**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра «Информационные системы»**

отчет

**по практической работе №1**

**по дисциплине «Программирование»**

Тема: Типы данных, определяемые пользователем. Структуры.

|  |  |  |
| --- | --- | --- |
| Студентка гр. 0323 |  | Сомова П.Д. |
| Преподаватель |  | Глущенко А.Г. |

Санкт-Петербург

2021

**Цель работы.**

Изучение и организация структур; получение практических навыков работы со структурами; определение преимуществ и недостатков использования структур.

**Основные теоретические положения.**

В языке С++ структуры представляют собой частный случай класса, сохраняя при этом синтаксис, унаследованный от языка си. Определение структуры осуществляется с использованием ключевого слова struct и выглядит следующим образом:

**struct** ИМЯ\_СТРУКТУРЫ

**{**

....//Поля структуры

**}** СПИСОК\_ОПИСАТЕЛЕЙ **;**

Структурную переменные возможно объявлять как при определении структуры, так и в других участках кода. В последнем случае - с использованием следующего синтаксиса:

**ИМЯ\_СТРУКТУРЫ** ИМЯ\_ПЕРЕМЕННОЙ;

Возможно и объявление массивов структурных пременных. При этом используется синтаксис, аналогичный обычным переменных.

Доступ к полям структурам осуществляется с использованием следующего синтаксиса:

**ИМЯ\_ПЕРЕМЕННОЙ**.ИМЯ\_ПОЛЯ

Поля структуры представляют собой обычные переменные, располагаемые в памяти ЭВМ последовательно (с учётом выравнивания по минимальный блокам памяти, зависящим от платформы и реализации).

Последовательное расположение полей структуры в памяти позволяет обращаться к полям структуры с испорльзованием указателей, при этом используется следующий синтаксис:

**УКАЗАТЕЛЬ\_НА\_ПЕРЕМЕННУЮ->**ИМЯ\_ПОЛЯ

В языке С++ сохраняется унаследованный от языка Си синтаксис определения структур с использованием ключевого слова typedef, однако никаких преимуществ перед определением структур без этого слова данный вариант не имеет.

В качестве поля структуры возможно и нерекурсивное использование других структур.

Структуры могут быть переданы в функцию как по значению, так и по ссылке (указателю), а также и возвращены ею. Это позволяет, в частности передавать по значению в функции массивы, а также возвращать их из функции, «упаковав» их в структуру.

**Постановка задачи.**

Создать массив структур, содержащий информацию о студентах: ФИО, пол, номер группы, номер в списке группы, оценки за прошедшую сессию (всего 3 экзамена и 5 дифференцированных зачетов), форма обучения, отметка времени о внесении или изменении данных. Ввод и изменение данных обо всех студентах должен осуществляться в файл students.

Написать функции, реализующие операции со структурами (ввод данных с клавиатуры):

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. N – инициализируется пользователем.
5. Вывод топа самых успешных студентов с наивысшим по рейтингу средним баллом за прошедшую сессию.
6. Вывод количества студентов мужского и женского пола.
7. Определение количества студентов, которые будут получать стипендию (стипендия начисляется, если у студента нет троек и очная форма обучения).
8. Вывод данных о студентах, которые не получают стипендию; учатся только на «хорошо» и «отлично»; учатся только на «отлично»;
9. Вывод данных о студентах, имеющих номер в списке – k.
10. Вывод всех записей, сделанных в день, который введет пользователь. Вывод всех записей, сделанных после полудня. Вывод всех записей, сделанных до полудня.

Сделать вывод по проделанной работе.

**Выполнение работы.**

Для решения поставленной задачи была написана программа на языке C++. Итоговый код программы представлен в приложении А.

Для хранения данных во внешнем файле был использован следующий формат:

ИМЯ\_СТУДЕНТА

ПОЛ\_СТУДЕНТА (1-мужской, 2-женский)

ГРУППА\_СТУДЕНТА

НОМЕР\_СТУДЕНТА\_В\_СПИСКЕ\_ГРУППЫ

ОЦЕНКИ\_СТУДЕНТА (каждая оценка представлена цифрой на новой строке)

ФОРМА\_ОБУЧЕНИЯ\_СТУДЕНТОВ (1-очная, 2-очно-заочная, 3- заочная)

ВРЕМЯ\_СОЗДАНИЯ/ИЗМЕНЕНИЯ\_ЗАПИСИ (В формате Unix-time)

Работа с файлом не осуществлена. Результаты тестирования представлены в приложении Б. При этом были получены результаты, соответствующие расчётным.

**Выводы.**

Структуры являются довольно удобным для использования элементом языка С++, хоть и уступают пости во всём классическим классам и служат, больше, для переносимости программ, написанных на языке Си.

Приложение А

Полный код программы

#include <iostream>  
#include <ctime>  
#include "istream"  
#include <limits>  
#include <fstream>  
#include <cstring>  
#define CLEARSCREEN "cls"  
using namespace std**;**struct student  
{  
 char studentName[**1001**]**;** short studentGender**;** char studentGroup[**21**]**;** short studentNoGroup**;** short exam1**;** short exam2**;** short exam3**;** short task1**;** short task2**;** short task3**;** short task4**;** short task5**;** short formEDU**;** time\_t modificationTime**;**}**;**unsigned int listSize**;**void getStudentInfo (student \*List**,** unsigned int studentNumber)  
{  
 student \*Student=List+studentNumber-**1;** cout << "Студент " << studentNumber << "\n"**;** cout << "ФИО студента: " << Student->studentName << "\n"**;** cout << "Пол студента:"**;** switch (Student->studentGender)  
 {  
 case **1**:  
 cout << "мужской\n"**;** break**;** case **2**:  
 cout << "женский\n"**;** break**;** }  
 cout << "Номер группы студента: " << Student-> studentGroup << "\n"**;** cout << "Номер студента в списке группы: " << Student-> studentNoGroup << "\n"**;** cout << "Оценка студента за первый экзамен: " << Student-> exam1 << "\n"**;** cout << "Оценка студента за второй экзамен: " << Student-> exam2 << "\n"**;** cout << "Оценка студента за третий экзамен: " << Student-> exam3 << "\n"**;** cout << "Оценка студента за первый зачет: " << Student-> task1 << "\n"**;** cout << "Оценка студента за второй зачет: " << Student-> task2 << "\n"**;** cout << "Оценка студента за третий зачет: " << Student-> task3 << "\n"**;** cout << "Оценка студента за четвертый зачет: " << Student->task4 << "\n"**;** cout << "Оценка студента за пятый зачет: " << Student-> task5 << "\n"**;** cout << "Форма обучения студента: "**;** switch (Student->formEDU)  
 {  
 case **1**:  
 cout << "очная\n"**;** break**;** case **2**:  
 cout << "очно-заочная\n"**;** break**;** case **3**:  
 cout << "заочная\n"**;** break**;** }  
 cout << "Дата внесения изменений в запись: " << asctime(localtime(&Student->modificationTime))**;**}  
  
int getNumAns (int minRange**,** int maxRange**,** const char \*Question)  
{  
 int answer**;** cout << Question**;** cin >> answer**;** if ((cin.fail())||(answer<minRange)||(answer>maxRange))  
 {  
 cin.clear()**;** cin.ignore(numeric\_limits<streamsize>::max()**,** '\n')**;** cout << "Некорректный ввод!\n"**;** answer= getNumAns(minRange**,** maxRange**,** Question)**;** return answer**;** }  
 else  
 {  
 cin.ignore(numeric\_limits<streamsize>::max()**,** '\n')**;** return answer**;** }  
}  
  
void changeStudentName (student \*Student**,** bool isNew)  
{  
 if (!isNew)  
 {  
 cout << "Текущие сведения о ФИО студента: " << Student-> studentName << "\n"**;** }  
 cout << "Введите ФИО студента: "**;** cin.getline (Student->studentName**, 1001**)**;**}  
  
void changeStudentGender (student \*Student**,** bool isNew)  
{  
 if (!isNew)  
 {  
 cout << "Текущие сведения о поле студента: "**;** switch (Student->studentGender)  
 {  
 case **1**:  
 cout << "?г?бЄ??\n"**;** break**;** case **2**:  
 cout << "???бЄЁ?\n"**;** break**;** }  
 }  
 cout << "Выберите пол студента: 1 - мужской 2 - женский \n" **;** Student->studentGender= getNumAns(**1, 2,** "Введите номер желаемого варианта: ")**;**}  
  
void changeStudentGroup (student \*Student**,** bool isNew)  
{  
 if (!isNew)  
 {  
 cout << "\nТекущие сведения группе студента: " << Student->studentGroup << "\n"**;** }  
 cout << "?ўВведите ноер группы студента: "**;** cin.getline (Student->studentGroup**, 21**)**;**}  
  
void changeStudentNumberGroup (student \*Student**,** bool isNew)  
{  
 if (!isNew)  
 {  
 cout << "\nТекущие сведения о номере студента в списке группы: " << Student->studentNoGroup << "\n"**;** }  
 Student->studentNoGroup = getNumAns(**0, 101,** "Введите номер студента в группе: ")**;**}  
  
void changeStudentGrades (student \*Student**,** short request)  
{  
 switch (request)  
 {  
 case **1**:  
 Student->exam1 = getNumAns(**2, 5,** "Пожалуйста, введите оценку студента за первый экзамен: : ")**;** break**;** case **2**:  
 Student->exam2 = getNumAns(**2, 5,** "Пожалуйста, введите оценку студента за второй экзамен: ")**;** break**;** case **3**:  
 Student->exam3 = getNumAns(**2, 5,** "Пожалуйста, введите оценку студента за третий экзамен: ")**;** break**;** case **4**:  
 Student->task1 = getNumAns(**2, 5,** "Пожалуйста, введите оценку студента за первый зачет: ")**;** break**;** case **5**:  
 Student->task2 = getNumAns(**2, 5,** "Пожалуйста, введите оценку студента за второй зачет: ")**;** break**;** case **6**:  
 Student->task3 = getNumAns(**2, 5,** "Пожалуйста, введите оценку студента за третий зачет: ")**;** break**;** case **7**:  
 Student->task4 = getNumAns(**2, 5,** "Пожалуйста, введите оценку студента за четвертый зачет: ")**;** break**;** case **8**:  
 Student->task5 = getNumAns(**2, 5,** "Пожалуйста, введите оценку студента за пятый зачет: ")**;** break**;** }  
}  
void studentGradesMenu (student \*Student)  
{  
 cout << "Оценки студента: \n"**;** cout << "Оценка студента за первый экзамен: " << Student-> exam1 << "\n"**;** cout << "Оценка студента за второй экзамен: " << Student-> exam2 << "\n"**;** cout << "Оценка студента за третий экзамен: " << Student-> exam3 << "\n"**;** cout << "Оценка студента за первый зачет: " << Student-> task1 << "\n"**;** cout << "Оценка студента за второй зачет: " << Student-> task2 << "\n"**;** cout << "Оценка студента за третий зачет: " << Student-> task3 << "\n"**;** cout << "Оценка студента за четвертый зачет: " << Student-> task4 << "\n"**;** cout << "Оценка студента за пятый зачет: " << Student-> task5 << "\n"**;** cout << "\\n\"Вы хотите изменить сведения о зачете или экзамене?\n"**;** cout << "1 - Изменить оценку за экзамен 2 - Изменить оценку за зачет 0 - Вернуться в предыдущее меню \n"**;** switch (getNumAns(**0, 2,** "Введите номер желаемого варианта: "))  
 {  
 case **1**:  
 switch (getNumAns(**1, 3,** "Введите номер экзамена, оценку за который желаете изменить: "))  
 {  
 case **1**:  
 changeStudentGrades(Student**, 1**)**;** break**;** case **2**:  
 changeStudentGrades(Student**, 2**)**;** break**;** case **3**:  
 changeStudentGrades(Student**, 3**)**;** break**;** }  
 Student->modificationTime=time (NULL)**;** studentGradesMenu (Student)**;** break**;** case **2**:  
 switch (getNumAns(**1, 5,** "Введите номер зачета, оценку за который желаете изменить: "))  
 {  
 case **1**:  
 changeStudentGrades(Student**, 4**)**;** break**;** case **2**:  
 changeStudentGrades(Student**, 5**)**;** break**;** case **3**:  
 changeStudentGrades(Student**, 6**)**;** break**;** case **4**:  
 changeStudentGrades(Student**, 7**)**;** break**;** case **5**:  
 changeStudentGrades(Student**, 8**)**;** break**;** }  
 Student->modificationTime=time (NULL)**;** studentGradesMenu (Student)**;** break**;** default:  
 break**;** }  
}  
  
  
void changeFormEDU (student \*Student**,** bool isNew)  
{  
 if (!isNew)  
 {  
 cout << "\nТекущие сведения о форме обучения студента: "**;** switch (Student->formEDU)  
 {  
 case **1**:  
 cout << "очная\n"**;** break**;** case **2**:  
 cout << "очно-заочная\n"**;** break**;** case **3**:  
 cout << "заочная\n"**;** break**;** }  
 }  
 cout << "Выберите форму обучения студента: 1 -Очная 2 - Очно-заочная 3-Заочная\n"**;** Student->formEDU = getNumAns(**1, 3,** "Введите номер желаемого варианта: ")**;**}  
  
  
void studentMenu (student \*List**,** unsigned int studentNumber)  
{  
 system (CLEARSCREEN)**;** getStudentInfo(List**,** studentNumber)**;** cout << "\n\nМеню работы с данными студента:\n"**;** cout << "1) Изменить сведения ФИО студента\n"**;** cout << "2) Изменить сведения о поле студента\n"**;** cout << "3) Изменить сведения о группе студента\n"**;** cout << "4) Изменить сведения о номере студента в списке группы\n"**;** cout << "5) Изменить сведения об оценках студента\n"**;** cout << "6) Изменить сведения о форме обучения студента\n"**;** cout << "0) Вернуться в главное меню\n\n"**;** student \*Student=List+studentNumber-**1;** switch (getNumAns (**0, 6,** "Введите номер желаемого варианта: "))  
 {  
 case **1**:  
 changeStudentName(Student**, 0**)**;** Student->modificationTime=time(NULL)**;** studentMenu (List**,** studentNumber)**;** break**;** case **2**:  
 changeStudentGender(Student**, 0**)**;** Student->modificationTime=time(NULL)**;** studentMenu (List**,** studentNumber)**;** break**;** case **3**:  
 changeStudentGroup (Student**, 0**)**;** Student->modificationTime=time(NULL)**;** studentMenu (List**,** studentNumber)**;** break**;** case **4**:  
 changeStudentNumberGroup(Student**, 0**)**;** Student->modificationTime=time(NULL)**;** studentMenu (List**,** studentNumber)**;** break**;** case **5**:  
 studentGradesMenu (Student)**;** studentMenu (List**,** studentNumber)**;** break**;** case **6**:  
 changeFormEDU(Student**, 0**)**;** Student->modificationTime=time(NULL)**;** studentMenu (List**,** studentNumber)**;** break**;** default:  
 break**;** }  
}  
  
void studentList (student \*List)  
{  
 system (CLEARSCREEN)**;** cout << "Список студентов в базе:\n\n"**;** for (unsigned int i=**0;** i<listSize**;** i++)  
 {  
 cout << i+**1** << ". " << (List+i)->studentName << "\n"**;** }  
 cout << "\n"**;** int answer=getNumAns(**0,** listSize**,** "Введите номер студента в списке, карточку которого вы хотите посмотреть (для возврата в главное меню введите 0): ")**;** if (answer)  
 {  
 studentMenu (List**,** answer)**;** studentList (List)**;** }  
}  
  
student\* addStudentsFromFile (student \*List)  
{  
 char fileName[**261**]**;** cout << "\nВведите имя (или путь) файла (по умолчанию - students): "**;** cin.getline (fileName**, 261**)**;** if (fileName[**0**]=='\0') strcpy(fileName**,** "students")**;** ifstream File**;** File.open (fileName)**;** if (File.is\_open())  
 {  
 while (!File.eof())  
 {  
 listSize++**;** List=(student\*)realloc(List**,** listSize\*sizeof(student))**;** student \*newStudent**;** newStudent=List+listSize-**1;** File.getline (newStudent->studentName**, 1001**)**;** File >> newStudent->studentGender**;** File.ignore(numeric\_limits<streamsize>::max()**,** '\n')**;** File.getline (newStudent->studentGroup**, 21**)**;** File >> newStudent->studentNoGroup**;** File.ignore(numeric\_limits<streamsize>::max()**,** '\n')**;** File >> newStudent->exam1**;** File.ignore(numeric\_limits<streamsize>::max()**,** '\n')**;** File >> newStudent->exam2**;** File.ignore(numeric\_limits<streamsize>::max()**,** '\n')**;** File >> newStudent->exam3**;** File.ignore(numeric\_limits<streamsize>::max()**,** '\n')**;** File >> newStudent->task1**;** File.ignore(numeric\_limits<streamsize>::max()**,** '\n')**;** File >> newStudent->task2**;** File.ignore(numeric\_limits<streamsize>::max()**,** '\n')**;** File >> newStudent->task3**;** File.ignore(numeric\_limits<streamsize>::max()**,** '\n')**;** File >> newStudent->task4**;** File.ignore(numeric\_limits<streamsize>::max()**,** '\n')**;** File >> newStudent->task5**;** File.ignore(numeric\_limits<streamsize>::max()**,** '\n')**;** File >> newStudent->formEDU**;** File.ignore(numeric\_limits<streamsize>::max()**,** '\n')**;** File >> newStudent->modificationTime**;** File.ignore(numeric\_limits<streamsize>::max()**,** '\n')**;** }  
 File.close()**;** }  
 else  
 {  
 cout << "Некорректный ввод!\n"**;** List=addStudentsFromFile(List)**;** }  
 return List**;**}  
  
void addListToFile (student \*List)  
{  
 char fileName[**261**]**;** cout << "\nВведите имя (или путь) файла (по умолчанию - students): "**;** cin.getline (fileName**, 261**)**;** if (fileName[**0**]=='\0') strcpy(fileName**,** "students")**;** ofstream File**;** File.open (fileName**,** std::ios\_base::trunc)**;** if (File.is\_open())  
 {  
 for (unsigned int i=**0;** i<listSize**;** i++)  
 {  
 student \*newStudent**;** newStudent=List+i**;** File << newStudent->studentName << "\n"**;** File << newStudent->studentGender << "\n"**;** File << newStudent->studentGroup << "\n"**;** File << newStudent->studentNoGroup << "\n"**;** File << newStudent->exam1 << "\n"**;** File << newStudent->exam2 << "\n"**;** File << newStudent->exam3 << "\n"**;** File << newStudent->task1 << "\n"**;** File << newStudent->task2 << "\n"**;** File << newStudent->task3 << "\n"**;** File << newStudent->task4 << "\n"**;** File << newStudent->task5 << "\n"**;** File << newStudent->formEDU << "\n"**;** if (i==listSize-**1**)  
 {  
 File << newStudent->modificationTime**;** }  
 else  
 {  
 File << newStudent->modificationTime << "\n"**;** }  
 }  
 File.close()**;** }  
 else  
 {  
 cout << "Некорректный ввод!\n"**;** List=addStudentsFromFile(List)**;** }  
}  
  
student\* addStudentFromTerminal (student \*List)  
{  
 system (CLEARSCREEN)**;** cout << "Добавление студента: \n\n"**;** listSize++**;** List=(student\*)realloc(List**,** listSize\*sizeof(student))**;** student \*Student=List+listSize-**1;** changeStudentName(Student**, 1**)**;** cout << '\n'**;** changeStudentGender(Student**, 1**)**;** cout << '\n'**;** changeStudentGroup(Student**, 1**)**;** cout << '\n'**;** changeStudentNumberGroup(Student**, 1**)**;** cout << '\n'**;** changeStudentGrades(Student**, 1**)**;** cout << '\n'**;** changeStudentGrades(Student**, 2**)**;** cout << '\n'**;** changeStudentGrades(Student**, 3**)**;** cout << '\n'**;** changeStudentGrades(Student**, 4**)**;** cout << '\n'**;** changeStudentGrades(Student**, 5**)**;** cout << '\n'**;** changeStudentGrades(Student**, 6**)**;** cout << '\n'**;** changeStudentGrades(Student**, 7**)**;** cout << '\n'**;** changeStudentGrades(Student**, 8**)**;** cout << '\n'**;** changeFormEDU(Student**, 1**)**;** Student->modificationTime=time (NULL)**;** return List**;**}  
  
void listGroup (student \*List**,** bool length**,** const char \*Group)  
{  
 system (CLEARSCREEN)**;** cout << "Список студентов в группе " << Group << ":\n\n" **;** int studentNumber=**0;** for (unsigned int i=**0;** i<listSize**;** i++)  
 {  
 if (!strcmp((List+i)->studentGroup**,** Group))  
 {  
 if (length)  
 {  
 getStudentInfo(List**,** i + **1**)**;** cout << std::endl**;** }  
 else  
 {  
 cout << i+**1** << ". " << (List+i)->studentName << "\n"**;** }  
 studentNumber++**;** }  
 }  
 cout << "\nВсего студентов в группе " << Group << ": " << studentNumber << "\n"**;** int answer=getNumAns(**0,** listSize**,** "Введите номер студента в списке, карточку которого вы хотите посмотреть (для возврата в главное меню введите 0): ")**;** if (answer)  
 {  
 studentMenu (List**,** answer)**;** listGroup (List**,** length**,** Group)**;** }  
}  
  
void listTop (student \*List**,** bool length)  
{  
 system (CLEARSCREEN)**;** cout << "Топ студентов с наивысшим средним баллом за прошедшую сессию:\n\n"**;** struct sortList  
 {  
 unsigned int number**;** double count**;** }**;** sortList \*SortList=(sortList\*)malloc(sizeof(sortList)\*listSize)**;** for (unsigned int i=**0;** i<listSize**;** i++)  
 {  
 (SortList+i)->number=i**;** (SortList+i)->count=static\_cast<float>((((List+i)->task1)+((List+i)->task2)+((List+i)->task3)+((List+i)->task4)+((List+i)->task5)+((List+i)->exam1)+((List+i)->exam2)+((List+i)->exam3))/**8**)**;** }  
 int sorted=**0;** int currentMaxUnsorted=listSize-**1;** while (!sorted)  
 {  
 sorted=**1;** for (int i=**0;** i<currentMaxUnsorted**;** i++)  
 {  
 if ((SortList+i)->count-(SortList+i+**1**)->count>**0**)  
 {  
 sortList Buffer**;** Buffer.number=(SortList+i)->number**;** Buffer.count=(SortList+i)->count**;** (SortList+i)->number=(SortList+i+**1**)->number**;** (SortList+i)->count=(SortList+i+**1**)->count**;** (SortList+i+**1**)->number=Buffer.number**;** (SortList+i+**1**)->count=Buffer.count**;** sorted=**0;** }  
 }  
 currentMaxUnsorted--**;** }  
 for (unsigned int i=listSize-**1,** j=**0;** i>=**0**&&j<**10;** i--**,** j++)  
 {  
 if (length)  
 {  
 getStudentInfo(List**,** (SortList + i)->number + **1**)**;** cout << std::endl**;** }  
 else  
 {  
 cout << ((SortList+i)->number)+**1;** cout << (List+((SortList+i)->number))->studentName << "\n"**;** }  
 }  
 cout << "\n\n"**;** int answer=getNumAns(**0,** listSize**,** "Введите номер студента в списке, карточку которого вы хотите посмотреть (для возврата в главное меню введите 0): ")**;** if (answer)  
 {  
 studentMenu (List**,** answer)**;** listTop (List**,** length)**;** }  
 free(SortList)**;**}  
  
void listGender (student \*List**,** bool length)  
{  
 system (CLEARSCREEN)**;** cout << "Список студентов мужского пола:\n"**;** int studentNumber=**0;** for (unsigned int i=**0;** i<listSize**;** i++)  
 {  
 if ((List+i)->studentGender == **1**)  
 {  
 if (length)  
 {  
 getStudentInfo(List**,** i + **1**)**;** cout << "\n\n"**;** }  
 else  
 {  
 cout << i+**1** << ". " << (List+i)->studentName << "\n"**;** }  
 studentNumber++**;** }  
 }  
 cout << "\nВсего студентов мужского пола: " << studentNumber << "\n"**;** cout << "\n\nСписок студентов женского пола: " << std::endl**;** studentNumber=**0;** for (unsigned int i=**0;** i<listSize**;** i++)  
 {  
 if ((List+i)->studentGender == **2**)  
 {  
 if (length)  
 {  
 getStudentInfo(List**,** i + **1**)**;** cout << "\n\n"**;** }  
 else  
 {  
 std::cout << i+**1** << ". " << (List+i)->studentName << std::endl**;** }  
 studentNumber++**;** }  
 }  
 std::cout << "\nВсего студентов женского пола: " << studentNumber << "\n"**;** int answer=getNumAns(**0,** listSize**,** "Введите номер студента в списке, карточку которого вы хотите посмотреть (для возврата в главное меню введите 0): ")**;** if (answer)  
 {  
 studentMenu (List**,** answer)**;** listGender(List**,** length)**;** }  
}  
  
