**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. Структуры

|  |  |  |
| --- | --- | --- |
| Студент(ка) гр. |  | Смольников И. Р. |
| 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.**

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. Чаще всего ограничиваются тем, как структуры представлены в языке С:

struct [имя\_типа] {

тип\_1 элемент\_1;

тип \_2 элемент\_2;

…

тип\_k элемент\_k;

} [ список\_описателей ];

Описание структуры начинается ключевым словом struct. 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 ‘;’.

При описании структуры память для размещения данных не выделяется. Работать с описанной структурой можно только после того, как будет определена переменная (переменные) этого типа данных, только при этом компилятор выделит необходимую память.

Для инициализации структуры значения ее элементов перечисляют в фигурных скобках в порядке их описания:

struct complex{

float real, im;

} data [2][2] = {

{{1,1}, {2,2}},

{{3,3}, {4,4}}

};

Все поля структурных переменных располагаются в непрерывной области памяти одно за другим. Общий объем памяти, занимаемый структурой, равен сумме размеров всех полей структуры. Для определения размера структуры следует использовать инструкцию sizeof().

Для того чтобы записать данные в структурную переменную, необходимо каждому полю структуры присвоить определенное значение. Для этого необходимо использовать оператор ‘’ («точка»):

struct Stack { // Cтек

float arr[100];

short topIndex;

};

…

Stack stack; // Объявляем переменную типа Stack

Stack.arr[0] = 1;

…

При доступе к определенному полю его следует рассматривать как обычную переменную, тип данных которой соответствует типу этого поля. Поля структур могут участвовать в качестве операндов любых выражений, допускающих использование операндов соответствующего типа данных.

Копирование данных из одной структурной переменной в другую осуществляется простой операцией присваивания, независимо от количества полей и размера структуры (это можно делать только в том случае, когда обе переменные одного и того же типа).

В программировании очень часто используются такие конструкции, как массивы структур. Например, сведения о студентах некоторой учебной группы можно хранить в массиве студентов:

t\_Student Gruppa\_N [30];

Был определен 30-элементный массив, каждый элемент которого предназначен для хранения данных одного студента. Получение доступа к данным некоторого студента из группы *N* осуществляется обычной индексацией переменной массива. Поскольку поля структуры могут быть любого типа данных, то они в свою очередь могут быть другой структурой или массивом других структур:

struct Stud

{

char FN[100];

short listNumber;

};

struct Group

{

int groupNumber;

short students;

Stud stud[30];

};

Но в структуре поля нельзя использовать элемент, тип которого совпадает с типом самой структуры, так как рекурсивное использование структур запрещено.

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».

Структуры можно использовать в качестве параметров функций, как и обычные переменные. Для структур поддерживаются все три механизма передачи данных: по значению, через указатели и по ссылке.

Передачу структур в функции по значению необходимо использовать аккуратно:

void WriteStudent ( t\_Student S )

{

cout << "Фамилия: " << S.Fam << endl;

cout << "Имя: " << S.Name << endl;

cout << "Год рождения: " << S.Year << endl;

if ( S.Sex )

cout << "Пол: " << "М\n";

else

cout << "Пол: " << "Ж\n";

cout << "Средний балл: " << S.Grade << endl;

}

Вызов такой функции сопровождается дополнительным расходом памяти для создания локальной переменной *S*и дополнительными затратами времени на физическое копирование данных из аргумента в параметр *S*. Учитывая то, что объем структур может быть очень большим, эти дополнительные затраты вычислительных ресурсов могут быть чрезмерными.

Предпочтительно использование передачи структуры по указателю или ссылке:

void WriteStudent ( t\_Student \*S )

{

cout << "Фамилия: " << S -> Fam << endl;

cout << "Имя: " << S -> Name << endl;

cout << "Год рождения: " << S -> Year << endl;

if ( S -> Sex )

cout << "Пол: " << "М\n";

else

cout << "Пол: " << "Ж\n";

cout << "Средний балл: " << S -> Grade << endl;

}

Фактической передачи данных в функцию не осуществляется. Дополнительные затраты памяти для создания локальной переменной небольшие – это адрес памяти (4 байта, независимо от размера самой структуры). Вызов такой функции будет происходить быстрее, а расход памяти будет существенно меньше, чем при передаче данных по значению.

