**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 и их внутреннее представление в памяти.

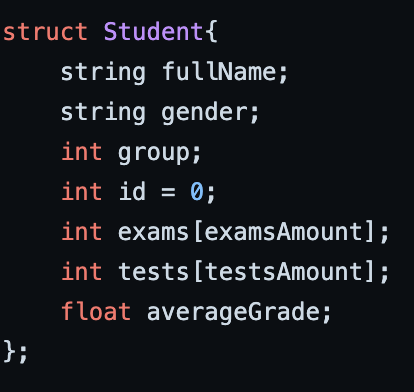
|  |  |  |
| --- | --- | --- |
| Студент гр. 3372 |  | Беляев К. В. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2023

**Цель работы.**

Работа со структурами

****

**Рабочий код:**

#include "iostream"

#include "string"

#include <fstream>

using namespace std;

const unsigned char examsAmount = 3;

const unsigned char testsAmount = 5;

int studentsAmount = 0;

const unsigned char studentsArrayLen = 60;

const unsigned char booksArrayLen = 30;

const unsigned char groupArrayLen = 30;

struct Student{

string fullName;

string gender;

int group;

int id = 0;

int exams[examsAmount];

int tests[testsAmount];

float averageGrade;

};

//main funcs;

void clearStream() {

cin.clear();

cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

}

void nextLine(int amount = 1) {

for (int repeat = 0; repeat < amount; repeat++) {

cout << "\n";

}

}

void showMenu(){

cout << "1. add new student"; nextLine();

cout << "2. edit student info"; nextLine();

cout << "3. show students"; nextLine();

cout << "4. show students in group"; nextLine();

cout << "5. show top 3 student"; nextLine();

cout << "6. show students amount by gender"; nextLine();

cout << "7. show students's grades info"; nextLine();

cout << "8. show Ks' students"; nextLine();

cout << "9. add students to file"; nextLine(2);

cout << "IDZ"; nextLine();

cout << "10. Взять книгу из библиотеки"; nextLine();

cout << "11. вернуть книгу в библиотеку"; nextLine();

cout << "12. купить книгу"; nextLine();

cout << "13. Сохранить изменения библиотеки в файл"; nextLine();

}

// student funcs;

// declarations

void addStudentToArray(Student student, Student \*students);

int countStudents(Student \*students);

int showStudentsByGroup(int groupNum, Student \*students, Student \*group);

bool isNormalGender(string gender);

int countStudents(Student \*students);

float avg( int \*exams, int \*tests);

Student findStudent(int group, int id, Student \*students);

//funcs

bool isNormalGender(string gender){

if ( gender == "M" || gender == "F"){

return true;

}

return false;

}

int countStudents(Student \*students){

studentsAmount = 0;

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

if (!(students + i)->fullName.empty()){

studentsAmount ++;

}

}

return studentsAmount;

}

float avg( int \*exams, int \*tests){

float sum = 0;

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

sum += float(\*(exams + i));

}

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

sum += float(\*(tests + i));

}

return (sum / 8.0);

}

Student findStudent(int group, int id, Student \*students){

studentsAmount = countStudents(students);

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

if ( (( students + i )->group == group) && (( students + i )->id == id) ) {

return \*( students + i );

}

}

return Student();

}

void showStudentInfo(Student student){

cout << "fullname: " << student.fullName; nextLine();

cout << "group: " << student.group; nextLine();

cout << "id: " << student.id; nextLine();

cout << "gender: " << student.gender; nextLine();

cout << "Exams: ";

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

cout << student.exams[i] << " ";

}

nextLine();

cout << "tests: ";

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

cout << student.tests[i] << " ";

}

nextLine();

cout << "avg Grade: " << student.averageGrade;

nextLine();

}

Student\* getAddress(int group, int id, Student \*students){

for (int i = 0; i < countStudents(students); i++){

if ((students + i)->id == id && (students + i)->group == group){

return (students + i);

}

}

return nullptr;

}

void editStudent(Student \*student, string parametr){ //#2

string genderParam;

if ( parametr == "fullname" ){

cout << "enter new data: "; getline(cin, student->fullName);

}else if( parametr == "gender" ){

cout << "enter new data: "; cin >> genderParam; clearStream();

if ( isNormalGender(genderParam) ){

student->gender = genderParam;

}else{

cout << "Че за гендер, введи нормальный";

}

}else if( parametr == "group" ){

cout << "enter new data: "; cin >> student->group; clearStream();

}else if( parametr == "id" ){

cout << "enter new data: "; cin >> student->id; clearStream();

}else if( parametr == "exams marks" ){

cout << "enter new data: ";

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

cin >> student->exams[i];

}

clearStream();

student->averageGrade = avg(student->exams, student->tests);

}else if( parametr == "tests marks" ){

cout << "enter new data: ";

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

cin >> student->tests[i];

}

clearStream();

student->averageGrade = avg(student->exams, student->tests);

}else{

cout << "Изменяемый параметр не существует";

}

}

void addStudents2File(Student \*students){

ofstream out;

out.open("/Users/kirillbelaev/CLionProjects/laba1Struct/db.txt");

if (out.is\_open())

{

for ( int i = 0; i < countStudents(students); i++ ){

out << (students + i)->fullName << endl;

out << (students + i)->gender << endl;

out << (students + i)->group << endl;

out << (students + i)->id << endl;

out << (students + i)->exams[0];

for ( int j = 1; j < examsAmount; j++ ){

out << " " << (students + i)->exams[j];

}

out << endl;

out << (students + i)->tests[0];

for ( int j = 1; j < testsAmount; j++ ){

out << " " << (students + i)->tests[j];

}

out << endl;

out << (students + i)->averageGrade << endl;

}

}

else{

cout << "Ошибка записи в файл";

}

out.close();

cout << "File has been written" << std::endl;

}

bool is\_Excellent(Student student){

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

if ( student.exams[i] != 5 ){

return false;

}

}

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

if (student.tests[i] != 5) {

return false;

}

}

return true;

}

bool is\_Good(Student student){

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

if ( student.exams[i] == 3 || student.exams[i] == 2){

return false;

}

}

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

if ( student.tests[i] == 3 || student.tests[i] == 2 ) {

return false;

}

}

return true;

}

bool is\_C\_Student(Student student){

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

if ( student.exams[i] < 4 ){

return true;

}

}

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

if ( student.tests[i] < 4 ) {

return true;

}

}

return false;

}

void showStudentsByGrades(Student \*students, int grade){

if ( grade == 5 ){

for ( int i = 0; i < countStudents(students); i++ ){

if ( is\_Excellent(\*(students + i)) ){

showStudentInfo(\*(students + i));

}

}

}else if( grade == 4 ){

for ( int i = 0; i < countStudents(students); i++ ){

if ( is\_Good( \*(students + i)) ){

showStudentInfo(\*(students + i));

}

}

}else{

for ( int i = 0; i < countStudents(students); i++ ){

if ( is\_C\_Student(\*(students + i)) ){

showStudentInfo(\*(students + i));

}

}

}

}

void readFile(Student students[]){

Student student;

ifstream fin("/Users/kirillbelaev/CLionProjects/laba1Struct/db.txt");

if ( !fin.is\_open() ){

cout << "file's not open";

}

else{

while (fin.peek() != EOF){

getline(fin, student.fullName);

getline(fin, student.gender);

fin >> student.group;

fin >> student.id;

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

fin >> student.exams[i];

}

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

fin >> student.tests[i];

}

fin >> student.averageGrade;

fin.get();

addStudentToArray(student, students);

}

}

fin.close();

}

int showStudentsByGroup(int groupNum, Student \*students, Student \*group){

studentsAmount = countStudents(students);

int index = 0;

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

if( ((students + i)->group) == groupNum ){

\*(group + index) = \*(students + i);

index++;

}

}

return index;

}

void showStudentsByID(int id, Student \*students){

for ( int i = 0 ; i < countStudents(students); i++ ) {

if ((students + i)->id == id) {

showStudentInfo(\*(students + i));