void listStipend (student \*List**,** bool length)  
{  
 system (CLEARSCREEN)**;** cout << "Список студентов, получающих стипендию: \n\n" **;** int studentNumber=**0;** for (unsigned int i=**0;** i<listSize**;** i++)  
 {  
 if (((((List+i)->task1)-**3**)>**0**)&&((((List+i)->task2)-**3**)>**0**)&&((((List+i)->task3)-**3**)>**0**)&&((((List+i)->task4)-**3**)>**0**)&&((((List+i)->task5)-**3**)>**0**)&&((((List+i)->exam1)-**3**)>**0**)&&((((List+i)->exam2)-**3**)>**0**)&&((((List+i)->exam3)-**3**)>**0**)&&((List+i)->formEDU==**1**))  
 {  
 if (length)  
 {  
 getStudentInfo(List**,** i + **1**)**;** cout << "\n"**;** }  
 else  
 {  
 cout << i+**1** << ". " << (List+i)->studentName << "\n"**;** }  
 studentNumber++**;** }  
 }  
 cout << "\nВсего студентов, получающих стипендию: " << studentNumber **;** int answer=getNumAns(**0,** listSize**,** "Введите номер студента в списке, карточку которого вы хотите посмотреть (для возврата в главное меню введите 0): ")**;** if (answer)  
 {  
 studentMenu (List**,** answer)**;** listStipend (List**,** length)**;** }  
}  
  
void listStipendLess (student \*List**,** bool length)  
{  
 system (CLEARSCREEN)**;** cout << "Список студентов, не получающих стипендию:\n\n" **;** int studentNumber=**0;** for (unsigned int i=**0;** i<listSize**;** i++)  
 {  
 if (((((List+i)->task1)-**3**)<=**0**)||((((List+i)->task2)-**3**)<=**0**)||((((List+i)->task3)-**3**)<=**0**)||((((List+i)->task4)-**3**)<=**0**)||((((List+i)->task5)-**3**)<=**0**)||((((List+i)->exam1)-**3**)<=**0**)||((((List+i)->exam2)-**3**)<=**0**)||((((List+i)->exam3)-**3**)<=**0**)||(!((List+i)->formEDU==**1**)))  
 {  
 if (length)  
 {  
 getStudentInfo(List**,** i + **1**)**;** std::cout << std::endl**;** }  
 else  
 {  
 std::cout << i+**1** << ". " << (List+i)->studentName << std::endl**;** }  
 studentNumber++**;** }  
 }  
 cout << "\nВсего студентов, не получающих стипендию: " << studentNumber**;** int answer=getNumAns(**0,** listSize**,** "Введите номер студента в списке, карточку которого вы хотите посмотреть (для возврата в главное меню введите 0): ")**;** if (answer)  
 {  
 studentMenu (List**,** answer)**;** listStipendLess (List**,** length)**;** }  
}  
  
void listFourFive (student \*List**,** bool length)  
{  
 system (CLEARSCREEN)**;** cout << "Список студентов, которые учатся только на \"хорошо\" и \"отлично\": \n\n"**;** int studentNumber=**0;** for (unsigned int i=**0;** i<listSize**;** i++)  
 {  
 if (((((List+i)->task1)-**3**)>**0**)&&((((List+i)->task2)-**3**)>**0**)&&((((List+i)->task3)-**3**)>**0**)&&((((List+i)->task4)-**3**)>**0**)&&((((List+i)->task5)-**3**)>**0**)&&((((List+i)->exam1)-**3**)>**0**)&&((((List+i)->exam2)-**3**)>**0**)&&((((List+i)->exam3)-**3**)>**0**))  
 {  
 if (length)  
 {  
 getStudentInfo(List**,** i + **1**)**;** cout << "\n"**;** }  
 else  
 {  
 cout << i+**1** << ". " << (List+i)->studentName << "\n"**;** }  
 studentNumber++**;** }  
 }  
 cout << "\nВсего студентов, которые учатся только на \"хорошо\" и \"отлично\": " << studentNumber << "\n\n"**;** int answer=getNumAns(**0,** listSize**,** "Введите номер студента в списке, карточку которого вы хотите посмотреть (для возврата в главное меню введите 0): ")**;** if (answer)  
 {  
 studentMenu (List**,** answer)**;** listFourFive (List**,** length)**;** }  
}  
  
void listFive (student \*List**,** bool length)  
{  
 system (CLEARSCREEN)**;** cout << "Список студентов, которые учатся только на \"отлично\": \n\n" **;** int studentNumber=**0;** for (unsigned int i=**0;** i<listSize**;** i++)  
 {  
 if (((((List+i)->task1)-**4**)>**0**)&&((((List+i)->task2)-**4**)>**0**)&&((((List+i)->task3)-**4**)>**0**)&&((((List+i)->task4)-**4**)>**0**)&&((((List+i)->task5)-**4**)>**0**)&&((((List+i)->exam1)-**4**)>**0**)&&((((List+i)->exam2)-**4**)>**0**)&&((((List+i)->exam3)-**4**)>**0**))  
 {  
 if (length)  
 {  
 getStudentInfo(List**,** i + **1**)**;** cout << "\n"**;** }  
 else  
 {  
 cout << i+**1** << ". " << (List+i)->studentName << "\n"**;** }  
 studentNumber++**;** }  
 }  
 cout << "\nВсего студентов, которые учатся только на \"отлично\": " << studentNumber <<"\n"**;** int answer=getNumAns(**0,** listSize**,** "Введите номер студента в списке, карточку которого вы хотите посмотреть (для возврата в главное меню введите 0): ")**;** if (answer)  
 {  
 studentMenu (List**,** answer)**;** listFive (List**,** length)**;** }  
}  
  