Передача по ссылке по эффективности эквивалентна передаче данных через указатель. Однако, поскольку при передаче данных по ссылке все адресные преобразования берет на себя компилятор, существенно упрощается программирование действий со структурами. При использовании ссылочных параметров структурных типов доступ к членам структуры осуществляется обычным способом – с помощью оператора «точка».

Недостатком этих способов является то, что случайные изменения значений полей структуры внутри функции отразятся на значении аргумента после окончания работы функции. Если необходимо предотвратить изменения переданных по адресу аргументов, можно при определении соответствующего параметра объявить его константой (использовать спецификатор const).

**Постановка задачи.**

Необходимо 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). Ввод и изменение данных обо всех студентах должен осуществляться в файл students Log.

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.   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.

Блок описания кода и использованных алгоритмов

1. Программа открывается с меню, в меню происходит выбор режима работы программы (Ввод и вывод студентов, редактирование и др.).
2. Ввод информации о студенте. Пользователь вводит ФИО, пол, номер группы, номер внутри группы и оценки студента. По завершении ввода программа предлагает ввести информацию об еще одном студенте. Если эта опция была выбрана, программа запускает ввод студента еще раз (второй раз вызывается функция ввода), в противном случае вызывается меню программы.
3. Вывод информации о студентах. Осуществляется вывод всех студентов, записанных в файле. Выводятся такие данные, как ФИО, пол, номер группы, номер внутри группы и оценки студента. По завершении вывода программа возвращается в главное меню.
4. Редактирование информации о студенте. Пользователь выбирает студента, данные которого требуется отредактировать (посредством ввода порядкого номера студента внутри файла). Затем он вводит новые данные об этом студенте. По завершении работы функции программа возвращается в главное меню.
5. Вывод информации обо всех студентах из одной группы. Пользователь задает номер группы, программа выводит данные всех студентов из этой группы. По завершении работы функции программа возвращается в главное меню.
6. Вывод рейтинга. Программа выводит рейтинг студентов, определяемый по средним оценкам студентов, студенты с наивысшей средней оценкой находятся выше в рейтинге. По завершении работы функции программа возвращается в главное меню.
7. Вывод количества студентов мужского и женского пола. Программа подсчитывает количество студентов мужского и женского пола и выводит их количество. По завершении работы функции программа возвращается в главное меню.
8. Вывод информации о стипендии студентов. Программа находит студентов с оценками «4» и выше и студентов со всеми «5», остальные студенты стипендию не получают. Далее программа выводит поочередно данные о студентах без стипендии, затем студентов с «4» и выше и студентов со всеми «5». По завершении работы функции программа возвращается в главное меню.
9. Вывод информации обо всех студентах, имеющий одинаковый номер внутри групп. Пользователь задает номер внутри группы, программа выводит данные всех студентов с этим номером. По завершении работы функции программа возвращается в главное меню.

