**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: Структуры.

|  |  |  |
| --- | --- | --- |
| Студентка гр. 2372 |  | Котлова П. В. |
| 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.**

Структуры представляют собой группы связанных между собой, как правило, разнотипных переменных, объединенных в единый объект, в отличие от массива, все элементы которого однотипны. В языке C++ структура является видом класса и обладает всеми его свойствами.  
Примечание  
 Класс - в ООП представляет собой шаблон для создания объектов, обеспечивающий начальные значения состояний: инициализация полей-переменных и реализация поведения функций или методов. ООП в данном курсе не рассматривается, но иногда будут упоминаться понятия, связанные с ним.

Для определения структуры применяется ключевое слово **struct**, а сам формат определения выглядит следующим образом:

struct [имя\_типа] {

тип\_1 элемент\_1;

тип \_2 элемент\_2;

…

тип\_k элемент\_k;

} [ список\_описателей ];

Каждая входящая в структуру переменная называется членом (полем, элементом) структуры и описывается типом данных и именем. Поля структуры могут быть любого типа данных. Их количество не лимитировано.

Вся эта конструкция является инструкцией языка программирования, поэтому после нее всегда должен ставиться символ ‘;’.

struct Student

{

int age;

char name[10];

int group;

};

Любая структурная переменная занимает в памяти определенное положение, характеризующееся конкретным адресом. Для работы с адресами структурных переменных (как и для простых переменных) можно использовать указатели. Указатели на структурные переменные определяются точно так же, как и для обычных переменных. Разыменование указателя (обращение к данным по адресу, хранящемуся в указателе) осуществляется также обычным образом.

Через указатели можно работать с отдельными полями структур. Для доступа к полю структуры через указатель используется оператор ‘->’ («стрелка»), а не «точка».

**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. При запуске программы считываются данные из *data.txt*. Затем пользователю выводится подсказка команды, выводящей меню доступных функций. Ожидается ввод команды с клавиатуры. (рис. 1).

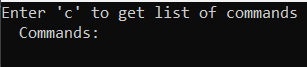


Рис .

1. Следующий шаг зависит от введенной команды, которую введет пользователь:
   1. “1”, то открывается меню создания новой записи (рис. 2).

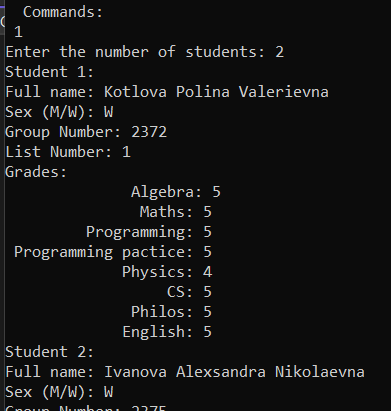


Рис 2.

* 1. “2”, Изменения в имеющуюся запись. Пользователь вводит номер студента, чьи данные нужно изменить (рис. 3).

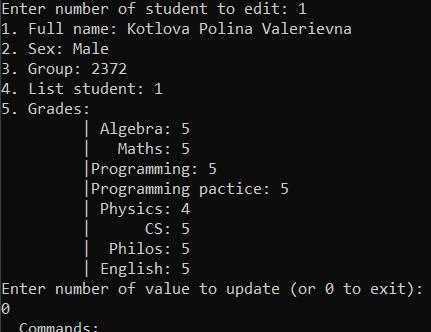
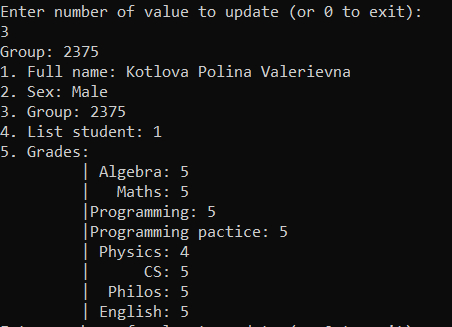


Рис. 3.

Затем вводится номер пункта, который нужно изменить:



Pиc. 4.

* 1. “3”, то выводятся все имеющиеся записи (рис. 4).

