МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Практическое занятие №11-12.

**ОСНОВЫ ООП.**

**Выполнил** студент:

Герасимов Константин Сергеевич

Тюмень-2023

**Работа в аудитории**

**1. Решение задачи 5.1**

***1.1 Постановка задачи***

Описать класс с именем «Студент», содержащий поля:

−фамилия и инициалы;

−место рождения;

−успеваемость (список (массив) из пяти элементов).

Описать класс с именем «Студенческая группа», содержащий поля:

−название группы;

−список (массив) студентов группы (список (массив) объектов класса «Студент»).

Написать программу, выполняющую следующие действия:

−ввод с клавиатуры данных класса типа «Студент»;

−вывод на дисплей фамилий и номеров групп для всех студентов, если средний балл студента больше 4.0, если таких студентов нет, вывести соответствующие сообщение;

−сортировку списка(массива) (одним из способов: методом «пузырька», методом вставки или методом выбора) студентов группы по месту рождения и вывод отсортированного списка(массива) (для сравнения объектов в классе «Студент» перегрузить операции сравнения).

***1.2 Текст программы на языке C++***

Программа была разбита на несколько файлов.  
5\_1.cpp:

#include <iostream>  
  
#include "Student.h"  
#include "Student\_Group.h"  
  
int main(){  
 Student\_Group group1;  
 group1.group\_name = "MOiAIS";  
 while (true){  
 int answer;  
 std::cout << "Write, what you want to do: " << std::endl  
 << "1. Add a student;" << std::endl  
 << "2. View a rating" << std::endl  
 << "3. Sort list of students." << std::endl;  
 std::cin >> answer;  
 switch (answer) {  
 case 1:{  
 std::string nameOfStudent, placeOfStudent;  
 int studentsGrades[5] = {0, 0, 0, 0, 0};  
 std::cout << "How's name of student? (ex. Johns\_B.)" << std::endl;  
 std::cin >> nameOfStudent;  
 std::cout << "Where he was born? (name of city, ex. Houston)" << std::endl;  
 std::cin >> placeOfStudent;  
 std::cout << "Write down his 5 grades in string (ex. 1 2 3 4 5): " << std::endl;  
 for (int & studentsGrade : studentsGrades){  
 std::cin >> studentsGrade;  
 }  
 Student student;  
 student.name = nameOfStudent;  
 student.place\_of\_birth = placeOfStudent;  
 for (int i = 0; i < student.grades.size(); i++){  
 student.grades[i] = studentsGrades[i];  
 }  
  
 group1.add\_student(student);  
 std::cout << "Student was successfully added!" << std::endl;  
 student.print\_info();  
 break;  
 }  
 case 2:{  
 group1.best\_rating();  
 break;  
 }  
 case 3:{  
 group1.sort();  
 break;  
 }  
 default:{  
 break;  
 }  
 }  
 }  
}

Student\_Group.h:

#ifndef STUDENT\_GROUP\_H  
#define STUDENT\_GROUP\_H  
  
#include <iostream>  
#include <vector>  
#include "Student.h"  
  
class Student\_Group {  
public:  
 std::string group\_name;  
 std::vector <Student> list\_of\_students;  
  
 void add\_student(const Student& name\_of\_student);  
 void print\_info();  
 void best\_rating();  
 void sort();  
};  
  
#endif //STUDENT\_GROUP\_H

Student.h

#ifndef STUDENT\_H  
#define STUDENT\_H  
  
#include <iostream>  
#include <vector>  
  
class Student {  
public:  
 std::string name;  
 std::string place\_of\_birth;  
 std::vector <int> grades = {0, 0, 0, 0, 0};  
  
 [[maybe\_unused]] void print\_info();  
 double mid\_grade();  
};  
  
#endif //STUDENT\_H

Student\_Group.cpp:

