**PROGRAMS:**

1. **Define a class named “fraction” whose objects represent rational numbers. Include integer data members num and den for storing numerator and denominator respectively.**

**Also include following:**

1. **Default constructor**
2. **Parametrised constructor with 2 arguments.**
3. **Copy constructor**
4. **Function eval\_func() for evaluating the rational number**
5. **Friend function display() for displaying the fraction in the form num/den**

**Input:**

#include<iostream>

#include<conio.h>

using namespace std;

class fraction {

int num;

int den;

public:

fraction() {

num = den = 1;

}

fraction(int n, int d) {

num = n;

den = d;

}

fraction(fraction &obj) {

num = obj.num;

den = obj.den;

}

void eval\_func() {

double result;

if (num == den) {

result = 1;

}

else if (den == 0) {

cout << "Invalid denominator" << endl;

result = num;

return;

}

else {

result = (double)num / den;

}

cout << "The result of the evaluation is " << result <<endl;

}

friend void display(fraction &obj);

};

void display(fraction& obj) {

cout <<"Result: "<<obj.num << "/" << obj.den << endl;

}

int main() {

int n, d;

cout << "Calling Default Constructor" << endl;

fraction obj;

display(obj);

cout << "Calling Parameterised Constructor" << endl;

cout << "Enter numerator and denominator :";

cin >> n >> d;

fraction obj2(n,d);

obj2.eval\_func();

display(obj2);

cout << "Calling Copy Constructor" << endl;

fraction obj3(obj2);

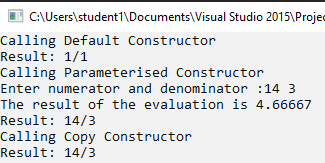
display(obj3);

\_getch();

return 0;

}

**Output:**

****

1. **Write a program to implement a banking system with data members balance and rate of interest.**

**Also include following:**

1. **Constructor to initialise the data members**
2. **Copy constructor**
3. **Make a deposit**
4. **Withdraw an amount**
5. **Calculate interest.**
6. **Destroy object using destructor.**

**Input:**

#include<iostream>

#include<conio.h>

#include<string>

using namespace std;

class BankAccount {

string name;

double balance;

public:

BankAccount() {

cout << "Enter the name of the account holder: ";

cin >> name;

balance = 0;

}

BankAccount(BankAccount &obj) {

name = obj.name;

balance = obj.balance;

}

void deposit(int amount) {

balance += amount;

cout << "Deposited : " << amount << endl;

cout << "Total Balance: " << balance << endl;

return;

}

void withdraw(int amount) {

balance -= amount;

cout << "Withdrawn : " << amount << endl;

cout << "Total Balance: " << balance << endl;

return;

}

void calculate\_interest(float interestRate, int years) {

double interest = 0.0;

interest = (balance \* interestRate \* years) / 100;

cout << "Interest amount : " << interest << endl;

return;

}

~BankAccount() {

cout << "destructor called";

}

friend void display(BankAccount&);

};

void display(BankAccount& obj) {

cout << "Namr : " << obj.name << endl;

cout << "Total Balance: " << obj.balance << endl;

}

int main() {

BankAccount customer1;

BankAccount customer2(customer1);

customer1.deposit(1000);

customer1.withdraw(500);

customer1.calculate\_interest(4.2, 2);

cout << "Object 1 " << endl;

display(customer1);

cout << "Object 2" << endl;

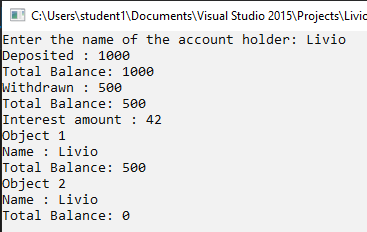
display(customer2);

\_getch();

return 0;

}

**Output:**

****

1. **Show the use of constructor with default arguments**

**Define a class named “box” with integer data members l, b, h. Use constructors with default arguments to initialize the variables.**

**Input:**

#include<iostream>

#include<conio.h>

using namespace std;

class box {

float l, b, h;

public:

box(float argL = 3.6, float argB = 4.8, float argH = 6.3) {

l = argL;

b = argB;

h = argH;

}

void display() {

cout << "Lenght: " << l << " Breadth: " << b << " Height: " << h << endl;

}

};

int main() {

float l, b, h;

cout << "Enter Lenght, Breadth & Height :";

cin >> l >> b >> h;

box cube;

cout << "Without passing args" << endl;

cube.display();

box cuboid(l,b,h);

cout << "Passing args" << endl;

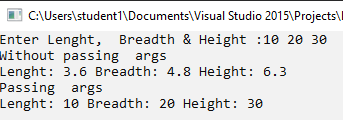
cuboid.display();

\_getch();

return 0;

}

**Output:**

****

1. **Write a C++ Program to Overload parameterised Constructor.**

**Input:**

#include<iostream>

#include<conio.h>

#include<string>

using namespace std;

class box {

float length, breadth, height, volume;

public:

box(float length)

{

volume = length\*length\*length;

}

box(float length, float breadth, float height)

{

volume = length\*breadth\*height;

}

void display() {

cout << "Volume : " << volume<<endl;

}

};

int main() {

float s, length, breadth, height, totalVolume;

cout << "Enter the Length of cube : ";

cin >> length;

box cube(length);

cube.display();

cout << "Enter the Length, breadth & height: ";

cin >> length >> breadth >> height;

box cuboid(length, breadth, height);

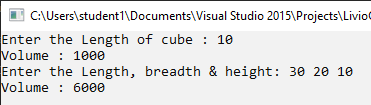
cuboid.display();

\_getch();

return 0;

}

**Output:**

****

1. **Write a recursive function power (base, exponent) that when invoked returns base^exponent.E.g. power (3.4)=3\*3\*3\*3.Exponent has to be greater or equal to 1.**

**Input:**

#include<iostream>

#include<conio.h>

using namespace std;

int power(int base, int exponent) {

if (exponent == 1)

return base;

return base\*power(base, exponent - 1);

}

int main() {

int base = 0, exponent = 0;

cout << "Enter the base value:\t";

cin >> base;

cout << "Enter the exponent:\t";

cin >> exponent;

if (exponent < 1) {

cout << "Please enter exponent greater or equal to 1";

}

else {

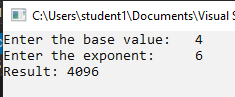
cout << "Result: " << power(base, exponent);

}

\_getch();

}

**Output:**

****

1. **Write a C++ Program to implement Linear Search using recursion.**

**Input:**

#include<iostream>

#include<conio.h>

using namespace std;

int linearSearch(int arr[], int searchElement, int index, int size) {

if (index == size) {

cout << "Element not found" << endl;

return 0;

}

if (searchElement == arr[index]) {

cout << "Element found at index: " << index + 1;

return index;

}

return linearSearch(arr, searchElement, index + 1, size);

}

int main() {

int size, arr[100], i = 0, searchElement ;

cout << "Enter size of Array: ";

cin >> size;

cout << "Enter elements of the array: "<<endl;

while (i < size)

{

cin >> arr[i];

i++;

}

cout << "Enter the element to be searched: ";

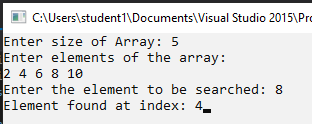
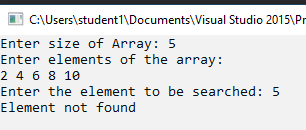
cin >> searchElement;

linearSearch(arr, searchElement, 0, size);

\_getch();

}

**Output:**

** **