Блок скриншотов работы программы

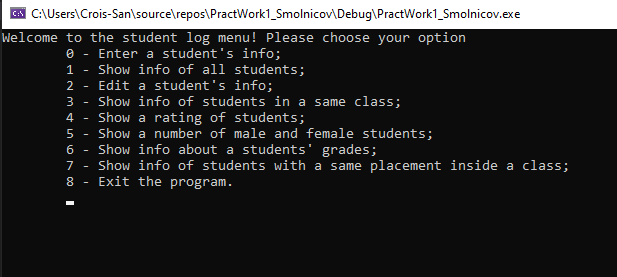


Рисунок Меню программы

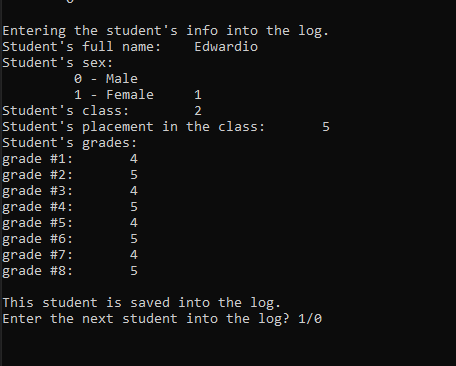


Рисунок Ввод данных о студенте

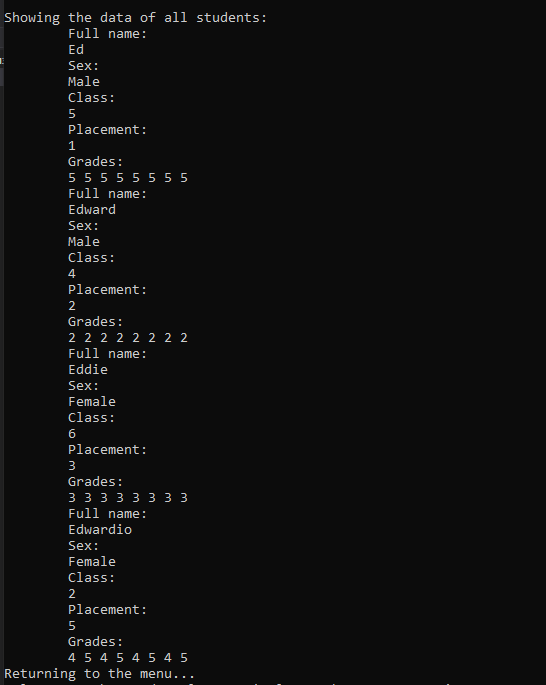


Рисунок Вывод данных о студентах

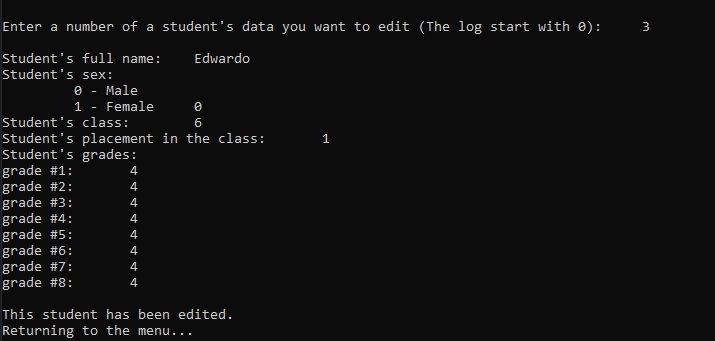


Рисунок Редактирование данных о студенте

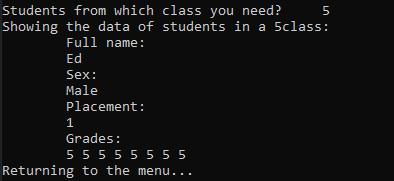


Рисунок Вывод данных обо всех студентах с одной группы

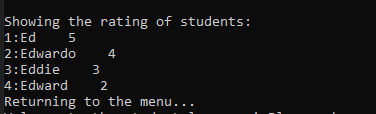


Рисунок Рейтинг студентов

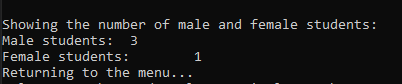


Рисунок Количество студентов мужского и женского пола

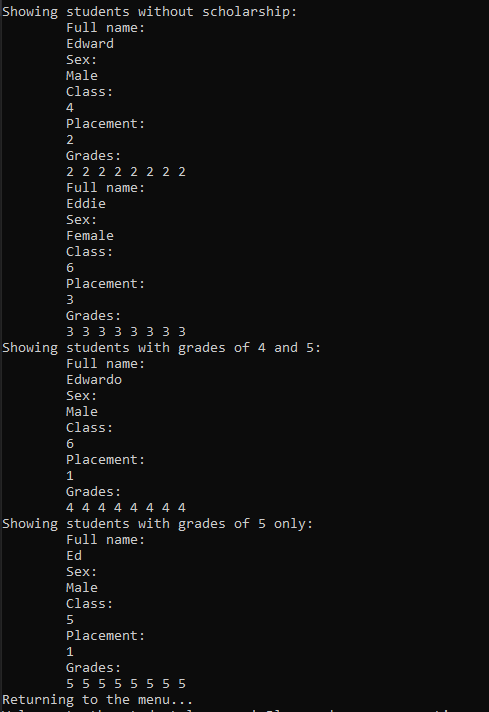


Рисунок Вывод информации о стипендиях студента

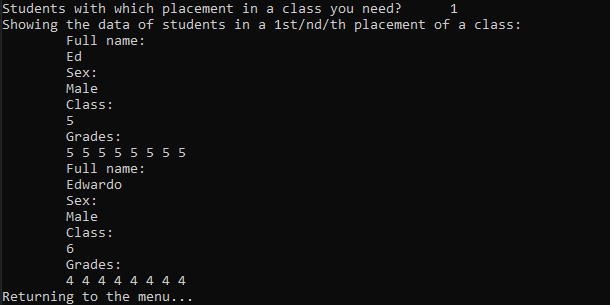


Рисунок Вывод данных обо всех студентах с одним номером внутри группы

Блок таблицы с тестовыми данными

Ed

0

5

1

5 5 5 5 5 5 5 5

Edward

0

4

2

2 2 2 2 2 2 2 2

Eddie

1

6

3

3 3 3 3 3 3 3 3

**Выводы.**

В ходе выполнения практической работы была создана программа для работы с массивом данных о студентах. Программа была реализована с помощью таких типов структур данных, как структуры, динамические массивы, динамические массивы структур и перечисления. Также программа задействовала функции работы с файловой системой. Эта программа имеет следующие функции: ввод, вывод и редактирование данных о студентах, а также специфицированный вывод данных о студентах.

Приложение А

рабочий код

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

enum Sex

{

Male, Female

};

struct Student

{

string fullName;

Sex sex;

int classNumber;

int classListNumber;

int grades[8];

};

void Menu();

void StudentInput()

{

cout << "\n" << "Entering the student's info into the log.";

Student s;

cout << "\n" << "Student's full name:\t";

cin >> s.fullName;

cout << "Student's sex:\n\t 0 - Male\n\t 1 - Female\t";

bool sexInput;

cin >> sexInput;

