OBJECT ORIENTED PROGRAMMING IN C++

**Index**

|  |  |  |
| --- | --- | --- |
| SR. No. | DATE | TITLE |
| 1. |  | Odd or Even |
| 2. |  | Array Element Manipulation |
| 3. |  | Character Printing |
| 4. |  | Calculator |
| 5. |  | Function Overloading and default parameters |
| 6. |  | Recursion |
| 7. |  | Inline Functions |
| 8. |  | Static Data Members and function |
| 9. |  | Constructor Overloading |
| 10. |  | Operator Overloading |
| 11. |  | Inheritance |
| 12. |  | Friend Function |
| 13. |  | File Operations |
|  |  |  |
|  |  |  |

Basic C++ Programs

Experiment -1

**Aim:** To illustrate the Basic Programs in C++.

**Algorithm:**

* Declare a variable.
* Get input for that variable.
* If exactly divisible by 2 print even.
* Else print odd.
* End.

**Program:**

Even Or Odd

#include<iostream>

using namespace std;

int main(){

int a;

cout<<"Enter a number";

cin>>a;

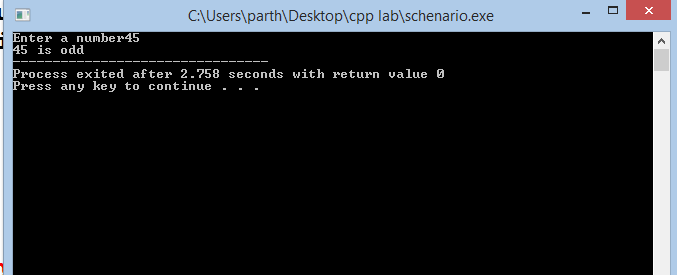
if(a%2 == 0){

cout<<a<<" is Even";}

else{

cout<<a<<" is odd";}

}



**Result:** The Basic C++ programs has been successfully executed.

Experiment -2

**Aim:** To illustrate the Basic Programs in C++.

**Algorithm:**

* Declare an array of required index.
* Get the values of different index using a loop of 1 increment
* To print the even position run loop i=0 to n increment of 2
* To print the odd position run loop i=1 to n increment of 2
* End

**Program:**

Array Element Manipulation

#include<iostream>

using namespace std;

int main(){

int n,i;

int a[10];

cout<<"enter the size of array";

std::cin >> n;

for(i=0;i<n;i++){

cout<<endl;

cin>>a[i];

}

cout<<"elements at the even position are:\n";

for(i=0;i<n;i=i+2){

cout<<a[i];

cout<<endl;

}

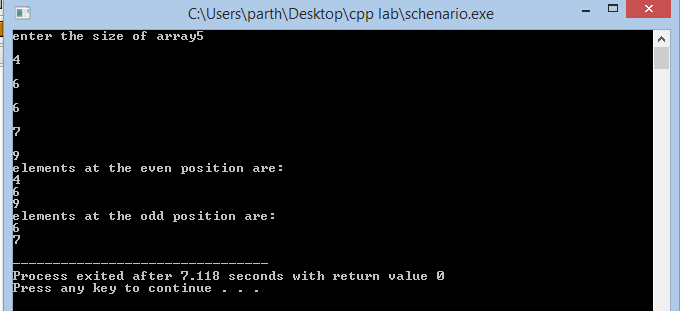
cout<<"elements at the odd position are:\n";

for(i=1;i<n;i+=2){

cout<<a[i];

cout<<endl; }

return 0;}



**Result:** The Basic C++ programs has been successfully executed.

Experiment -3

**Aim:** To illustrate the Basic Programs in C++.

**Algorithm:**

* Declare a character.
* Get the value of the character.
* Declare a integer to get no of times.
* Pass the character and times to a print function.
* Print times time using a loop.
* End.

