written as pointer notation.

```
# include < iostream.h >
void main ( )
{
    int A [5] = { 20, 35, 25, 22, 27 }
    for (int i = 0; i < 5; i + + )
    cout << "\n" << *(A + i);

}
```

At one stage the value of i is 2. Therefore A + i is 2, i.e., two locations from the zero location. The *(A+i) will print the data stored at that location.

15.3.2 Pointer to String constant

Consider the following example:

```
# include < iostream.h >
void main ( )
{
    char stu1 [ ] = "work as an array";
    char *stu2 = "work as a pointer";
    cout << stu 1;
    cout << stu2; // display both the strings
    stu1 ++; / / it is a wrong statement
    stu2 + +;
    cout << stu2; / / it prints "ork as a pointer"

}
```

stu 1 + + is a wrong statement because stu 1 is a pointer which holds the address of zero location (& stu1 [0]). It is a constant in a program.

15.3.3 Pointer to Structure

Consider the following program

```
struct student
{
    char name [30];
```

```
        int rn;
    };
```

The statement

```
student st;
```

declares st as the variable of the structure student.

The statement

```
student *ptr;
```

declares a pointer variable ptr to a student.

That data members using ptr can be referred to as

```
ptr -> name;
ptr -> rn;
```

Another way of referring the data member is

```
(*ptr) . name;
(*ptr) . rn;
```

15.3.4 Pointer to Objects

Object pointers are useful in creating objects at run time. Object pointers can be used to access the public members of an object. Consider a class student defined as follows:

```
class student
{
    char name [20];
    int rn;
public :
    int marks;
    void getdata ( );
    void putdata ( );
};
void student : : getdata ( )
{
        cin >> name;
        cin >> rn;
        cin >> marks;
    }
    void student : : putdata (  )
    {
        cout << “Name” << << name << “\n”;
        cout << “Roll no” << rn << “\n”;
        cout << “Marks” << marks << “\n”;
    }
```

Declare an object st and pointer ptr as follows:

```
student st;
student *ptr;
```

We can refer to the member functions and data member of student in two ways.

(i) using dot operator

```
st. marks = 90;
st. getdata ( );
```

(ii) using arrow operator and object pointer

```
ptr -> marks = 90;
ptr -> getdata (  );
```

another way to represent is (*ptr)

```
(*ptr) . marks = 90;
(*ptr) . getdata (  );
```

15.4 This Pointer

C++ uses a unique keyword called *this* to represent the object that invokes a member function. *This* is a pointer that points to the object for which *this* function was called. For example;

```
class ABC

{
        int rn;
    public:
        void getdata ( )
        {
        cin >> this -> rn;
        }
        void putdata ( )
        {
        cout << this -> rn;
        };
    void main ( )
    {
        ABC A, B;
        A . getdata ( );
        A . putdata ( );
        B . getdata ( );
        B . putdata ( );
    }
```

When a getdata ( ) or putdata ( ) function is called through object A, *this* has the address of object A. Similarly, when a getdata ( ) or putdata ( ) function is called through object B, *this* has the address of object B.

In-Text Questions 15.1

1. How is a pointer variable different from ordinary variable ?

2. A pointer is :

   (i) the address of a variable
   (ii) an indication of the variable to be accessed next
   (iii) a variable for string addresses
   (iv) the data type of address variable

3. If a pointer ptr points to a variable A, write a statement that represents the content of A but does not use its name.

4. The expression *ptr can be said to

(i) be a pointer to ptr

(ii) refer to the contents of ptr

(iii) refer to the value of the variable pointed to by ptr

(iv) dereference ptr

5. The expression int* can be said to

   (i) be a pointer pointing to variable int

   (ii) refer to contents of int

   (iii) be a pointer to a int type value.

6. Fill in the blanks:

   (a) A pointer variable stores the .... of another variable

   (b) The .... operator is called address operator.

   (c) To declare t_age pointer to integer, we should write ....

   (d) If A is an integer variable than .... gives its address.

   (e) If ptr is a pointer to an integer variable, the number stored in it is given by ....

7. State whether the following are True or False.

   (a) A pointer is a address of the variable.

   (b) Dereferencing operator (*) is a unary operator. It is different from the multiplication operator (*) which needs two operands.

   (c) this pointer points to the objects that is currently used to invoke a function.

15.5 What you have learnt

Having gone through the present lesson you learnt the use of pointer in variable array, structure variable and class object. A special pointer is available in C++ called this pointer. We have learnt the use of this special pointer also.

15.6 Terminal Questions

1. If arr is an array of integers, why is the expression arr++ not legal?

2. What is the difference between arr [4] and *(arr+4)?

3. If a structure defined has pointer variable, how can it access the members of the structure ? Explain if by taking an example.

4. How a data member and member function present in public in class accessed through pointer object? Explain it by taking an example.

5. What is this pointer? Explain briefly?

15.7 Feedback to In-Text Questions

In-text Questions 15.1

1. The pointer variable stores the address whereas ordinary variable stores the value.

2. (i)

3. *ptr

4. (iv)

5. (iii)

6. (a) address
   (b) &
   (c) int *t_age;
   (d) &A
   (e) *ptr

7. (a) T
   (b) T
   (c) T