s.sex = static\_cast<Sex>(sexInput);

cout << "Student's class:\t";

cin >> s.classNumber;

cout << "Student's placement in the class:\t";

cin >> s.classListNumber;

cout << "Student's grades:\t\n";

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

{

cout << "grade #" << i+1 <<":\t";

cin >> s.grades[i];

}

ofstream studentLog;

studentLog.open("Student Log.txt", ios::app);

if (studentLog.is\_open())

{

studentLog << s.fullName<< "\n";

studentLog << s.sex << "\n";

studentLog << s.classNumber << "\n";

studentLog << s.classListNumber << "\n";

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

{

studentLog << s.grades[i] << " ";

}

studentLog << "\n";

studentLog.close();

cout << "\n" << "This student is saved into the log.";

}

else

{

cout << "\n" << "Error opening file.";

}

bool confirmation;

cout << "\n" << "Enter the next student into the log? 1/0\t";

cin >> confirmation;

if (confirmation)

{

StudentInput();

}

else

{

cout << "\n" << "Returning to the menu...\n";

Menu();

}

}

int StudentCount()

{

ifstream studentLog("Student Log.txt");

if (studentLog.is\_open())

{

int temp = 0;

string data;

while (!studentLog.eof())

{

getline(studentLog, data);

temp++;

}

studentLog.close();

int n;

n = temp / 5;

return n;

}

else

{

cout << "\n" << "Error opening file.";

return 0;

}

}

void StudentOutput()

{

ifstream studentLog("Student Log.txt");

if (studentLog.is\_open())

{

cout << "\n" << "Showing the data of all students:";

int count = StudentCount();

Student\* s = new Student[count];

if (count > 0)

{

int temp = 0;

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

{

studentLog >> s[i].fullName;

studentLog >> temp;

s[i].sex = static\_cast<Sex>(temp);

studentLog >> s[i].classNumber;

studentLog >> s[i].classListNumber;

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

{

studentLog >> s[i].grades[j];