void listNumber (student \*List**,** bool length**,** int Number)  
{  
 system (CLEARSCREEN)**;** cout << "Список студентов, имеющих номер " << Number << "в списке группы:\n\n"**;** int studentNumber=**0;** for (unsigned int i=**0;** i<listSize**;** i++)  
 {  
 if ((List+i)->studentNoGroup==Number)  
 {  
 if (length)  
 {  
 getStudentInfo(List**,** i + **1**)**;** cout << "\n"**;** }  
 else  
 {  
 cout << i+**1** << ". " << (List+i)->studentName << "\n"**;** }  
 studentNumber++**;** }  
 }  
 cout << "\nВсего студентов, имеющих номер " << Number << " в списке группы: " << studentNumber << "\n\n"**;** int answer=getNumAns(**0,** listSize**,** "Введите номер студента в списке, карточку которого вы хотите посмотреть (для возврата в главное меню введите 0): ")**;** if (answer)  
 {  
 studentMenu (List**,** answer)**;** listNumber (List**,** length**,** Number)**;** }  
}  
  
void listTime (student \*List**,** bool length**,** tm Begin**,** int Noon)  
{  
 system (CLEARSCREEN)**;** cout << "Список студентов, записи о которых сделаны в указанный промежуток времени:\n\n"**;** tm End**;** End.tm\_sec=Begin.tm\_sec**;** End.tm\_min=Begin.tm\_min**;** End.tm\_hour=Begin.tm\_hour**;** End.tm\_mday=Begin.tm\_mday**;** End.tm\_mon=Begin.tm\_mon**;** End.tm\_year=Begin.tm\_year**;** End.tm\_isdst=Begin.tm\_isdst**;** switch (Noon)  
 {  
 case **0**:  
 End.tm\_sec=**59;** End.tm\_min=**59;** End.tm\_hour=**23;** break**;** case **1**:  
 End.tm\_sec=**59;** End.tm\_min=**59;** End.tm\_hour=**11;** break**;** case **2**:  
 End.tm\_sec=**59;** End.tm\_min=**59;** End.tm\_hour=**23;** Begin.tm\_sec=**00;** Begin.tm\_min=**00;** Begin.tm\_hour=**12;** break**;** }  
 time\_t beginTime=mktime(&Begin)**;** time\_t endTime=mktime(&End)**;** int studentNumber=**0;** for (unsigned int i=**0;** i<listSize**;** i++)  
 {  
 if ((((List+i)->modificationTime)>=beginTime)&&(((List+i)->modificationTime)<=endTime))  
 {  
 if (length)  
 {  
 getStudentInfo(List**,** i + **1**)**;** cout << std::endl**;** }  
 else  
 {  
 cout << i+**1** << ". " << (List+i)->studentName << std::endl**;** }  
 studentNumber++**;** }  
 }  
 cout << "\nВсего студентов, записи о которых сделаный в указанный промежуток времени: " << studentNumber << "\n\n"**;** int answer=getNumAns(**0,** listSize**,** "Введите номер студента в списке, карточку которого вы хотите посмотреть (для возврата в главное меню введите 0): ")**;** if (answer)  
 {  
 studentMenu (List**,** answer)**;** listTime (List**,** length**,** Begin**,** Noon)**;** }  
}  
  
tm getTmDate ()  
{  
 tm Day**;** cout << "Введите дату: (формат: ДД.ММ.ГГГГ): "**;** char date[**11**]**;** std::cin.getline(date**, 11**)**;** Day.tm\_sec=**0;** Day.tm\_min=**0;** Day.tm\_hour=**0;** Day.tm\_mday=(date[**0**]-**48**)\***10**+(date[**1**]-**48**)**;** Day.tm\_mon=(date[**3**]-**48**)\***10**+(date[**4**]-**48**)-**1;** Day.tm\_year=(date[**6**]-**48**)\***1000**+(date[**7**]-**48**)\***100**+(date[**8**]-**48**)\***10**+(date[**9**]-**48**)-**1900;** Day.tm\_isdst=-**1;** if (mktime(&Day)<**0**)  
 {  
 cout << "Некорректный ввод!\n"**;** Day=getTmDate()**;** }  
 return Day**;**}  
  
void listFilter (student \*List**,** bool length)  
{  
 system (CLEARSCREEN)**;** cout << "Вывести список студентов,соотвествующих критериям:\n"**;** cout << "1) Вывести список студентов определенной группы\n"**;** cout << "2) Вывести топ студентов с наивысшим средним баллом за прошедшую сессию\n"**;** cout << "3) Вывод список студентов по гендеру\n"**;** cout << "4) Вывести список студентов, которые будут получать стипендию по итогам прошедшей сессии\n"**;** cout << "5) Вывести список студентов, которые не получают стипендию\n"**;** cout << "6) Вывести список студентов, которые учатся на \"хорошо\" и \"отлично\"\n"**;** cout << "7) Вывести список студентов, которые учатся на \"отлично\"\n"**;** cout << "8) Вывести список студентов, имеющий определенный номер в списке группы\n"**;** cout << "9) Вывод всех записей, сделанных/измененных в конкретный промежуток времени\n"**;** cout << "0) Возврат в главное меню\n\n"**;** switch (getNumAns(**0, 9,** "Введите номер желаемого варианта: "))  
 {  
 case **1**:  
 {  
 cout << "Введите номер группы: "**;** char Group[**21**]**;** cin.getline(Group**, 21**)**;** listGroup(List**,** length**,** Group)**;** listFilter(List**,** length)**;** break**;** }  
 case **2**:  
 listTop(List**,** length)**;** listFilter(List**,** length)**;** break**;** case **3**:  
 listGender(List**,** length)**;** listFilter(List**,** length)**;** break**;** case **4**:  
 listStipend(List**,** length)**;** listFilter(List**,** length)**;** break**;** case **5**:  
 listStipendLess(List**,** length)**;** listFilter(List**,** length)**;** break**;** case **6**:  
 listFourFive(List**,** length)**;** listFilter(List**,** length)**;** break**;** case **7**:  
 listFive(List**,** length)**;** listFilter(List**,** length)**;** break**;** case **8**:  
 {  
 int number=getNumAns(**0, 9999,** "Введите номер студента в списке группы: ")**;** listNumber(List**,** length**,** number)**;** listFilter(List**,** length)**;** break**;** }  
 case **9**:  
 {  
 tm Day=getTmDate()**;** int answer=getNumAns(**0, 2,** "Вывести список записей, сделанных:\n\n0) в течение дня\n1) до полудня\n2) после полудня\n\nВведите желаемый вариант: ")**;** listTime(List**,** length**,** Day**,** answer)**;** listFilter(List**,** length)**;** break**;** }  
 default:  
 break**;** }  
}  
  