#include "Student\_Group.h"  
  
void Student\_Group::add\_student(const Student& name\_of\_student) {  
 list\_of\_students.push\_back(name\_of\_student);  
}  
  
void Student\_Group::print\_info() {  
 std::cout << "Group: " << group\_name << std::endl << "List of students: " << std::endl;  
 for (auto & list\_of\_student : list\_of\_students){  
 std::cout << list\_of\_student.name << std::endl;  
 }  
}  
  
void Student\_Group::best\_rating() {  
 int counter = 0;  
 for (auto & student : list\_of\_students){  
 if (student.mid\_grade() >= 4.0){  
 ++counter;  
 std::cout << student.name << " mid range - " << student.mid\_grade() << std::endl;  
 }  
 }  
 if (!counter){  
 std::cout << "Best list is empty! " << std::endl;  
 }  
}  
  
void Student\_Group::sort() {  
 Student value;  
 for (int i = 0; i < list\_of\_students.size()-1; i++){  
 if (list\_of\_students[i].place\_of\_birth[0] > list\_of\_students[i+1].place\_of\_birth[0]){  
 value = list\_of\_students[i];  
 list\_of\_students[i] = list\_of\_students[i+1];  
 list\_of\_students[i+1] = value;  
 }  
 }  
}

Student.cpp:

#include "Student.h"  
  
[[maybe\_unused]] void Student::print\_info() {  
 std::cout << "Name: " << name << "; City: " << place\_of\_birth << "; Grades: ";  
 for (int i = 0; i < 5; i++){  
 if (i < 4){  
 std::cout << grades[i] << ", ";  
 } else {  
 std::cout << grades[i] << ". " << std::endl;  
 }  
 }  
}  
  
double Student::mid\_grade() {  
 double sum = 0;  
 double mid;  
 for (int i = 0; i < 5; i++){  
 sum = sum + grades[i];  
 }  
 mid = sum / 5;  
 return mid;  
}

***1.3 Результат тестирования программы на языке C++***

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**2. Решение задачи 5.3**

***2.1 Постановка задачи***

***Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание***

***2.2 Текст программы на языке C++***

#include <iostream>  
#include <vector>  
#include <string>  
  
using namespace std;  
  
class Book {  
public:  
 string nameOfBook;  
 string authorName;  
 int year = 0;  
 string category;  
  
 void show\_info() const{  
 cout << "Book name: " << nameOfBook << "; Author: " << authorName << "; Year: " << year << endl;  
 }  
};  
  
class Library {  
public:  
 string nameOfLibrary;  
 vector<Book> listOfBooks;  
  
 void add\_books\_in\_library(const Book& book1){  
 listOfBooks.push\_back(book1);  
 }  
 void show\_books(const string& author){  
 for (auto & listOfBook : listOfBooks){  
 if (author == listOfBook.authorName){  
 listOfBook.show\_info();  
 } else {  
 cout << "There is no more books from this author yet..." << endl;  
 }  
 }  
 }  
 void sort\_books(){  
 Book value;  
 for (int i = 0; i < listOfBooks.size()-1; i++){  
 if (listOfBooks[i].category[0] > listOfBooks[i+1].category[0]){  
 value = listOfBooks[i];  
 listOfBooks[i] = listOfBooks[i+1];  
 listOfBooks[i+1] = value;  
 }  
 }  
 for (const auto & listOfBook : listOfBooks){  
 listOfBook.show\_info();  
 }  
 }  
  
};  
  
int main(){  
 Library library1;  
 library1.nameOfLibrary = "Book\_house";  
 while (true){  
 int answer;  
 cout << "Choose what you need:" << endl  
 << "1. Add a book;" << endl  
 << "2. Find a book;" << endl  
 << "3. Sort the book list." << endl;  
 cin >> answer;  
 switch (answer) {  
 case 1:{  
 string book\_name, author\_name, cat;  
 int year\_cycl;  
 cout << "Write down name of your book: (use \_ instead ' ', ex. Crime\_and\_Punishment)" << endl;  
 cin >> book\_name;  
 cout << "Write down name of author: (use \_ instead ' ', ex. Fedor\_Dostoevsky)" << endl;  
 cin >> author\_name;  
 cout << "Write down the book's year: " << endl;  
 cin >> year\_cycl;  
 cout << "Write down the category: " << endl;  
 cin >> cat;  
  
