МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа № 8

по дисциплине: «Объектно-ориентированное программирование»

Выполнил: ст. группы ПВ-211
Медведев Дмитрий Сергеевич

Проверил:
Буханов Дмитрий Геннадьевич
Харитонов Сергей Дмитриевич

Белгород 2023 г.

Создание шаблонов классов в C++.

Вариант 8

Цель работы: Получение теоретических знаний о шаблонах классов в C++. Получение практических навыков по созданию классов-шаблонов C++.

Задание

1. Изучить теоретические сведения о шаблонах классов в C++.
2. Разработать программу в соответствии с заданным вариантом задания.
3. Оформить отчет.

Задание 1

Реализовать шаблон класса в соответствии с указанным вариантом. Предусмотреть необходимые методы для работы со структурой данных, указанной в варианте. Предусмотреть исключительные ситуации, которые могут возникнуть в процессе работы.

4-арное дерево

```
#include <iostream>
#include <utility>
#include "vector"

using namespace std;

template<typename T>
class FourNodeTreeElement {
private:
    int index;
    bool isActive;
    T data;
public:
    explicit FourNodeTreeElement(int i) {
        index = i;
        isActive = false;
    }

    bool checkIfIsActive() {
        return isActive;
    }

    void setData(T newData) {
        isActive = true;
```

```
        data = newData;
    }

    T getData() {
        return data;
    }

    int getChildIndex(int childPosition) {
        return (index + 1) * 4 + childPosition;
    }

    int getParentIndex() {
        int parentIndex = (index - 4) / 4;

        if (parentIndex > 0) {
            return parentIndex;
        } else {
            return index;
        }
    }
};

template<typename T>
class FourNodeTree {
private:

public:
    vector<FourNodeTreeElement<T>> treeElements;

    FourNodeTree<T>() {
        int baseSize = 1000;
        treeElements = vector<FourNodeTreeElement<T>>(baseSize);
        for (int i = 0; i < baseSize; i++) {
            treeElements[i] = FourNodeTreeElement<T>(i);
        }
    }

    void insertData(T data) {
        for (int i = 0; i < treeElements.size(); i++) {
            if (!treeElements[i].checkIfIsActive()) {
                treeElements[i].setData(data);
                return;
            }
        }

        throw overflow_error(treeElements.size());
    }

    T getDataByIndex(int index) {
        return treeElements[index];
    }
};
```

