МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «Програмна інженерія та інформаційні технології управління»

Звіт з лабораторної роботи №3

З предмету «Алгоритмізація та програмування»

Виконав

Студент групи КН-36а

Рубан Ю.Д.

Перевірив: Ст. в. Смолін П. О.

Харків 2017

Використання засобів Стандартної бібліотеки C++

1 Завдання на лабораторну роботу

1.1 Вектор векторів для представлення двовимірного масиву

Розв'язати завдання 1.2 першої лабораторної роботи з використанням вектору векторів Стандартної бібліотеки. Для ініціалізації рядків використати ініціалізатори як фактичні параметри конструкторів векторів. Для всіх циклів, у яких не потрібне явне значення індексу, використовувати циклічну конструкцію **for**, побудовану на діапазоні.

1.2 Представлення й обробка даних про студентів з використанням засобів Стандартної бібліотеки

Створити клас для представлення даних про студента. Клас повинен містити такі елементи даних:

1. номер залікової книжки (**unsigned int**);
2. прізвище (рядок типу std::string);
3. оцінки за останню сесію (std::vector).

Всередині класу реалізувати функції доступу, а також об'явити конструктор, який ініціалізує елементи даних, та функції, які здійснюють:

- обчислення показника, за величиною якого здійснюється сортування відповідно до індивідуального завдання;
- перевірку умови, яка використовується для пошуку даних відповідно до індивідуального завдання.

Здійснити опис класу для представлення групи студентів. В об'єкті такого класу повинні зберігатись дані про студентів у вигляді вектора об'єктів класу, який представляє студента. Клас повинен містити перевантажені операції введення-виведення, а також функції-елементи, які здійснюють

- сортування масиву за ознакою, яка наведена в індивідуальному завданні (з використанням алгоритму sort());
- пошуку даних про студентів, які відповідають умові, наведеній в індивідуальному завданні (з використанням алгоритму for_each()).

Розмістити об'єкти класу "Студент" в черзі з пріоритетом, з якої вилучати об'єкти в порядку зменшення середнього балу.

Індивідуальне завдання:

17	За збільшенням довжини прізвища	З середнім балом в інтервалі "4" - "5"

Хід виконання роботи

1.1 Вектор векторів для представлення двовимірного масиву

Розв'язано завдання 1.2 першої лабораторної роботи з використанням вектору векторів Стандартної бібліотеки. Для ініціалізації рядків використано ініціалізатори як фактичні параметри конструкторів векторів. Для всіх циклів, у яких не потрібне явне значення індексу, використано циклічну конструкцію **`for`**, побудовану на діапазоні.

Код програми:

```
#include<iostream>
#include<vector>
#include<algorithm>
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.resize(m);
		for (int j = 0; j < m; j++)
		{
			p[j].resize(n);
		}
	}
	Matrix(const Matrix& A)
	{
		n = A.n;
		m = A.m;
		p = A.p;
	}
	vector<T>& operator[](int index)
	{
		if (index<0)
		{
			throw ex(1);
		}
		return p[index];
	}
	friend ostream& operator<<(ostream&os, Matrix &M)
	{
		for (vector<T>i : M.p)
		{
			for (T j : i)
			{
				os << j << " ";
			}
			os << endl;
		}
		return os;
	}
	friend istream& operator >> (istream&is, Matrix&M)
	{
```

```
                for (vector<T>&i : M.p)
                {
                        for (T &j : i)
                        {
                                is >> j;
                        }
                }
                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];
                        }
                }
        }
        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; }
        vector<vector<T>>& getP(){ return p;}
private:
        int n;
        int m;
        vector<vector<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;
	cout << "ind " << endl;
	ind(M);
	cout << M;
	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 (vector<T> i : A.getP())
	{
		for (T j : i)
		{
			if (min > j)
				min = j;
		}
	}
	return min;
}
template<class T>
void ind(Matrix<T>&A)
{
	for_each(A.getP().begin(), A.getP().end(), [](vector<T>&i)
	{
		for_each(i.begin(), i.end(), [](T& j)
		{
			if (j == 0)
			{
				j = 1;
			}
		});
	});
}
```

1.2 Представлення й обробка даних про студентів з використанням засобів Стандартної бібліотеки

Створено клас для представлення даних про студента. Клас містить такі елементи даних:

1. номер залікової книжки (**unsigned int**);
2. прізвище (рядок типу std::string);
3. оцінки за останню сесію (std::vector).

Всередині класу реалізувано функції доступу, а також об'явлено конструктор, який ініціалізує елементи даних, та функції, які здійснюють:

- обчислення показника, за величиною якого здійснюється сортування відповідно до індивідуального завдання;

- перевірку умови, яка використовується для пошуку даних відповідно до індивідуального завдання.

Здійснено опис класу для представлення групи студентів. В об'єкті такого класу зберігаються дані про студентів у вигляді вектора об'єктів класу, який представляє студента. Клас містить перевантажені операції введення-виведення, а також функції-елементи, які здійснюють

- сортування масиву за ознакою, яка наведена в індивідуальному завданні (з використанням алгоритму sort());
- пошуку даних про студентів, які відповідають умові, наведеній в індивідуальному завданні (з використанням алгоритму for_each()).

Розміщено об'єкти класу "Студент" в черзі з пріоритетом, з якої вилучаються об'єкти в порядку зменшення середнього балу.

Код програми:

```
#include<iostream>
#include<string>
#include<vector>
#include<algorithm>
#include<functional>
#include<queue>
using namespace std;
class Student
{
private:
	unsigned int bookNum;
	string surname;
	vector<double>marks;
	double avg;
public:
	Student() {}
	Student(string sName, unsigned int book, vector<double>mark)
	{
		surname = sName;
		bookNum = book;
		marks = mark;
		avg = 0;
		for (double &i : marks)
		{
			avg += i;
		}
		avg /= marks.size();
	}
	double getAvg() { return avg; }
	bool condition(Student &a)
	{
		return a.surname.length() > surname.length();
	}
	unsigned int getBookNum() { return bookNum; }
	string getSurname() { return surname; }
	vector<double> getMarks() { return marks; }
	void search()
	{
		if (avg >= 4 && avg <= 5)
		{
			cout << *this <<endl;
		}
	}
	friend ostream& operator<<(ostream& os, Student& obj)
	{
		os << "Surname: " << obj.surname << ", book number " << obj.bookNum<< ", avg mark is: " <<
obj.avg;
		return os;
	}
```

```
};
class Group
{
private:
	vector<Student>group;
public:
	Group(vector<Student> obj)
	{
		group = obj;
	}
	Group() {}
	void indSort()
	{
		sort(group.begin(), group.end(), mem_fun_ref(&Student::condition));
	}
	vector<Student> indSearch()
	{
		vector<Student> search;
		for_each(group.begin(), group.end(), [&](Student&a)
		{
			if (a.getAvg() >= 4 && a.getAvg() <= 5)
				search.push_back(a);
		});
		return search;
	}
	friend ostream& operator<<(ostream& os, Group& obj)
	{
		for (Student &i : obj.group)
		{
			os << i << endl;
		}
		return os;
	}
};
struct compareAvg
{
	bool operator()(Student &a, Student &b)
	{
		return a.getAvg() < b.getAvg();
	}
};
int main()
{
	Student f("gosha", 555, { 4,4,5,3,4,2 });
	Student s("bob", 123, { 4,5,5,5,4,4 });
	Student t("abraham", 157, { 2,4,4,3,4,2 });
	Student o("T", 463, { 4,5,5,4,5,5 });
	Student i("trex", 666, { 2,2,2,2,3,2 });
	Group gr({ f,s,t,o,i });
	cout << "initial group" << endl;
	cout << gr << endl;
	gr.indSort();
	cout << "sorted group" << endl;
	cout << gr<< endl;
	cout << "Students with 4<=avg mark<=5 "<< endl;
	Group bestStudents;
	bestStudents = gr.indSearch();
	cout << bestStudents;
	cout << endl;
	priority_queue < Student, vector<Student>,compareAvg > que;
	que.push(f);
	que.push(s);
	que.push(t);
	que.push(o);
	que.push(i);
	Student temp;
	cout << "Students in the queue are sorted in descending order of the average mark " << endl;
	while ( !que.empty())
	{
		temp = que.top();
		cout << temp << endl;
		que.pop();
	}
	system("pause");
	return 0;

}
```

Висновки

У даній лабораторній роботі я навчився використовувати засоби стандартної бібліотеки C++. Вивчив теоретичний матеріал та додатково дізнався про лямбда вирази, які можуть спростити деякі моменти зв’язані з використанням фукціональних об’єктів і предикатів.