 Book book\_cycl;  
 book\_cycl.nameOfBook = book\_name;  
 book\_cycl.authorName = author\_name;  
 book\_cycl.year = year\_cycl;  
 book\_cycl.category = cat;  
  
 library1.add\_books\_in\_library(book\_cycl);  
 cout << "Your book has successfully added!" << endl;  
 break;  
 }  
 case 2:{  
 string author\_name;  
 cout << "Write down author name to find books: " << endl;  
 cin >> author\_name;  
 library1.show\_books(author\_name);  
 break;  
 }  
 case 3:{  
 library1.sort\_books();  
 break;  
 }  
 default:{  
 break;  
 }  
 }  
 }  
}

***2.3 Результат тестирования программы на языке C++***

***Изображение выглядит как текст, снимок экрана

Автоматически созданное описание***Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**3. Решение задачи 5.4**

***3.1 Постановка задачи***

Изображение выглядит как текст, Шрифт, снимок экрана, алгебра

Автоматически созданное описание

***3.2 Текст программы на языке C++***

#include <iostream>

#include <vector>

using namespace std;

class Matrix {

private:

int rows;

int cols;

vector<vector<double> > data;

public:

Matrix(int rows, int cols) : rows(rows), cols(cols), data(rows, vector<double>(cols)) {}

void print() {

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

cout << data[i][j] << " ";

}

cout << endl;

}

}

void set(int row, int col, double value) {

data[row][col] = value;

}

double get(int row, int col) {

return data[row][col];

}

Matrix operator+(const Matrix& other) const {

Matrix result(rows, cols);

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

result.set(i, j, data[i][j] + other.data[i][j]);

}

}

return result;

}

Matrix operator-(const Matrix& other) const {

Matrix result(rows, cols);

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

result.set(i, j, data[i][j] - other.data[i][j]);

}

}

return result;

}

Matrix operator+(double scalar) const {

Matrix result(rows, cols);

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

result.set(i, j, data[i][j] + scalar);

}

}

return result;

}

Matrix operator-(double scalar) const {

Matrix result(rows, cols);

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

result.set(i, j, data[i][j] - scalar);

}

}

return result;

}

bool operator==(const Matrix& other) const {

if (rows != other.rows || cols != other.cols) {

return false;

}

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

if (data[i][j] != other.data[i][j]) {

return false;

}

}

}

return true;

}

bool operator!=(const Matrix& other) const {

return !(\*this == other);

}

Matrix transpose() const {

Matrix result(cols, rows);

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

result.set(j, i, data[i][j]);

}

}

return result;

}

Matrix inverse() const {

if (rows != cols) {

cerr << "Error: matrix is not square" << endl;

return Matrix(rows, cols);

}

Matrix result(rows, cols\*2);

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

result.set(i, j, data[i][j]);

}

result.set(i, i+cols, 1.0);

}

for (int i = 0; i < rows; i++) {

if (result.get(i, i) == 0.0) {

cerr << "Error: matrix is not invertible" << endl;

return Matrix(rows, cols);

}

double pivot = result.get(i, i);

for (int j = 0; j < cols\*2; j++) {

result.set(i, j, result.get(i, j) / pivot);

}

for (int j = 0; j < rows; j++) {

if (j == i) {

continue;

}

double factor = result.get(j, i);

for (int k = 0; k < cols\*2; k++) {

result.set(j, k, result.get(j, k) - factor \* result.get(i, k));

}

}

}

Matrix inverse(rows, cols);

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

inverse.set(i, j, result.get(i, j+cols));

}

}

return inverse;

