**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

|  |  |  |
| --- | --- | --- |
| Студентка гр. 2373 |  | Басова К.В. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2023

**Цель работы.**

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

**Основные теоретические положения.**

Структуры представляют собой группы связанных между собой, как правило, разнотипных переменных, объединенных в единый объект, в отличие от массива, все элементы которого однотипны. В языке C++ структура является видом класса и обладает всеми его свойствами. 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.

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}}

};

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.

Через указатели можно работать с отдельными полями структур. Для доступа к полю структуры через указатель используется оператор ‘’.

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.

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.

**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). Данные о студентах хранятся в файле.

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»;

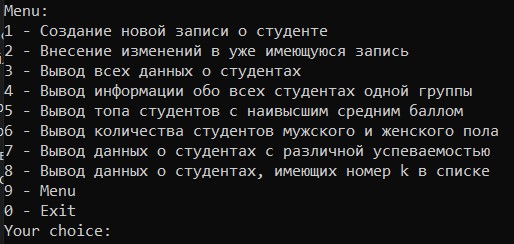
8.   Вывод данных о студентах, имеющих номер в списке – *k*.

**Выполнение работы.**

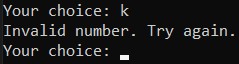
Код программы представлен в приложении А.

Блок описания кода и использованных алгоритмов

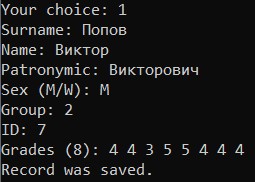
1. При запуске программы появляется окно, в котором выводится меню допустимых операций под номерами. Пользователю предлагается ввести любой номер операции от 0 до 9:



2. При попытке ввода символа, не входящего в диапазон выбора, пользователю предлагается ввести номер ещё раз:



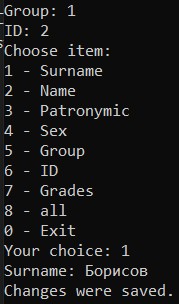
3. Операция 1 – создание новой записи. Пользователю предлагается по очереди внести сведения о студенте. Введённые данные сохраняются в структуру.



При вводе недопустимых оценок, выводится сообщение о том, что данные не могут быть сохранены.



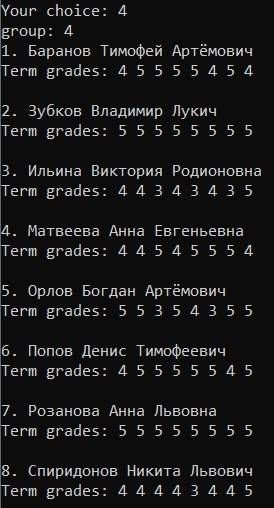
4. Операция 2 – внесение изменений в запись. Пользователю предлагается изменить либо 1 пункт записи, либо всю запись. Чтобы определить, какую запись необходимо изменить, вводится номер группы и порядковый номер нужного студента.



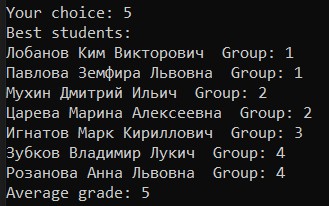
5. Операции 3 – 8 – вывод данных о студентах, классифицированных по определённому признаку.

3 – вывод данных о всех студентах.

4 – вывод данных о студентах выбранной группы. На экран выводится порядковый номер, ФИО и оценки студентов. Пример выводимой информации:



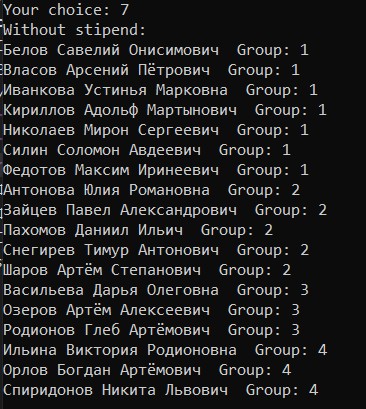
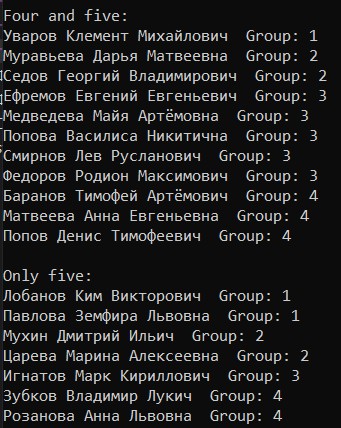
5 – список студентов с наивысшим средним баллом. Выводятся ФИО, номер группы и средний балл лучших студентов.



