**Experiment #12**

**OBJECTIVE**

**To become familiar with the use of the pointers.**

**Pointers**

A pointer provides the way of accessing a variable (or a more complex kind of data, such as an array) without referring to the variable directly. The mechanism used for this is the address of the variable.

**Uses of the Pointer variable**

Some reasons to use the pointers are

1. To return more than one value from the function.
2. To pass array and function more conveniently frm one function to another.
3. To manipulate the arrays more easily by moving the pointer to them.
4. To create complex data structure, such as linked lists and binary tress.

Pointer are declared in the same manner as we are declaring the simple variables, but the name of the variable must be followed by the \* sign.

e.g. int \* a;

**Example (1)**

#include<iostream>

using namespace std;

#include<conio.h>

int main()

{

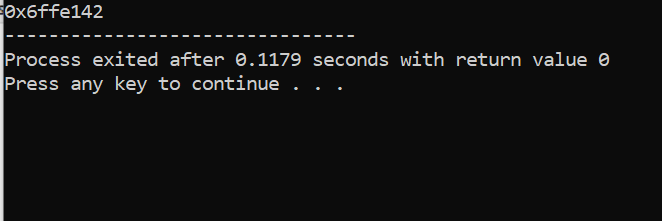
int a=2, \*b;

b=&a;

cout<<b; // displays the address of the variable a.

cout<<\*b; // displays the value of the variable a .

}



**Example (2)**

// program for returning more than one value from the function

#include<iostream>

using namespace std;

#include<conio.h>

void function(int \*a, int \*b)

{

\*a=100;

\*b=200;

}

main( )

{

int y,z;

function(&y,&z);

cout<<y<<z;

getch();

}



**Dynamic Allocation operators**

C++ provides two dynamic allocation operators: new and delete. These operators are used to allocate and free memory at run time. The new operator allocates memory and returns a pointer to the start of it the delete operator frees memory previously allocated using new. The general from of new and delete are shown here:

p\_var =new type;

delete p\_var;

Here p\_var is pointer variable that receives a pointer to memory that is large large enough to hold an item of type type.

**Example**

#include<iostream.h>  
#include<conio.h>  
main()

{  
int \*ptr;

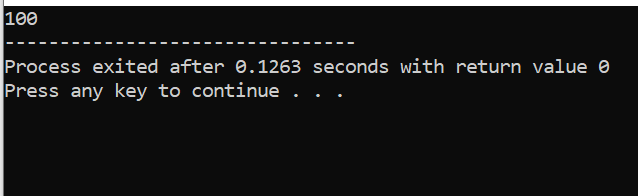
ptr = new int ;

\*ptr=100;

cout<<\*ptr;

delete ptr;

}



**Lab Tasks:**

**Q1** Dispaly the output of the following program:

#include<iostream.h>  
main()  
{  
int a=10,b=20,\*c,\*d;

c=&a;

d=&b;

cout<<a<<b<<endl;

cout<<c<<d<<endl;

cout<<\*c<<\*d<<endl;}

**Q2** Distinguish between near and far pointer?

**Q3** What mathematical operation that can be performed on the pointers?

**Q 4** int I,j;

int \*p = &i, \*q;

p= &j;

What will be

i) &\*&i ii) \*&\*&j iii) \*\*&p

**Q 5** If

int arr[] = { 2, 39, 14,36};

int \*ptr = arr;

Develop a code to explain the difference between

++\*ptr , \*ptr++, (\*ptr)++, ++ptr, ++\*ptr++

**QUESTION NUMBER : 01:**

Dispaly the output of the following program:

**PROGRAM:**

#include<iostream>

using namespace std;

main()

{

int a=10,b=20,\*c,\*d;

c=&a;

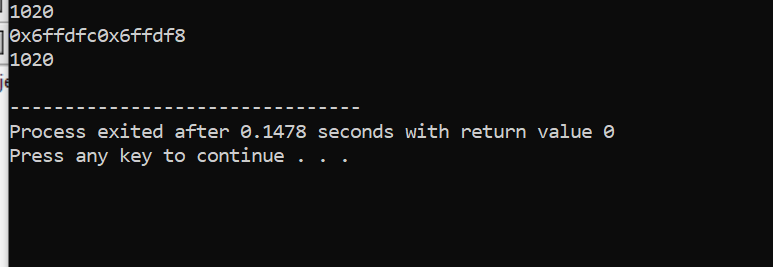
d=&b;

cout<<a<<b<<endl;

cout<<c<<d<<endl;

cout<<\*c<<\*d<<endl;}

**OUTPUT:**



**QUESTION NUMBER : 02:**

Distinguish between near and far pointer?

**ANSWER:**

**Near pointer** is used to store 16 bit addresses means within current segment on a 16 bit machine. The limitation is that we can only access 64kb of data at a time.

**A far pointer** is typically 32 bit that can access memory outside current segment.  To use this, compiler allocates a segment register to store segment address, then another register to store offset within current segment.

**QUESTION NUMBER : 03:**

What mathematical operation that can be performed on the pointers?

**ANSWER:**

You can perform a limited number of arithmetic operations on pointers. These operations are: **Increment and decrement**. Addition and subtraction.

**QUESTION NUMBER : 04:**

Dispaly the output of the following program:

**PROGRAM:**

int I,j;

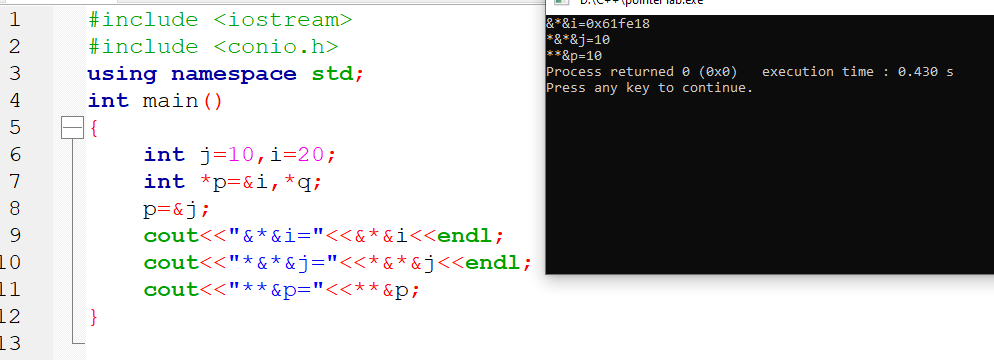
int \*p = &i, \*q;

p= &j;

What will be

i) &\*&i ii) \*&\*&j iii) \*\*&p

**OUTPUT:**



**QUESTION NUMBER : 05:**

If

int arr[] = { 2, 39, 14,36};

int \*ptr = arr;

Develop a code to explain the difference between

++\*ptr , \*ptr++, (\*ptr)++, ++ptr, ++\*ptr++

**PROGRAM:**