}

}

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

{

cout << "\n\t" << "Full name:";

cout << "\n\t" << s[i].fullName;

cout << "\n\t" << "Sex:";

switch (s[i].sex)

{

case 0:

cout << "\n\t" << "Male";

break;

case 1:

cout << "\n\t" << "Female";

break;

default:

break;

}

cout << "\n\t" << "Class:";

cout << "\n\t" << s[i].classNumber;

cout << "\n\t" << "Placement:";

cout << "\n\t" << s[i].classListNumber;

cout << "\n\t" << "Grades:\n\t";

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

{

cout << s[i].grades[j] << " ";

}

}

}

else

{

cout << "\n\t" << "The log is empty.";

}

delete[] s;

}

else

{

cout << "\n" << "Error opening file.";

}

cout << "\n" << "Returning to the menu...\n";

Menu();

}

void StudentEdit()

{

int studentNum = 0;

int count = StudentCount();

cout << "\n" << "Enter a number of a student's data you want to edit (The log start with 0):\t";

cin >> studentNum;

if (studentNum < count)

{

Student\* s = new Student[count];

fstream studentLog;

studentLog.open("Student Log.txt", ios::in);

if (studentLog.is\_open())

{

bool temp;

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

{

studentLog >> s[i].fullName;

studentLog >> temp;

s[i].sex = static\_cast<Sex>(temp);

studentLog >> s[i].classNumber;

studentLog >> s[i].classListNumber;

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

{

studentLog >> s[i].grades[j];

}

}

}

else

{

cout << "\n" << "Error opening file.";

}

studentLog.close();

cout << "\n" << "Student's full name:\t";

cin >> s[studentNum].fullName;

cout << "Student's sex:\n\t 0 - Male\n\t 1 - Female\t";

bool sexInput;

cin >> sexInput;

s[studentNum].sex = static\_cast<Sex>(sexInput);

cout << "Student's class:\t";

cin >> s[studentNum].classNumber;

cout << "Student's placement in the class:\t";

cin >> s[studentNum].classListNumber;

cout << "Student's grades:\t\n";

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

{

cout << "grade #" << i + 1 << ":\t";

cin >> s[studentNum].grades[i];

}

studentLog.open("Student Log.txt", ios::out);

if (studentLog.is\_open())

{

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

{

studentLog << s[i].fullName << "\n";

studentLog << s[i].sex << "\n";

studentLog << s[i].classNumber << "\n";

studentLog << s[i].classListNumber << "\n";

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

{

studentLog << s[i].grades[j] << " ";

}

studentLog << "\n";

}

studentLog.close();

cout << "\n" << "This student has been edited.";

}

else

{

cout << "\n" << "Error opening file.";

}

delete[] s;

}

else

{

cout << "The number is out of range. A student with this number doesn't exist";

}

cout << "\n" << "Returning to the menu...\n";

Menu();

}

void StudentSameClass()

{

cout << "Students from which class you need?\t";

int studentClass;

cin >> studentClass;

ifstream studentLog("Student Log.txt");

if (studentLog.is\_open())

{

cout << "Showing the data of students in a " << studentClass << "class:";

bool classFound = 0;

int count = StudentCount();

Student\* s = new Student[count];

bool temp;

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

{

studentLog >> s[i].fullName;

studentLog >> temp;

s[i].sex = static\_cast<Sex>(temp);

studentLog >> s[i].classNumber;

studentLog >> s[i].classListNumber;

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

{

studentLog >> s[i].grades[j];

}

}

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

{

if (studentClass == s[i].classNumber)

{

classFound = 1;

cout << "\n\t" << "Full name:";

cout << "\n\t" << s[i].fullName;

cout << "\n\t" << "Sex:";

switch (s[i].sex)

{

case 0:

cout << "\n\t" << "Male";

break;

case 1:

cout << "\n\t" << "Female";

break;

default:

break;

}

cout << "\n\t" << "Placement:";

cout << "\n\t" << s[i].classListNumber;

cout << "\n\t" << "Grades:\n\t";

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

{

cout << s[i].grades[j] << " ";

}

}

}

if(!classFound)

{

cout << "\n" << "Class hasn't been found.";

}

delete[] s;