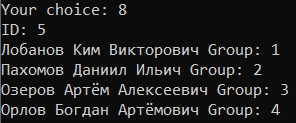
6 – вывод количества студентов мужского и женского пола.



7 – вывод данных о студентах с различной успеваемостью (те, кто не получает стипендию; учится на «хорошо» и «отлично»; учится только на «отлично»).

8 – список студентов с номером k в группе (k вводится пользователем с клавиатуры).



6. Операция 9 – повторный вывод меню на экран, 0 – выход из программы.

**Выводы.**

Структуры позволяют сделать простой и удобной работу с данными различных типов, которые связаны друг с другом определённым образом (например, информация о ФИО, группе и оценках студента). При этом есть возможность анализировать и изменять части структуры по отдельности, а также создавать массивы структур для работы с большим количеством упорядоченных данных.

Приложение А

рабочий код

#include <iostream>

#include <fstream>

#include <string>

#include <Windows.h>

using namespace std;

struct Profile // Профайл студента

{

char surname[20], name[20], patron[20]; // ФИО

char sex; // Пол: M/W

char group; // Номер группы

int id; // Номер студента в списке

int grades[8]; // Оценки за экзамены и диф. зачёты

};

int countStudents() // Функция посчёта количества студентов

{

ifstream file("Database.txt");

if (file.is\_open())

{

int temp = 0;

string data;

while (!file.eof()) // Пока не достигнут конец файла, идёт чтение

{

getline(file, data); // Чтение строки файла

temp++;

}

file.close();

int n;

n = temp;

return n;

}

else

{

cout << "Error"; // Если не удалось прочитать файл, выдаётся ошибка

return 0;

}

}

int newRecord(Profile student[], int& c) { // Функция создания новой записи о студенте

bool no = 0;

cout << "Surname: ";

cin >> student[c].surname;

cout << "Name: ";

cin >> student[c].name;

cout << "Patronymic: ";

cin >> student[c].patron;

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

cin >> student[c].sex;

while (true) {

if ((student[c].sex != 'M') and (student[c].sex != 'W')) { // Проверка на правилоность ввода пола

cout << "Invalid data. Try again.\n";

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

cin >> student[c].sex;

}

else

break;

}

cout << "Group: ";

cin >> student[c].group;

cout << "ID: ";

cin >> student[c].id;

cout << "Grades (8): ";

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

cin >> student[c].grades[k];

if ((student[c].grades[k] < 3) or (student[c].grades[k] > 5))

no++; // Счётчик увеличивается, если введённые оценки недопустимы

}

if (no == 0) {

for (int i = 0; i < c; i++) { // Смещаем порядковые номера одногруппников, следующих за новым студентом

if (student[i].group == student[c].group) {

if ((student[i].id == student[c].id) or (student[i].id > student[c].id))

student[i].id++;

}

}

c++;

cout << "Record was saved.\n";

}

else

cout << "Found invalid number. Record can't be saved.\n";

return 0;

