**EXPERIMENT-1**

**AIM:** Write a case study of lexical analyzer using Lex.

1. **Introduction to Lexical Analyzer:**

Lexical analysis is an important phase of compilation operated at beginning. The language processor that are written in the form of sentences, it takes the modified code from them. It removes the white code and source code by that it breaks the syntax into different tokens.

It generates the error when it found some invalid tokens. It works closely with the syntax analyzer. It reads streams of character from the source code, checks for legal tokens, and passes the data to the syntax analyzer whenever it demands for it. A **lexer** can be class implementation, having a constructor which takes a string as an input parameter (representing the source code to perform lexical analysis on). It shows a method to recognize and then it returns the next token in the input. The aim of the lexical analyzer is to read the input characters of the source program, group them into lexemes, and produce as output a sequence of tokens for each lexeme in the source program. The lexical analysis phase of a compiler performs the task of reading the source program as a file of character and dividing up into the tokens. It usually implements as a subroutine or co-routine of parser. It is front end of a compiler.

1. **Why we need Lexical Analyzer:**

Its main task is to read input character and produce as output a sequence of tokens that parser uses for syntax analysis.

* It separates the input source code into tokens.
* Removing out the unnecessary white spaces from the source code.
* Removing the comments.
* Keeping tracks of line numbers.
* Preprocessing of macros.
* The attribute influences the translation of tokens.

1. **Literature Survey:**An exploration on lexical analysis is published by IEEE. It was published in[2016International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)](https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7731602).The word lexical in lexical analysis, its meaning is extracted from the word “lexeme”. A lexical analyzer is used in various applications like text editors, information retrieval system, pattern recognition programs and language compilers. In this paper to discussion is about language compilers. Lexical analyzer being the first phase of the compilation process it deals with the processing of input language. Discussion is also extended to multi-core environment. Input to the lexical analyzer is the source program itself. It scans the source program character by character and produces meaningful sequence called Tokens. Due to its scanning property this phase is also called scanner. Lexical analyzer stores the tokens in symbol table and as well as sends it to next phase as an input.
2. **Advantages:**

* This technique used to implement lexical analyzers can also be applied to other areas such as query languages and information retrieval system.
* It looked for patterns corresponding to imperfections in the string of line segments.
* It can utilize the best known pattern-matching algorithms and thereby create efficient lexical analyzers for people who are not experts in pattern-matching technique.
* Scanners perform pattern matching process.

1. **Disadvantages:**

* There are several reasons for separating the analysis phase of compiling into lexical analysis and parsing.
* Simpler design of the parser.
* Unnecessary tokens can be removed.
* The lexical analysis phase is the most time consuming phase in compilation.
* Portability of the compiler is isolated.

1. **Conclusion:**

It breaks the syntax into tokens. It removes the white space and source code. It takes the modified code for language processor that are written in sentences. The attribute so lexical analyzer influences the translation of tokens. It also removes the comment in the compiler design. Correlate the error messages from the compiler. To specify the lexical analyzer, the lex is used. It is also time consuming in compiling the program. It decreases the portability of the compiler, especially when we are designing it.



**EXPERIMENT 2**

**AIM:** Write a case study of Virtual Machines

1. **Introduction:**

A Virtual Machine is a software or a computer file that provides an application environment to any guests to meet some specific requirement. It enables morphing of a system on another system, where the first system is Virtual image that is the need of the guests whereas the latter one is the dedicated physical system also called as host. There can be multiple guests on a single host.

A software called Hypervisor provides the virtual environment, looks and controls the complete emulation of the Virtual Machine on the host. A Virtual Machine have the control over host system and provides.

1. **Why to Use Virtual Machine**:

* A Virtual Machine is widely used to run different Operating System on a single physical machine. It enables the user to get hands on different software environment without having different physical resources.
* A Virtual Machine is also helpful in checking of whether the program, file or software we are using is damaged or incompatible or not.
* In the recent world of research and development, Virtual Machines provide an edge to the researchers, computer scientists, to test different things and they can make mistakes even huge ones too. They make the research flexible.
* Virtual Machines enable a user to test on different stuff without having any prerequisites for technological requirements mostly on the hardware requirements.
* Virtual Machines are used to reconstruct or clone different machines for specific requirement.

1. **Literature Survey:**

Survey Paper Topic – A Survey on Virtual Machine Security

Author Name – Jenni Susan Reuben, Helsinki University of Technology

Summary – This paper is concerned about the security problems in virtualization technologies. Main focus is on some open security vulnerabilities that virtual machine produce in the surroundings. This paper does not provide a great solution to all the security threats but provides an idea to avoid these threats. Paper includes both the benefits and drawbacks in terms of security that virtualization provides. Resource sharing and Isolation are the main two benefits that VM provides. It also depicts the architecture of Virtual Machine and explains the terms like full and para virtualization along with application, hardware support, resource virtualization. Security vulnerabilities like VM monitoring from the host, Denial of Service etc... are explained and also the solution to these problems are provided. The overall paper has presented security flaws in VM.

1. **Advantages:**

1.Familiar interfaces.