**Program:**

Character Printing

#include<iostream>

using namespace std;

void printit(char,int);

int main(){

char a;

cout<<"enter a charcter";

cin>>a;

int t;

cout<<"enter no of time\n";

cin>>t;

printit(a,t);}

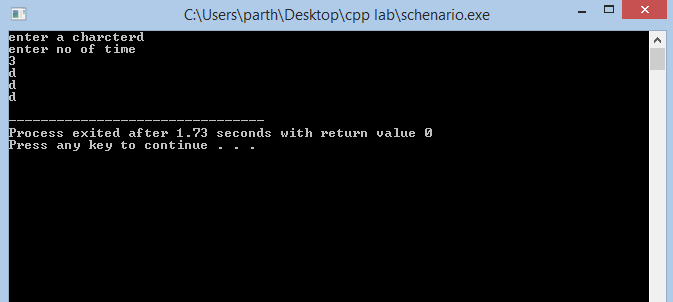
void printit(char c,int times){

int i;

for(i=0;i<times;i++){

cout<<c;

cout<<endl;}}



**Result:** The Basic C++ programs has been successfully executed.

Experiment -4

**Aim:** To illustrate the Basic Programs in C++.

**Algorithm:**

* Declare two variables.
* Take the input.
* Take choice option based on operation in a new variable.
* Switch the variable using switch case and carry out the operations
* End

**Program:**

Calculator

#include<iostream>

using namespace std;

int main(){int a,b,c,s;

cout<<"enter a number\n";

cin>>a;

cout<<"enter second number\n";

cin>>b;

cout<<"enter your operation:\n1.Add\n2.Subtract\n3.Multiply\n4.Divide";

cin>>s;

switch(s){

case 1:

cout<<a+b;

break;

case 2:

cout<<a-b;

break;

case 3:

cout<<a\*b;

break;

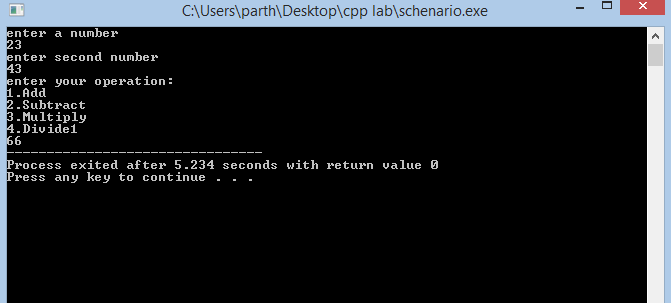
case 4:

cout<<a/b;

break;

default:

break;}}



**Result:** The Basic C++ programs has been successfully executed.

EXPERIMENT-5

Default Parameters

**Aim:** To illustrate the Default Parameters Program in C++.

**Algorithm:**

* Define the function having default parameters of height =10
* Call the function with l and b alone
* Now, call the function with l,b, and h
* Print the volume
* End

**Program:**

#include<iostream>

using namespace std;

void vol(int a,int b,int c=10){

cout<<"volume is:"<<a\*b\*c;

}

int main(){

int l,b,h;

cout<<"enter l&b";

cin>>l>>b;

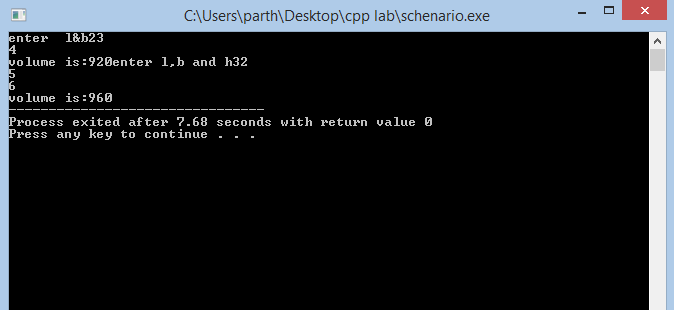
vol(l,b);

cout<<"enter l,b and h";

cin>>l>>b>>h;

vol(l,b,h);

return 0;}



**Result:** The program for the Default parameters has been successfully executed.

EXPERIMENT-6

Inline Function

**Aim:** To illustrate the inline function Program in C++.

**Algorithm:**

* Take the input from user as integer
* Define inline function which return the 0 or 1
* If it returns 0 , the no is even
* Else the number is odd
* End