}

else

{

cout << "\n" << "Error opening file.";

}

cout << "\n" << "Returning to the menu...\n";

Menu();

}

void StudentRating()

{

ifstream studentLog("Student Log.txt");

if (studentLog.is\_open())

{

cout << "\n" << "Showing the rating of students:";

bool classFound = 0;

int count = StudentCount();

Student\* s = new Student[count];

bool temp;

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

{

studentLog >> s[i].fullName;

studentLog >> temp;

s[i].sex = static\_cast<Sex>(temp);

studentLog >> s[i].classNumber;

studentLog >> s[i].classListNumber;

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

{

studentLog >> s[i].grades[j];

}

}

float \*gradeAverages= new float[count]();

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

{

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

{

gradeAverages[i] += s[i].grades[j];

}

gradeAverages[i] = gradeAverages[i] / 8;

}

float temp1 = 0;

string temp2 = "";

for (int i = 0; i < count-1;i++)

{

for (int j = i+1; j < count;j++)

{

if (gradeAverages[i] < gradeAverages[j])

{

temp1 = gradeAverages[j];

gradeAverages[j] = gradeAverages[i];

gradeAverages[i] = temp1;

temp2 = s[j].fullName;

s[j].fullName = s[i].fullName;

s[i].fullName = temp2;

}

}

}

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

{

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

cout << s[i].fullName << " " << gradeAverages[i];

}

delete[] gradeAverages;

delete[] s;

}

else

{

cout << "\n" << "Error opening file.";

}

cout << "\n" << "Returning to the menu...\n";

Menu();

}

void StudentSexCount()

{

ifstream studentLog("Student Log.txt");

if (studentLog.is\_open())

{

cout << "\n" << "Showing the number of male and female students:";

int M = 0, F = 0;

int count = StudentCount();

Student\* s = new Student[count];

bool temp;

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

{

studentLog >> s[i].fullName;

studentLog >> temp;

s[i].sex = static\_cast<Sex>(temp);

studentLog >> s[i].classNumber;

studentLog >> s[i].classListNumber;

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

{

studentLog >> s[i].grades[j];

}

}

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

{

if (s[i].sex == Male)

{

M++;

}

else

{

F++;

}

}

cout << "\n" << "Male students:\t" << M;

cout << "\n" << "Female students:\t" << F;

delete[] s;

}

else

{

cout << "\n" << "Error opening file.";

}

cout << "\n" << "Returning to the menu...\n";

Menu();

}

void StudentGrades()

{

ifstream studentLog("Student Log.txt");

if (studentLog.is\_open())

{

int count = StudentCount();

Student\* s = new Student[count];

bool temp;

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

{

studentLog >> s[i].fullName;

studentLog >> temp;

s[i].sex = static\_cast<Sex>(temp);

studentLog >> s[i].classNumber;

studentLog >> s[i].classListNumber;

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

{

studentLog >> s[i].grades[j];

}

}

bool\* hasThrees = new bool[count]();

bool\* hasFours = new bool[count]();

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

{

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

{

if (s[i].grades[j] < 4)

{

hasThrees[i] = 1;

}

if (s[i].grades[j] < 5)

{

hasFours[i] = 1;

}

}

}

cout << "\n" << "Showing students without scholarship:";

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

{

if (hasThrees[i])

{

cout << "\n\t" << "Full name:";

cout << "\n\t" << s[i].fullName;

cout << "\n\t" << "Sex:";

switch (s[i].sex)

{

case 0:

cout << "\n\t" << "Male";

break;

case 1:

cout << "\n\t" << "Female";

break;

default:

break;

}

cout << "\n\t" << "Class:";

cout << "\n\t" << s[i].classNumber;

cout << "\n\t" << "Placement:";

cout << "\n\t" << s[i].classListNumber;

cout << "\n\t" << "Grades:\n\t";

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

{

cout << s[i].grades[j] << " ";

}

}

}

cout << "\n" << "Showing students with grades of 4 and 5:";

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

{

if (!hasThrees[i] && hasFours[i])

{

cout << "\n\t" << "Full name:";

cout << "\n\t" << s[i].fullName;

cout << "\n\t" << "Sex:";

switch (s[i].sex)

