MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII AL REPUBLICII MOLDOVA
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică
Departamentul Informatică și Ingineria Sistemelor

Raport

pentru lucrarea de laborator Nr.4

la cursul de " Programarea orientata pe obiecte"

Efectuat: Studentul gr. SI-191 **Comanac Artiom**
Verificat: **Mititelu Vitalie**

Chișinău – 2020

LUCRARE DE LABORATOR NR. 4

Tema: Moștenirea și compoziția

Scopul lucrării:

- studierea moștenirii, avantajele și dezavantajele
- studierea compoziției
- studierea regulilor de definire a moștenirii și compoziției
- studierea formelor de moștenire
- studierea iniţializatorilor
- principiul de substituție
- moștenire și compoziție

Varianta 9

a) Să se creeze o ierarhie a claselor *presă – ziar*, *revistă* și *ziar electronic*. Determinați câmpurile: denumirea ziarului, tirajul, indexul de abonare, periodicitatea de publicare. Pentru setarea câmpurilor textuale, utilizați operatorul *new*. Definiți fluxul de ieșire și fluxul de intrare, constructorul de copiere, operatorul de atribuire prin funcția corespunzătoare a clasei de bază.

b) Să se creeze clasa *Procesor*, care conține informația despre denumirea procesorului, frecvența lui, tehnologiile de producere utilizate și mărimea memoriei. Determinați clasa *computer*, care este compusă dintr-un procesor și alte componente. Pentru setarea câmpurilor textuale, utilizați operatorul *new*. Definiți constructorii, funcția fluxului de ieșire și alte funcții necesare.

Realizarea punctului *a*

main.cpp

```
/*
	Comanac Artiom		SI-191
	LAB #4, a
*/

#include "wrapper.h"

int main() {
	NewsPaper newspaper, np;
	Magazine magazine;
	Online online;

	cin >> newspaper;
	cout << endl;

	NewsPaper copy(newspaper);
	cout << newspaper << endl;
```

```
        cout << "Type: " << newspaper.type() << endl;
        cout << newspaper << endl;

        np = newspaper;
        cout << "Type (Copy): " << np.type() << endl;
        cout << np << endl;
}
```

wrapper.h

```
#pragma once

#include <iostream>
#include <string>
#include <windows.h>

using namespace std;

class Press {
private:
        string title = "";
        int edition = 0;
        int index = 0;
        string period = "";

public:
        Press();
        Press(const Press&);
        Press(string, int, int, int);

        Press& operator=(const Press&); //Press = Press

        friend ostream& operator<<(ostream&, Press&); //cout << Press
        friend istream& operator>>(istream&, Press&); //cin >> Press
};

//ziar
class NewsPaper : public Press {
public:
        string type();
};

//revista
class Magazine : public Press {
        string type();
};

//ziar electronic
class Online : public Press {
        string type();
};
```

functions.cpp

```
#include "wrapper.h"

/*
```

```
	Constructor implicit
*/
Press::Press() {
	this->title = "";
	this->edition = 0;
	this->index = 0;
	this->period = "";
}

/*
	Constructor de copiere
*/
Press::Press(const Press& p) {
	this->title = p.title;
	this->edition = p.edition;
	this->index = p.index;
	this->period = p.period;
}

/*
	Constructor general
*/
Press::Press(string title, int edition, int index, int period) {
	this->title = title;
	this->edition = edition;
	this->index = index;
	this->period = period;
}

/*
	Operator:
		Press = Press
*/
Press& Press::operator=(const Press& p) {
	if (this == &p)
		return *this;

	this->title = p.title;
	this->edition = p.edition;
	this->index = p.index;
	this->period = p.period;

	return *this;
}

/*
	Operator:
		cout << Press;
*/
ostream& operator<<(ostream& out, Press& p) {
	cout << "Title: " << p.title << endl;
	cout << "Edition: " << p.edition << endl;
	cout << "Index: " << p.index << endl;
	cout << "Period: " << p.period << endl;

	return out;
}

/*
	Operator:
		cin >> Press;
*/
istream& operator>>(istream& in, Press& p) {
```

```
        cout << "Title: ";
        cin >> p.title;

        cout << "Edition: ";
        cin >> p.edition;

        cout << "Index: ";
        cin >> p.index;

        cout << "Period: ";
        cin >> p.period;

        return in;
}

/*
        Tipul de presa (ziar)
*/
string NewsPaper::type() {
        return "NewsPaper";
}

/*
        Tipul de presa (revista)
*/
string Magazine::type() {
        return "Magazine";
}

/*
        Tipul de presa (ziar electronic)
*/
string Online::type() {
        return "Online";
}
```

