**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №1**

**по дисциплине «Программирование»**

Тема: Типы данных, определяемые пользователем

|  |  |  |
| --- | --- | --- |
| Студент гр. 2372 |  | Соколовский В.Д. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2023

**Цель работы.**

Изучение и организация структур. Получение практических навыков работы со структурами. Определение преимуществ и недостатков использования структур.

**Основные теоретические положения.**

Структура – группа связанных между собой, как правило, разнотипных переменных, объединенных в единый объект, в отличие от массива, все элементы которого строго однотипны. В языке программирования C++ структура является видом класса и обладает всеми его свойствами.

Для определения структуры применяется ключевое слово *struct*. Каждая входящая в структуру переменная называется членом (полем, элементом) структуры и описывается типом данных и именем. Поля структуры могут быть любого типа данных.

На структуры во многом похожи объединения. Объединения также хранят набор элементов, но в отличие от структуры все элементы объединения имеют нулевое смещение. То есть разные элементы занимают в памяти один и тот же участок.

Объединения объявляются ключевым словом *union*. В памяти члены объединения располагаются по одному адресу. Объединения не могут хранить одновременно несколько различных значений, они позволяют интерпретировать несколькими различными способами содержимое одной и той же области памяти.

Перечисления служат для определения пользовательских типов данных, значения которых принадлежат одному из элементов в некотором списке именованных целочисленных констант. Перечисление задается ключевым словом *enum*.

Для упрощения чтения программы, можно задать некоторому типу данных новое имя с помощью ключевого слова *typedef*. Объявления *typedef* можно использовать для создания более коротких или более понятных имен для типов, уже определенных в языке C/C++ или объявленных пользователем.

**Постановка задачи.**

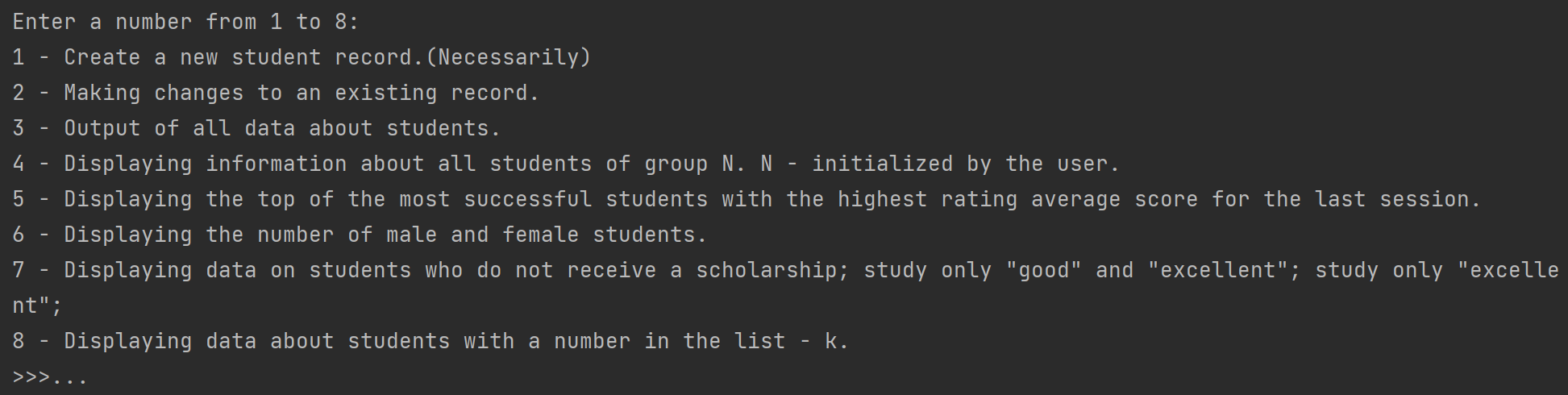
Необходимо создать массив структур, содержащий информацию о студентах: ФИО, пол, номер группы, номер в списке группы, оценки за прошедшую сессию.