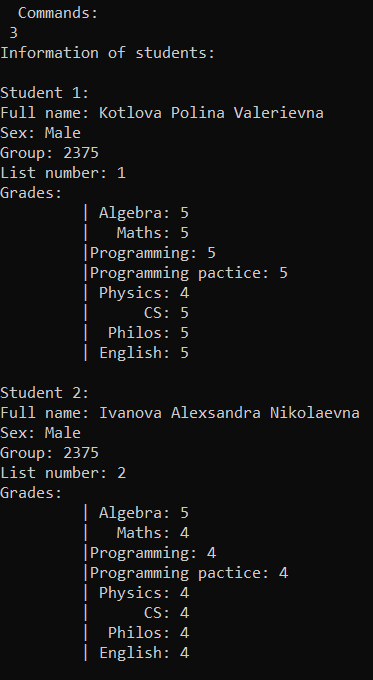


Рис. 4.

* 1. “4”, то ожидается ввод номера группы, после чего выводятся записи с номером группы, соответствующим введенному (рис. 5).

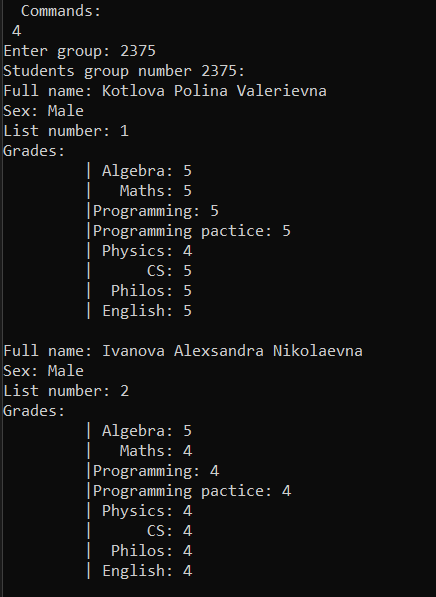


Рис. 5.

* 1. “5”, пользователь вводит количество лучших студентов в топе, после чего выводится топ студентов с наивысшим по рейтингу средним баллом за прошедшую сессию (рис. 6).

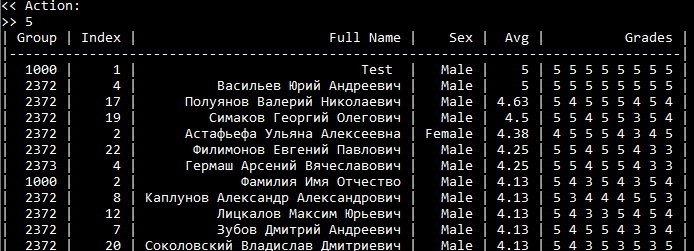


Рис. 6.

* 1. “6”, то выводится количество студентов мужского и женского полов соответственно (рис. 7).

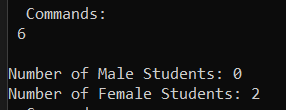


Рис. 7.

* 1. “7”, то выводится информация о студентах с их успеваемостью (отличники, хорошисты, ударники) (рис. 8).

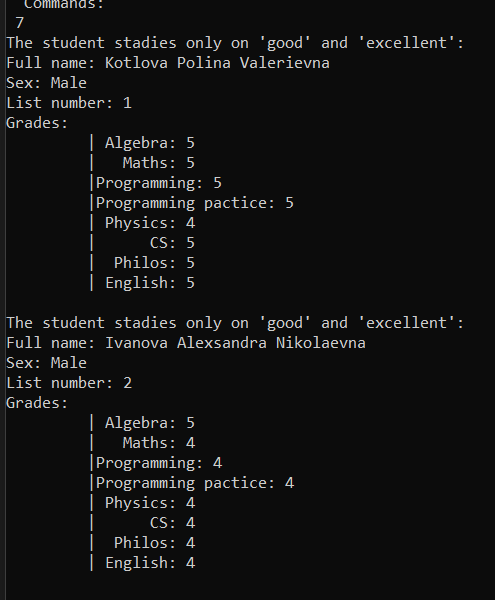


Рис. 8.

* 1. “8”, то ожидается ввод номера студента в группе, после чего выводятся все записи с соответствующим номером (рис. 9).

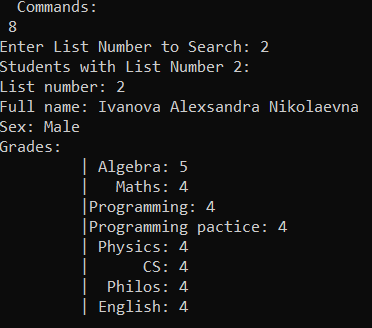


Рис. 9.

* 1. “с”, то выводится меню доступных команд (рис. 10).

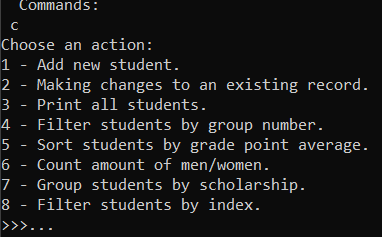


Рис. 10.

* 1. Любой символ или набор символов, отличный от существующий команд, то будет выведено сообщение об ошибке (рис. 11).



Рис. 11.

**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

// ConsoleApplication5.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#include <iostream>

#include <iomanip>

#include <fstream>

#include <Windows.h>

#include <vector>

#include <cmath>

#include <fstream>

#include <string>

#include <algorithm>

using namespace std;

const int maxStud = 40;

const string subjects[] = {

"Algebra", "Maths", "Programming", "Programming pactice", "Physics", "CS", "Philos", "English"

};

struct Student {

string fullName;

string sex;

unsigned short group{};

unsigned short listNum{};

unsigned short grades[8]{};

};

void loadFromFile(Student students[], int& numStudents) //загрузка данных из файла

{

ifstream in;

in.open("data.txt");

while (!in.eof())

{

in >> students[numStudents].fullName;

in >> students[numStudents].sex;

in >> students[numStudents].group;

in >> students[numStudents].listNum;

numStudents++;

}

numStudents--;

in.close();

}

void saveToFile(Student students[], int& numStudents) // Запись данных в файл

{

ofstream out;

out.open("data.txt");

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

{

out << students[i].fullName << endl << students[i].sex << endl << students[i].group << endl << students[i].listNum << endl;

}

out.close();

}

void addStudents(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 << "Full name: ";

getline(cin.ignore(), students[i].fullName);

cout << "Sex (M/W): ";

cin >> students[i].sex;

cout << "Group Number: ";

cin >> students[i].group;

cout << "List Number: ";

cin >> students[i].listNum;

cout << "Grades:\n ";

for (int j = 0; j < 8; ++j) {

cout << setw(20) << subjects[j] << ": ";

cin >> students[i].grades[j];

if (students[i].grades[j] <= 2) {

cout << students[i].fullName << "Expelled. Not saved to the DB.\n";

break;

}

}

}

}

int 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 0;

}

Student& student = students[studentIndex - 1];

unsigned short input = 0;

do {

cout << "1. " << "Full name: " << student.fullName << endl;

cout << "2. " << "Sex: " << (student.sex = "M" ? "Male" : "Female") << endl;

cout << "3. " << "Group: " << student.group << endl;

cout << "4. " << "List student: " << student.listNum << endl;

cout << "5. " << "Grades: " << endl;

for (int i = 0; i < 8; ++i) {

cout << setw(10) << '|' << setw(8);

cout << subjects[i] << ": " << student.grades[i] << endl;

}

cout << "Enter number of value to update (or 0 to exit):\n";

cin >> input;

switch (input) {

case 0: break;

case 1: {

cout << "Full name: ";

getline(cin.ignore(), student.fullName);

break;

}

case 2: {

cout << "Sex (M/W): ";

cin >> student.sex;

break;

}

case 3: {

cout << "Group: ";

cin >> student.group;

break;

}

case 4: {

cout << "List student: ";

cin >> student.listNum;

break;

}

case 5: {

cout << "Grades:\n ";

for (int j = 0; j < 8; ++j) {

cout << setw(20) << subjects[j] << ": ";

cin >> student.grades[j];

if (student.grades[j] <= 2) {

cout << student.fullName << "Expelled. Not saved to the DB.\n";

break;

}

}

}

}

} while (input != 0);

students[studentIndex - 1] = student;

return 0;

}

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 << "Full name: " << students[i].fullName << endl;

cout << "Sex: " << (students[i].sex = "M" ? "Male" : "Female") << endl;