{

case 0:

cout << "\n\t" << "Male";

break;

case 1:

cout << "\n\t" << "Female";

break;

default:

break;

}

cout << "\n\t" << "Class:";

cout << "\n\t" << s[i].classNumber;

cout << "\n\t" << "Placement:";

cout << "\n\t" << s[i].classListNumber;

cout << "\n\t" << "Grades:\n\t";

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

{

cout << s[i].grades[j] << " ";

}

}

}

cout << "\n" << "Showing students with grades of 5 only:";

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

{

if (!hasThrees[i] && !hasFours[i])

{

cout << "\n\t" << "Full name:";

cout << "\n\t" << s[i].fullName;

cout << "\n\t" << "Sex:";

switch (s[i].sex)

{

case 0:

cout << "\n\t" << "Male";

break;

case 1:

cout << "\n\t" << "Female";

break;

default:

break;

}

cout << "\n\t" << "Class:";

cout << "\n\t" << s[i].classNumber;

cout << "\n\t" << "Placement:";

cout << "\n\t" << s[i].classListNumber;

cout << "\n\t" << "Grades:\n\t";

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

{

cout << s[i].grades[j] << " ";

}

}

}

delete[] hasFours;

delete[] hasThrees;

delete[] s;

}

else

{

cout << "\n" << "Error opening file.";

}

cout << "\n" << "Returning to the menu...\n";

Menu();

}

void StudentPlacement()

{

cout << "Students with which placement in a class you need?\t";

int studentPlacement;

cin >> studentPlacement;

ifstream studentLog("Student Log.txt");

if (studentLog.is\_open())

{

cout << "Showing the data of students in a " << studentPlacement << "st/nd/th placement of a class:";

bool classFound = 0;

int count = StudentCount();

Student\* s = new Student[count];

bool temp;

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

{

studentLog >> s[i].fullName;

studentLog >> temp;

s[i].sex = static\_cast<Sex>(temp);

studentLog >> s[i].classNumber;

studentLog >> s[i].classListNumber;

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

{

studentLog >> s[i].grades[j];

}

}

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

{

if (studentPlacement == s[i].classListNumber)

{

classFound = 1;

cout << "\n\t" << "Full name:";

cout << "\n\t" << s[i].fullName;

cout << "\n\t" << "Sex:";

switch (s[i].sex)

{

case 0:

cout << "\n\t" << "Male";

break;

case 1:

cout << "\n\t" << "Female";

break;

default:

break;

}

cout << "\n\t" << "Class:";

cout << "\n\t" << s[i].classNumber;

cout << "\n\t" << "Grades:\n\t";

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

{

cout << s[i].grades[j] << " ";

}

}

}

if (!classFound)

{

cout << "\n" << "This placement in a class hasn't been found.";

}

delete[] s;

}

else

{

cout << "\n" << "Error opening file.";

}

cout << "\n" << "Returning to the menu...\n";

Menu();

}

void Menu()

{

cout << "Welcome to the student log menu! Please choose your option";

cout << "\n\t0 - Enter a student's info;";

cout << "\n\t1 - Show info of all students;";

cout << "\n\t2 - Edit a student's info;";

cout << "\n\t3 - Show info of students in a same class;";

cout << "\n\t4 - Show a rating of students;";

cout << "\n\t5 - Show a number of male and female students;";

cout << "\n\t6 - Show info about a students' grades;";

cout << "\n\t7 - Show info of students with a same placement inside a class;";

cout << "\n\t8 - Exit the program.\n\t";

int choice = 0;

cin >> choice;

switch (choice)

{

case 0:

StudentInput();

break;

case 1:

StudentOutput();

break;

case 2:

StudentEdit();

break;

case 3:

StudentSameClass();

break;

case 4:

StudentRating();

break;

case 5:

StudentSexCount();

break;

case 6:

StudentGrades();

break;

case 7:

StudentPlacement();

break;

case 8:

exit;

break;

default:

cout << "\n" << "Nonexistant option";

exit;

break;

}

}

int main()

{

Menu();

return 0;

}