**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. |  | Симаков Г.О. |
| 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. 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), 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.   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*.  
    
  \* Данные студентов должны храниться в файле.

**Выполнение работы.**

Код программы представлен в приложении А.

Приложение А

рабочий код

#include <iostream>  
#include <Windows.h>  
#include <stdio.h>  
#include <math.h>  
#include <stdlib.h>  
#include <fstream>  
#include <iomanip>  
using namespace std;  
  
struct Student {  
 string Fam;  
 string Name;  
 string Patronymic;  
 char Sex;  
 int Group;  
 int idInGroup;  
 int Grades[8]{ 0, 0, 0, 0, 0, 0, 0, 0 };  
 float avg = 0.0f;  
}S;  
  
  
  
/\*void Copy()  
{  
 ifstream reading("Input.txt");  
 ofstream record("Buffer.txt", ios::out); // открытие файла для ввода  
  
 if (reading){  
 if (record) {  
  
 int N;  
 reading >> N;  
 record << N << endl;  
  
 for (int i = 0; i < N; i++) {  
 reading >> S.Fam;  
 record << S.Fam << '\n';  
  
 reading >> S.Name;  
 record << S.Name << '\n';  
  
 reading >> S.Patronymic;  
 record << S.Patronymic << '\n';  
  
 reading >> S.Sex;  
 record << S.Sex << '\n';  
  
 reading >> S.Group;  
 record << S.Group << '\n';  
  
 reading >> S.idInGroup;  
 record << S.idInGroup << '\n';  
  
 for (int j = 0; j < 8; j++) {  
 reading >> S.Grades[j];  
 record << S.Grades[j] << ' ';  
 }  
  
   
 }  
 }  
 else {  
 cout << "Ошибка открытия файла";  
 }  
 }  
 else {  
 cout << "Ошибка открытия файла";  
 }  
 reading.close();  
 record.close();  
}\*/  
  
void readFile(Student\* (&S))  
{  
 ifstream reading("Input.txt");  
  
 if (reading) {  
 int N;  
 reading >> N;  
 S = new Student[N]; // Выделяем память  
 for (int i = 0; i < N; i++) {  
 reading >> S[i].Fam;  
 reading >> S[i].Name;  
 reading >> S[i].Patronymic;  
 reading >> S[i].Sex;  
 reading >> S[i].Group;  
 reading >> S[i].idInGroup;  
 for (int j = 0; j < 8; j++) {  
 reading >> S[i].Grades[j];  
 }  
 }  
 }  
 else {  
 cout << "Открыть файл не удалось! \n";  
 }  
 reading.close();  
}  
  
int AmountOfStudent()  
{  
 ifstream reading("Input.txt");  
 int N;  
 if (reading) {  
 reading >> N;  
 }  
 else {  
 cout << "Открыть файл не удалось! \n";  
 }  
  
 reading.close();  
  
 return N;  
}  
  
void Print()  
{  
 ifstream reading("Input.txt");  
  
 if (reading) {  
 int N;  
 reading >> N;  
  
 cout << "Количество студентов: " << N << endl << endl;  
  
 for (int i = 0; i < N; i++) {  
  
 reading >> S.Fam;  
 reading >> S.Name;  
 reading >> S.Patronymic;  
 reading >> S.Sex;  
 reading >> S.Group;  
 reading >> S.idInGroup;  
 for (int j = 0; j < 8; j++) {  
 reading >> S.Grades[j];  
 }  
  
 cout << "Student №" << i + 1 << ":" << endl;  
 cout << "FIO: " << S.Fam << " " << S.Name << " " << S.Patronymic << endl;  
 cout << "SEX:" << S.Sex << endl;  
 cout << "GROUP: " << S.Group << endl;  
 cout << "ID IN GROUP: " << S.idInGroup << endl;  
 cout << "GRADES: ";  
 for (int j = 0; j < 8; j++) {  
 cout << " " << S.Grades[j];  
 }  
 cout << endl;  
 cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  
 }  
  
 cout << "Данные считаны!\n";  
 }  
 else {  
 cout << "Открыть файл не удалось! \n";  
 }  
}  
  
void changeStudent(Student\* S)  
{  
 readFile(S);  
  
 int N = AmountOfStudent();  
 cout << "Выберети номер студента из базы (от 1 до " << N << "): ";  
 int dn;  
 cin >> dn;  
 --dn;  
 if (dn >= 0 && dn < N) {  
 cout << endl << endl << "FIO: " << S[dn].Fam << " " << S[dn].Name << " " << S[dn].Patronymic << endl;  
 cout << "SEX: " << S[dn].Sex << endl;  
 cout << "Group: " << S[dn].Group << endl;  
 cout << "Id in group: " << S[dn].idInGroup << endl;  
 cout << "GRADES: ";  
 for (int j = 0; j < 8; j++) {  
 cout << " " << S[dn].Grades[j];  
 }  
 cout << endl;  
  
 cout << endl << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_NEW\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";  
 cout << endl << endl << "NEW Surname: ";  
 cin >> S[dn].Fam;  
 cout << endl << "NEW Name: ";  
 cin >> S[dn].Name;  
 cout << endl << "NEW Patronymic: ";  
 cin >> S[dn].Patronymic;  
 cout << endl << "NEW Sex (M/F): ";  
 cin >> S[dn].Sex;  
 cout << endl << "NEW Group: ";  
 cin >> S[dn].Group;  
 cout << endl << "NEW Id in group: ";  
 cin >> S[dn].idInGroup;  
 cout << endl << "NEW Grades: ";  
 for (int j = 0; j < 8; j++) {  
 cin >> S[dn].Grades[j];  
 }  
 cout << endl;  
  
 cout << "\t\tData changed" << endl << endl;  
 }  
 else {  
 cout << "Student not found, please try again" << endl;  
 }  
  
 ofstream record("Input.txt"); // открытие файла для ввода  
  
 if (record) {  
 record << N << endl;  
 for (int i = 0; i < N; i++) {  
 record << S[i].Fam << endl;  
 record << S[i].Name << endl;  
 record << S[i].Patronymic << endl;  
 record << S[i].Sex << endl;  
 record << S[i].Group << endl;  
 record << S[i].idInGroup << endl;  
 for (int j = 0; j < 8; j++) {  
 record << S[i].Grades[j] << " ";  
 }  
 record << endl;  
 }  
 }  
 else {  
 cout << "Открыть файл не удалось! \n";  
 }  
  
 record.close();  
  
}  
  
void AddStudent()  
{  
 int N = AmountOfStudent() + 1;  
  
 ofstream record("Input.txt", ios::app);  
 ofstream \_record("Input.txt", ios::out| ios::in);  
  
 if (\_record) {  
 \_record << N << endl;  
 }  
 else {  
 cout << "Ошибка открытия файла!\n";  
 }  
  
  
 cout << "\t\tВведите данные студента";  
 cout << endl << endl << "Surname: ";  
 cin >> S.Fam;  
 cout << endl << "Name: ";  
 cin >> S.Name;  
 cout << endl << "Patronymic: ";  
 cin >> S.Patronymic;  
 cout << endl << "Sex (M/F): ";  
 cin >> S.Sex;  
 cout << endl << "Group: ";  
 cin >> S.Group;  
 cout << endl << "Id in group: ";  
 cin >> S.idInGroup;  
 cout << endl << "8 grades per semester: ";  
 for (int j = 0; j < 8; j++) {  
 cin >> S.Grades[j];  
 }  
 cout << endl << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Data added\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n";  
  
 if (record) {  
 record << endl;  
  
  
 record << S.Fam << endl;  
 record << S.Name << endl;  
 record << S.Patronymic << endl;  
 record << S.Sex << endl;  
 record << S.Group << endl;  
 record << S.idInGroup << endl;  
 for (int j = 0; j < 8; j++) {  
 record << S.Grades[j] << " ";  
 }  
 record << endl;  
 }  
 else {  
 cout << "Открыть файл не удалось! \n";  
 }  
  
 record.close();  
 \_record.close();  
  
}  
  
void studOfGroup(Student\* S)  
{  
 readFile(S);  
  
 int N = AmountOfStudent();  
 int cnt = 0;  
 cout << "Введите номер группы:" << " ";  
 int k;  
 cin >> k;  
 bool ExitFlag = false;  
 system("cls");  
 cout << "\t\tStudents of group " << k << ":" << endl << endl;  
 for (int i = 0; i < N; i++) {  
 if (S[i].Group == k && k >= 0) {  
 cnt++;  
  
 cout << "Student №" << i + 1 << ":" << endl;  
 cout << "FIO: " << S[i].Fam << " " << S[i].Name << " " << S[i].Patronymic << endl;  
 cout << "SEX:" << S[i].Sex << endl;  
 cout << "GROUP: " << S[i].Group << endl;  
 cout << "ID IN GROUP: " << S[i].idInGroup << endl;  
 cout << "GRADES: ";  
 for (int j = 0; j < 8; j++) {  
 cout << " " << S[i].Grades[j];  
 }  
 cout << endl;  
 cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  
 }  
  
 }  
 if (cnt == 0) {  
 cout << "Такой группы нет!\n\n\n";  
 }  
}  
  
void studOfId(Student\* S)  
{  
 readFile(S);  
 int cnt = 0;  
 int N = AmountOfStudent();  
 cout << "Введите номер студента в группе:" << " ";  
 int id;  
 cin >> id;  
 system("cls");  
 cout << "\t\tStudents with number " << id << ":" << endl << endl;  
 for (int i = 0; i < N; i++) {  
 if (S[i].idInGroup == id && id >= 0) {  
 cnt++;  
 cout << "Student №" << i + 1 << ":" << endl;  
 cout << "FIO: " << S[i].Fam << " " << S[i].Name << " " << S[i].Patronymic << endl;  
 cout << "SEX:" << S[i].Sex << endl;  
 cout << "GROUP: " << S[i].Group << endl;  
 cout << "ID IN GROUP: " << S[i].idInGroup << endl;  
 cout << "GRADES: ";  
 for (int j = 0; j < 8; j++) {  
 cout << " " << S[i].Grades[j];  
 }  
 cout << endl;  
 cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  
 }  
 }  
 if (cnt == 0) {  
 cout << "Студента с таким номером нет!\n\n\n";  
 }  
}  
  
void topOfStud(Student\* S)  
{  
 readFile(S);  
 int N = AmountOfStudent();  
 for (int i = 0; i < N; i++) {  
 int sum = 0;  
 for (int j = 0; j < 8; j++) {  
 sum += S[i].Grades[j];  
 }  
 S[i].avg = (float)sum / 8;  
 }  
  
 for (int i = 0; i < N - 1; i++) {  
 for (int j = 0; j < N - i - 1; j++) {  
 if (S[j].avg < S[j + 1].avg) {  
 swap(S[j], S[j + 1]);  
 }  
 }  
 }  
 cout << "\t\t\t\tTop 3 students:\n\n";  
 for (int i = 0; i < 3; i++) {  
 cout << endl;  
 cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ # " << i + 1 << " whith avg = (" << S[i].avg << ")\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;;  
 cout << "Student №" << i + 1 << ":" << endl;  
 cout << "FIO: " << S[i].Fam << " " << S[i].Name << " " << S[i].Patronymic << endl;  
 cout << "SEX:" << S[i].Sex << endl;  
 cout << "GROUP: " << S[i].Group << endl;  
 cout << "ID IN GROUP: " << S[i].idInGroup << endl;  
 }  
 cout << endl << endl;  
}  
  
void sortBySex(Student\* S)  
{  
 readFile(S);  
 int N = AmountOfStudent();  
 int m = 0, w = 0;  
 for (int i = 0; i < N; i++) {  
 if (S[i].Sex == 'M') {  
 m += 1;  
 }  
 else {  
 w += 1;  
 }  
 }  
 cout << "\t\t\t\tNUMBER OF STUDENTS \_ " << N << endl << endl;  
 cout << "\tNumber of male = " << m << '\n' << '\n';  
 cout << "\tNumber of female = " << w << endl << endl << endl;  
  