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

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

cout << "Grades: " << endl;

for (int j = 0; j < 8; ++j) {

cout << setw(10) << '|' << setw(8);

cout << subjects[j] << ": " << students[i].grades[j] << 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].group == N) {

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

cout << "Sex: " << (students[i].sex = "M" ? "Male" : "Female") << endl;

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

cout << "Grades: " << endl;

for (int j = 0; j < 8; ++j) {

cout << setw(10) << '|' << setw(8);

cout << subjects[j] << ": " << students[i].grades[j] << endl;

}

cout << "\n";

}

}

}

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

const int topCount = 5;

struct StudentRating {

int index;

float rating;

} topStudents[topCount];

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

topStudents[i].index = -1;

topStudents[i].rating = 0;

}

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

float sum = 0;

for (int j = 0; j < 8; j++) {

sum += students[i].grades[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].fullName << ", rating: " << topStudents[i].rating

<< endl;

}

}

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

int numMale = 0, numFemale = 0;

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

if (students[i].sex == "M") {

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";

bool grad3 = false, grad4 = false, grad5 = false;

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

int minGrade = 5;

for (int j = 0; j < 8; j++) {

minGrade = min(minGrade, students[i].grades[j]);

}

if (minGrade == 3) {

cout << "The student doesnt receive a scholarship:" << endl;

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

cout << "Sex: " << (students[i].sex = "M" ? "Male" : "Female") << endl;

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

cout << "Grades: " << endl;

for (int j = 0; j < 8; ++j) {

cout << setw(10) << '|' << setw(8);

cout << subjects[j] << ": " << students[i].grades[j] << endl;

}

cout << "\n";

}

else if (minGrade == 4) {

cout << "The student stadies only on 'good' and 'excellent': " << endl;

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

cout << "Sex: " << (students[i].sex = "M" ? "Male" : "Female") << endl;

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

cout << "Grades: " << endl;

for (int j = 0; j < 8; ++j) {

cout << setw(10) << '|' << setw(8);

cout << subjects[j] << ": " << students[i].grades[j] << endl;

}

cout << "\n";

}

else {

cout << "The student stadies only on 'excellent':" << endl;

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

cout << "Sex: " << (students[i].sex = "M" ? "Male" : "Female") << endl;

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

cout << "Grades: " << endl;

for (int j = 0; j < 8; ++j) {

cout << setw(10) << '|' << setw(8);

cout << subjects[j] << ": " << students[i].grades[j] << endl;

}

cout << "\n";

}

}

cout << "\n";

}

void studentsByListNum(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 << "Full name: " << students[i].fullName << endl;

cout << "Sex: " << (students[i].sex = "M" ? "Male" : "Female") << endl;

cout << "Grades: " << endl;

for (int j = 0; j < 8; ++j) {

cout << setw(10) << '|' << setw(8);

cout << subjects[j] << ": " << students[i].grades[j] << endl;

}

cout << "\n";

}

}

}

int main() {

int num;

Student students[maxStud];

loadFromFile(students, num);

cout << "Enter 'c' to get list of commands\n";

while (true) {

char number;

cout << " Commands:\n ";

cin.sync();

cin >> number;

if (number == '0') break;

switch (number) {

case '1': {

addStudents(students, num);

break;

}

case '2': {

editStudents(students, num);

break;

}

case '3': {

listStudents(students, num);

break;

}

case '4': {

int N;

cout << "Enter group: ";

cin >> N;

groupStudents(students, num, N);

break;

}

case '5': {

int size;

cout << "Enter number of top: ";

cin >> size;

topStudents(students, num, size);

break;

}

case '6': {

NumberByGender(students, num); // 6

break;

}

case '7': {

scholarshipStudents(students, num); // 7

break;

}

case '8': {

int input;

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

cin >> input;

studentsByListNum(students, num, input); // 8

break;

}

case 'c': {

cout << "Choose an action:\n";

cout << "1 - Add new student.\n";

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

cout << "3 - Print all students.\n";

cout << "4 - Filter students by group number.\n";

cout << "5 - Sort students by grade point average.\n";

cout << "6 - Count amount of men/women.\n";

cout << "7 - Group students by scholarship.\n";

cout << "8 - Filter students by index.\n";

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

break;

}

default:

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

}

saveToFile(students, num);

}

return 0;

}