2.Isolation – each OS can run individually with its own virtual resources.

3.High Availability – if one VM server fails then you can easily grab the data from another VM server.

4.Scalability – you can easily connect or disconnect resources.

5.Backup – you can always have backup using VMDK tool.

6.Reduced cost – VM supports hardware sharing thus saving cost and also you can run multiple OS on a single system.

1. **Disadvantages:**

1.Difficult Access – you cannot access the hardware directly.

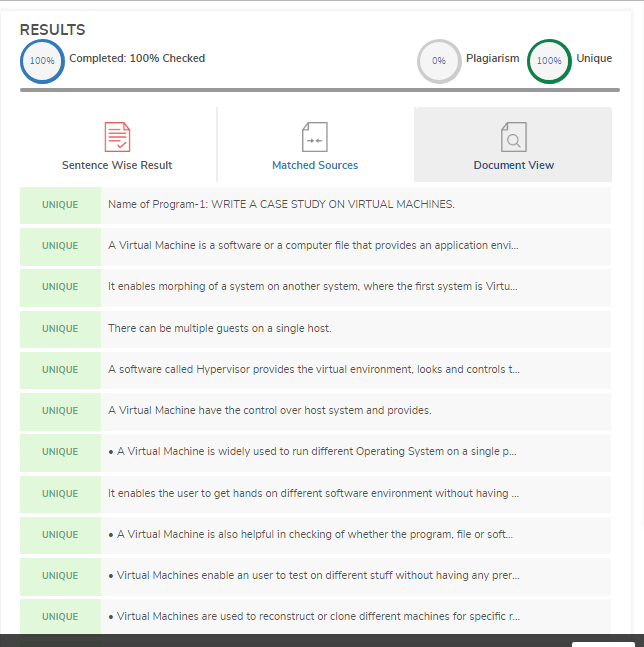
2.More RAM utilization – since each VM occupies separate memory area for the same copies.

3.More disk space utilization – since all the files for each OS is installed on each VM.

4.Less Efficient – A VM is less efficient as compared to the actual machine when it tries to indirectly access the host Hard Drive.

1. **Conclusion:**

Virtual Machine which can also be said as virtualization is a very mandatory need for today’s technology since it provides a whole lot of benefits which a system can easily afford without risking any of its parts. Although Virtual Machines have some drawbacks regarding security but still have huge benefits. Virtualization is a powerful solution to reduce the operational costs in today’s computing but if done wrong it becomes a threat to the environment.



EXPERIMENT-3

**AIM:** Given a square matrix of size N×N, Design a program to calculate the absolute difference between the sums of its diagonals.

* + - 1. **Source Code:**

#include <bits/stdc++.h>

using namespace std;

const int MAX = 100;

void DiagonalDiff(int mat[][MAX], int n)

{

int principal = 0, secondary = 0;

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

for (int j = 0; j < n; j++) {

if (i == j) {

principal += mat[i][j];

if ((i + j) == (n - 1))

secondary += mat[i][j];

}

}

}

cout<<"Difference = "<<(principal - secondary)<<endl;

}

int main()

{

int a[][MAX] = { { 5, 2, 3, 4 }, { 5, 6, 7, 8 },

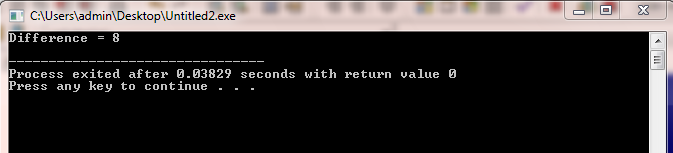
{ 1, 2, 3, 4 }, { 1, 6, 7, 8 } };

DiagonalDiff(a, 4);

return 0;

}

* + - 1. **Output:**

******

EXPERIMENT-4

**AIM:** Write a program that reads three dimensional array and display it.

**Source Code:**

#include<iostream>

using namespace std;

int main()

{

cout<<"Enter a 2x2x2 multidimensional array:"<<endl;

int a[2][2][2];

for(int i=0;i<2;i++)

{for(int j=0;j<2;j++)

{for(int k=0;k<2;k++)

{cin>>a[i][j][k]; }}}

cout<<"The array is:"<<endl;

for(int i=0;i<2;i++)

{

for(int j=0;j<2;j++)

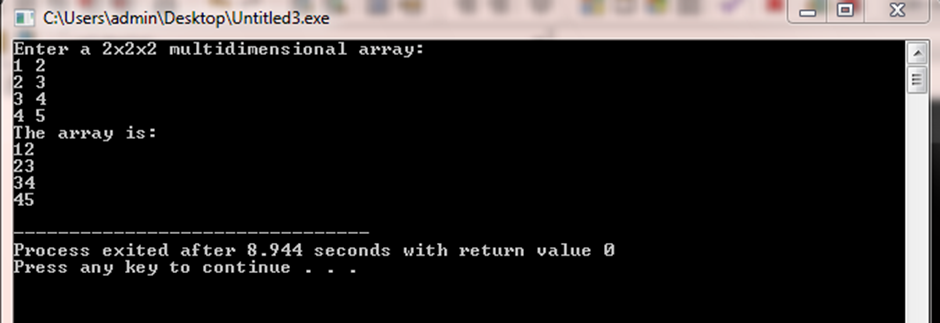
{for(int k=0;k<2;k++)