}

int changeRecord(Profile student[], int c) { // Функция изменения записи о студенте

int num;

char gr;

int found = 0;

cout << "Group: ";

cin >> gr; // Вводим номер группы нужного студента

cout << "ID: ";

cin >> num; // Вводим порядковый номер студента в группе

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

if ((student[i].group == gr) and (student[i].id == num)) { // Находим нужного студента

found = 1;

cout << "Choose item: \n1 - Surname \n2 - Name \n3 - Patronymic \n";

cout << "4 - Sex \n5 - Group \n6 - ID \n7 - Grades \n8 - all \n0 - Exit";

cout << "\nYour choice: ";

int item\_n;

cin >> item\_n; // Выбираем, что нужно изменить

switch (item\_n) {

case 0: break;

case 1:

cout << "Surname: ";

cin >> student[i].surname;

break;

case 2:

cout << "Name: ";

cin >> student[i].name;

break;

case 3:

cout << "Patronymic: ";

cin >> student[i].patron;

break;

case 4:

cout << "Sex: ";

cin >> student[i].sex;

break;

case 5:

cout << "Group: ";

cin >> student[i].group;

break;

case 6:

cout << "ID: ";

cin >> student[i].id;

break;

case 7:

cout << "Grades: ";

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

cin >> student[i].grades[g];

}

break;

case 8:

cout << "Surname: ";

cin >> student[i].surname;

cout << "Name: ";

cin >> student[i].name;

cout << "Patronymic: ";

cin >> student[i].patron;

cout << "Sex: ";

cin >> student[i].sex;

cout << "Group: ";

cin >> student[i].group;

cout << "ID: ";

cin >> student[i].id;

cout << "Grades: ";

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

cin >> student[i].grades[g];

}

break;

}

cout << "Changes were saved.\n";

break;

}

}

if (found == 0)

cout << "Student not found\n";

return 0;

}

int showAllStudents(Profile student[], int c) { // Функция вывода данных о всех студентах

int temp = 0;

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

cout << i + 1 << ". " << student[i].surname << " " << student[i].name << " " << student[i].patron << "\n";

cout << "Group:" << student[i].group << " ID:" << student[i].id << "\n";

cout << "Grades: ";

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

cout << student[i].grades[k] << " ";

cout << "\n";

temp++;

}

if (temp == 0)

cout << "No records were found" << endl;

return 0;

}

int showGroup(Profile student[], int c) { // Функция вывода данных о студентах одной группы

int temp = 0;

char n;

cout << "group: ";

cin >> n;

for (int i = 0; i < c; i++) // Проходим по всем студентам

{

if (student[i].group == n) // Сравниваем номер группы текущего студента с введённым номером

{

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

cout << student[i].surname << " " << student[i].name << " " << student[i].patron << "\n";

cout << "Term grades: ";

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

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

cout << "\n\n";

temp++;

}

}

if (temp == 0) // Если ни одной записи не найдено

cout << "No records were found" << endl;

return 0;

}

int showBestStudents(Profile student[], int c) { // Функция вывода данных о студента

int temp = 0, sum = 0, maxg = 0;

float aver;

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

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

sum += student[i].grades[k];

}

if (sum >= maxg) {

maxg = sum;

}

if (sum == 40)

break;

temp++;

sum = 0;

}

aver = maxg / 8.0;

cout << "Best students: \n";

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

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

sum += student[i].grades[k];

}

if (sum == maxg) {

cout << student[i].surname << " " << student[i].name << " " << student[i].patron;

cout << " Group: " << student[i].group << "\n";

}

sum = 0;

}

if (temp == 0) // Если ни одной записи не найдено

cout << "No records were found" << endl;

else

cout << "Average grade: " << aver << "\n";

return 0;

}

int countSex(Profile student[], int c) { // Функция подсчёта студентов женского и мужского пола

int temp = 0, male = 0, female = 0;

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

if (student[i].sex == 'M')

male++;

else

female++;

temp++;

}

if (temp == 0) // Если ни одной записи не найдено

cout << "No records were found" << endl;

else

cout << "Male: " << male << " " << "Female: " << female << "\n";

return 0;

}

int sortStipend(Profile student[], int c) { // Функция вывода данных о студентах с различной успеваемостью

int temp = 0, count4 = 0, count3 = 0;

string three[30], four[30], five[30]; // Создаём массивы для сортировки студентов по оценкам

int i3 = 0, i4 = 0, i5 = 0;

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

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

if (student[i].grades[k] == 3) { // Если среди оценок есть 3

count3++;

break;

}

else if (student[i].grades[k] == 4) { // Если среди оценок есть 4

count4++;

