**VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES**

**VIVEKANANDA SCHOOL OF INFORMATION TECHNOLOGY**



**BACHELOR OF COMPUTER APPLICATION**

**Practical-VII C++ Lab File**

**BCA-271**

**Guru Gobind Singh Indraprastha University   
Sector - 16C Dwarka, Delhi – 110078**



**SUBMITTED TO:                                               SUBMITTED BY:**

Dr. Shalini Bhartiya                                                   Karthik Nair

Assistant Professor

VSIT

1. WAP to implement ‘Inline function’

#include<iostream>

using namespace std;

inline void printsum(int num1, int num2) {

cout<<num1+num2<<endl;

}

int main(){

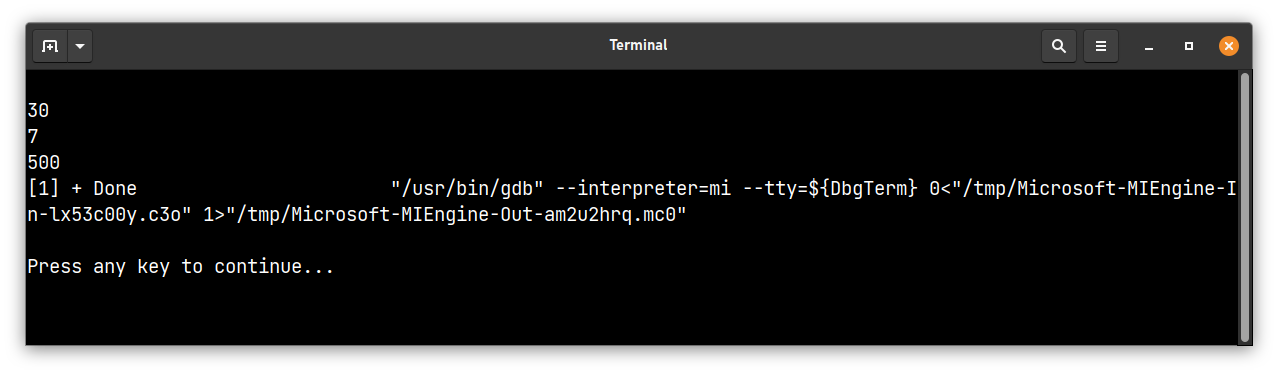
printsum(10,20);

printsum(2,5);

printsum(100,400);

return 0;

}



2. WAP to implement call by reference and return by reference using class. [Hint.

Assume necessary functions]

#include <iostream>

using namespace std;

class MyClass {

public:

int x;

void setValue(int &val) {

val = x;

}

int& getValue() {

return x;

}

};

int main() {

MyClass obj;

obj.x = 5;

int val;

obj.setValue(val);

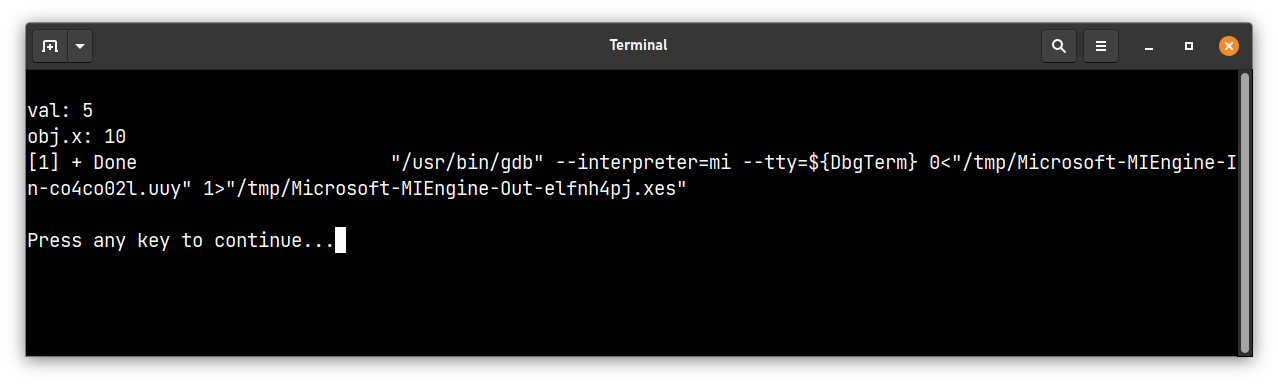
std::cout << "val: " << val << std::endl; // Outputs "val: 5"

obj.getValue() = 10;

std::cout << "obj.x: " << obj.x << std::endl; // Outputs "obj.x: 10"

return 0;