{cout<<a[i][j][k]; }

cout<<endl;

}}}

**Output:**



EXPERIMENT-5

**AIM:** Design a webpage in php.

**1. Source Code:**

<html>

<head>

<title> Login </title>

<style>

body {background-image:url(bg.jpg);background-position: center;background-repeat: no-repeat;background-size: cover;height: 100%;}

h1 {color:Blue;}

.login {background:white;width:300px;height:350px;border-radius:25px;}

</style>

</head>

<body>

<center>

<?php

$un = $\_POST['un'];

$pass = $\_POST['pass'];

if(isset($\_POST["submit"]))

{

if($un == 'Admin' && $pass == '123')

{

echo "<br/><br/><div class='login'><br/><br/><h1><font color='Green'>Access Granted</font><br/><br/></h1><h2> Welcome Admin </h2></div>";

exit;

}

else

{

echo "<br/><br/><div class='login'><br/><br/><h1><font color='Red'>Unauthorized Access</font><br/><br/></div>";

exit;

}

}

?>

<br/><br/><div class="login">

<br/><br/><h1> Login </h1><br/><br/>

<form method="POST" action="Login.php">

<input type="text" name="un" placeholder="Enter Username" /><br/><br/>

<input type="password" name="pass" placeholder="Enter Password" /><br/><br/>

<input type="submit" name="submit" value="Login" />

</form>

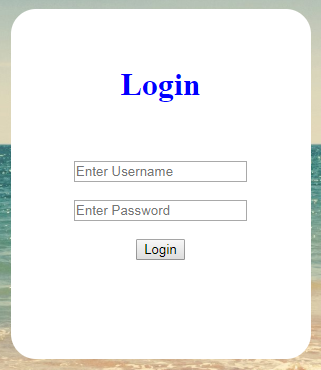
</div>

</center>

</body>

</html>

**2. Output:**



EXPERIMENT-6

**AIM:** write a program in java to implement exception handling.

**1. Source Code:**

import java.util.Scanner;

class Division {

public static void main(String[] args) {

int a, b, result;

Scanner input = new Scanner(System.in);

System.out.println("Input two integers");

a = input.nextInt();

b = input.nextInt();

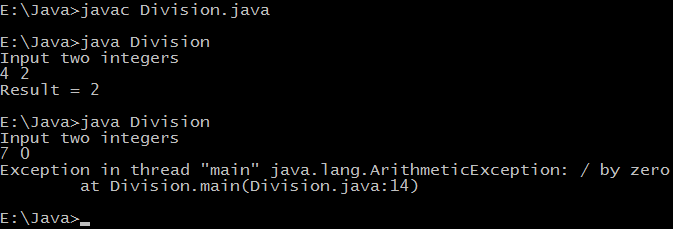
result = a / b;

System.out.println("Result = " + result);

}

}

**2. Output:**



EXPERIMENT-7

**AIM:** WAP to count vowels and consonants in a string using pointer.

**Source Code:**

#include<iostream>

using namespace std;

int main()

{

int cnt=0,cnt1=0;

char a[1000];

cin>>a;

char\*ptr;

ptr=a;

int i=0;

while(\*(ptr+i)!='\0')

{

if(\*(ptr+i)=='a'||\*(ptr+i)=='e'||\*(ptr+i)=='i'||\*(ptr+i)=='o'||\*(ptr+i)=='u')

cnt++;

else cnt1++;

i++;

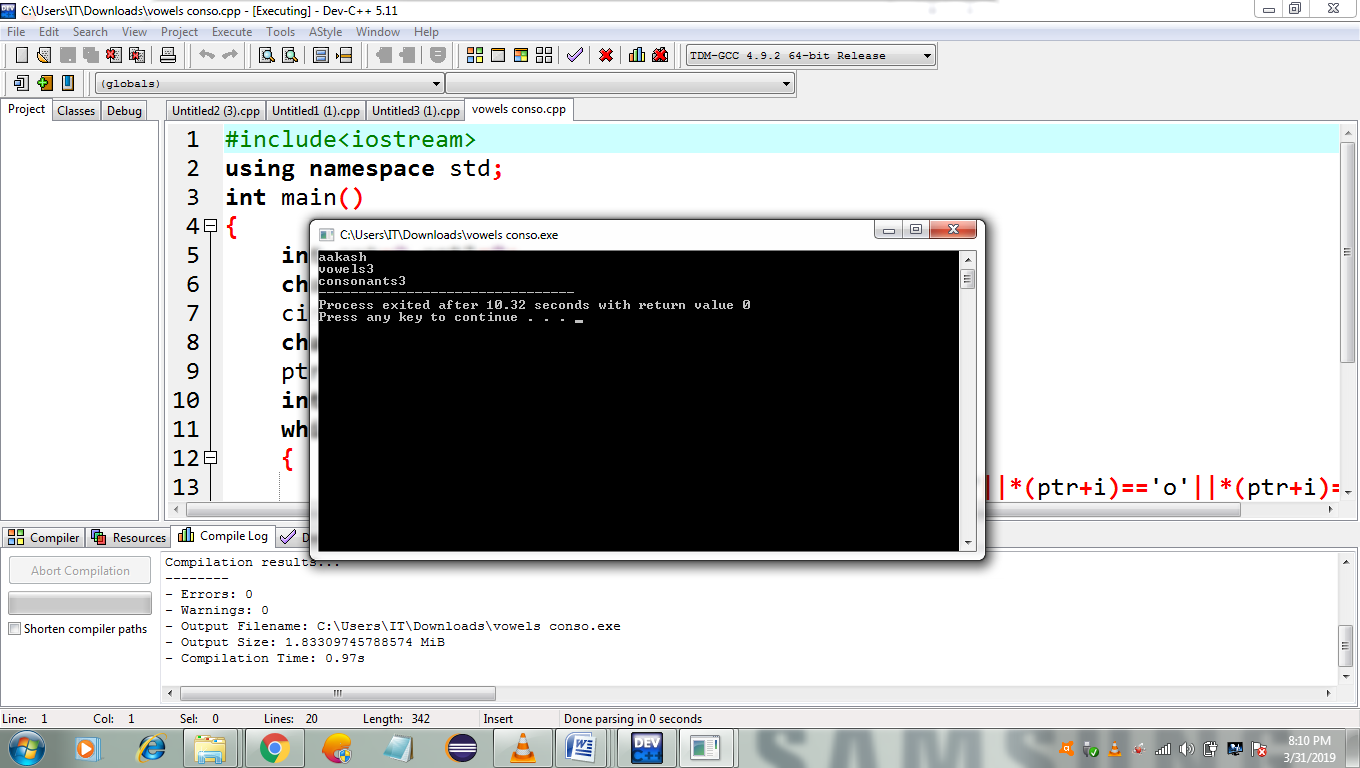
}

cout<<"vowels"<<cnt<<endl;

cout<<"consonants"<<cnt1;

}

**OUTPUT:**

****

EXPERIMENT-8

**AIM:** WAP to find the sum if array elements using dynamic memory location.

**Source Code:**

package array.java;

importjava.util.Scanner;

classSumDemo{

public static void main(String args[]){

Scanner scanner = new Scanner(System.in);

int[] array = new int[10];

int sum = 0;

System.out.println("Enter the elements:");

for (inti=0; i<10; i++)

{

array[i] = scanner.nextInt();

}

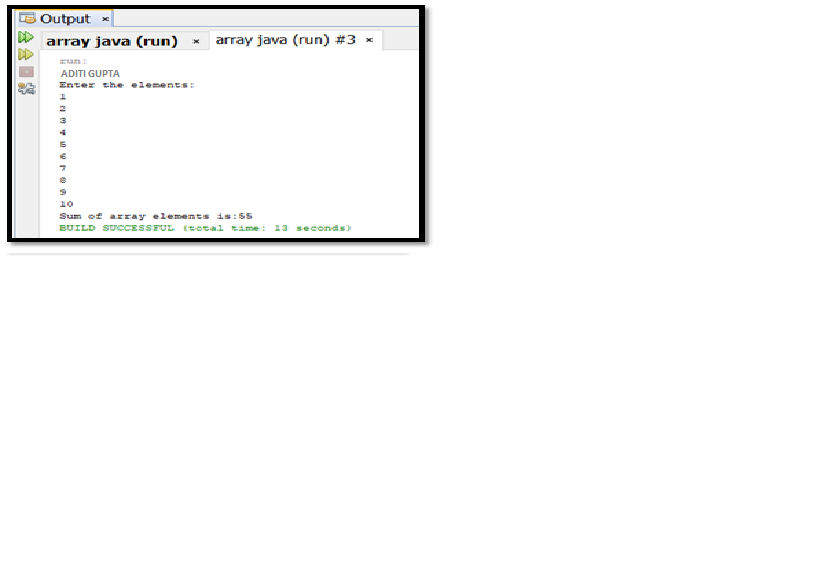
for(intnum : array) {

sum = sum+num;

}System.out.println("Sum of array elements is:"+sum);

}}

**Output:**

****

EXPERIMENT-9

**AIM:** WAP to implement concepts of encapsulation in JAVA/C#.

**Source Code:**

// c++ program to explain Encapsulation

#include<iostream>

using namespace std;

class Test

{ private: // data hidden from outside world

int x;

public:

// function to set value of

// variable x

void set(int a)

{

x =a;

}

// function to return value of

// variable x

int get()

{ return x; }

};

int main()

{

Test obj;

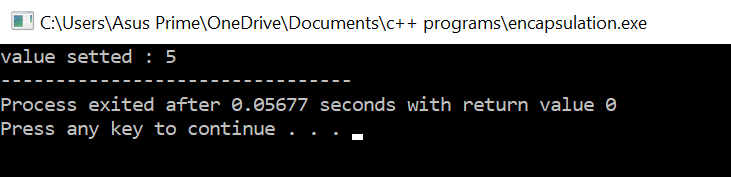
obj.set(5);

cout<<"value setted : "<<obj.get();

return 0;

}

**OUTPUT:**

****

EXPERIMENT-10

**AIM:** WAP to implement concepts of inheritance in JAVA/C#.

**Source Code:**

**IMPLEMENTATION OF SINGLE INHERITANCE**

#include <iostream>

using namespace std;

// base class

class Vehicle {

public:

Vehicle()

{

cout<< "This is a Vehicle" <<endl;

}

};

// sub class derived from two base classes

class Car: public Vehicle{ };

// main function

int main()

{

// creating object of sub class will

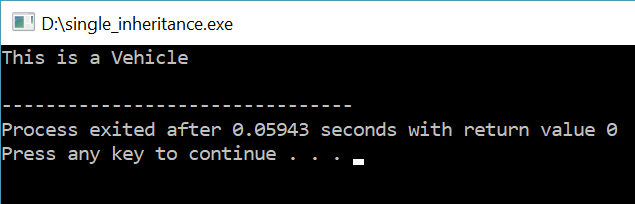
// invoke the constructor of base classes

Car obj;

return 0;

}

**Output:-**

****

**IMPLEMENTATION OF MULTIPLE INHERITANCE**

// C++ program to explain multiple inheritance

#include <iostream>

using namespace std;

class Vehicle {

public:

Vehicle()

{

cout<< "This is a Vehicle" <<endl;

}

};

classFourWheeler

{

public:

FourWheeler()

{

cout<< "This is a 4 wheeler Vehicle" <<endl;

}

};

class Car: public Vehicle, public FourWheeler{};

int main()

{

// creating object of sub class will

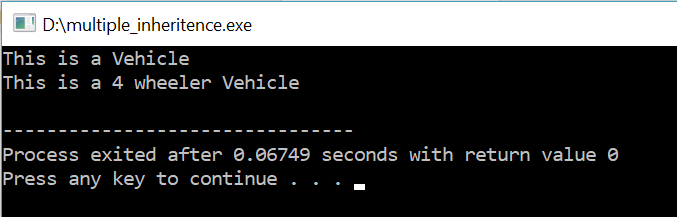
// invoke the constructor of base classes

Car obj;

return 0;

}

**Output:-**

****

**IMPLEMENTATION OF MULTILEVEL INHERITANCE**

// C++ program to implement Multilevel Inheritance

#include <iostream>

using namespace std;

class Vehicle

{

public:

Vehicle()

{

cout<< "This is a Vehicle" <<endl;

}

};

classfourWheeler: public Vehicle

{ public:

fourWheeler()

{

cout<<"Objects with 4 wheels are vehicles"<<endl;

}

};

// sub class derived from two base classes

class Car: public fourWheeler

{

public:

car()

{

cout<<"Car has 4 Wheels"<<endl;

}

};

int main()

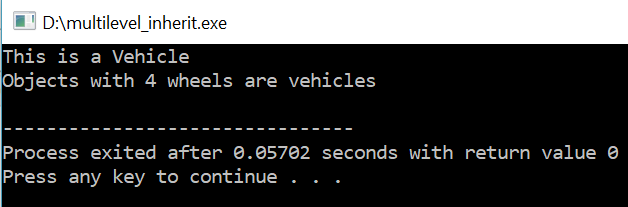
{

Car obj;

return 0;

}

**Output:-**

****

**IMPLEMENTATION OF HIERARCHICAL INHERITANCE**

// C++ program to implement Hierarchical Inheritance

#include <iostream>

usingnamespacestd;

classVehicle

{

  public:

    Vehicle()

    {

      cout<< "This is a Vehicle"<<endl;

    }

};

classCar: publicVehicle {};

classBus: publicVehicle {};

intmain()

{

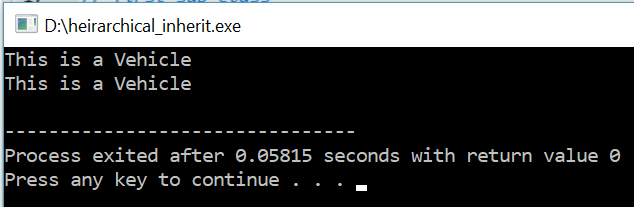
    Car obj1;

    Bus obj2;

    return0;

}

**OUTPUT:-**



**IMPLEMENTATION OF HYBRID INHERITANCE**

// C++ program for Hybrid Inheritance

#include <iostream>

using namespace std;

class Vehicle

{

public:

Vehicle()

{ cout<< "This is a Vehicle" <<endl; }

};

class Fare

{ public:

Fare()

{

cout<<"Fare of Vehicle\n";

}

};

class Car: public Vehicle {};

class Bus: public Vehicle, public Fare {};

int main()

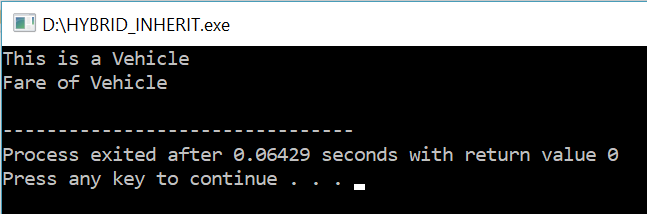
{

Bus obj2;

return 0;

}

**OUTPUT:-**



EXPERIMENT-11

**AIM:** WAP to implement concepts of polymorphism in JAVA/C#.

**Implementation of function overloading**

**Source code:-**

#include<iostream>

using namespace std;

class Test

{

public:

int square(int a) {cout<<"answer:"<<a\*a<<endl;}

double square(double a) {cout<<"answer:"<<a\*a<<endl;}

long square(long a) { cout<<"answer:"<<a\*a<<endl;}

float square(float a) {cout<<"answer:"<<a\*a<<endl;}

};

int main()

{

Test obj;

obj.square(3);

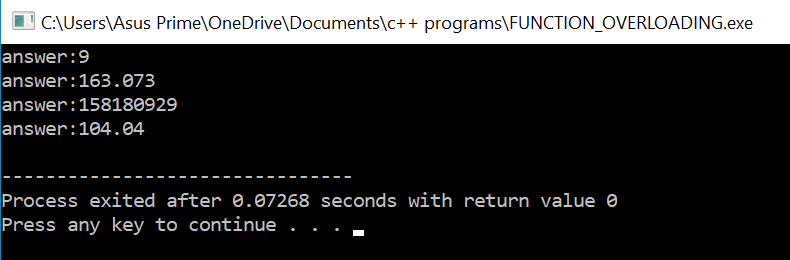
obj.square(12.77);

obj.square(12577);

obj.square(10.2);

return 0;

}

****

**Implementation of operator overloading (pre and post (++)operator )**

#include <iostream>

using namespace std;

class Test

{

private:

intnum;

public:

Test(): num(8){}

void operator ++()

{

num = num+2;

}

void operator ++(int)

{

num=num+3;

}

void Print() {

cout<<"The Count is: "<<num;

}

};

int main()

{

Test tt;

++tt; // calling of a function "void operator ++()"

tt.Print();

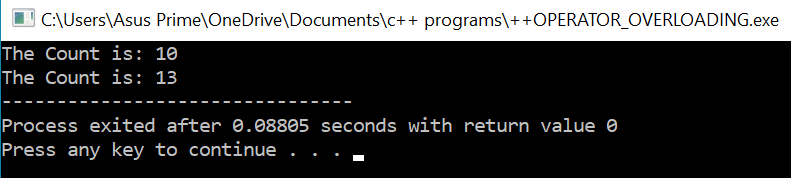
cout<<endl;

tt++;

tt.Print();

return 0;

}

****

**Implementation of operator overloading (+ operator )**

#include<iostream>

using namespace std;

class Complex

{

private:

int real, imag;

public:

Complex(int r = 0, inti =0)

{

real = r;

imag = i;

}

// This is automatically called when '+' is used with between two Complex objects

Complex operator + (Complex const&obj)

{

Complex r;

r.real = real + obj.real;

r.imag = imag + obj.imag;

return r;

}

void print()

{

cout<< real << " + i" <<imag<<endl;

}

};

int main()

{

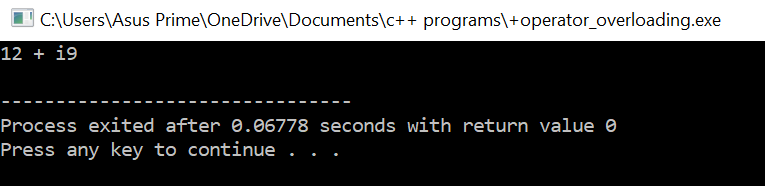
Complex c1(10, 5), c2(2, 4);

Complex c3 = c1 + c2; // call to "operator+"

c3.print();

}

**OUTPUT:-**

****

**Implementation of operator overloading (--operator )**

#include <iostream>

using namespace std;

class Test

{

private:

intnum;

public:

Test(): num(8){}

void operator --() { num = num-2; }

void Print() { cout<<"The Count is: "<<num; }

};

int main()

{

Test tt;

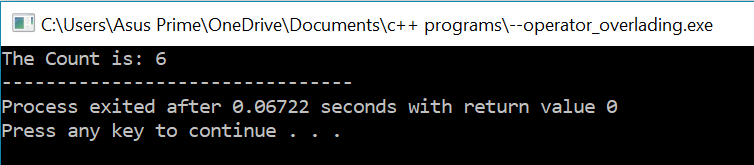
--tt; // calling of a function "void operator ++()"

tt.Print();

return 0;

}

**Output:-**

****

**Implementation of overriding**

#include<iostream>

using namespace std;

class Base

{ public:

inta,b;

void input()

{ cout<<" base class display,enter the values:\t";

cin>>a>>b;

}

void sum()

{ int s=a+b;

cout<< "\nBase class\t";

cout<<"sum="<<s;

}

};

classDerived:public Base

{ public:

inta,b,c;

void input()

