

OOP

Assignment#02



Name: Talib Husain

Roll # 21F-9070

Task-1

#include<iostream>

#include<string>

using namespace std;

class bank {

private:

static string bankname;

static int branchcode;

static string city;

string name;

int accountnumber;

char type\_of\_account;

double balance;

public:

bank(char account\_type, int acoountnumber, string bankname, int branchcode, string city, string customername, double balance);

bank() {

this->accountnumber = 0;

this->balance = 0;

this->name = "";

this->type\_of\_account = ' ';

this->bankname = "";

this->branchcode = 0;

this->city = "";

}

void setBank\_Details(char account\_type, int acoountnumber, string bankname, int branchcode, string city, string customername, double balance);

void deposit(double balance);

void withdraw(double withdraw);

void display();

};

bank::bank(char account\_type, int accountnumber, string bankname, int branchcode, string city, string customername, double balance) {

this->accountnumber = accountnumber;

this->balance = balance;

this->name = customername;

this->type\_of\_account = account\_type;

this->bankname = bankname;

this->branchcode = branchcode;

this->city = city;

}

void bank::deposit(double balance) {

this->balance += balance;

cout << "New Balance: " << this->balance << endl;

}

void bank::withdraw(double withdraw) {

this->balance -= withdraw;

cout << "Your Remaining Balance: " << this->balance << endl;

}

void bank::setBank\_Details(char account\_type,int accountnumber, string bankname, int branchcode, string city, string customername, double balance) {

this->accountnumber = accountnumber;

this->balance = balance;

this->name = customername;

this->type\_of\_account = account\_type;

this->bankname = bankname;

this->branchcode = branchcode;

this->city = city;

}

void bank::display() {

cout << "Name of Depositor: " << this->name << endl;

cout << "Name of Bank: " << this->bankname << endl;

cout << "Name of City: " << this->city << endl;

cout << "Account Balance: " << this->balance << endl;

}

string bank::bankname = "";

int bank::branchcode = 0;

string bank::city = "";

int main() {

string bankname, city, name;

int branchcode, accountnumber;

char type\_of\_account;

double balance, deposit, withdraw;

int choose;

cout << "Enter a Bank Name: ";

getline(cin, bankname);

cout << "Enter a Bank CIty: ";

getline(cin, city);

cout << "Enter a Branch Code: ";

cin >> branchcode;

cin.ignore();

cout << "Enter a Customer Name: ";

getline(cin, name);

cout << "Enter a Account Number: ";

cin >> accountnumber;

cout << "Enter a Type of Account: ";

cin >> type\_of\_account;

cout << "Enter a Balance: ";

cin >> balance;

bank b(type\_of\_account,accountnumber, bankname, branchcode, city, name, balance);

cout << "Enter a Amount to deposit: ";

cin >> deposit;

b.deposit(deposit);

cout << "Enter a Amount to Withdraw: ";

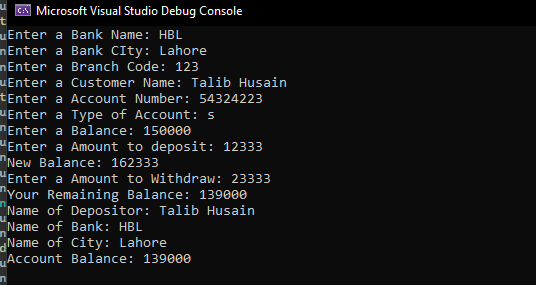
cin >> withdraw;

b.withdraw(withdraw);

b.display();

}

Output



Task-2

In C++ this keyword is actually a pointer who has the address of a object

that has been created in main function. We use this keyword or pointer in

parameterized constructors to distiguish between class member variables and

formal para meters of a function. When we use this pointer the compiler

understand that we are talking about class member variables. e.g

class person{

string name;

int height;

char gender;

public:

person(string name, int height, char gender){

this->name = name;

this->height = height;

this->gender = gender;

}

};

If we don't use this keyword in above example than it does not make sense

to compiler that of which name, height, gender we are talking about is it

class members or formal parameters of constructor.

Task-3

#include<iostream>

using namespace std;

class HotDogStand {

int stands\_id;

int sold\_hotDogs;

static int totalSold;

public:

HotDogStand(int, int);

void justSold();

int getSoldHotDog();

static int getTotalSold();

};

int HotDogStand::totalSold = 0;

HotDogStand::HotDogStand(int stands\_id, int sold\_hotDogs) {

this->stands\_id = stands\_id;

this->sold\_hotDogs = sold\_hotDogs;

this->HotDogStand::totalSold += sold\_hotDogs;

justSold();

}

void HotDogStand::justSold() {

this->sold\_hotDogs++;

this->HotDogStand::totalSold++;

}

int HotDogStand::getSoldHotDog() {

return this->sold\_hotDogs;

};

int HotDogStand::getTotalSold() {

return HotDogStand::totalSold;

}

int main() {

int stands\_id;

int sold\_hotDogs;

cout << "Enter hot dog stand id: ";

cin >> stands\_id;

cout << "Enter sold hot dogs: ";

cin >> sold\_hotDogs;

HotDogStand s1(stands\_id, sold\_hotDogs);

cout << "Enter hot dog stand id: ";

cin >> stands\_id;

cout << "Enter sold hot dogs: ";

cin >> sold\_hotDogs;

HotDogStand s2(stands\_id, sold\_hotDogs);

cout << "Enter hot dog stand id: ";

cin >> stands\_id;

cout << "Enter sold hot dogs: ";

cin >> sold\_hotDogs;

HotDogStand s3(stands\_id, sold\_hotDogs);

cout << endl;

cout << "Hot Dogs sold by stand 1: " << s1.getSoldHotDog() << endl;

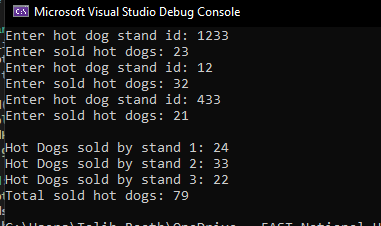
cout << "Hot Dogs sold by stand 2: " << s2.getSoldHotDog() << endl;

cout << "Hot Dogs sold by stand 3: " << s3.getSoldHotDog() << endl;

cout << "Total sold hot dogs: " << s3.getTotalSold() << endl;

}

Output



Task-4

/\*We use pointer funtions to store address of a function. So, instead by directly using the function we use function by

a pointer who has tha address of that function. e.g\*/

#include<iostream>

using namespace std;

int sum(int a ,int b){

return a + b;

}

int fun(int(\*p[])(int ,int )){

int i=0;

for(i=0; i < 5; i++){

cout << p[i](i+1,i+2);

cout << endl;

}

cout<<"Fun is Displied";

return 0;

}

int main(){

int (\*funPtr[])(int,int) = {sum,sum,sum,sum,sum};

fun(funPtr);

}

/\*we ca do this in array as well to store multiple addresses.\*/

Task-5

#include <iostream>

using namespace std;

int fib(int x) {

if ((x == 1) || (x == 0)) {

return(x);

}

else {

return(fib(x - 1) + fib(x - 2));

}

}

int main() {

int x, i = 0;

cout << "Enter the number of Fibonacci Series: ";

cin >> x;

cout << "Fibonnaci Series : ";

while (i < x) {

cout << " " << fib(i);

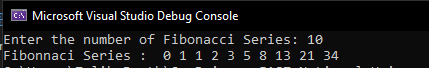
i++;

}

return 0;

}

Output



Graphical Representation