Написать функции, реализующие операции со структурами:

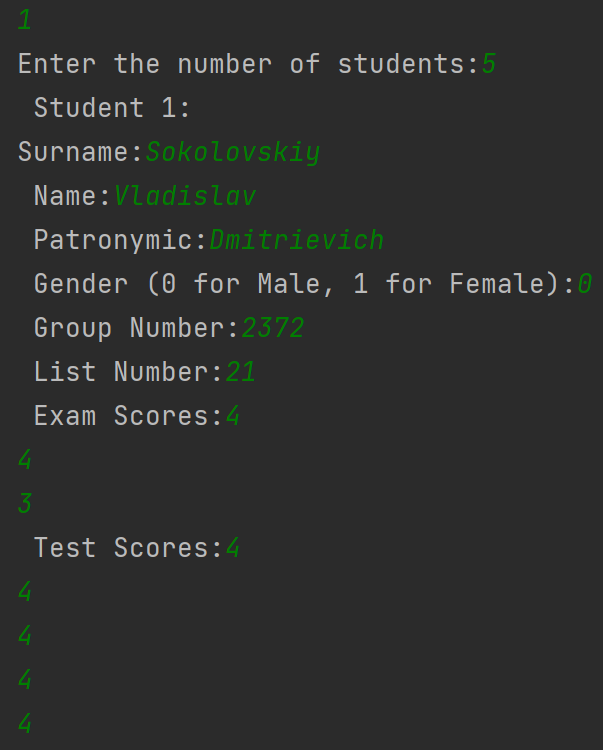
1. Создание новой записи о студенте.
2. Внесение изменений в уже имеющуюся запись.
3. Вывод всех данных о студентах.
4. Вывод информации обо всех студентах группы N.
5. Вывод самых успешных студентов с наивысшим по рейтингу средним баллом за прошедшую сессию.
6. Вывод количества студентов мужского и женского пола.
7. Вывод данных о студентах, которые не получают стипендию; учатся только на «хорошо» и «отлично»; учатся только на «отлично».
8. Вывод данных о студентах, имеющих номер в списке – k.

**Выполнение работы.**

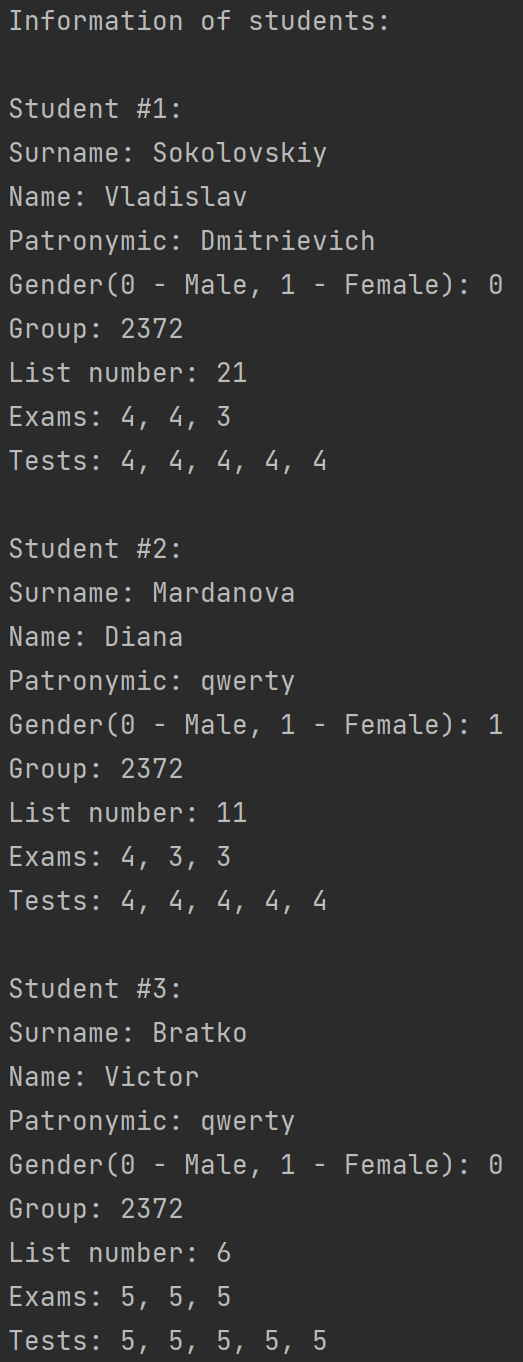
Код программы представлен в приложении А.



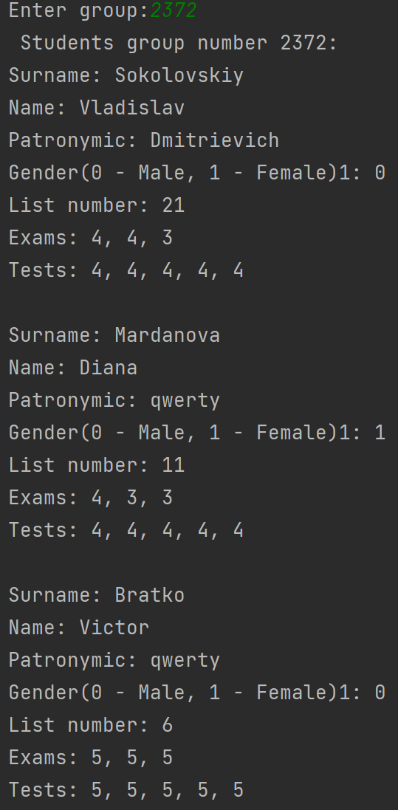
1. Следующий шаг зависит от введенной команды, если пользователь ввёл:
   1. Создания новой записи(вводить с клавиатуры):



* 1. “2”, то открывается редактирование записи о студенте.
  2. “3”, то выводятся все имеющиеся записи.



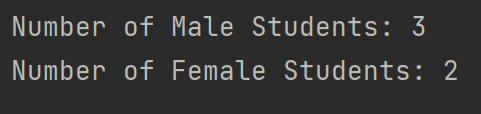
* 1. “4”, то ожидается ввод номера группы, после чего выводятся записи с номером группы, соответствующим введенному.



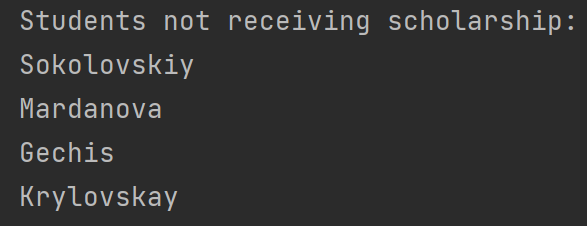
* 1. “5”, то выводится таблица, отсортированная по значениям среднего балла.



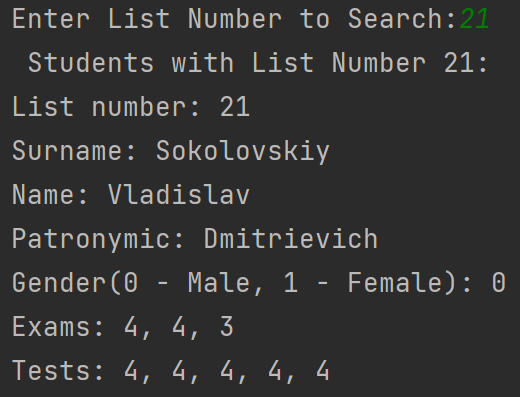
* 1. “6”, то выводится количество студентов мужского и женского полов соответственно.



* 1. “7”, то выводится список кто не получает стипендию из студентов.



* 1. “4”, то ожидается ввод номера студента в группе, после чего выводятся все записи с соответствующим номером.



**Выводы.**

В ходе работы были изучены структур, получены практические навыки работы с ними. Также были определены преимущества и недостатки использования структур.

Приложение А

рабочий код

#include <iostream>

#include <string>

using namespace std;

const int maxStudents = 50;

const int numExams = 3;

const int numTests = 5;

enum Gender {

Male,

Female

};

struct ExamScores {

float exam[numExams];

float test[numTests];

};

struct Student {

string name;

string surname;

string patr;

Gender gender;

int groupNum;

int listNum;

ExamScores scores;

};

void createStudents(Student students[], int& numStudents){

cout << "Enter the number of students: ";

cin >> numStudents;

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

cout << "Student " << i + 1 << ":\n";

cout << "Surname: ";

cin >> students[i].surname;

cout << "Name: ";

cin >> students[i].name;

cout << "Patronymic: ";

cin >> students[i].patr;

cout << "Gender (0 for Male, 1 for Female): ";

int inputG;

cin >> inputG;

students[i].gender = static\_cast<Gender>(inputG); //Преобразует выражение в тип type-id,

// основываясь только на типах, присутствующих в выражении.

