МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
Кафедра «Програмна інженерія та інформаційні технології управління»

Звіт з лабораторної роботи №2
З предмету «Алгоритмізація та програмування»

Виконав
Студент групи КН-36а
Рубан Ю.Д.
Перевірив:
Ст. в.
Смолін П. О.

Харків
2017

Тема: Використання поліморфізму та шаблонів у C++

1.1 Ієрархія класів

Реалізувати класи "Людина", "Громадянин", "Студент", "Співробітник". В кожному класі визначити віртуальну функцію виведення даних про об'єкт на екран. Створити масив указівників на різні об'єкти ієрархії. В циклі для кожного об'єкта вивести на екран рядок даних про нього.

1.2 Використання поліморфізму

Створити клас для розв'язання завдання 1.2 шостої лабораторної роботи попереднього семестру. Клас повинен містити принаймні дві функції-члена - функцію, яка повертає значення відповідно до індивідуального завдання, а також суто віртуальну функцію, яка викликається з попередньої та визначає ліву частину рівняння або функцію для дослідження (відповідно до завдання).

Клас розташувати в окремому заголовному файлі. У відповідному файлі реалізації здійснити визначення однієї з двох функцій класу.

В іншій одиниці трансляції створити похідний клас із визначенням конкретної функції, яка підлягає дослідженню. У функції main() створити об'єкт похідного класу та здійснити виконання індивідуального завдання.

Примітка: Для обчислення першої (другої) похідної слід додати окремі функції-члени базового класу.

1.3 Узагальнений клас для представлення двовимірного масиву

Переробити клас, створений у завданні 1.3 попередньої лабораторної роботи, на узагальнений (шаблон класу). Реалізувати зовнішню узагальнену функцію знаходження мінімального елемента масиву. В функції `main()` створити масиви цілих, дійсних і простих дробів (раніше створений клас). Для цих трьох масивів здійснити перевірку роботи функції знаходження мінімального значення серед елементів масиву.

У функції `main()` здійснити тестування всіх можливостей класу з перехопленням можливих винятків, знайти мінімальне значення серед елементів масиву, а також розв'язати індивідуальну задачу.

Примітка: для того, щоб можна було знаходити мінімальне значення у масиві дробів, у класі "Простий дріб" необхідно перевантажити операції порівняння.

Хід виконання роботи

1.1 Ієрархія класів

Реалізувано класи "Людина", "Громадянин", "Студент", "Співробітник". В кожному класі визначено віртуальну функцію виведення даних про об'єкт на екран. Створено масив указівників на різні об'єкти ієрархії. В циклі для кожного об'єкта виведено на екран рядок даних про нього.

Код програми:

```
#include<iostream>
using namespace std;

class human
{
private:
	int age;
	char *name;
	char gender;
protected:
	int getAge() { return age; }
	char* getName() { return name; }
	char getGender() { return gender; }
public:
	virtual void info()=0;
	human(int a, char n[], char g) :age(a), name(n), gender(g) {}
};
class citizen : public human
{
private:
	char*status;
protected:
	char *getStatus() { return status; }
public:
	citizen(int a, char n[], char g, char st[]) :human(a, n, g),status(st){}
	virtual void info()
	{
		cout <<"age: " <<getAge() << " name: " << getName() << " gender: " << getGender() << " status: " << status << endl;
	}
};
class student : public citizen
{
private:
	int year;
	char*specialty;
public:
	student(int a, char n[], char g, int y, char sp[],char stat[]) :citizen(a, n, g, stat), year(y), specialty(sp) {}
	virtual void info()
	{
		cout <<"age: "<< getAge() << " name: " << getName() << " gender: " << getGender() << " year: " << year<<" specialty:
"<<specialty <<" status "<<getStatus() << endl;
	}
};
class employee : public citizen
{
private:
	char*prof;
	int salary;
public:
	employee(int a, char n[], char g, char p[], int sal, char stat[]) :citizen(a, n, g,stat), prof(p), salary(sal) {}
	virtual void info()
	{
		cout <<"age: "<<getAge() << " name: " << getName() << " gender: " << getGender() << " proffession: " << prof << "
salary: " << salary <<" status "<<getStatus() <<endl;
	}
};
int main()
{
	human *arr[5];
	arr[0] = new student(19, "bob", 'm', 2, "KN","going to sell his soul to pass the exam");
	arr[1] = new employee(25, "ambal", 'm', "security", 30000,"bodybuilder");
	arr[2] = new citizen(12, "fiona", 'w', "complicated");
	arr[3] = new student(22, "polina", 'w', 6, "history","hard learning");
	arr[4] = new employee(30, "vasya", 'm', "cleaner", 10000,"work hard");
	for (int i = 0; i < 5; i++)
	{
		arr[i]->info();
	}
	system("pause");
	return 0;
}
```