}



3. WAP to implement friend function by taking some real life example

#include <iostream>

using namespace std;

class Distance {

private:

int meter;

// friend function

friend int addFive(Distance);

public:

Distance() : meter(0) {}

};

// friend function definition

int addFive(Distance d) {

//accessing private members from the friend function

d.meter += 5;

return d.meter;

}

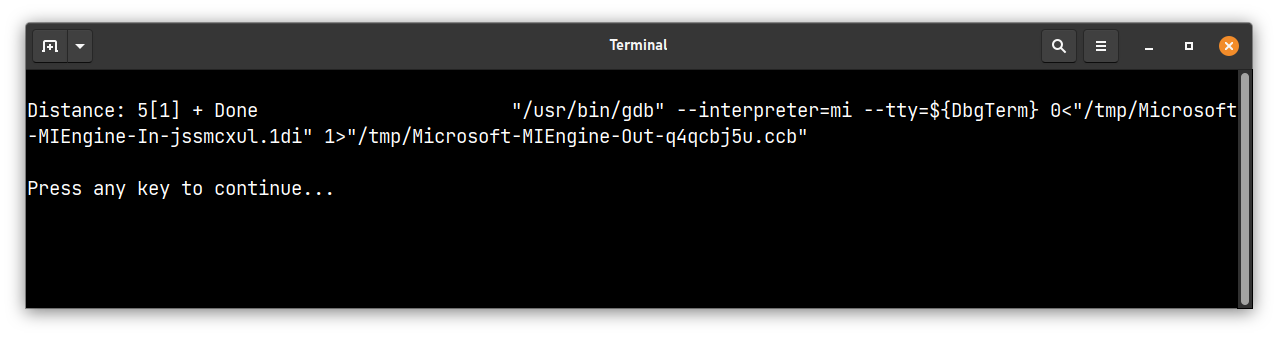
int main() {

Distance D;

cout << "Distance: " << addFive(D);

return 0;

}



4. WAP to implement ‘Function Overloading’

#include <iostream>

using namespace std;

int add(int a, int b) {

return a + b;

}

float add(double a, double

b) {

return a + b;

}

int main() {

cout << add(1, 2) << endl;

cout << add(1.2, 2.3) << endl;

return 0;

}



5. WAP to implement Parameterized Constructor, Copy Constructor and Destructor

#include <iostream>

#include <string>

class MyClass {

private:

std::string data;

public:

// Parameterized Constructor

MyClass(std::string d) {

data = d;

std::cout << "Creating object with data: " << data << std::endl;

}

// Copy Constructor

MyClass(const MyClass &other) {

data = other.data;

std::cout << "Copying object with data: " << data << std::endl;

}

// Destructor

~MyClass() {

std::cout << "Destroying object with data: " << data << std::endl;

}

std::string getData() { return data; }

};

int main() {

MyClass obj1("Hello");

MyClass obj2 = obj1; // obj2 is constructed using the copy constructor

std::cout << obj2.getData() << std::endl;

return 0;

}



6. WAP to show the usage of constructor in base and derived classes, in multiple inheritance.

#include <iostream>

using namespace std;

class Base1 {

public:

Base1() {

cout << "Base1 constructor called" << endl;

}

};

class Base2 {

public:

Base2() {

cout << "Base2 constructor called" << endl;

}

};

class Derived: public Base1, public Base2 {

public:

Derived() {

cout << "Derived constructor called" << endl;

}

};

int main() {

Derived obj;

return 0;

}



7. WAP to show the implementation of ‘containership’

#include <iostream>

using namespace std;

class Container {

private:

int x;

public:

