**Ministerul Educației al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică și Microelectronică**

**Departamentul Ingineria Software și Automatică**

**Raport**

Lucrarea de laborator nr.4

Disciplina Programarea orietntată pe obiecte:

Tema: Moştenirea şi compoziţia

Varianta 5

**Efectuat**: st.gr. TI-207 Bunescu Gabriel

**Verificat lect.univ.Lisnic Inga**.

Chișinău 2021

**Scopul lucrării:**

1. Studierea moșteniri, avantajele și dezavantajele;
2. Studierea compoziției;
3. Studierea regulilor de definire a moștenirii și compoziției;
4. Studierea formelor de moștenire;
5. Studierea inițializatorilor;
6. Principiul de substituției;
7. Moștenirea și compoziția – ce să alegem;

**Sarcina:**

**Varianta: 5**

а) De creat clasa *lichid,* care conţine informaţie despre denumire (pointer la cîmp), densitate. Determinaţi constructorii, destructorii, operatorii fluxului de intrare. Creaţi a clasă derivată – băuturi alcoolice, care conţin tărie. Determinaţi funcţiile predestinate densităţii şi tăriei.

b) De creat clasa *buton*, care conţine un text anumit. Determinaţi constructorii şi metodele de acces. Creaţi clasa *fereastra,* care conţine un buton şi coordonatele ferestrei. Definiţi constructorii şi destructorii. Definiţi clasa derivată fereastra cu buton şi cu un mesaj. Definiţi constructorii, destructorii şi operatorii fluxului de ieşire.

**Mersul lucrării:**

**Listingul programului:**

**a)**

#include <iostream>

#include <stdio.h>

#include <string.h>

class Lichid{

public:

std::string denumire;

double densitate;

Lichid(){

printf("Constructorul a fost apelat\n");

}

Lichid(std::string a, double b){

this->denumire = a;

this->densitate = b;

}

~Lichid(){

printf("\nLichidul a fost eliminat\n");

}

void modificareDensitate(double a){

this->densitate = a;

}

friend std::ostream& operator<<(std::ostream& output, const Lichid &D){ output << D.denumire << std::endl << D.densitate << std::endl; }

friend std::istream &operator>>( std::istream &input,Lichid &D )

{ printf("\nIntrodu Denumirea: "); input >> D.denumire;

printf("\nIntrodu Densitatea: ");input >> D.denumire; }

};

class Alcool:public Lichid{

private:

double tarie;

public:

Alcool(){

printf("Constructorul a fost apelat: ");

}

Alcool(std::string a, double b, double c){

this->denumire = a;

this->densitate = b;

this->tarie = c;

}

void modificareTarie(double a){

this->tarie = a;

}

friend std::ostream& operator<<(std::ostream& output, const Alcool &D){

output << D.denumire << " - Denumirea\n" << D.densitate << " - Densitatea\n" <<

D.tarie << " - Tarie\n" << std::endl;

}

friend std::istream &operator>>( std::istream &input,Alcool &D )

{

printf("\nIntrodu Denumirea: "); input >> D.denumire;

printf("\nIntrodu Densitatea: "); input >> D.densitate;

printf("\nIntrodu taria: "); input >> D.tarie; }

};

int main(){

Alcool A;

std::cin >> A;

std::cout << A;

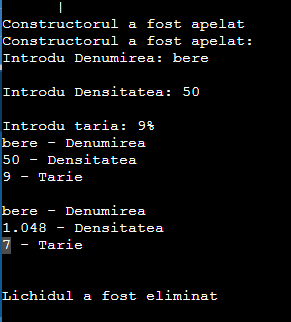
A.modificareDensitate(1.048);

A.modificareTarie(7);

std::cout << A;

return 0;

}

****

**Figura.1. Rezultatul afișării.**

**b)**

#include <iostream>

#include <stdio.h>

#include <string.h>

class Buton{

public:

std::string text;

Buton(){

printf("Constructorul a fost apelat\n");

}

Buton(std::string a){

this->text = a;

}

~Buton(){

printf("\nButonul a fost eliminat\n");