**Program:**

#include <iostream>

using namespace std;

inline int oddeven(int a){

return ((a%2== 0)?0:1);}

int main(){int n,r;

cout<<"enter a number\n";

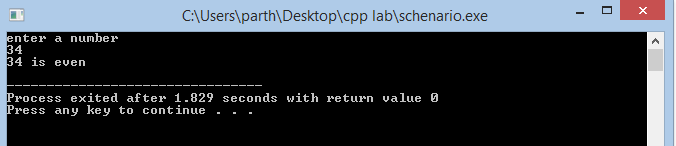
cin>>n;

r=oddeven(n);

if(r == 0){

cout<<n<<" is even\n";}

else{cout<<n<<" is odd\n";}}

**Result:** The program for the inline function has been successfully executed.

EXPERIMENT-7

Recursion in c++

**Aim:** To illustrate the recursion Program in C++.

**Algorithm:**

* Read the number as integer
* Call the function with (2,range) for even numbers
* Call the function with (1,range) for odd numbers
* Print the numbers
* End

**Program:**

#include <iostream>

using namespace std;

void printoddeven(int,int);

void printoddeven(int c,int range){

if(c>range){

return;}

cout<<c<<"\t";

printoddeven(c+2,range);}

int main(){

int a;

cout<<"Enter a number\n";

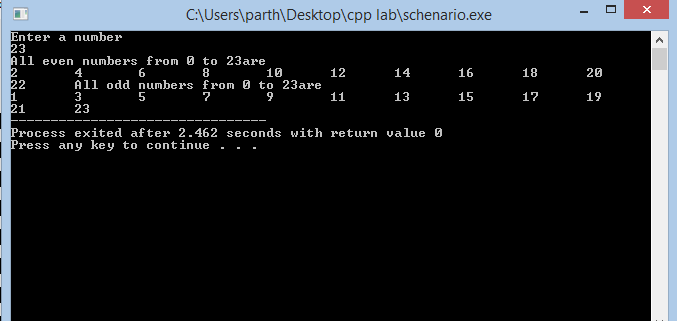
cin>>a;

cout<<"All even numbers from 0 to "<<a<<"are\n";

printoddeven(2,a);

cout<<"All odd numbers from 0 to "<<a<<"are\n";

printoddeven(1,a);}

**Result:** The program for the recursion has been successfully executed.

EXPERIMENT-8

Static data member and static member function

**Aim:** To illustrate the static data member and static member function Program in C++.

**Algorithm:**

* Start the program
* Declare the class name box
* Define its constructor
* Define the member function called volume
* The objectcount and getcount both will be static
* end

**Program:**

#include <iostream>

using namespace std;

class Box

{public:

static int objectCount;

Box(double l=2.0, double b=2.0, double h=2.0)

{cout <<"Constructor called." << endl;

length = l;

breadth = b;

height = h;

objectCount++; }

double Volume(){return length \* breadth \* height;}

static int getCount(){

return objectCount;}

private:

double length; // Length of a box

double breadth; // Breadth of a box

double height; // Height of a box};

int Box::objectCount = 0;

int main(void){ cout << "Inital Stage Count: " << Box::getCount() << endl;

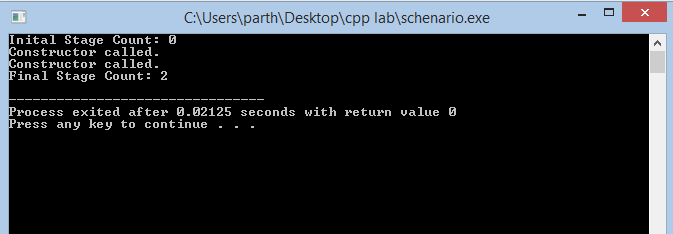
Box Box1(3.3, 1.2, 1.5); // Declare box1

Box Box2(8.5, 6.0, 2.0); // Declare box2

cout << "Final Stage Count: " << Box::getCount() << endl;

return 0;