}

}

}

void countStudentsByGender(Student \*students){

int MaleAmount = 0;

int FemaleAmount = 0;

for ( int i = 0; i < countStudents(students); i++ ){

if ( (students + i)->gender == "M") {

MaleAmount++;

}else {

FemaleAmount++;

}

}

cout << "Male amount: " << MaleAmount ;

nextLine();

cout << "Female amount: " << FemaleAmount ;

}

void sortStudentsByGrades(Student \*students){

studentsAmount = countStudents(students);

for (int i = 0; i < studentsAmount / 2; i++){

for ( int j = i; j < studentsAmount - i - 1; j++ ){

if ( (students + j)->averageGrade < (students + j + 1)->averageGrade){

swap(\*(students + j), \*(students + j + 1) );

}

}

for ( int j = studentsAmount - 2 - i; j > i + 1; j-- ){

if ( (students + j)->averageGrade > (students + j + 1)->averageGrade){

swap(\*(students + j), \*(students + j - 1) );

}

}

}

}

void addStudentToArray(Student student, Student students[]){

students[countStudents(students)] = student;

}

// IDZ;

struct Book{

string title;

string author;

int publicationYear;

int pages;

int studentCard = 0;

};

//funcs;

int countBooks(Book \*books){

int booksAmount = 0;

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

if (!(books + i)->title.empty()){

booksAmount ++;

}else{

break;

}

}

return booksAmount;

}

void addBookToArray(Book book, Book books[]){

books[countBooks(books)] = book;

}

Book \*is\_bookExists(Book \*books, string title, string author){

for ( int i = 0; i < countBooks(books); i++ ){

if ( (books + i)->title == title && (books + i)->author == author ){

return (books + i);

}

}

return nullptr;

}

void readFile(Book books[]){

Book book;

ifstream fin("/Users/kirillbelaev/CLionProjects/laba1Struct/libdb.txt");

if ( !fin.is\_open() ){

cout << "file's not open";

}

else{

while (fin.peek() != EOF){

getline(fin, book.title);

getline(fin, book.author);

fin >> book.publicationYear;

fin >> book.pages;

fin >> book.studentCard;

fin.get();

addBookToArray(book, books);

}

}

fin.close();

}

void addBooks2File(Book \*books){

ofstream out;

out.open("/Users/kirillbelaev/CLionProjects/laba1Struct/libdb.txt");

if (out.is\_open())

{

for ( int i = 0; i < countBooks(books); i++ ){

out << (books + i)->title << endl;

out << (books + i)->author << endl;

out << (books + i)->publicationYear << endl;

out << (books + i)->pages << endl;

out << (books + i)->studentCard << endl;

}

}

else{

cout << "Ошибка записи в файл";

}

out.close();

cout << "File has been written" << std::endl;

}

void showBookInfo(Book book) {

cout << "title: " << book.title; nextLine();

cout << "author: " << book.author; nextLine();

cout << "year of publication: " << book.publicationYear; nextLine();

cout << "pages amount: " << book.pages; nextLine();

cout << "student card: " << book.studentCard; nextLine();

}

//----------------------------------------------------------------------------------------------------------------------

int main(){

int index;

int problem;

int groupNum, id;

string editingParam;

Student students[studentsArrayLen];

Student group[groupArrayLen];

Book books[booksArrayLen];

Book \*bookPTR;

string title, author;

Student student;

Student \*studentPTR;

readFile(students);

readFile(books);

for ( int repeats = 0; repeats < 100; repeats++ ){

nextLine();

showMenu();

cout << "choose problem: "; cin >> problem; clearStream(); nextLine();

switch (problem) {

case 1:

cout << "fullName: "; getline(cin, student.fullName);

cout << "gender: "; cin >> student.gender; clearStream();

cout << "group: "; cin >> student.group; clearStream();

cout << "id: "; cin >> student.id; clearStream();

cout << "enter exam's grades by Space: ";

for (int i = 0; i < examsAmount; i++) cin >> student.exams[i];

clearStream();