// т.е. преобразуется int в 'gender'

cout << "Group Number: ";

cin >> students[i].groupNum;

cout << "List Number: ";

cin >> students[i].listNum;

cout << "Exam Scores: ";

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

cin >> students[i].scores.exam[j];

}

cout << "Test Scores: ";

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

cin >> students[i].scores.test[j];

}

}

}

void editStudents(Student students[], int& numStudents){

int studentIndex;

cout << "Enter number of student to edit: ";

cin >> studentIndex;

if(studentIndex < 1 || studentIndex > numStudents){

cout << "Error!" << endl;

return;

}

Student &student = students[studentIndex - 1];

cout << "Enter a new information of student:" << endl;

cout << "Surname: ";

cin >> student.surname;

cout << "Name: ";

cin >> student.name;

cout << "Patronymic: ";

cin >> student.patr;

cout << "Gender(0 - Male, 1 - Female): ";

int inputG;

cin >> inputG;

student.gender = static\_cast<Gender>(inputG);

cout << "Group: ";

cin >> student.groupNum;

cout << "List student: ";

cin >> student.listNum;

cout << "Exams: ";

cin >> student.scores.exam[0] >> student.scores.exam[1] >> student.scores.exam[2];

cout << "Tests: ";

cin >> student.scores.test[0] >> student.scores.test[1] >> student.scores.test[2] >> student.scores.test[3]

>> student.scores.test[4];

}

void listStudents(Student students[], int& numStudents){

cout << "Information of students:" << endl;

cout << "\n";

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

cout << "Student #" << i + 1 << ":" << endl;

cout << "Surname: " << students[i].surname << endl;

cout << "Name: " << students[i].name << endl;

cout << "Patronymic: " << students[i].patr << endl;

cout << "Gender(0 - Male, 1 - Female): " << students[i].gender << endl;

cout << "Group: " << students[i].groupNum << endl;

cout << "List number: " << students[i].listNum << endl;

cout << "Exams: " << students[i].scores.exam[0] << ", " << students[i].scores.exam[1] << ", "

<< students[i].scores.exam[2] << endl;

cout << "Tests: " << students[i].scores.test[0] << ", " << students[i].scores.test[1] << ", "

<< students[i].scores.test[2] << ", " << students[i].scores.test[3]

<< ", " << students[i].scores.test[4] << endl;

cout << "\n";

}

}

void groupStudents(Student students[], int& numStudents, int N){

cout << "Students group number " << N << ":" << endl;

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

if(students[i].groupNum == N){

cout << "Surname: " << students[i].surname << endl;

cout << "Name: " << students[i].name << endl;

cout << "Patronymic: " << students[i].patr << endl;

cout << "Gender(0 - Male, 1 - Female)1: " << students[i].gender << endl;

cout << "List number: " << students[i].listNum << endl;

cout << "Exams: " << students[i].scores.exam[0] << ", " << students[i].scores.exam[1] << ", "

<< students[i].scores.exam[2] << endl;

cout << "Tests: " << students[i].scores.test[0] << ", " << students[i].scores.test[1] << ", "

<< students[i].scores.test[2] << ", " << students[i].scores.test[3]

<< ", " << students[i].scores.test[4] << endl;

cout << "\n";

}

}

}

void topStudents(Student students[], int& numStudents, int size) {

const int topCount = 5;

struct StudentRating {

int index;

float rating;

} topStudents[topCount]; // список описателя

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

topStudents[i].index = -1;

topStudents[i].rating = 0;

}

// вычисление рейтинга для каждого студента

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

float sum = 0;

for(int j = 0; j < 3; j++){

sum += students[i].scores.exam[j];

}

for(int j = 0; j < 5; j++){

sum += students[i].scores.test[j];

}

float rating = sum / 8;