Container(int x) : x(x) {};

void show() { cout << "x: " << x << endl;}

};

class Containership {

private:

Container container;

public:

Containership(int x) : container(x) {};

void show() { container.show(); }

};

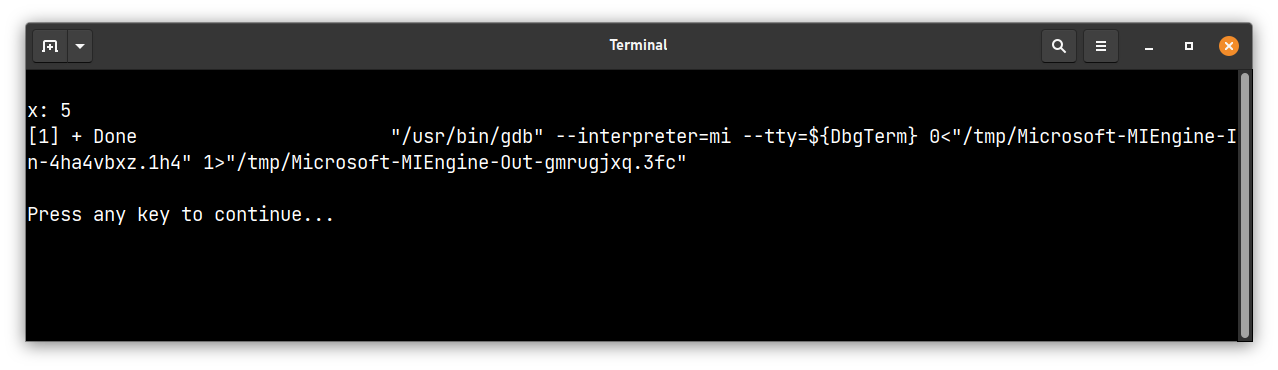
int main() {

Containership obj(5);

obj.show();

return 0;

}



8. WAP to show swapping using template function (Generic)

#include <iostream>

using namespace std;

template <typename T>

void my\_swap(T &a, T &b) {

T temp = a;

a = b;

b = temp;

}

int main() {

int x = 5, y = 10;

cout << "Before swapping: x = " << x << ", y = " << y << endl;

my\_swap(x, y);

cout << "After swapping: x = " << x << ", y = " << y << endl;

string s1 = "hello", s2 = "world";

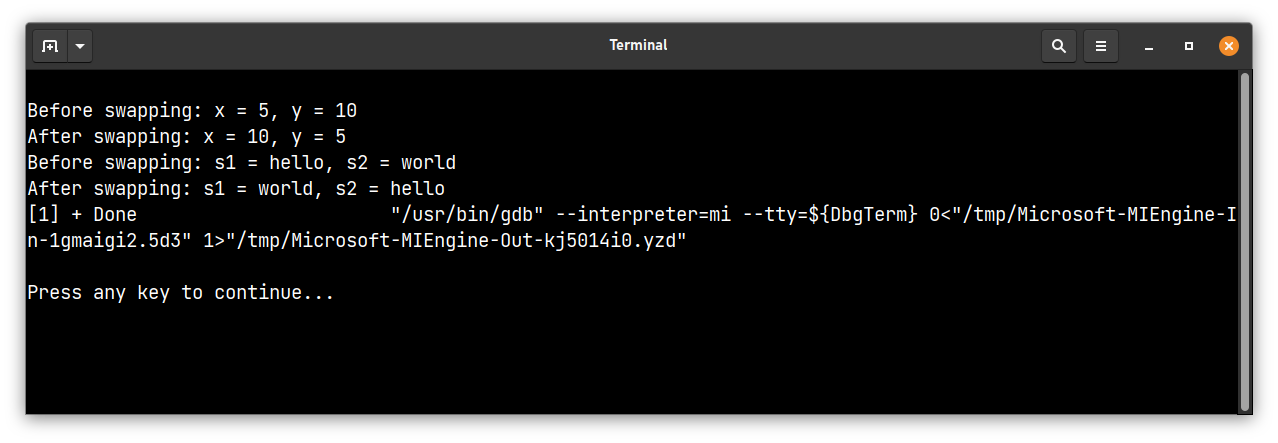
cout << "Before swapping: s1 = " << s1 << ", s2 = " << s2 << endl;

my\_swap(s1, s2);

cout << "After swapping: s1 = " << s1 << ", s2 = " << s2 << endl;

return 0;

}



9. WAP to implement ‘Exception Handling’

#include <iostream>

using namespace std;

int divide(int a, int b) {

if (b == 0) {

throw "Division by zero exception";

}

return a / b;

}

int main() {

int x = 10, y = 0;

try {

int result = divide(x, y);

cout << "Result: " << result << endl;

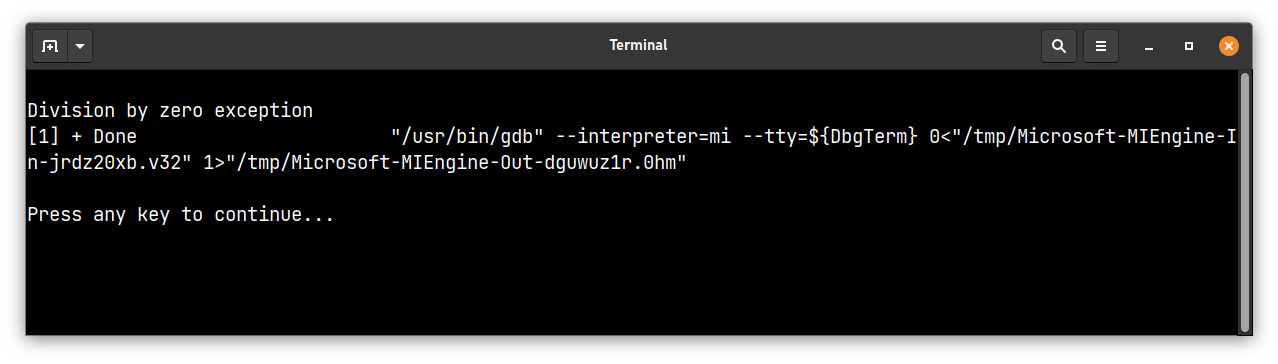
} catch (const char\* msg) {

cerr << msg << endl;

}

return 0;

}



10. WAP to read and write values through object using file handling

#include <iostream>

#include <fstream>

using namespace std;

class student

{

private:

char name[30];

int age;

public:

void getData(void)

{ cout<<"Enter name:"; cin.getline(name,30);

cout<<"Enter age:"; cin>>age;

}

void showData(void)

{

cout<<"Name:"<<name<<",Age:"<<age<<endl;

}

};

int main()

{

student s;

ofstream file;

file.open("abc.txt",ios::out);

if(!file)

{

cout<<"Error in creating file.."<<endl;

return 0;

}

cout<<"\nFile created successfully."<<endl;

s.getData();

file.write((char\*)&s,sizeof(s));

file.close();

cout<<"\nFile saved and closed succesfully."<<endl;

ifstream file1;

file1.open("abc.txt",ios::in);

if(!file1){

cout<<"Error in opening file..";

return 0;

}

file1.read((char\*)&s,sizeof(s));

s.showData();

//close the file

file1.close();

return 0;

};



11. Create a class employee which have name, age and address of employee, include

functions getdata() and showdata(), getdata() takes the input from the user, showdata()

display the data in following format:

Name:

Age:

Address:

#include <iostream>

using namespace std;

class Employee {

private:

string name;

int age;

string address;

public:

void getdata() {

cout << "Enter name: ";

cin >> name;

cout << "Enter age: ";

cin >> age;

cout << "Enter address: ";

cin.ignore();

getline(cin, address);

}

void showdata() {

cout << "Name: " << name << endl;

cout << "Age: " << age << endl;

cout << "Address: " << address << endl;

}

};

int main() {

Employee emp;

emp.getdata();

emp.showdata();

return 0;

}



12. Write a class called CAccount which contains two private data elements, an integer

accountNumber and a floating point accountBalance, and three member functions:

 A constructor that allows the user to set initial values for accountNumber and

accountBalance and a default constructor that prompts for the input of the

values for the above data numbers.