1.2 Використання поліморфізму

Створено клас для розв'язання завдання 1.2 шостої лабораторної роботи попереднього семестру.

3	Найменший корінь

Клас містить дві функції-члена - функцію, яка повертає значення відповідно до індивідуального завдання, а також суто віртуальну функцію, яка викликається з попередньої та визначає ліву частину рівняння або функцію для дослідження (відповідно до завдання).

Клас розташовано в окремому заголовному файлі. У відповідному файлі реалізації здійснено визначення однієї з двох функцій класу.

В іншій одиниці трансляції створено похідний клас із визначенням конкретної функції, яка підлягає дослідженню. У функції main() створено об'єкт похідного класу та здійснено виконання індивідуального завдання.

Код програми:

Головний клас(заголовний файл):

```
#ifndef header
#define header
class ind
{
public:
	double root(double begin, double end,double step=0.00001);
protected:
	virtual double function(double x) = 0;
};

#endif
```

Головний клас(файл реалізації):

```
#include"ind.h"
double ind::root(double begin, double end, double step)
{
	for (double i = begin; i <= end; i += step)
	{
		if (function(i)*function(i + step) < 0)
		{
			return i;
		}
	}
}
```

Похідний клас(заголовний файл):

```
#ifndef SIN
#define SIN
#include"ind.h"
class Sin :public ind
{
protected:
	virtual double function(double x);
};
#endif
```

Похідний клас(файл реалізації):

```
#include"sin.h"
#include"Math.h"
double Sin::function(double x)
{
	return sin(x);
}
```

Функція main():

```
#include<iostream>
#include"sin.h"
using namespace std;

int main()
{
	Sin obj;
	cout <<"min root: "<<obj.root(0, 4) << endl;
	system("pause");
	return 0;
}
```

1.3 Узагальнений клас для представлення двовимірного масиву

Перероблено клас, створений у завданні 1.2 попередньої лабораторної роботи, на узагальнений (шаблон класу). Реалізовано зовнішню узагальнену функцію знаходження мінімального елемента масиву. В функції `main()` створено масиви цілих, дійсних і простих дробів (раніше створений клас). Для цих трьох масивів здійснено перевірку роботи функції знаходження мінімального значення серед елементів масиву.

У функції `main()` здійснено тестування всіх можливостей класу з перехопленням можливих винятків, знайдено мінімальне значення серед елементів масиву, а також розв'язано індивідуальну задачу.