Задание 2

Дан файл, в котором храниться структуры следующего типа: кафедра->группа->студент->сессия->предметы->оценки, предметы могут быть разных типов, от этого зависит тип оценки (диф. зачет, зачет, экзамен, прослушал). Каждая структура описывает одного студента. Построить дерево оценок всех студентов одной кафедры, группы. Реализовать сортировку внутри группы, по оценкам студентов, и сортировку предметов по типу внутри студента.

```
enum TypeOfGrade {
    DifferentiatedCredit,
    SimpleCredit,
    Exam,
    Listened
};

class Subject {
public:
    TypeOfGrade gradeType;
    int grade;

    Subject(TypeOfGrade type) {
        gradeType = type;
        grade = 0;
    }

    void setGrade(int Grade) {
        grade = Grade;
    }
};

class Session {
public:
    FourNodeTree<Subject> subjects;

    float getAverageGrade() {
        float sum = 0;
        int numOfSubjects = 0;
        for (int i = 0; i < subjects.treeElements.size(); i++) {
            if (subjects.treeElements[i].checkIfIsActive() &&
(subjects.treeElements[i].getData().gradeType ==

TypeOfGrade::Exam ||

subjects.treeElements[i].getData().gradeType ==

TypeOfGrade::DifferentiatedCredit)) {
                sum += subjects.treeElements[i].getData().grade;
                numOfSubjects++;
            }
        }

        return sum / numOfSubjects;
    }

    vector<Subject> sortSessionByType() {
        vector<Subject> listOfSubjects;
        for (int i = 0; i < subjects.treeElements.size(); i++) {
```

```
            if (subjects.treeElements[i].checkIfIsActive()) {

listOfSubjects.push_back(subjects.treeElements[i].getData());
            }
        }

        for (int i = 0; i < listOfSubjects.size(); i++) {
            for (int j = 0; j < listOfSubjects.size() - (i + 1); j++) {
                if (subjects.treeElements[i].getData().gradeType >
                    subjects.treeElements[i + 1].getData().gradeType) {
                    swap(subjects.treeElements[i], subjects.treeElements[i
+ 1]);
                }
            }
        }

        return listOfSubjects;
    }
};

class Student {
public:
    string name;

    explicit Student(string Name) {
        name = std::move(Name);
    }

    FourNodeTree<Session> sessions;

    float getAverageGrade() {
        return sessions.treeElements[0].getData().getAverageGrade();
    }
};

class Group {
public:
    string name;

    explicit Group(string Name) {
        name = std::move(Name);
    }

    FourNodeTree<Student> students;

    vector<Student> getSortedListByGrades() {
        vector<Student> listOfStudents;
        for (int i = 0; i < students.treeElements.size(); i++) {
            if (students.treeElements[i].checkIfIsActive()) {

listOfStudents.push_back(students.treeElements[i].getData());
            }
        }

        for (int i = 0; i < listOfStudents.size(); i++) {
```

```
            for (int j = 0; j < listOfStudents.size() - (i + 1); j++) {
                if (students.treeElements[i].getData().getAverageGrade() >
                        students.treeElements[i +
1].getData().getAverageGrade()) {
                    swap(students.treeElements[i], students.treeElements[i
+ 1]);
                }
            }
        }

        return listOfStudents;
    }
};

class Department {
public:
    string name;

    explicit Department(string Name) {
        name = std::move(Name);
    }

    FourNodeTree<Group> groups;
};

int main() {
    Subject operationsResearch(TypeOfGrade::DifferentiatedCredit);
    Subject systemModeling(TypeOfGrade::SimpleCredit);
    Subject objectOrientedProgramming(TypeOfGrade::Exam);
    Subject english(TypeOfGrade::SimpleCredit);
    Subject informationTheory(TypeOfGrade::DifferentiatedCredit);

    Session semester3;
    operationsResearch.grade = 5;
    semester3.subjects.insertData(operationsResearch);
    systemModeling.grade = 1;
    semester3.subjects.insertData(systemModeling);
    objectOrientedProgramming.grade = 5;
    semester3.subjects.insertData(objectOrientedProgramming);
    english.grade = 1;
    semester3.subjects.insertData(english);
    informationTheory.grade = 5;
    semester3.subjects.insertData(informationTheory);

    Student student1("Dmitry");
    student1.sessions.insertData(semester3);

    operationsResearch.grade = 4;
    systemModeling.grade = 1;
    objectOrientedProgramming.grade = 5;
    english.grade = 1;
    informationTheory.grade = 4;
    Student student2("Ilya");
    student2.sessions.insertData(semester3);
```

```
    operationsResearch.grade = 5;
    systemModeling.grade = 1;
    objectOrientedProgramming.grade = 5;
    english.grade = 1;
    informationTheory.grade = 4;
    Student student3("Stanislav");
    student2.sessions.insertData(semester3);

    Group group1("PV-211");
    group1.students.insertData(student1);
    group1.students.insertData(student2);
    group1.students.insertData(student3);

    Department department("POVTAS");
    department.groups.insertData(group1);

    vector<Student> sortedStudentList = group1.getSortedListByGrades();

    for (int i = 0; i < sortedStudentList.size(); i++) {
        cout << sortedStudentList[i].name << ", ";
    }
    cout << "\b\n";

    vector<Subject> sortedSubjectList = semester3.sortSessionByType();

    for (int i = 0; i < sortedSubjectList.size(); i++) {
        cout << sortedSubjectList[i].gradeType << ", ";
    }
}
```

Вывод: в ходе лабораторной работы мы получили теоретические знания о шаблонах классов в C++, а также практические навыки по созданию классов-шаблонов C++.