 A function called inputTransaction, which reads a character value for

transactionType(‘D’ for deposit and ‘W’ for withdrawal), and a floating point

value for transactionAmount, which updates accountBalance.

A function called printBalance, which prints on the screen the accountNumber and

accountBalance.

#include <iostream>

using namespace std;

class CAccount {

private:

int accountNumber;

float accountBalance;

public:

// constructor that allows user to set initial values

CAccount(int accountNumber, float accountBalance) {

this->accountNumber = accountNumber;

this->accountBalance = accountBalance;

}

// default constructor that prompts for input

CAccount() {

cout << "Enter account number: ";

cin >> accountNumber;

cout << "Enter account balance: ";

cin >> accountBalance;

}

void inputTransaction() {

char transactionType;

float transactionAmount;

cout << "Enter transaction type (D for deposit, W for withdrawal): ";

cin >> transactionType;

cout << "Enter transaction amount: ";

cin >> transactionAmount;

if (transactionType == 'D') {

accountBalance += transactionAmount;

}

else if (transactionType == 'W') {

accountBalance -= transactionAmount;

}

else {

cout << "Invalid transaction type!" << endl;

}

}

void printBalance() {

cout << "Account number: " << accountNumber << endl;

cout << "Account balance: " << accountBalance << endl;

}

};

int main(){

CAccount account(123, 1500);

account.inputTransaction();

account.printBalance();

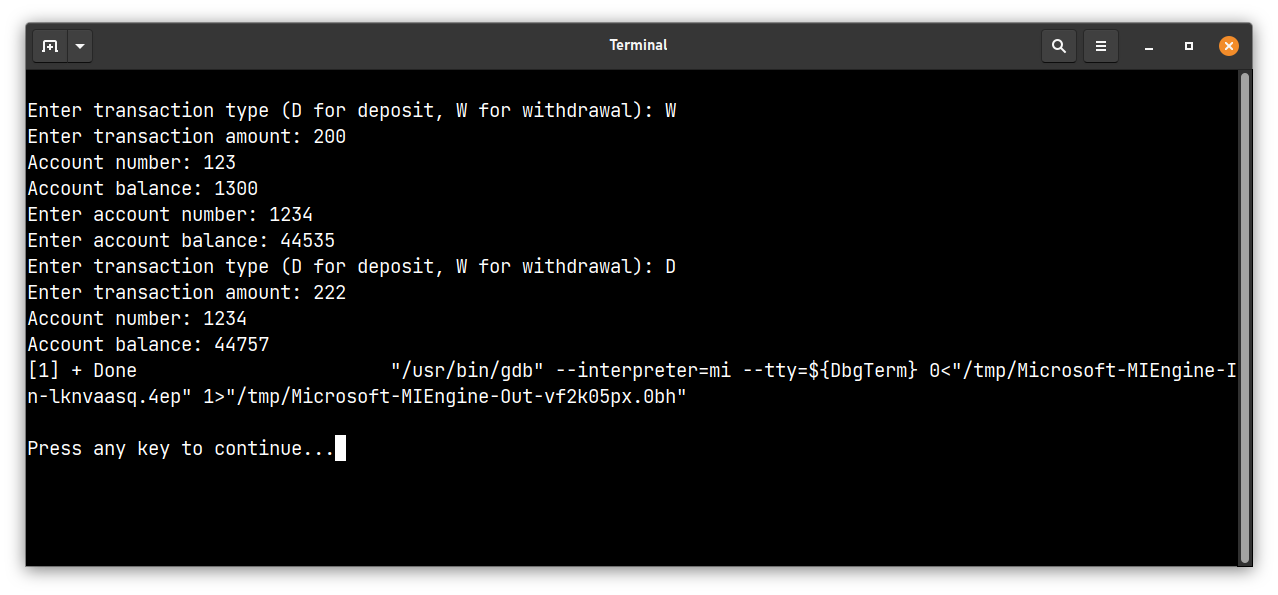
CAccount account1;

account1.inputTransaction();

account1.printBalance();

return 0;