}



**Result:** The program for the static data member and static member function has been successfully executed.

**EXPERIMENT-9**

**Constructor And Destructor**

**Aim:** To illustrate the constructor and Destructor in C++.

**Algorithm:**

* Construct a class named Complex
* Declare re\_ and im\_ as private data member
* Declare Constructor and Destructor of given class in public
* Declare norm()and print() functions
* Call the Constructor from main function.
* End

**Program:**

#include<iostream>

#include<math.h>

using namespace std;

class Complex

{

private:

double re\_, im\_;

public:

Complex(): re\_(0.0), im\_(0.0) // Default parameter of the constructor

{cout <<"Constructor: (" << re\_ << ", " << im\_ << ")" << endl; }

~Complex()//destru­­ctor call

{ cout << "Destructor: (" << re\_ << ", " << im\_ << ")" << endl;}

double norm()

{ return sqrt(re\_\*re\_ + im\_\*im\_);}

void print() {

cout << "|" << re\_ << "+i" << im\_ << "| = " << norm() << endl; }

void set(double re, double im) {

re\_ = re; im\_ = im; }

};

int main()

{Complex c; // Default constructor

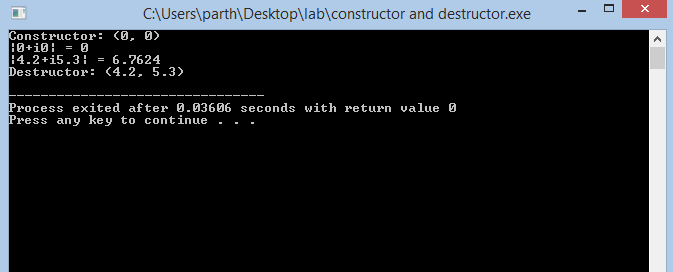
c.print(); // Print initial values

c.set(4.2, 5.3); // Set components

c.print();

return 0;

} // Destructor



**Result:** The program for Constructor and Destructor has been successfully executed

**EXPERIMENT-10**

**Operator Overloading**

**Aim:** To illustrate the operator overloading in C++.

**Algorithm:**

* Construct a class named Complex
* Declare re\_ and im\_ as private data member
* Declare Constructor and Destructor of given class in public
* Declare display to print value
* End

**Program:**

#include <iostream>

#include <cstring>

#include<stdlib.h>

using namespace std;

struct String{

char \*str; };

String operator+(const String &s1,const String &s2){

String s3;

s3.str=(char\*)malloc(sizeof(strlen(s1.str)+strlen(s2.str)+1));

strcpy(s3.str,s1.str);

strcat(s3.str,s2.str);

return s3; }

int main(){

String fname,lname,name;

fname.str=strdup("Parth ");

lname.str=strdup("Patel ");

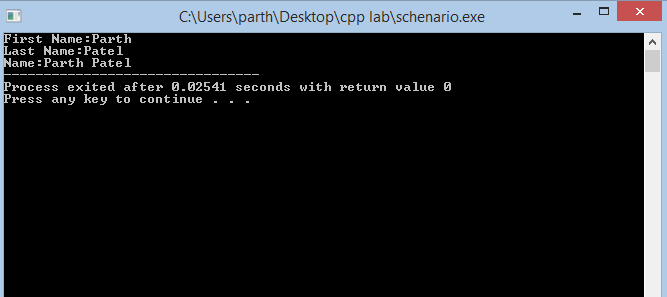
name=fname+lname;//operator overloading for +

cout<<"First Name:"<<fname.str<<"\n";

cout<<"Last Name:"<<lname.str<<"\n";

cout<<"Name:"<<name.str;

return 0;}



**Result:** The program for Binary Operator Overloading has been successfully executed.

EXPERIMENT-11

Inheritance in C++

**Aim:** To illustrate the Inheritance in C++.

Algorithm:

* Construct a class named Base
* Declare Constructor and Destructor of given class in public
* Make a derived class and create its constructor and destructor.
* Print the conformation massage
* End

Program:

Simple Inheritance:

#include <iostream>

using namespace std;

class base

{public:

base()

{cout << "Constructing base\n";}

~base()

{cout << "Destructing base\n";

