CHAPTER 7

POINTERS

After completing this lesson, you will be able to:

- Define pointers
- Understand memory addresses
- Know the use of reference operator &
- Know the use of deference operator *
- Declare variables of pointer types
- Initialize the pointers

1.1 INTRODUCTION TO POINTERS

Q1. What are pointers in C++? Write down its uses and advantage in C++

Ans:

Pointers

Pointers are powerful feature of C++ that differentiates it from other programming languages. With the help of pointers, C++ gives users the power to manipulate the data in the computer's memory directly.

Uses of pointers

1) Pointers are used in C++ program to access the memory and manipulate the address. The variable name refers to that memory space that is occupied by it.
2) Pointers are used to store the address of a variable. The width of the memory address/location depends on the computer architecture. If the computer architecture is 16-bit then it means that it can have 216 memory bits or 2 bytes wide. Similarly, for 32-bit and 64-bit architecture, we need to have pointers with size 4 byes and 8 bytes

Advantage of Pointer

The main advantage of pointer is it can save memory and run faster because it does not have to duplicate the date

1.1.1 POINTER VARIABLE

Q2. What is meant by pointer variable? Explain with the help of program

Ans

Pointer Variable

Pointer variable is a variable that points to a specific address in the memory pointed by another variable.

Purpose of point variable

Point variable holds the address of variable. In memory, every variable has an address assigned to it by the compiler and if a programmer wants to access that address, another variable called "pointer" is needed.

Declare a pointer

For the declaration of pointer, an asterisk (*) symbol is used.

Examples of Declaring Pointers

Consider the following pointers declaration of the integer variable marks, floating point variable percentage and character type variable name

Int * marks :

Float * percentage :

Char * name :

1.1.2 MEMORY ADDRESSES

Q3. Describe the concept of memory addresses in declaring a variable.

Ans

When writing a program, variables are needed to be declared. Declaration of variable is simple and its declaration tells the computer to reserve space in memory for this variable. The name of the variable refers to that memory space. This task is automatically performed by the operating system during runtime. When variables are declared programmers store data in these variables. Therefore, everything a programmer declares has an address. It is analogous to the home address of some person. Using pointer variables, one can easily find out the address of a particular variable.

1.1.3 REFERENCE OPERATOR (&) OR ADDRESS OPERATOR

Q4. Explain reference operator or address operator with the help of program

Ans:

Reference Operator OR Address Operator

As pointers are the variables which hold the addresses of other variables. Therefore, while assigning addresses to them, a programmer needs a special type of operator called reference or address operator that is denoted by ampersand (&) symbol. This provides address of a memory location.

Example

To understand it, consider the following segment of code

```
Float x = 6.5;
```

Float *fPointer;

fPointer = &x: //assign address of x to fPointer ;

Conceptually, the above lines of code can be pictorially represented as

Figure 7.1: Use of the Pointer Variable

Here, the variable x has a float value 6.5 that is stored at location 2000. The pointer variable fPointer points to the variable x by holding its address 2000 as its value. In this case, if we want to print the value of the variable x it will display 6.5 but if we print the value of fPointer. It will display 2000. Note that, the value of fPointer may differ from 2000 depending upon memory availability.

Program

Consider the following program

```
//use of pointer in a program
#include <iostream.h>
#include<conio.h>
Int main( )
{
Float x=6.5 ;
Float *fPointer;
fPointer=&x;
cout < < "The address of x =" < < &x < < end1;
cout < < "The value of x = " < < x < < end1;
```

```
cout < < "The value of fPointer= " < < fPointer;
getch( )
return 0;
}
```

Output of the Program

The address of x = 2000

The value of x = 6.5

The value of fPointer= 2000

Explanation

The output of the program is shown along with the program. It is notable that this program may results some other output on another computer depending upon the availability of the memory.

1.1.4 DIFFERENCE OPERATOR (*)

Q5. What is dereference operator in C++? Explain with the help of program

Ans

Deference Operator

If we want to store the value of the variable through the pointer then we need a special type of operator called deference operator denoted by (*)

Program

Consider the following program to demonstrate the use of deference operator (*)

```
/* use of deference operator (*) in a program*/
```

```
#include<iostream.h>
#include<conio.h>
Int main ( )
{
Int n= 200;
Int *Pn;          //defines a pointer to n
Pn=&n;            //Pn stores the address of 'n'
Int valueN;
valueN=*Pn;
cout < < "The address of n=" < < &n < < end1;
cout < <"The value of n=" < < n < < end1;
cout < < "The value of Pn=" < < Pn < < end1;
cout < < "The value of (*Pn)=" (*Pn) < < end1;
cout < < "The value of valueN=" < < valueN;
getch( )
return 0;
}
```

Output of the program

The address of n=0x8fc5fff4

The value of n = 200

The value of Pn = 0x8fc5fff4

The value of (*Pn) = 200

The value of valueN=200

Explanation

In this example, the instructions at lines 10 and 14 make use of the deference operator (*) and thus access the actual values of the original variable n which is pointed out by the pointer Pn. In the pointers, the ampersand operator (&) is the reference operator and can be read as address of and (*) is the deference operator that can be read as value pointed by. These operators are complementary of each other and have opposite meanings. A variable referenced with & can be dereferencing with (*).

1.1.5 DECLARNG VARIABLES OF POINTER TYPES

Q6. Explain declaring variables of pointer types in C++.

Answer

Declaration of pointer

The declaration of pointer is simple and is similar to the declaration of a regular with a minor difference of the use of an asterisk (*) symbol between the data types and the variable name

General Format

Consider the following general format

Data type *nameVariable;

Here data type is the type of the value of that variable to which this pointer will point. This data type is not the type of the pointer itself but the type of the data the pointer points to

Examples

To understand it, consider the following examples of pointers declarations

Int * total Marks;

Char *Name;

Float *percentage;

Explanation of Examples

In the above examples three pointers totalMarks, Name and percentage are declared. Each one is intended to point to a different data type. All these pointer occupy space in memory. The first pointer points to an int the second to a char and the last one to a float

It Is notable that the asterisk (*) symbol used between the data type and the name of the variable is the indication for the pointer declaration and is not used for multiplication

Ways to place the Asterisk

There are three ways to place the asterisk. These are placing next to the data type, the variable name or in the middle. All these variations are shown in the above declaration of variables totalMarks, Name and percentage.

Ways to Place the Asterisk

There are three ways to place the asterisk. These are placing next to data type, the variable name or in the middle. All these variations are shown in the above declarations of variables totalMarks, Name and percentage

Pointing of Same Data Type Pointer

A pointer of type int can only points to a variable of type int and cannot to some other types of variable. Same is the case for other data types as well, but there is a special type of pointer named void pointer or generic pointer that can point to an objects of any data type, its declaration is similar to the declaration of normal pointers with the only difference of the use of void keyword as the pointer type

Declaration of void Pointer

Consider the following statement of declaration of the void pointer

```
Void *pointerVoid;   // pointerVoid is a void pointer
```

As a void pointer can point to objects of any data type, therefore, consider the following statements

```
Int x;
Float y;
Void *pointerVoid;
Pointervoid = &X;        // valid
pointerVoid = &Y;        // valid
```

In this segment of code the void pointer, pointerVoid stores the address of both integer variable X and floating point variable Y. the compiler decides at run time that the address of which type should be assigned to this pointer.

1.1.6 POINTER INITIALIZATION

Q7. What is meant by pointer initialization in C++? Explain in detail.

Answer

Pointer Initialization

Assigning values to pointers at declaration time is called pointe initialization

As we know that the values of pointers are the addresses of other variables, therefore, sometimes when we declare pointers we may want to explicitly specify to which variables they will point

Examples

Consider the following segment of code to understand the concept of pointer initialization

Float temperature:

Float *Ptemperature = &temperature:

Here, Ptemperature is a pointer variable to a floating point variable. As this pointer is created with the statement 'float * Ptemperature' is immediately the address of a float variable 'temperature' is assigned to it. The behavior of the above code is being equivalent to the following code.

Float Temperature:

Float * Ptemperature:

It should be considered that at the moment of declaring a pointer, the asterisk (*) indicates only that it is a pointer variable and not the deference operator

Program

Consider the following program to explain the concept of pointer initialization

```
/* pointer initialization program */?/

#include <iostream>

#include <Conio.h>

Int main ()

{

Float temperature:

Float *Ptemperature = &Temperature:

Cout<<"The address of Temperature is ="

<<Temperature<<End1:

Cout<<"the value of (*Ptemperature) is="
```

```
<<Ptemperature<<end1:

Getch():

Return 0:

}
```

Output of the Program

The address of temperature is = 0*8f98fff2

The value of Ptemperature is = 0*8f98fff2

Q8. What does null pointer mean? Explain by giving examples

Answer

Null Pointer

Sometimes we need to initialize a pointer to zero. Such pointers are called null pointers and they do not point to anything. Null pointers can be defined by assigning address 0 to them

Examples

Consider the following initialization to demonstrate null pointer

```
Int *Nptr:          // defines Null pointer

Nptr = 0:           // this assigns address 0 to Nptr
```

Key Points

- A pointer is a variable that points to or refers to another variable
- Some tasks in C++ are easier to do with pointers. Such as accessing the address of variables indirectly

- Like normal variables Pointer variables are also declared in programs before using them to declare a pointer variable, the data types of the variables is followed by (*operator).
- While using pointers in C++ programs address of operator denoted by ampersand & symbol is used to assign the address of variables to the pointer variable
- In C++ programs, pointers are initialized to addresses of other variables like the initialization of ordinary variables.