Demonstrarea și testarea

```
NewsPaper newspaper, np;
Magazine magazine;
Online online;

cin >> newspaper;
cout << endl;

NewsPaper copy(newspaper);
cout << newspaper << endl;

cout << "Type: " << newspaper.type() << endl;
cout << newspaper << endl;

np = newspaper;
cout << "Type (Copy): " << np.type() << endl;
cout << np << endl;
```

```
Title: ZiarTOP
Edition: 30000
Index: 123456
Period: daily

Title: ZiarTOP
Edition: 30000
Index: 123456
Period: daily

Type: NewsPaper
Title: ZiarTOP
Edition: 30000
Index: 123456
Period: daily

Type (Copy): NewsPaper
Title: ZiarTOP
Edition: 30000
Index: 123456
Period: daily
```

Realizarea punctului *b*

main.cpp

```
/*
	Comanac Artiom		SI-191
	LAB #4, b
*/

#include "wrapper.h"

int main() {
	Computer homepc(
		"Exedl-PC @exedl",
		"Intel Core i7-8550U",
		1.80,
		"VT-X, AES, SSE, SSE2, SSE3",
		8
	);

	//cin >> homepc;

	cout << homepc << endl;

	//change cpu
	Processor new_cpu("Intel Core i9-8550u", 2.40, "VT-X, AES, SSE, SSE2, SSE3", 16);
	homepc.newcpu(new_cpu);

	cout << homepc << endl;

	//change cpu to custom (cin)
	Processor custom_cpu;
	cin >> custom_cpu;
	homepc.newcpu(custom_cpu);

	cout << endl << homepc;
}
```

wrapper.h

```
#pragma once

#include <iostream>
#include <string>
#include <windows.h>

using namespace std;

class Processor {
private:
	string title = "";
	float freq = 0.00;
	string technologies = "";
	float memory = 0.00;

public:
	Processor();
	Processor(string, float, string, float);

	Processor& operator=(const Processor&); //Processor = Processor
```

```
	friend ostream& operator<<(ostream&, Processor&); //cout << Processor
	friend istream& operator>>(istream&, Processor&); //cin >> Processor
};

class Computer {
private:
	string title = "";
	Processor cpu;
public:
	Computer();
	Computer(string, string, float, string, float);

	void newcpu(Processor);

	Computer& operator=(const Computer&); //Computer = Computer

	friend ostream& operator<<(ostream&, Computer&); //cout << Computer
	friend istream& operator>>(istream&, Computer&); //cin >> Computer
};
```

functions.cpp

```
#include "wrapper.h"

/*
	Constructor implicit
*/
Processor::Processor() {
	this->title = "";
	this->freq = 0.00;
	this->technologies = "";
	this->memory = 0.00;
}

/*
	Constructor general
*/
Processor::Processor(string title, float freq, string technologies, float memory) {
	this->title = title;
	this->freq = freq;
	this->technologies = technologies;
	this->memory = memory;
}

/*
	Operator:
		Processor = Processor
*/
Processor& Processor::operator=(const Processor& p) {
	if (this == &p)
		return *this;

	this->title = p.title;
	this->freq = p.freq;
	this->technologies = p.technologies;
	this->memory = p.memory;

	return *this;
}
```

```
/*
	Operator:
		cout << Processor
*/
ostream& operator<<(ostream& out, Processor& p) {
	cout << "Title: " << p.title << endl;
	cout << "Frequency (GHz): " << p.freq << endl;
	cout << "Technologies: " << p.technologies << endl;
	cout << "Memory (MB): " << p.memory << endl;

	return out;
}

/*
	Operator:
		cin >> Processor
*/
istream& operator>>(istream& in, Processor& p) {
	cout << "Title: ";
	cin >> p.title;

	cout << "Frequency (GHz): ";
	cin >> p.freq;

	cout << "Technologies: ";
	cin >> p.technologies;

	cout << "Memory (MB): ";
	cin >> p.memory;

	return in;
}

/*
	Constructor implicit
*/
Computer::Computer() {
	this->title = "Default PC @User";
	Processor _cpu("Intel", 2.10, "SSE / SSE2 / SSE3", 16);
	this->cpu = _cpu;
}

/*
	Constructor general
*/
Computer::Computer(string pc_title, string cpu_title, float freq, string technologies,
float memory) {
	this->title = pc_title;
	Processor _cpu(cpu_title, freq, technologies, memory);
	this->cpu = _cpu;
}

/*
	Schimbarea procesorului
*/
void Computer::newcpu(Processor _cpu) {
	this->cpu = _cpu;
}

/*
	Operator:
		Computer = Computer
*/
```

```
Computer& Computer::operator=(const Computer& pc) {
	if (this == &pc)
		return *this;

	this->title = pc.title;
	this->cpu = pc.cpu;

	return *this;
}

/*
	Operator:
		cout << Computer
*/
ostream& operator<<(ostream& out, Computer& p) {
	cout << "PC Title: " << p.title << endl;
	cout << p.cpu;

	return out;
}

/*
	Operator:
		cin >> Computer
*/
istream& operator>>(istream& in, Computer& p) {
	cout << "PC Title: ";
	cin >> p.title;

	string cpu_title, technologies;
	float freq, memory;

	cout << "CPU Title: ";
	cin >> cpu_title;

	cout << "Frequency (GHz): ";
	cin >> freq;

	cout << "Technologies: ";
	cin >> technologies;

	cout << "Memory (MB): ";
	cin >> memory;

	Processor _cpu(cpu_title, freq, technologies, memory);
	p.cpu = _cpu;

	return in;
}
```

Demonstrarea și testarea

```
Computer homepc(
    "Exedl-PC @exedl",
    "Intel Core i7-8550U",
    1.80,
    "VT-X, AES, SSE, SSE2, SSE3",
    8
);

//cin >> homepc;

cout << homepc << endl;

//change cpu
Processor new_cpu("Intel Core i9-8550u",
homepc.newcpu(new_cpu);

cout << homepc << endl;

//change cpu to custom (cin)
Processor custom_cpu;
cin >> custom_cpu;
homepc.newcpu(custom_cpu);

cout << endl << homepc;
```

```
PC Title: Exedl-PC @exedl
Title: Intel Core i7-8550U
Frequency (GHz): 1.8
Technologies: VT-X, AES, SSE, SSE2, SSE3
Memory (MB): 8

PC Title: Exedl-PC @exedl
Title: Intel Core i9-8550u
Frequency (GHz): 2.4
Technologies: VT-X, AES, SSE, SSE2, SSE3
Memory (MB): 16

Title: AMD_FX-5800k
Frequency (GHz): 4.20
Technologies: VT-X
Memory (MB): 4

PC Title: Exedl-PC @exedl
Title: AMD_FX-5800k
Frequency (GHz): 4.2
Technologies: VT-X
Memory (MB): 4
```

Concluzii

Efectuând aceasta lucrarea de laborator au fost obținute cunoștințele in diferite domenii limbajului C++: crearea claselor, moștenirea și compoziția lor, utilizarea constructorilor in compoziții de clase.