}  
  
void sortByGrades(Student\* S, int l)  
{  
 readFile(S);  
 int N = AmountOfStudent();  
 int cnt = 0;  
 for (int i = 0; i < N; i++) {  
 int grade = 5;  
 for (int j = 0; j < 8; j++)  
 grade = min(S[i].Grades[j], grade);  
  
 if (grade == l) {  
 cnt++;  
 cout << "Student №" << i + 1 << ":" << endl;  
 cout << "FIO: " << S[i].Fam << " " << S[i].Name << " " << S[i].Patronymic << endl;  
 cout << "SEX:" << S[i].Sex << endl;  
 cout << "GROUP: " << S[i].Group << endl;  
 cout << "ID IN GROUP: " << S[i].idInGroup << endl;  
 cout << "GRADES: ";  
 for (int j = 0; j < 8; j++) {  
 cout << " " << S[i].Grades[j];  
 }  
 cout << endl;  
 cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  
 }  
 }  
 if (cnt == 0) {  
 cout << "Студент не найден!\n\n\n";  
 }  
  
}  
  
  
  
  
int main()  
{  
 int amountOfStudent = AmountOfStudent();  
 Student\* S = new Student[amountOfStudent];  
  
 int cnt = 0;  
 SetConsoleCP(1251);  
 SetConsoleOutputCP(1251);  
  
 while (true) {  
 int choice;  
 cout << "Кол-во Студентов:\t" << AmountOfStudent() << endl << endl;  
 cout << "Введите номер меню:" << endl;  
 cout <<  
 "0.Выход\n"  
 "1.Вывести все данные\n"  
 "2.Добавить данные студента\n"  
 "3.Изменить данные студента\n"  
 "4.Вывести студентов группы N\n"  
 "5.Вывести студента с номером k\n"  
 "6.Вывести топ 3 студента\n"  
 "7.Вывести кол-во мужчин и женщин\n"  
 "8.Вывести по успеваемости\n" << endl;  
  
 cin >> choice;  
 switch (choice)  
 {  
 case 0:  
 system("cls");  
 cout << "Выход из программы...\n\n\n";  
 return 0;  
  
 case 1:  
 system("cls");  
 Print();  
 system("pause");  
 system("cls");  
 break;  
 case 2:  
 system("cls");  
 AddStudent();  
 system("pause");  
 system("cls");  
 break;  
 case 3:  
 system("cls");  
 changeStudent(S);  
 system("pause");  
 system("cls");  
 break;  
 case 4:  
 system("cls");  
 studOfGroup(S);  
 system("pause");  
 system("cls");  
 break;  
 case 5:  
 system("cls");  
 studOfId(S);  
 system("pause");  
 system("cls");  
 break;  
 case 6:  
 system("cls");  
 topOfStud(S);  
 system("pause");  
 system("cls");  
 break;  
 case 7:  
 system("cls");  
 sortBySex(S);  
 system("pause");  
 system("cls");  
 break;  
 case 8:  
 system("cls");  
 int us;  
 cout << endl <<  
 "3. Students without a scholarship\n"  
 "4. Students which got 4s and 5s\n"  
 "5. Students which got only 5s\n";  
 cin >> us;  
 switch (us) {  
 case 3:  
 system("cls");  
 sortByGrades(S, us);  
 system("pause");  
 system("cls");  
 break;  
 case 4:  
 system("cls");  
 sortByGrades(S, us);  
 system("pause");  
 system("cls");  
 break;  
 case 5:  
 system("cls");  
 sortByGrades(S,us);  
 system("pause");  
 system("cls");  
 break;  
 }  
 break;  
 default:  
 system("cls");  
 cout << "Неверно выбран номер меню!\n\n";  
 system("pause");  
 system("cls");  
 break;  
  
 }  
  
  
 }  
}