}

void modificareText(double a){

this->text = a;

}

void afisareText(double a){

printf("%s\n", text);

}

};

class Punct{

public:

double x,y;

Punct(){

printf("Punctul a fost apelat\n");

}

Punct(double a, double b){

this->x = a;

this->y = b;

}

};

class Fereastra{

public:

Punct A,B,C,D;

Buton buton;

Fereastra(){

printf("Constructorul a fost apelat: ");

}

Fereastra(Punct a, Punct b, Punct c, Punct d, Buton e){

this->A.x = a.x;

this->A.y = a.y;

this->B.x = b.x;

this->B.y = b.y;

this->C.x = c.x;

this->C.y = c.y;

this->D.x = d.x;

this->D.y = d.y;

this->buton.text = e.text;

}

~Fereastra(){

printf("Deconstructorul a fost apelat");

}

};

class FereastracuMesaj:public Fereastra{

private:

std::string mesaj;

public:

FereastracuMesaj(){

printf("\nConstructorul a fost apelat");

}

FereastracuMesaj(Punct a, Punct b, Punct c, Punct d, Buton e, std::string f){

this->A.x = a.x;

this->A.y = a.y;

this->B.x = b.x;

this->B.y = b.y;

this->C.x = c.x;

this->C.y = c.y;

this->D.x = d.x;

this->D.y = d.y;

this->buton.text = e.text;

this->mesaj = f;

}

~FereastracuMesaj(){

printf("Deconstructorul a fost apelat\n");

}

friend std::ostream& operator<<(std::ostream& output, const FereastracuMesaj &E){

printf("Punctul A: ("); output << E.A.x << " , " << E.A.y << (") \n");

printf("Punctul B: ("); output << E.B.x << " , " << E.B.y << (") \n");

printf("Punctul C: ("); output << E.C.x << " , " << E.C.y << (") \n");

printf("Punctul D: ("); output << E.D.x << " , " << E.D.y << (") \n");

output << "Textul butonului: " << E.buton.text;

output << "\nMesajul Ferestrei: " << E.mesaj;

return output;

}

};

int main(){

std::string whatever1 = "Button";

std::string whatever2 = "Message";

FereastracuMesaj A(Punct(0,0), Punct(10,0), Punct(0,5), Punct(10,5), whatever1,

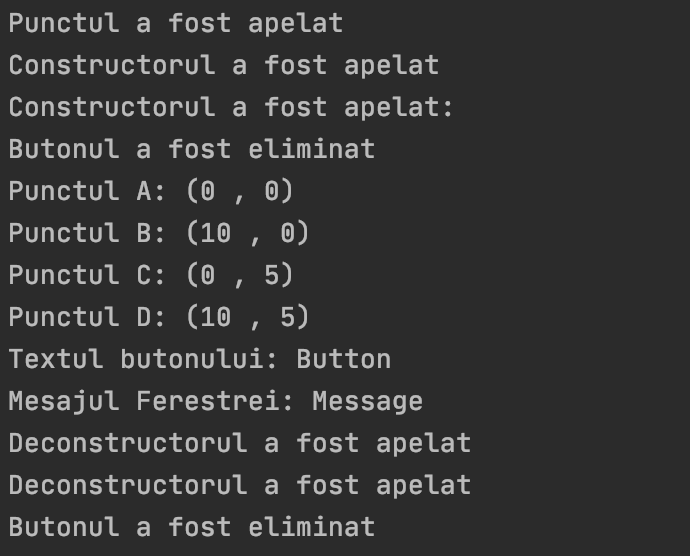
whatever2);

std::cout << A;

printf("\n");

return 0;

}

****

**Figura.2. Rezultatul afișării.**

**Concluzie:**

Elaborând această lucrare de laborator eu am obținut cunoștințe despre moștenire și compoziție, am aflat ce înseamnă moștenirea ce mecanisme stă la baza ei, care sunt regulile de definire a acestei și ce avantaje și dezavantaje are și faptul ca moștenirea există de două tipuri:simpla și multipla, am aflat ce înseamnă compoziția, și care este definirea a compoziției.