}



13. Define a class Counter which contains an int variable count defined as static and a static

function Display () to display the value of count. Whenever an object of this class is

created count is incremented by 1. Use this class in main to create multiple objects of

this class and display value of count each time

#include <iostream>

using namespace std;

class Counter {

private:

static int count; //static variable

public:

Counter() {

count++;

}

static void Display() {

cout << "Count: " << count << endl;

}

};

//initializing the static variable

int Counter::count = 0;

int main() {

Counter c1, c2, c3;

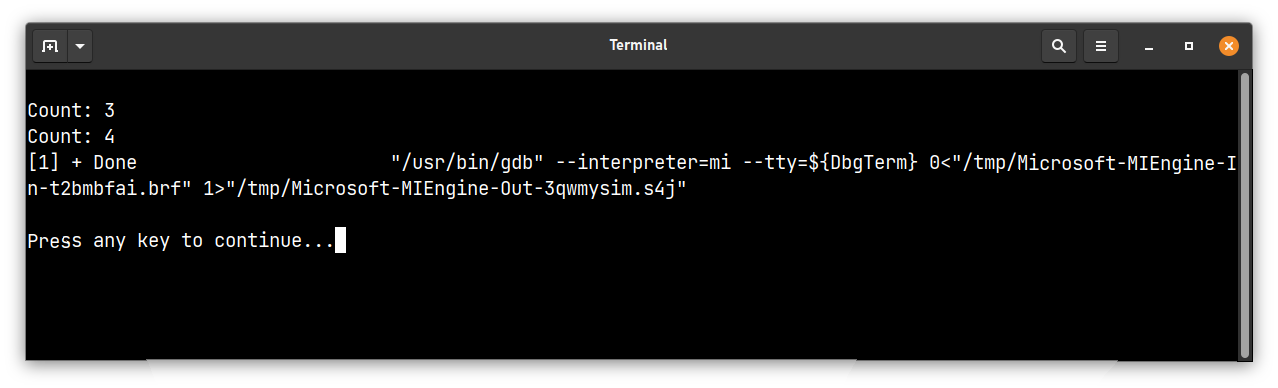
Counter::Display(); //Count: 3

Counter c4;

Counter::Display(); //Count: 4

return 0;

}



14. WAP to add and subtract two complex numbers using classes

#include <iostream>

using namespace std;

class Complex {

private:

double real;

double imag;

public:

Complex() : real(0), imag(0) {}

Complex(double real, double imag) {

this->real = real;

this->imag = imag;

}

// Overloading '+' operator to add two Complex numbers

Complex operator+(const Complex &c) {

return Complex(real + c.real, imag + c.imag);

}

// Overloading '-' operator to subtract two Complex numbers

Complex operator-(const Complex &c) {

return Complex(real - c.real, imag - c.imag);

}

void print() {

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

}

};

int main() {

Complex c1(1, 2);

Complex c2(3, 4);

Complex c3 = c1 + c2;

cout << "c1 + c2 = ";

c3.print();

Complex c4 = c1 - c2;

cout << "c1 - c2 = ";

c4.print();

return 0;

}



15. WAP to implement += and = operator

#include <iostream>

using namespace std;

class MyNumber {

private:

int number;

public:

MyNumber(int number) {

this->number = number;

}

// Overloading '+=' operator

MyNumber& operator+=(const MyNumber &other) {

this->number += other.number;

return \*this;

}

// Overloading '=' operator

MyNumber& operator=(const MyNumber &other) {

this->number = other.number;

return \*this;

}

void print() {

cout << number << endl;

}

};

int main() {

MyNumber n1(5);

MyNumber n2(7);

n1 += n2; //n1 = n1 + n2;

cout << "n1 += n2: ";

n1.print();

MyNumber n3(10);

MyNumber n4 = n3; //using '=' operator

cout << "n4 = n3: ";

n4.print();

return 0;

}

