**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 | Шавлохова А. А. |  |
| 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.   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.

Приложение А

рабочий код

#include <iostream>

#include <ctime>

#include <fstream>

#include <string>

#include < stdio.h >

using namespace std;

struct Profile

{

string full\_name; // Фамилия Имя Отчество

char sex; // Пол: M/W

int group; // Номер группы

int id; // Номер студента в списке

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

string date;

float average;

};

void creatingRecord() {

Profile student;

cout << "ФИО: ";

cin.ignore();

getline(cin, student.full\_name);

cout << "Группа: ";

cin >> student.group;

cout << "Номер в списке группы: ";

cin >> student.id;

cout << "Пол (Ж/М): ";

cin >> student.sex;

for (int i = 0; i < 8; i++) cin >> student.grades[i];

int temp = 0;

for (int i = 0; i < 8; i++) // Так оценка 2 не может стоять в ведомости, идёт проверка

{

if (student.grades[i] == 2)

temp++; // Найдена оценка 2, увеличиваем счётчик

}

if (temp == 0) // Если двоек не обнаружено

{

ofstream database; // Создаем поток записи в файл

// Открываем файл student с флагом записи в конец файла с сохранением содержимого файла

database.open("students.txt", ios::app);

if (!database.is\_open()) // Если файл открыть не удалось

cout << '\n' << "Ошибка сохранения!"; // Выдаём соответсвующее сообщение

else // Если файл успешно был открыт поочереди записываем всю информацию в файл.

{

database << student.full\_name << '\n';

database << student.group << '\n' << student.id << '\n' << student.sex << '\n';

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

database << student.grades[i] << " ";

database << '\n';

database.close();

cout << '\n' << "Профиль сохранен в datebase.\n";

}

}

else // Если у студента есть неуд, информация о нем, не может быть сохранена, т.к. он должен быть отчислен

cout << '\n' << "Этот студент должен быть отчислен. Профиль не сохранен в datebase.";

}

int countStudents()

{

ifstream database("students.txt");

if (database.is\_open())

{

int temp = 0;

string data;

while (!database.eof())

{

getline(database, data);

temp++;

}

database.close();

int n;

n = temp / 5;

return n;

}

else return 0;

}

void showData()

{

ifstream database("students.txt"); // Создаем поток считывания информации из файла и открываем файл student

if (!database.is\_open()) // Если файл открыть не удалось

cout << "Ошибка открытия файла";

else

{

int temp;

temp = countStudents();

if (temp == 0)

cout << "Файл пуст";

else

{

string data;

int t = 0;

while (!database.eof())

{

if (t == 0) {

getline(database, data); // Считываем очередную строчку

cout << "ФИО: " << data << '\n'; // Выводим строку на экран

t += 1;

}

else if (t == 1) {

getline(database, data); // Считываем очередную строчку

cout << "Группа: " << data << '\n'; // Выводим строку на экран

t += 1;

}

else if (t == 2) {

getline(database, data); // Считываем очередную строчку

cout << "Номер с группе: " << data << '\n'; // Выводим строку на экран

t += 1;

}

else if (t == 3) {

getline(database, data); // Считываем очередную строчку

cout << "Пол (Ж/М): " << data << '\n'; // Выводим строку на экран

t += 1;

}

else {

getline(database, data);

cout << "Оценки: " << data << '\n'; // Выводим строку на экран

t = 0;

}

}

}

database.close();

}

}

void showGroup()

{

int n; // Переменная, которая будет содержать информацию о введенном пользователем номере группы

cout << "Введите номер группы:" << endl;

cin >> n;

char litter;

ifstream database("students.txt");

if (!database.is\_open())

cout << "Ошибка открытия файла!";

else

{

int size;

size = countStudents();

if (size == 0)

cout << "Datebase пуст." << endl;

else

{

Profile\* student = new Profile[size];

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

{

getline(database, student[i].full\_name);

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

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

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

database.get(litter);

database.get(litter);

}

database.close(); // Закрываем файл

int temp = 0;

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

{

if (student[i].group == n) // Фильтруем данные по необходимому признаку

{

cout << "Номер в списке группы: ";

cout << student[i].id << '\n';

cout << "ФИО: ";

cout << student[i].full\_name << '\n';

cout << "Пол (Ж/М): ";

cout << student[i].sex << '\n';

cout << "Оценки за семестр: ";

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 << "Записи не найдены" << endl;

delete[] student; // Освобождаем динамическую память

}

}

}

void top() {

char litter;

ifstream database("students.txt");

if (!database.is\_open())

cout << "Ошибка открытия файла!";

else

{

int size, count;

int k = 0;

size = countStudents();

if (size == 0)

cout << "Datebase пуст." << endl;

else

{

Profile\* student = new Profile[size];

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

{

getline(database, student[i].full\_name);

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

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

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

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

}

database.get(litter);

database.get(litter);

student[i].average = k / 8;

k = 0;

}

database.close(); // Закрываем файл

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

{

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

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

if (student[j].average < student[j + 1].average)

swap(student[j], student[j + 1]);

}

}

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

{

count = i + 1;

cout << count << '\n';

cout << student[i].full\_name << '\n';

}

delete[] student; // Освобождаем динамическую память

}

}

}

void morw() {

char litter;

ifstream database("students.txt");

if (!database.is\_open())

cout << "Ошибка открытия файла!";

else

{

int size, W = 0, M = 0;

size = countStudents();

if (size == 0)

cout << "Datebase пуст." << endl;

else

{

Profile\* student = new Profile[size];

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

{

getline(database, student[i].full\_name);

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

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

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

}

database.get(litter);

database.get(litter);

if (student[i].sex == 'W') {

W += 1;

}

else {

M += 1;

}

}

database.close(); // Закрываем файл

cout << "Сколько WOMEN: " << W << '\n';

cout << "Сколько MEN: " << M << '\n';

delete[] student; // Освобождаем динамическую память

}

}

}

void showIdGroup()

{

int n; // Переменная, которая будет содержать информацию о введенном пользователем номере в списке группы(id)

char litter;

cout << "Введите номер в списке группы:" << endl;

cin >> n;

ifstream database("students.txt");

if (!database.is\_open())

cout << "Ошибка открытия файла!";

else

{

int size;

size = countStudents();

if (size == 0)

cout << "Datebase пуст." << endl;

else

{

Profile\* student = new Profile[size];

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

{

getline(database, student[i].full\_name);

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

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

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

database.get(litter);

database.get(litter);

}

database.close(); // Закрываем файл

int temp = 0;

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

{

if (student[i].id == n) // Фильтруем данные по необходимому признаку

{

cout << "Номер группы: ";

cout << student[i].group << '\n';

cout << "ФИО: ";

cout << student[i].full\_name << '\n';

cout << "Пол (Ж/М): ";

cout << student[i].sex << '\n';

cout << "Оценки за семестр: ";

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 << "Записи не найдены" << endl;

delete[] student; // Освобождаем динамическую память

}

}

}

void grades() {

char litter;

ifstream database("students.txt");

if (!database.is\_open())

cout << "Ошибка открытия файла!";

else

{

int size;

size = countStudents();

if (size == 0)

cout << "Datebase пуст." << endl;

else

{

Profile\* student = new Profile[size];

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

{

getline(database, student[i].full\_name);

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

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

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

}

database.get(litter);

database.get(litter);

}

database.close(); // Закрываем файл

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

{

int three = 0, four = 0, five = 0;

int\* three1 = &three, \* four1 = &four, \* five1 = &five;

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

if (student[i].grades[j] == 3) {

three = 1;

break;

}

else if (student[i].grades[j] == 4)

four = 1;

else if (student[i].grades[j] == 5)

five += 1;

}

if (\*three1 == 1) {

cout << "Студенты без стипендии: \n";

cout << "Номер в списке группы: ";

cout << student[i].id << '\n';

cout << "ФИО: ";

cout << student[i].full\_name << '\n';

cout << "Пол (Ж/М): ";

cout << student[i].sex << '\n';

cout << "Оценки за семестр: ";

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

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

cout << "\n\n";

continue;

}

else if (\*four1 == 1) {

cout << "Студенты, обучающиеся на хорошо/отлично: \n";

cout << "Номер в списке группы: ";

cout << student[i].id << '\n';

cout << "ФИО: ";

cout << student[i].full\_name << '\n';

cout << "Пол (Ж/М): ";

cout << student[i].sex << '\n';

cout << "Оценки за семестр: ";

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

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

}

cout << "\n\n";

continue;

}

else if (\*five1 == 8) {

cout << "Студенты, обучающиеся на отлично: \n";

cout << "Номер в списке группы: ";

cout << student[i].id << '\n';

cout << "ФИО: ";

cout << student[i].full\_name << '\n';

cout << "Пол (Ж/М): ";

cout << student[i].sex << '\n';

cout << "Оценки за семестр: ";

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

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

}

cout << "\n\n";

continue;

}

}

delete[] student; // Освобождаем динамическую память

}

}

}

int main() {

int r;

setlocale(0, "");

while (true) {

cout << " Меню\n" <<

"1 - Создать новую запись о студенте\n" <<

"2 - Внести изменения в уже имеющуюся запись\n" <<

"3 - Вывести все данные о студентах\n" <<

"4 - Вывести информацию обо всех студентах определенной(введенный пользователем) группы\n" <<

"5 - Вывести топ самых успешных студентов с наивысшим по рейтингу средним баллом за прошедшую сессию\n" <<

"6 - Вывести количество студентов мужского и женского пола\n" <<

"7 - Вывести данные о студентах, которые не получают стипендию; учатся только на «хорошо» и «отлично»; учатся только на «отлично»\n" <<

"8 - Вывести данные о студентах, имеющих определенный номер(введенный пользователем) в списке \n" <<

"0 - Выйти из программы\n";

cout << "Ваш выбор: ";

cin >> r;

cin.sync();

if (r == 0) break;

switch (r) {

case 1:

system("cls");

cout << "Создание новой записи о студенте:\n";

creatingRecord();

system("pause");

system("cls");

break;

case 2:

system("cls");

cout << "Внесение изменений в уже имеющуюся запись:\n";

creatingRecord();

system("pause");

system("cls");

break;

case 3:

system("cls");

cout << "Вывод всех данных о студентах:\n";

showData();

system("pause");

system("cls");

break;

case 4:

system("cls");

cout << "Вывод информации обо всех студентах группы N:\n";

showGroup();

system("pause");

system("cls");

break;

case 5:

system("cls");

cout << "Вывод топа самых успешных студентов с наивысшим по рейтингу средним баллом за прошедшую сессию:\n";

top();

system("pause");

system("cls");

break;

case 6:

system("cls");

cout << "Вывод количества студентов мужского и женского пола:\n";

morw();

system("pause");

system("cls");

break;

case 7:

system("cls");

cout << "Вывод данных о студентах, которые не получают стипендию; учатся только на «хорошо» и «отлично»; учатся только на «отлично»:\n";

grades();

system("pause");

system("cls");

break;

case 8:

system("cls");

cout << "Вывод данных о студентах, имеющих номер в списке – k:\n";

showIdGroup();

system("pause");

system("cls");

break;

}

}

return 0;

}

Условный пример содержания пункта «Выполеннеие работы»

|  |  |
| --- | --- |
| Ввод пользователем и обработка данных | Работа алгоритма и вывод на экран |
| Меню | |
| При запуске программы перед пользователем появляется окно с меню, где он может выбрать тип будущей вводимой последовательности | Меню: |

Продолжение Таблицы

|  |  |
| --- | --- |
| Создание новой записи | |
| При вводе пользователем корректного значения пункта меню и выбора создания новой записи, пользователь может добавить данные о студенте, но только на английском языке. | Как только все данные заданы, пользователь может еще раз просмотреть все данные о студенте и в случае, если видит ошибку, сразу же ее исправить без дополнительного вызова меню: |

Продолжение Таблицы

|  |  |
| --- | --- |
| Информация о студентах из группы | |
| Если пользователь хочет классифицировать студентов по полу, по группам, по номеру в списке, то он может вывести список о всех студентах | Пример, когда пользователь выбрал список данных по конкретной группе, ввел номер группы, а студентов в этой группе пока что не имеется:    Пример, когда пользователь выбрал студентов по их номеру группы: |

Продолжение Таблицы

|  |  |
| --- | --- |
| Рейтинг студентов | |
| Пользователь в главном меню может выбрать такую опцию как рейтинг студентов, чтобы наглядно увидеть список студентов. | Выводится номер в топе, ФИО. Рейтинг складывается из всех оценок за сессию (диф зачеты экзамены): |

Продолжение Таблицы

|  |  |
| --- | --- |
| Стипендия | |
| Стипендию пользователь может узнать в двух случаях: он может выбрать непосредственно пункт в меню, где ему выйдет список студентов, которые будут получать стипендию. | Пример вывода данных о студентах: |

Продолжение Таблицы

|  |  |
| --- | --- |
| Количество учеников женского и мужского пола | |
| Пользователь может просмотреть количество учеников женского и мужского пола: | Как только все данные заданы, пользователь может просмотреть количество учеников женского и мужского пола: |

**Вывод**

Я изучила структуры; получила практические навыки работы со структурами; определила преимущества и недостатки использования структур.