void listFilterChoise (student \*List)  
{  
 bool answer=getNumAns(**1, 2,** "\nВыберите желаемый формат списков:\n1)Краткий\n2)Длинный\n\nВведите желаемый вариант: ")-**1;** listFilter(List**,** answer)**;**}  
  
student\* mainMenu (student \*List)  
{  
 system (CLEARSCREEN)**;** cout << "Главное меню:\n\n"**;** cout << "1) Вывести полный список студентов\n"**;** cout << "2) Добавить студента из терминала\n"**;** cout << "3) Добавить студентов из файла\n"**;** cout << "4) Вывести список студентов по фильтру\n"**;** cout << "5) Вывод всех данных в файл\n"**;** cout << "0) Выход из программы\n\n"**;** switch (getNumAns(**0, 5,** "Введите номер желаемого варианта: "))  
 {  
 case **1**:  
 studentList(List)**;** List=mainMenu(List)**;** break**;** case **2**:  
 List=addStudentFromTerminal(List)**;** List=mainMenu(List)**;** break**;** case **3**:  
 List=addStudentsFromFile(List)**;** List=mainMenu(List)**;** break**;** case **4**:  
 listFilterChoise(List)**;** List=mainMenu(List)**;** break**;** case **5**:  
 addListToFile(List)**;** List=mainMenu(List)**;** break**;** default:  
 break**;** }  
 return List**;**}  
  
  
int main()  
{  
 setlocale(**0,** "")**;** listSize=**0;** student \*List=NULL**;** List=mainMenu (List)**;** free(List)**;** return **0;**}

Приложение Б

результаты тестирования

C:\Users\admin\CLionProjects\LB1\cmake-build-debug\LB1.exe

Главное меню:

1) Вывести полный список студентов

2) Добавить студента из терминала

3) Добавить студентов из файла

4) Вывести список студентов по фильтру

5) Вывод всех данных в файл

0) Выход из программы

Введите номер желаемого варианта:2

2

Введите ФИО студента:Денисюк Татьяна Викторовна

Денисюк Татьяна Викторовна

Выберите пол студента: 1 - мужской 2 - женский

Введите номер желаемого варианта:2

2

?Введите ноер группы студента:123

123

Введите номер студента в группе:5

5

Пожалуйста, введите оценку студента за первый экзамен: :5

5

Пожалуйста, введите оценку студента за второй экзамен:4

4

Пожалуйста, введите оценку студента за третий экзамен:5

5

Пожалуйста, введите оценку студента за первый зачет:4

4

Пожалуйста, введите оценку студента за второй зачет:5

5

Пожалуйста, введите оценку студента за третий зачет:4

4

Пожалуйста, введите оценку студента за четвертый зачет:5

5

Пожалуйста, введите оценку студента за пятый зачет:4

4

Выберите форму обучения студента: 1 -Очная 2 - Очно-заочная 3-Заочная

Введите номер желаемого варианта:1

1

1) Вывести полный список студентов

2) Добавить студента из терминала

3) Добавить студентов из файла

4) Вывести список студентов по фильтру

5) Вывод всех данных в файл

0) Выход из программы

Введите номер желаемого варианта:1

1

Список студентов в базе:

1. "Денисюк Татьяна-

Введите номер студента в списке, карточку которого вы хотите посмотреть (для возврата в главное меню введите 0):1

1

Оценка студента за третий экзамен: 5

Оценка студента за первый зачет: 4

Оценка студента за второй зачет: 5

Оценка студента за третий зачет: 4

Оценка студента за четвертый зачет: 5

Оценка студента за пятый зачет: 4

Форма обучения студента: очная

Дата внесения изменений в запись: Fri Jun 04 13:29:00 2021

Меню работы с данными студента:

1) Изменить сведения ФИО студента

2) Изменить сведения о поле студента

3) Изменить сведения о группе студента

4) Изменить сведения о номере студента в списке группы

5) Изменить сведения об оценках студента

6) Изменить сведения о форме обучения студента

0) Вернуться в главное меню

Введите номер желаемого варианта: