POINTER VARIABLES

**A pointer** is a [programming language](http://en.wikipedia.org/wiki/Programming_language) [data type](http://en.wikipedia.org/wiki/Data_type) whose value refers directly to (or "**points** to") another value stored elsewhere in the [computer memory](http://en.wikipedia.org/wiki/Computer_memory) using its [address](http://en.wikipedia.org/wiki/Memory_address).

A pointer variable (or pointer in short) is basically the same as the other variables, which can store a piece of data, but unlike normal variable pointers store a memory address.

DECLARATION OF POINTER VARIABLES

type \*ptr;

for example,

int \*ptr;

here ptr is a pointer variable which points to an integer variable, or which stores the address of an integer variable.

Pointers are extremely powerful because they allow you to access addresses and manipulate their contents. But they are also extremely complex to handle.

To get the value pointed to by a pointer, you need to use the de-referencing operator '\*'

For example to get the value of the variable that is pointed by ptr, we need to use the \*ptr in any expression.

REFERENCE VARIABLES

**A reference is an alias, or an alternate name to an existing variable.**

A reference is similar to a pointer. In many cases, a reference can be used as an alternative to pointer, in particular, for the function parameter, at the place of formal arguments.

The main use of references is found in function formal parameters to support pass-by-reference. In a reference variable is passed into a function, the function works on the original copy (instead of a clone copy in pass-by-value mechanism). Changes inside the function are reflected outside the function.

C/C++ use '&' to denote the **address-o**f operator in an expression. C++ assigns an additional meaning to '&' in declaration statements to declare a reference variable.

USING ‘&’ IN EXPRESSIONS

The meaning of symbol '&' is different in an expression and in a declaration. When it is used in an expression, '&' denotes the **address-of** operator, which returns the address of a variable, e.g., if ‘n’ is a variable of integer type, then &n returns the address of the variable n.

USING ‘&’ IN DECLARATION STATEMENTS

When '&' is used in a declaration (including function formal parameters), it is part of the type identifier and is used to declare a reference variable (or reference or alias or alternate name). It is used to provide another name, or another reference, or alias to an existing variable.

The syntax for creating a reference variable is as follow:

*type* &*new\_variable* = *existing\_variable*;

Remember that we need to initialize the reference variable during declaration.

EXAMPLE :

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<iostream.h>

int main()

{

int i=10; //i is a normal variable

int &j=i; //j is a reference variable

int \*k; //k is a pointer variable

k=&i; //k now points to i

//j is a reference of i

//k is a pointer to integer that stores the address of i

cout<<"Address of i = "<<(&i)<<endl;

cout<<"Address of j = "<<(&j)<<endl;

cout<<"Address of i (using k) = "<<k<<endl;

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

In the above program as we can see that the address of i is same as that of address of j, which means the reference variable accesses the same memory location as that of the original variable.

Also, printing the address of variable i using the pointer k(which points to the variable i) prints the same address as that was originally printed by using i. This program clearly illustrates the properties of the pointer variable and the reference variables.

PARAMETER PASSING BY USING PASS BY POINTER TECHNIQUE

Consider the following program to swap two number in which we create a function for swapping the two numbers.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*program in C++ illustrating the swapping of two numbers

using a function using pass by reference by pointer arguments

\*/

#include<iostream.h>

#include<conio.h>

void swap(int\*,int\*);

int main()

{

int a,b;

a=15;

b=20;

clrscr();

cout<<"variables before swapping : "<<endl;

cout<<"a = "<<a<<endl;

cout<<"b = "<<b<<endl;

swap(&a,&b);

cout<<"variables after swapping : "<<endl;

cout<<"a = "<<a<<endl;

cout<<"b = "<<b<<endl;

getch();

return 0;

}

void swap(int \*ptra,int \*ptrb)

{

int tmp;

tmp=\*ptra;

\*ptra=\*ptrb;

\*ptrb=tmp;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PARAMETER PASSING BY USING PASS BY REFERENCE TECHNIQUE

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

program in c++ to illustrate the swapping of two integers

using a function by implementing call by reference using

reference variables

\*/

#include<iostream.h>

#include<conio.h>

void swap(int &,int &);

int main()

{

int a,b;

clrscr();

a=15;

b=20;

cout<<"values of variables before swapping : "<<endl;

cout<<"a = "<<a<<endl;

cout<<"b = "<<b<<endl;

swap(a,b);

cout<<"values of variables after swapping : "<<endl;

cout<<"a = "<<a<<endl;

cout<<"b = "<<b<<endl;

getch();

return 0;

}

void swap(int &refa, int &refb)

{

int tmp;

tmp=refa;

refa=refb;

refb=tmp;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/