24k-0918 **DSA** LAB 01 TASKS

TASK-1

#include <iostream>

using namespace std;

class BankAccount{

    double\* balance;

    public:

        BankAccount(){

            balance = new double(0);

        }

        BankAccount(int *b*){

            balance = new double(*b*);

        }

        BankAccount(const BankAccount& *BA*){

            balance = new double(\**BA*.balance); // deep copy

        }

        double getBalance(){

             return \*balance;

        }

        void deduct(double *amount*) {

            \*balance -= *amount*;

        }

        ~BankAccount(){

            delete balance;

        }

};

int main() {

    BankAccount account1;

    cout<<account1.getBalance()<<endl;

    BankAccount account2(1000);

    cout<<account2.getBalance()<<endl;

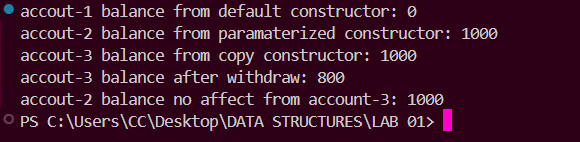
    BankAccount account3 = account2;

    account3.deduct(200);

    cout<<account3.getBalance()<<endl;

    cout<<account2.getBalance()<<endl;

}



TASK-2

#include <iostream>

using namespace std;

#include <cstring>

class Exam{

    char \*name;

    char \*date;

    double \*score;

    public:

        Exam(string *n*, string *d*, double *s*){

            name = new char[*n*.size()+1];

            strcpy(name, *n*.c\_str());

            date = new char[*d*.size()+1];

            strcpy(date, *d*.c\_str());

            score = new double(*s*);

        }

        void display(){

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

            cout<<"Date:"<<date<<endl;

            cout<<"Score: "<<\*score<<endl;

        }

        ~Exam(){

            delete[] name;

            delete[] date;

            delete score;

        }

*//changin name to show same memory address*

        void newName(string *n*){

            strcpy(name, *n*.c\_str());

        }

};

int main(){

    Exam ex1("batool", "20-8-2024", 99.9);

    ex1.display();

    cout<<endl;

    Exam ex2 = ex1; *// shallow copy*

    ex2.display();

    cout<<endl

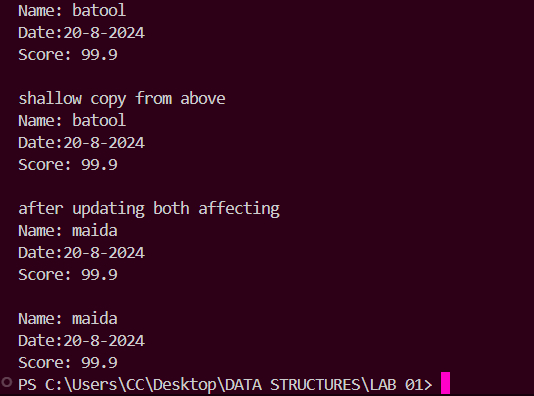
    ex1.newName("maida");

    ex1.display();

    cout<<endl;

    ex2.display();

}



TASK-3

#include <iostream>

using namespace std;

class Box {

    int\* size;

    public:

        Box(int *s*) {

            size = new int(*s*);

        }

        Box(const Box& *other*) {

            size = new int(\**other*.size);

        }

        Box operator = (const Box& *b*){

            if(this == &*b*){

                return \*this;

            }

            delete size;

            this->size = *b*.size;

            return \*this;

        }

        int getSize(){

            return \*size;

        }

        void setSize(int *s*){

            \*size = *s*;

        }

        ~Box() {

            delete size;

        }

};

int main(){

    Box box1(10);

    Box box2 = box1; *// deep copy*

    cout<<box1.getSize()<<endl;

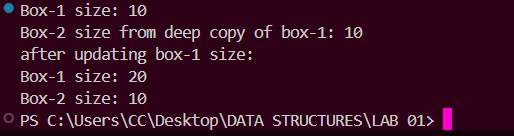
    cout<<box2.getSize()<<endl;

    box1.setSize(20);

    cout<<box1.getSize()<<endl;

    cout<<box2.getSize()<<endl;

}



TASK-4

#include <iostream>

using namespace std;

class Product

{

    string name;

    double price;

    int stock;

public:

    Product(string *n*, double *p*, int *s*) : name(*n*), price(*p*), stock(*s*) {};

    void applyDiscount(double *percent*)

    {

        price -= price \* (*percent* / 100.0);

    }

    void display()

    {

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

        cout << "Price: " << price << endl;

        cout << "Stock: " << stock << endl;

    }

};

class Bundle

{

    Product \*\*prod;

    int count;

    int capacity;

    double discount;

public:

    Bundle(double *d*, int *cap* = 20)

    {

        discount = *d*;

        capacity = *cap*;

        count = 0;

        prod = new Product \*[capacity];

    }

    void addProd(Product \**p*)

    {

        if (count < capacity)

        {

            prod[count++] = *p*;

        }

    }

    void applyDiscount()

    {

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

        {

            prod[i]->applyDiscount(discount);

        }

    }

    void display()

    {

        cout << "\n--- Bundle Details ---" << endl;

        cout << "Discount: " << discount << "%" << endl;

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

        {

            prod[i]->display();

        }

        cout << "----------------------" << endl;

    }

*// shallow copy*

    Bundle(const Bundle &*b*)

    {

        discount = *b*.discount;

        capacity = *b*.capacity;

        count = *b*.count;

        prod = *b*.prod; *// same product*

    }

*// deep copy*

    Bundle(const Bundle &*b*, bool *deepCopy*)

    {

        discount = *b*.discount;

        capacity = *b*.capacity;

        count = *b*.count;

        if (*deepCopy*)

        {

            prod = new Product \*[capacity];

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

            {

                prod[i] = new Product(\**b*.prod[i]);

            }

        }

    }

    ~Bundle()

    {

        delete[] prod;

    }

};

int main()

{

*// Create some products*

    Product \*p1 = new Product("Laptop", 1000, 5);

    Product \*p2 = new Product("Phone", 500, 10);

    Product \*p3 = new Product("Headphones", 100, 20);

*// Original bundle*

    Bundle original(10); *// 10% discount*

    original.addProd(p1);

    original.addProd(p2);

    original.addProd(p3);

    cout << "Original Bundle:" << endl;

    original.display();

*// Shallow copy*

    Bundle shallowCopy = original;

*// Deep copy*

    Bundle deepCopy(original, true);

*// Apply discount to original*

    original.applyDiscount();

    cout << "\nAfter applying discount to ORIGINAL bundle:" << endl;

    cout << "Original:";

    original.display();

    cout << "Shallow Copy (affected):";

    shallowCopy.display();

    cout << "Deep Copy (independent):";

    deepCopy.display();

*// Cleanup*

    delete p1;

    delete p2;

    delete p3;

    return 0;

}