{ cout<<"\nDerived class display,enter values:\t";

cin>>a>>b>>c; }

void sum()

{cout<< "\nDerived Class\n";

int s=a+b+c;

cout<<"SUM="<<s;}

};

int main()

{ Base b; //Base class object

Derived d;

b.input(); //Derived class object

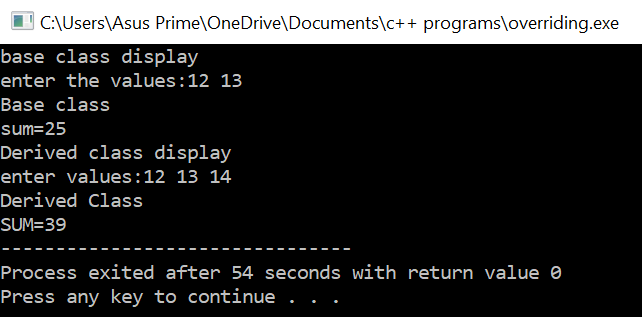
b.sum();

d.input();

d.sum();

}

**OUTPUT:-**

****

EXPERIMENT-12

**AIM:** WAP in C++ to implement different parameter passing methods.

**Implementation of call by address**

#include <iostream>

using namespace std;

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

int main () {

// local variable declaration:

int a = 100;

int b = 200;

cout<< "Before swap, value of a :" << a <<endl;

cout<< "Before swap, value of b :" << b <<endl;

swap(&a, &b);

cout<< "After swap, value of a :" << a <<endl;

cout<< "After swap, value of b :" << b <<endl;

return 0;

}

void swap(int \*x, int \*y)

{

int temp;

temp = \*x;

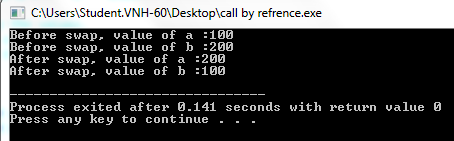
\*x = \*y;

\*y = temp;

return;

}

**OUTPUT:-**



**Implementation of call by reference**

#include <iostream>

using namespace std;

// function declaration

void swap(int&x, int&y);

int main () {

// local variable declaration:

int a = 100;

int b = 200;

cout<< "Before swap, value of a :" << a <<endl;

cout<< "Before swap, value of b :" << b <<endl;

swap(a, b);

cout<< "After swap, value of a :" << a <<endl;

cout<< "After swap, value of b :" << b <<endl;

return 0;

}

void swap(int&x, int&y) {

int temp;

temp = x;

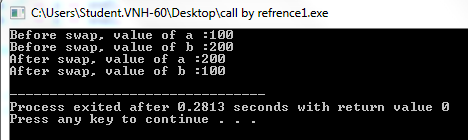
x = y;

y = temp;

return;

}

**OUTPUT:-**



**Implementation of call by value**

#include <iostream>

using namespace std;

void swap(int x, int y)

{

int temp;

temp = x;

x = y;

y = temp;

return;

}

int main () {

// local variable declaration:

int a = 100;

int b = 200;

cout<< "Before swap, value of a :" << a <<endl;

cout<< "Before swap, value of b :" << b <<endl;

// calling a function to swap the values.

swap(a, b);

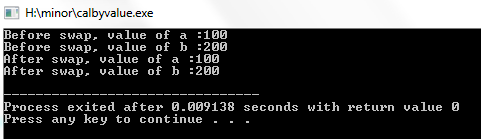
cout<< "After swap, value of a :" << a <<endl;

cout<< "After swap, value of b :" << b <<endl;

return 0;

}

**OUTPUT:-**

****

**Implementation of actual and formal parameters**

#include<iostream>

using namespace std;

voidfunc(int a, int b)

{

a += b;

cout<<"In func formal parameters: a and b = "<< a<< ", " << b;

}

int main()

{

int x = 5, y = 7;

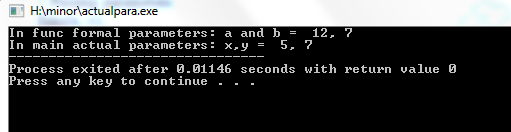
func(x, y);

cout<<"\nIn main actual parameters: x,y = "<< x << ", " <<y;

return 0;

}

**Output:-**

****

EXPERIMENT-13

**AIM:** WAP to find transpose of a matrix.

**Source Code:**

#include <iostream>

using namespace std;

int main()

{

int a[10][10], trans[10][10], r, c, i, j;

cout << "Enter rows and columns of matrix: ";

cin >> r >> c;

cout << endl << "Enter elements of matrix: " << endl;

for(i = 0; i < r; ++i)

for(j = 0; j < c; ++j)

{

cout << "Enter elements a" << i + 1 << j + 1 << ": ";

cin >> a[i][j];

} cout << endl << "Entered Matrix: " << endl;

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

for(j = 0; j < c; ++j)

{cout << " " << a[i][j];

if(j == c - 1)

cout << endl << endl} }

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

for(j = 0; j < c; ++j)

{trans[j][i]=a[i][j]:}}

cout << endl << "Transpose of Matrix: " << endl;

for(i = 0; i < c; ++i)