}};

class derived: public base

{public:

derived()

{cout << "Constructing derived\n";}

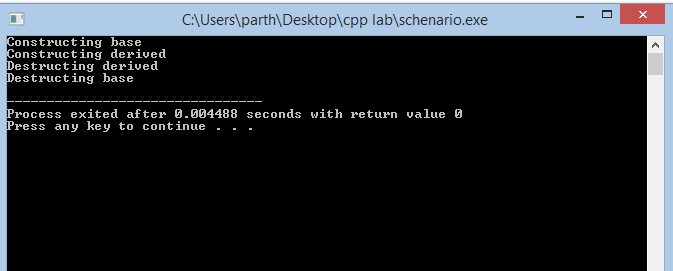
~derived()

{cout << "Destructing derived\n";

}};

int main()

{derived ob;}



Multiple Inheritance

#include <iostream>

using namespace std;

class base

{public:

base()

{cout << "Constructing base\n";

}

~base()

{cout << "Destructing base\n";}};

class derived1 : public base

{public:

derived1()

{cout << "Constructing derived1\n";

}

~derived1()

{cout << "Destructing derived1\n";

}};

class derived2: public derived1

{public:

derived2()

{cout << "Constructing derived2\n";

}

~derived2()

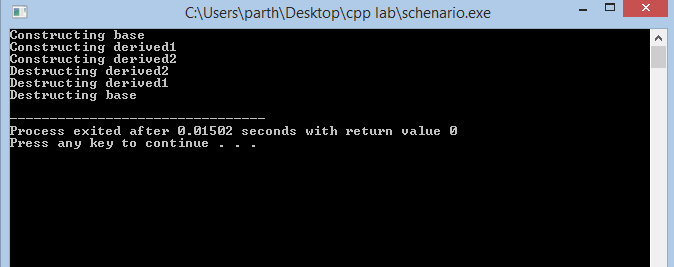
{cout << "Destructing derived2\n";

}};

int main()

{derived2 ob;

}



Multilevel Inheritance

#include <iostream>

using namespace std;

class base1

{public:

base1()

{cout << "Constructing base1\n";

}

~base1()

{cout << "Destructing base1\n";}};

class base2

{public:

base2()

{cout << "Constructing base2\n";

}

~base2()

{cout << "Destructing base2\n";}};

class derived: public base1, public base2

{public:

derived()

{cout << "Constructing derived\n";

}

~derived()

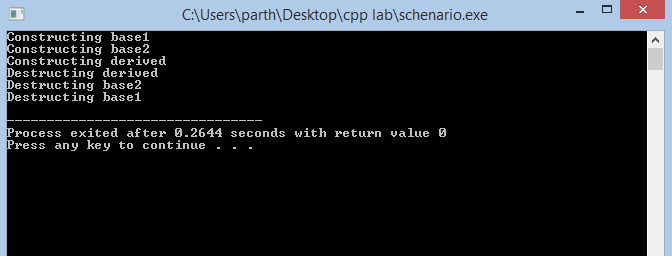
{cout << "Destructing derived\n";

}};

int main()

{derived ob;

}



Hybrid Inheritance

#include<iostream>

#include<conio.h>

using namespace std;

int a,b,c,d,e;

class A

{public:

void getab() {

cout<<"\n ENter a and b value:";

cin>>a>>b; }};

class B:public A {

public:

void getc() {

cout<<"Enter c value:";

cin>>c; }};

class C {protected:

public:

void getd() {

cout<<"Enter d value:";

cin>>d; }};

class D:public B,public C {

protected:

public:

void result() {

getab(); getc();