Код програми:

```
#include<iostream>
using namespace std;
//////////////////////////////////////////
template<class T>
class Matrix
{
public:
	class ex
	{
		char*indexEx = "Bad Index";
		char*matrixSameEx = "Matrixes is not same";
		char*multiplyingEx = "cannot multyply matrixes";
		char*sizeEx = "wrong matrix size";
	public:
		ex(int i)
		{
			switch (i)
			{
			case 1:
				cout << indexEx << endl; break;
			case 2:
				cout << matrixSameEx << endl; break;
			case 3:
				cout << multiplyingEx << endl; break;
			case 4:
				cout << sizeEx << endl; break;
			}
		}
	};
	Matrix(int lines, int colons)
	{
		if (lines <= 0 || colons <= 0) { throw ex(4); }
		n = colons;
		m = lines;
		p = new T*[m];
		for (int i = 0; i < m; i++)
		{
			p[i] = new T[n];
		}
	}
	Matrix(const Matrix& A)
	{
		n = A.n;
		m = A.m;
		p = new T*[m];
		for (int i = 0; i < m; i++)
		{
			p[i] = new T[n];
		}
		for (int i = 0; i < m; i++)
		{
			for (int j = 0; j < n; j++)
			{
				p[i][j] = A.p[i][j];
			}
		}
	}
	~Matrix()
```

```
{
		for (int i = 0; i < m; i++)
		{
			delete p[i];
		}
		delete p;
}
T* operator[](int index)
{
		if (index<0)
		{
			throw ex(1);
		}
		return p[index];
}
friend ostream& operator<<(ostream&os, Matrix &M)
{
		int col = M.n;
		int line = M.m;
		for (int i = 0; i < line; i++)
		{
			for (int j = 0; j < col; j++)
			{
				os << M[i][j] << " ";
			}
			os << endl;
		}
		return os;
}
friend istream& operator >> (istream&is, Matrix&M)
{
		T a;
		for (int i = 0; i < M.m; i++)
		{
			for (int j = 0; j < M.n; j++)
			{
				is >> a;
				M[i][j] = a;
			}
		}
		return is;
}
Matrix operator+(Matrix &rv)
{
		if (this->m != rv.m || this->n != rv.n)
		{
			throw ex(2);
		}
		Matrix<T> temp(m, n);
		Matrix<T> th = *this;
		for (int i = 0; i < m; i++)
		{
			for (int j = 0; j < n; j++)
			{
				temp[i][j] = th[i][j] + rv[i][j];
			}
		}
		return temp;
}
Matrix operator-(Matrix &rv)
{
		if (this->m != rv.m || this->n != rv.n)
		{
			throw ex(2);
		}
		Matrix<T> temp(m, n);
		Matrix<T> th = *this;
		for (int i = 0; i < m; i++)
		{
			for (int j = 0; j < n; j++)
			{
				temp[i][j] = th[i][j] - rv[i][j];
			}
		}
		return temp;
}
Matrix operator*(Matrix&rv)
{
		if (this->n != rv.m)
		{
			throw ex(3);
		}
		int newN = rv.n;
		int newM = this->m;
		Matrix<T> th = *this;
		Matrix<T> temp(newM, newN);
		for (int i = 0; i < newM; i++)
			for (int j = 0; j < newN; j++)
			{
				temp[i][j] = 0;
				for (int k = 0; k < this->n; k++)
				{
					temp[i][j] += th[i][k] * rv[k][j];
				}
			}
		return temp;
}
Matrix operator*(T&k)
{
		int newM = this->m;
		int newN = this->n;
		Matrix<T> temp(newM, newN);
		for (int i = 0; i < newM; i++)
		{
```

```
                                for (int j = 0; j < newN; j++)
                                {
                                        temp[i][j] = this->p[i][j] * k;
                                }
                        }
                        return temp;

                }
                int getM()const { return m; }
                int getN()const { return n; }
private:
                int n;
                int m;
                T **p;
};
template<class T>
T min(Matrix<T>A);
template<class T>
void ind(Matrix<T>&A);
////////////////////////////////////////////
int main()
{
                int m, n;
                cout << "enter m, n" << endl;
                cin >> m >> n;
                Matrix<double>M(m,n);
                system("cls");
                cout << "enter elements of (" << m << ", " << n << ") matrix" << endl;
                cin >> M;
                system("cls");
                cout << "min elem " << min(M) << endl;
                system("pause");
                system("cls");
                cout << "enter m, n for second marix" << endl;
                cin >> m >> n;
                Matrix<int>M1(m, n);
                system("cls");
                cout << "enter elements of (" << m << ", " << n << ") matrix2" << endl;

                cin >> M1;
                system("cls");
                cout << "min elem " << min(M1) << endl;
                system("pause");
                return 0;
}
////////////////////////////////////////////
template<class T>
T min(Matrix<T>A)
{
                T min = A[0][0];
                for (int i = 0; i < A.getM(); i++)
                {
                        for (int j = 0; j < A.getN(); j++)
                        {
                                if (min > A[i][j])
                                {
                                        min = A[i][j];
                                }
                        }
                }
                return min;
}
template<class T>
void ind(Matrix<T>&A)
{
                for (int i = 0; i < A.getM(); i++)
                {
                        for (int j = 0; j < A.getN(); j++)
                        {
                                if (A[i][j]==0)
                                {
                                        A[i][j] = 1;
                                }
                        }
                }
}
```

Висновок:

У даній лабораторній роботі я навчився користуватися узагальненням для класів і функцій. А також дізнався про ієрархію класів, що таке успадкування, і що таке поліморфізм.