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

if(topStudents[j].index == -1 || rating > topStudents[j].rating){

for(int k = topCount - 1; k > j; k--){

topStudents[k] = topStudents[k - 1];

}

topStudents[j].index = i;

topStudents[j].rating = rating;

break;

}

}

}

cout << "Top students:" << endl;

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

if(topStudents[i].index == -1){

break;

}

int index = topStudents[i].index;

cout << "Student " << index + 1 << ": " << students[index].surname << ", rating: " << topStudents[i].rating

<< endl;

}

}

void genderStudents(Student students[], int& numStudents){

int numMale = 0, numFemale = 0;

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

if(students[i].gender == Male){

numMale++;

}

else{

numFemale++;

}

}

cout << "\n";

cout << "Number of Male Students: " << numMale << endl;

cout << "Number of Female Students: " << numFemale << endl;

}

void scholarshipStudents(Student students[], int& numStudents){

cout << "Students not receiving scholarship:" << "\n";

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

if(students[i].scores.exam[0] < 4){

cout << students[i].surname << endl;

}

else if(students[i].scores.exam[1] < 4){

cout << students[i].surname << endl;

}

else if(students[i].scores.exam[2] < 4){

cout << students[i].surname << endl;

}

else if(students[i].scores.test[0] < 4){

cout << students[i].surname << endl;

}

else if(students[i].scores.test[1] < 4){

cout << students[i].surname << endl;

}

else if(students[i].scores.test[2] < 4){

cout << students[i].surname << endl;

}

else if(students[i].scores.test[3] < 4){

cout << students[i].surname << endl;

}

else if(students[i].scores.test[4] < 4){

cout << students[i].surname << endl;

}

}

cout << "\n";

}

void studentsListNum(Student students[], int& numStudents, int k){

cout << "Students with List Number " << k << ":" << endl;

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

if(students[i].listNum == k){

cout << "List number: " << students[i].listNum << endl;

cout << "Surname: " << students[i].surname << endl;

cout << "Name: " << students[i].name << endl;

cout << "Patronymic: " << students[i].patr << endl;

cout << "Gender(0 - Male, 1 - Female): " << students[i].gender << endl;

cout << "Exams: " << students[i].scores.exam[0] << ", " << students[i].scores.exam[1] << ", "

<< students[i].scores.exam[2] << endl;

cout << "Tests: " << students[i].scores.test[0] << ", " << students[i].scores.test[1] << ", "

<< students[i].scores.test[2] << ", " << students[i].scores.test[3]

<< ", " << students[i].scores.test[4] << endl;

cout << "\n";

}

}

}

int main(){

Student students[maxStudents];

int numStudents;

while(true){

char number;

cout << "\nEnter a number from 1 to 8:\n";

cout << "1 - Create a new student record.(Necessarily)\n";

cout << "2 - Making changes to an existing record.\n";

cout << "3 - Output of all data about students.\n";

cout << "4 - Displaying information about all students of group N. N - initialized by the user.\n";

cout << "5 - Displaying the top of the most successful students with the highest rating average score for the last session.\n";

cout << "6 - Displaying the number of male and female students.\n";

cout << "7 - Displaying data on students who do not receive a scholarship; study only \"good\" and \"excellent\"; study only \"excellent\";\n";

cout << "8 - Displaying data about students with a number in the list - k.\n";

cout << ">>>...\n";

cin.sync();

cin >> number;

switch (number) {

case '1':{

createStudents(students,numStudents); // 1

break;

}

case '2': {

editStudents(students, numStudents); // 2

break;

}

case '3': {

listStudents(students, numStudents); // 3

break;

}

case '4': {

int N;

cout << "Enter group: ";

cin >> N;

groupStudents(students, numStudents, N); // 4

break;

}

case '5': {

int size;

cout << "Enter number of top: ";

cin >> size;

topStudents(students, numStudents, size); // 5

break;

}

case '6': {

genderStudents(students, numStudents); // 6

break;

}

case '7': {

scholarshipStudents(students, numStudents); // 7

break;

}

case '8': {

int k;

cout << "Enter List Number to Search: ";

cin >> k;

studentsListNum(students, numStudents, k); // 8

break;

}

default:{

cout << "Error" << "\n";

}

}

}

}