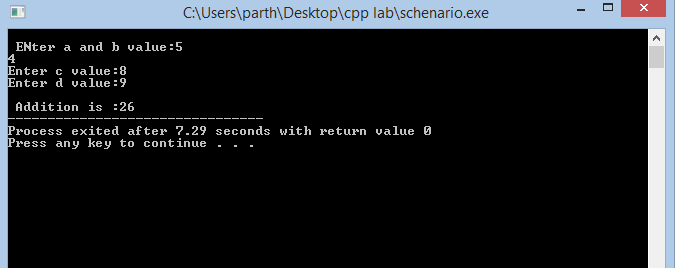
getd(); e=a+b+c+d;

cout<<"\n Addition is :"<<e; }};

int main() {

D d1;

d1.result();}



**Result:** The programs for Inheritance has been successfully executed.

EXPERIMENT-12

Friend Function

**Aim:** To illustrate the Friend Function in C++.

**Algorithm:**

* Construct two classes called Rectangle and Square
* Declare Friend function named display in Rectangle
* Declare friend display in the square class also.
* for finding the area.
* End

**Program:**

#include <iostream>

using namespace std;

class Square; // forward declaration

class Rectangle {

int width, height;

public:

Rectangle(int w = 1, int h = 1):width(w),height(h){}

friend void display(Rectangle &, Square &);};

class Square {

int side;

public:

Square(int s = 1):side(s){}

friend void display(Rectangle &, Square &);};

void display(Rectangle &r, Square &s) {

cout << "Rectangle: " << r.width \* r.height << endl;

cout << "Square: " << s.side \* s.side << endl;}

int main () {

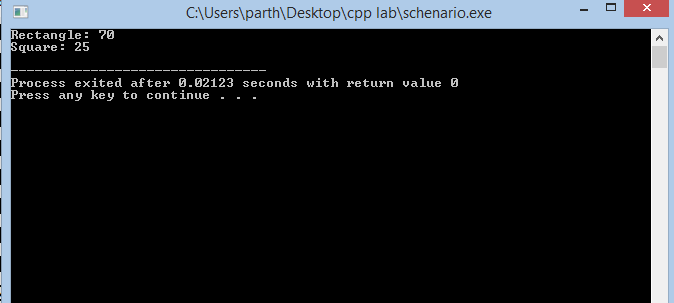
Rectangle rec(7,10);

Square sq(5);

display(rec,sq);

return 0;

}



**Result:** The program for Friend Function has been successfully executed.

EXPERIMENT-13

File Handling

**Aim:** To illustrate the File Handling in C++.

**Algorithm:**

* Create a file called file.txt
* Write two lines into the file
* Close the file file.txt
* Read the line from user
* Write that data into the file
* Print the data of the file
* End

**Program:**

#include<iostream>

#include<fstream>

using namespace std;

int main() {

ofstream ofile; // declaring an object of class ofstream

ofile.open("file.txt"); // open "file.txt" for writing data

ofile << "This is a line " << endl;

ofile << "This is second line" << endl;

cout << "Data written to file" << endl;

ofile.close(); // close the file

char line[100];

fstream file; // declare an object of fstream class

file.open("file.txt", ios :: out | ios :: app); // open file in append mode

if (file.fail()) { // check if file is opened successfully

// file opening failed

cout << "Error Opening file ... " << endl;

}

else {

cout << "Enter a line : ";

cin.getline(line, 100);

file << line << endl; // Append the line to the file

cout << "Line written into the file" << endl; }

char data[100]; // buffer to store a line read from file

ifstream ifile; // declaring an object of class ifstream

ifile.open("file.txt"); // open "file.txt" for reading

cout << "Reading data from a file :-" << endl << endl;

while (!ifile.eof()) { // while the end of file [ eof() ] is not reached

ifile.getline(data, 100); // read a line from file

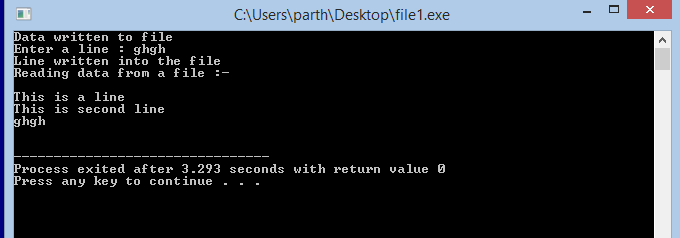
cout << data << endl; // print the file to console

}

ifile.close(); // close the file

return 0;

}



**Result:** The program for File Handling has been successfully executed.