}

};

int main() {

Matrix m(2, 3);

m.set(0, 0, 1.0);

m.set(0, 1, 2.0);

m.set(0, 2, 3.0);

m.set(1, 0, 4.0);

m.set(1, 1, 5.0);

m.set(1, 2, 6.0);

cout << "Matrix m:" << endl;

m.print();

Matrix n(2, 3);

n.set(0, 0, 2.0);

n.set(0, 1, 4.0);

n.set(0, 2, 6.0);

n.set(1, 0, 8.0);

n.set(1, 1, 10.0);

n.set(1, 2, 12.0);

cout << "Matrix n:" << endl;

n.print();

Matrix sum = m + n;

cout << "m + n:" << endl;

sum.print();

Matrix difference = m - n;

cout << "m - n:" << endl;

difference.print();

Matrix scalarSum = m + 2.0;

cout << "m + 2:" << endl;

scalarSum.print();

Matrix scalarDifference = m - 2.0;

cout << "m - 2:" << endl;

scalarDifference.print();

Matrix transposeM = m.transpose();

cout << "Transpose of m:" << endl;

transposeM.print();

Matrix inverseM = m.inverse();

cout << "Inverse of m:" << endl;

inverseM.print();

bool isEqual = m == n;

cout << "m == n: " << isEqual << endl;

bool isNotEqual = m != n;

cout << "m != n: " << isNotEqual << endl;

return 0;

}

***3.3 Результат тестирования программы на языке C++***

Изображение выглядит как снимок экрана, текст

Автоматически созданное описание

**Домашняя работа**

1. ***Тестовое задание***

Какой из следующих методов является конструктором класса "Книга"?

a) void Book();

b) Book();

c) Book(string title);

d) string Book();

* 1. ***Ответ***

c) Book(string title)

1. ***Тестовое задание***

Какой модификатор доступа используется по умолчанию для свойств и методов класса в C++?

a) private

b) public

c) protected

d) static

1. ***Ответ***
2. Private

***7. Практическое задание***

Разработайте класс "Банковский счет", который представляет собой счет в банке. Класс должен иметь следующие свойства и методы:

Свойства:

- Номер счета (accountNumber) - целое число

- Баланс счета (balance) - число с плавающей точкой

- Имя владельца счета (ownerName) – строка

Так же разработайте методы, которые выводят информацию о счете, сумму пополнения и сумму снятия.

***7.1 Листинг программы***

#include <iostream>  
#include <string>  
  
class BankAccount {  
private:  
 int accountNumber;  
 double balance;  
 std::string ownerName;  
  
public:  
 BankAccount(int accNumber, double initialBalance, std::string owner) {  
 accountNumber = accNumber;  
 balance = initialBalance;  
 ownerName = owner;  
 }  
  
 void deposit(double amount) {  
 balance += amount;  
 }  
  
 void withdraw(double amount) {  
 if (balance >= amount) {  
 balance -= amount;  
 } else {  
 std::cout << "Ошибка: на счете недостаточно средств." << std::endl;  
 }  
 }  
  
 void displayAccountInfo() {  
 std::cout << "Номер счета: " << accountNumber << std::endl;  
 std::cout << "Имя владельца: " << ownerName << std::endl;  
 std::cout << "Баланс счета: $" << balance << std::endl;  
 }  
};

***7.2 Примерные входные данные***

int main() {  
 *// Создание объекта класса "Банковский счет"* BankAccount account(12345, 0.00, "John Doe");  
  
 *// Внесение депозита на счет* account.deposit(500.00);  
 std::cout << "Пополнение счета на $500.00" << std::endl;  
  
 *// Снятие средств со счета* account.withdraw(200.00);  
 std::cout << "Снятие со счета $200.00" << std::endl;  
  
 *// Вывод информации о счете* std::cout << std::endl << "Обновленная информация о счете:" << std::endl;  
 account.displayAccountInfo();  
  
 return 0;  
}

***7.3 Пример работы***

***Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание***