**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.

#include<iostream>

#include<conio.h>

using namespace std;

class fraction {

int num;

int den;

public:

//default constructor

fraction() {

num = 0;

den = 0;

}

//parametrised constructor with two arguments

fraction(int n, int d) {

num = n;

den = d;

}

//copy constructor

fraction(fraction &obj) {

num = obj.num;

den = obj.den;

}

// evaluating rational numbers

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 function to display the

friend void display(fraction &obj);

};

void display(fraction& obj){

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

}

int main() {

int pause = 0, num, den;

fraction obj;

cout << "Enter the numerator: ";

cin >> num;

cout << "Enter the denominator: ";

cin >> den;

fraction obj2(num,den);

fraction obj3(obj2);

cout << "Default constructor: ";

display(obj);

cout << "Parametrised constructor: ";

display(obj2);

cout << "Copy constructor: ";

display(obj3);

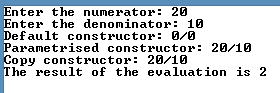
obj2.eval\_func();

\_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.

#include<iostream>

#include<conio.h>

#include<string>

using namespace std;

class BankAccount {

string name;

double balance;

public:

//default constructor

BankAccount() {

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

getline(cin,name);

balance = 0;

}

//copy constructor

BankAccount(BankAccount &obj) {

balance = obj.balance;

name = obj.name;

}

~BankAccount(){

cout<< "destructor called";

}

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;

}

// calculate interest

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

double interest = 0.0;

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

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

return;

}

friend void display(BankAccount&);

};

void display(BankAccount& obj) {

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

cout<< "TOTAL BALANCE : "<< obj.balance << endl;

}

int main() {

BankAccount customer1;

BankAccount customer2(customer1);

cout << "\n--- Object 1 ---" << endl;

customer1.deposit(1000);

customer1.withdraw(500);

customer1.calculate\_interest(10,2);

display(customer1);

cout << "\n--- Object 2 ---" << endl;

customer2.deposit(2000);

customer2.withdraw(500);

customer1.calculate\_interest(10, 4);

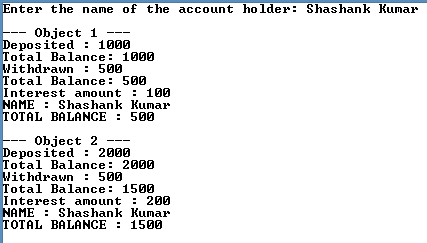
display(customer2);

\_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.

#include<iostream>

#include<conio.h>

using namespace std;

class box {

int l, b, h;

public:

box(int argL = 3, int argB = 4, int argH = 6){

l = argL;

b = argB;

h = argH;

}

void display() {

cout << "l: " << l << " b: " << b << " h: " << h << endl;

}

};

int main() {

int side;

cout << "Enter the length of the side of the square box: ";

cin >> side;

box rect;

box square(side, side, side);

cout << "Reactangle: " << endl;

rect.display();

cout << "Square: " << endl;

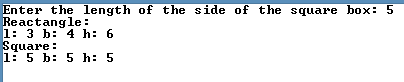
square.display();

\_getch();

return 0;

}

**Output:**



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

#include<iostream>

#include<string>

using namespace std;

class vehicle {

string name;

int no\_of\_wheels;

double cost;

public:

//Passing only two arguments to the constructor

vehicle(string nameV, int wheels) {

name = nameV;

no\_of\_wheels = wheels;

cout << "Enter the cost of the vehicle : ";

cin >> cost;

}

//Passing three argument to the constructor

vehicle(string nameV, int wheels, int costI) {

name = nameV;

no\_of\_wheels = wheels;

cost = costI;

}

void display() {

cout << "The "<< name << " has " << no\_of\_wheels << " wheels and " << "the cost of the vehicle is Rs." << cost << endl;

}

};

int main() {

int wheels, costI;

double costD;

string name;

cout << "OBJECT 1:"<< endl;

cout << "Enter the model name of the vehicle : ";

cin >> name;

cout << "Enter the number of wheels the vehicle has : ";

cin >> wheels;

cout << "Enter the cost of the vehicle : ";

cin >> costI;

vehicle obj1(name, wheels, costI);

cout << "\nOBJECT 2:"<< endl;

cout << "Enter the model name of the vehicle : ";

cin >> name;

cout << "Enter the number of wheels the vehicle has : ";

cin >> wheels;

vehicle obj2(name, wheels);

cout << "\nOBJECT 1:"<< endl;

obj1.display();

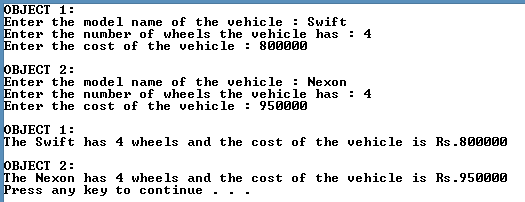
cout << "\nOBJECT 2:"<< endl;

obj2.display();

return 0;

}

**Output:**



1. Write a C++ program to join two strings using classes. Make use of dynamic Memory Allocation.

#include <conio.h>

#include <string.h>

using namespace std;

class Strings

{

char \*str;

int length;

public:

Strings(){

length = 0;

str = new char[length +1];

}

Strings(char\*s){

length = strlen(s);

str = new char[length + 1];

strcpy\_s(str, length, s);

}

void concat(Strings &a, Strings &b){

length = a.length + b.length;

delete str;

str = new char[length + 1];

strcpy(str, length, a.str);

strcat\_s(str, length +1, b.str);

}

void display(){

cout << "Concatenated string: " << str << endl;

}

};

int main(){

char \*str = "Hello";

Strings str1(str), str2("World"), s1;

s1.concat(str1,str2);

s1.display();

return 0;

}

**Output:**