cout << "enter test's grades by Space: ";

for (int i = 0; i < testsAmount; i++) cin >> student.tests[i];

clearStream();

student.averageGrade = avg(student.exams, student.tests);

addStudentToArray(student, students);

break;

case 2:

cout << "Edit student info"; nextLine();

cout << "group: "; cin >> groupNum; clearStream();

cout << "id: "; cin >> id; clearStream();

studentPTR = getAddress(groupNum, id, students);

if (studentPTR) {

if (!studentPTR->fullName.empty()) {

cout << "Введите параметр, который хотите изменить( fullname, gender, group, id, exams marks, tests marks ): ";

getline(cin, editingParam);

editStudent(studentPTR, editingParam);

}

}else{

cout << "такого студента нет"; nextLine();

}

break;

case 3:

cout << "find student:"; nextLine();

cout << "group: "; cin >> groupNum; clearStream();

cout << "id: "; cin >> id; clearStream();

student = findStudent(groupNum, id, students);

if ( !student.fullName.empty() ){

nextLine();

showStudentInfo(student);

}else {

cout << "Студент не найден";

}

break;

case 4:

cout << "find students in group: "; cin >> groupNum; clearStream(); nextLine();

index = showStudentsByGroup(groupNum, students, group);

if ( !index ){

cout << "no such group";

}else{

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

showStudentInfo(group[i]);

nextLine();

}

}

break;

case 5:

sortStudentsByGrades(students);

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

showStudentInfo(students[i]);

nextLine();

}

break;

case 6:

countStudentsByGender(students);

nextLine();

break;

case 7:

cout << "На повышке: ";

nextLine();

showStudentsByGrades(students, 5);

nextLine();

cout << "На стипе: ";

nextLine();

showStudentsByGrades(students, 4);

nextLine();

cout << "Без стипы: ";

nextLine();

showStudentsByGrades(students, 3);

break;

case 8:

cout << "Введите номер в группе: "; cin >> id; clearStream();

showStudentsByID(id, students);

break;

case 9:

cout << "Сохранение изменений в файл"; nextLine();

addStudents2File(students);

break;

case 10:

cout << "book title: ";

getline(cin, title);

cout << "book author: ";

getline(cin, author);

bookPTR = is\_bookExists(books, title, author);

if ( bookPTR ){

cout << "student group: "; cin >> groupNum; clearStream();

cout << "student id: "; cin >> id; clearStream();

student = findStudent(groupNum, id, students);

if ( !student.fullName.empty() ){

bookPTR->studentCard = student.group \* 100 + student.id;

}else{

cout << "No such student";

}

}else{

cout << "No such book";

}

break;

case 11:

cout << "book title: ";

getline(cin, title);

cout << "book author: ";

getline(cin, author);

bookPTR = is\_bookExists(books, title, author);

if ( bookPTR ){

cout << "student group: "; cin >> groupNum; clearStream();

cout << "student id: "; cin >> id; clearStream();

student = findStudent(groupNum, id, students);

if ( !student.fullName.empty() ){

bookPTR->studentCard = 0;

}else{

cout << "No such student";

}

}else{

cout << "No such book";

}

break;

case 12:

cout << "book title: ";

getline(cin, title);

cout << "book author: ";

getline(cin, author);

bookPTR = is\_bookExists(books, title, author);

if ( bookPTR ){

cout << "student group: "; cin >> groupNum; clearStream();

cout << "student id: "; cin >> id; clearStream();

student = findStudent(groupNum, id, students);

if ( !student.fullName.empty() ){

int len = countBooks(books);

\*bookPTR = Book();

swap(\*bookPTR, books[len - 1]);

}else{

cout << "No such student";

}

}else{

cout << "No such book";

}

break;

case 13:

addBooks2File(books);

break;

default:

for ( int i = 0; i < countStudents(students); i++ ){

showStudentInfo(\*(students + i));

nextLine();

}

for ( int i = 0; i < countBooks(books); i++ ){

showBookInfo(\*(books + i));

nextLine();

}

}

}

}