{ for(j = 0; j < r; ++j)

{

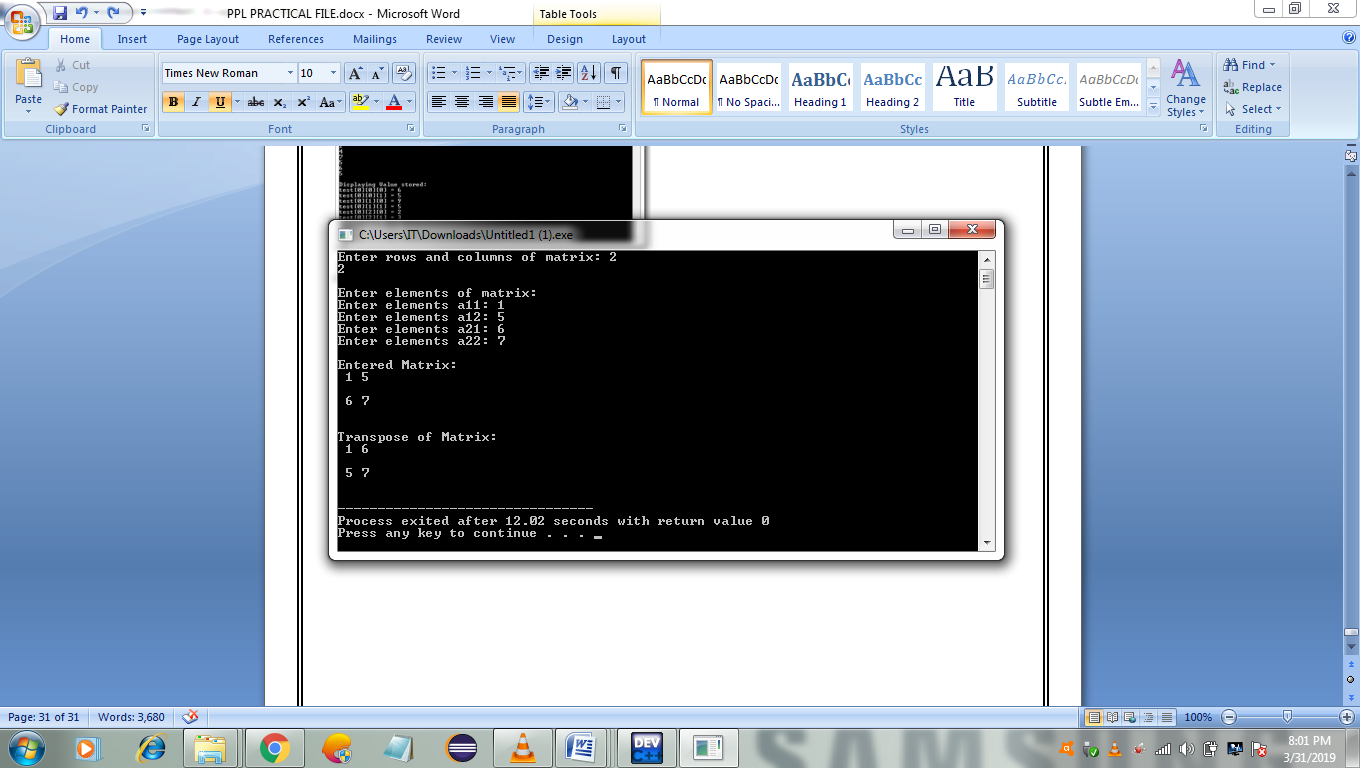
cout << " " << trans[i][j];

if(j == r - 1)

cout << endl << endl;

}}return 0;}

Output:

****

EXPERIMENT-14

**AIM:** WAP in java to implement concurrent execution of a job using threads.

**Source Code:**

class NewThread extends Thread

{

public void run()

{

for(int i=0;i&lt;=3;i++)

System.out.println(Thread.currentThread().getName());

}

}

public class NewClass1

{

public static void main(String [] args)

{

NewThread t1=new NewThread();

t1.setName(&quot;t1&quot;);

NewThread t2=new NewThread();

t2.setName(&quot;t2&quot;);

NewThread t3=new NewThread();

t3.setName(&quot;t3&quot;);

t1.start();

t2.start();

t3.start();

}}

**Output:**

t1

t2

t1

t3

t2

t3

t3

t1

t2

EXPERIMENT-15

**AIM:** WAP that demonstrates the use of THIS pointer.

**Source Code:**

#include <iostream>

using namespace std;

class Box {

public:

// Constructor definition

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

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

length = l;

breadth = b;

height = h;

}

double Volume() {

return length \* breadth \* height;

}

int compare(Box box) {

return this->Volume() > box.Volume();

}

private:

double length; // Length of a box

double breadth; // Breadth of a box

double height; // Height of a box

};

int main(void) {

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

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

if(Box1.compare(Box2)) {

cout << "Box2 is smaller than Box1" <<endl;

} else {

cout << "Box2 is equal to or larger than Box1" <<endl;

}

return 0;

}