}

}

if ((count4 == 0) and (count3 == 0)) { // Если среди оценок нет ни 3, ни 4

five[i5] = string(student[i].surname) + " " + string(student[i].name) + " " + string(student[i].patron);

five[i5] += " Group: ";

five[i5] += student[i].group; // Запись в массив с отличниками

i5++;

}

else if (count3 == 0) { // Нет 3, но есть 4

four[i4] = string(student[i].surname) + " " + string(student[i].name) + " " + string(student[i].patron);

four[i4] += " Group: ";

four[i4] += student[i].group; // Запись в массив хорошистов

i4++;

}

else { // Если есть 3

three[i3] = string(student[i].surname) + " " + string(student[i].name) + " " + string(student[i].patron);

three[i3] += " Group: ";

three[i3] += student[i].group; // Запись в массив троечников

i3++;

}

count4 = 0;

count3 = 0;

temp++;

}

if (temp == 0) // Если ни одной записи не найдено

cout << "No records were found" << endl;

else { // Вывод данных о студентов в соответствии с категориями

cout << "Without stipend:\n";

for (int i = 0; i < i3; i++)

cout << three[i] << "\n";

cout << "\nFour and five:\n";

for (int i = 0; i < i4; i++)

cout << four[i] << "\n";

cout << "\nOnly five:\n";

for (int i = 0; i < i5; i++)

cout << five[i] << "\n";

}

return 0;

}

int sortID(Profile student[], int c) { // Функция вывода данных о студентах с выбранным порядковым номером

int temp = 0, k;

cout << "ID: ";

cin >> k;

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

if (student[i].id == k) {

cout << student[i].surname << " " << student[i].name << " " << student[i].patron;

cout << " Group: " << student[i].group << "\n";

}

temp++;

}

if (temp == 0) // Если ни одной записи не найдено

cout << "No records were found" << endl;

return 0;

}

int main(){

SetConsoleCP(1251); // Установка русского языка

SetConsoleOutputCP(1251);

Profile student[50]; // Создание массива структур

int size;

size = countStudents(); // Подсчёт кол-ва студентов в файле

ifstream file("Database.txt");

if (!file.is\_open())

cout << "Error!";

else

{

if (size == 0)

cout << "The file is empty." << endl; // Если кол-во строк в файле = 0, то он пустой

for (int i = 0; i < size; i++) // Считываем данные всех студентов в массив структур

{

file >> student[i].surname >> student[i].name >> student[i].patron;

file >> student[i].sex >> student[i].group >> student[i].id;

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

file >> student[i].grades[j];

}

file.close(); // Закрываем файл

}

string Menu = "Menu: \n1 - Создание новой записи о студенте \n2 - Внесение изменений в уже имеющуюся запись \

\n3 - Вывод всех данных о студентах \n4 - Вывод информации обо всех студентах одной группы \

\n5 - Вывод топа студентов с наивысшим средним баллом \n6 - Вывод количества студентов мужского и женского пола \

\n7 - Вывод данных о студентах c различной успеваемостью \n8 - Вывод данных о студентах, имеющих номер k в списке \

\n9 - Menu \n0 - Exit\n";

cout << Menu;

cout << "Your choice: ";

string menu\_n;

cin >> menu\_n; // Выбор пункта меню

while (menu\_n != "0") { // Работа с программой продолжится, пока не будет введён 0

if (menu\_n == "1")

newRecord(student, size);

else if (menu\_n == "2")

changeRecord(student, size);

else if (menu\_n == "3")

showAllStudents(student, size);

else if (menu\_n == "4")

showGroup(student, size);

else if (menu\_n == "5")

showBestStudents(student, size);

else if (menu\_n == "6")

countSex(student, size);

else if (menu\_n == "7")

sortStipend(student, size);

else if (menu\_n == "8")

sortID(student, size);

else if (menu\_n == "9")

cout << Menu;

else

cout << "Invalid number. Try again.\n"; // Если введено недопустимое значение

cout << "Your choice: ";

cin >> menu\_n; // Повторный ввод значения после выполнения операции

}

return 0;

}