**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 гр. 3372 |  | Кипень В. А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2024

**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.

**Выполнение 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 пункт, который требуется выполнить | Меню:    Проверка на ввод символов, которые не входят в диапазон выбора: |
| После меню с навигацией, выводится информация о студентах, которые нашлись в файле на момент запуска программы, и предложение добавить нового студента |  |
| Создание записи | |
| Пункт 1, создание записи о студенте, путём ввода данных с клавиатуры | Введённые данные:    Сохранённые данные: |
| Изменение записи | |
| Пункт 2, изменение поля в записи, путём поиска студента по id, с последующим вводом поля, которое необходимо изменить, и новым значением этого поля |  |
| Вывод данных о всех студентах | |
| Пункт 3, вывод информации о студентах на экран |  |
| Вывод данных о студентах из группы | |
| Пункт 4, вывод информации о студентах определённой группы (вводится пользователем) на экран |  |
| Вывод топа студентов по среднему баллу | |
| Пункт 5, вывод информации о студентах, отсортированных по убыванию среднего балла на экран |  |
| Вывод количества студентов мужского и женского пола | |
| Пункт 6, вывод количества студентов мужского и женского пола на экран |  |
| Вывод студентов, в зависимости от их успеваемости | |
| Пункт 7, вывод информации о студентах, которые не получают стипендию/учатся на 4/отличников на экран |  |
| Вывод данных о студентах, имеющий определённый номер в списке группы | |
| Пункт 8, вывод информации о студентах, имеющих номер в списке, который введет пользователь, на экран |  |

**Выводы.**

Реализовано:  
 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.

Приложение А

рабочий код

#include <Windows.h>

#include <iostream>

#include <fstream>

#include <ctime>

#include <time.h>

#include <cmath>

#include <cstdlib>

using namespace std;

#define EXAMS\_COUNT 3

#define TESTS\_COUNT 5

#define MAX\_NAME\_SIZE 10

#define TIMESTAMP\_SIZE 20

#define STUDENT\_SIZE 7

#define MAX\_DATA\_SIZE max(MAX\_NAME\_SIZE, TIMESTAMP\_SIZE) // максимальная длина строки в файле students.txt

void getCurrentTime(char\* formatTime) {

struct tm \* timeinfo;

time\_t tstamp;

time(&tstamp);

timeinfo = localtime(&tstamp);

strftime(formatTime, TIMESTAMP\_SIZE, "%Y-%m-%d.%X", timeinfo);

}

int getRandomValueFromRange(int limit) {

return rand() % (limit + 1);

}

struct Student {

unsigned int id;

char name[MAX\_NAME\_SIZE];

char gender;

unsigned int groupNumber;

short unsigned int groupListNumber;

short unsigned int grades[EXAMS\_COUNT + TESTS\_COUNT];

float avgScore;

char timestamp[TIMESTAMP\_SIZE];

void calculateAverageScore() {

this->avgScore = 0.0f;

for (int i = 0; i < EXAMS\_COUNT + TESTS\_COUNT; i++) {

avgScore += this->grades[i];

}

this->avgScore = round((this->avgScore / (EXAMS\_COUNT + TESTS\_COUNT)) \* 100.f) / 100.f;

}

};

int countStudents() {

ifstream database("students.txt");

if (database.is\_open()) {

int temp = 0;

char data[MAX\_DATA\_SIZE];

while (database.getline(data, MAX\_DATA\_SIZE))

{

temp++;

}

database.close();

int n;

n = temp / STUDENT\_SIZE;

return n;

}

return 0;

}

void showStudent(Student &student) {

cout << "\nStudent id: " << student.id << "\n"

<< " name: " << student.name << "\n"

<< " gender: " << student.gender << "\n"

<< " groupNumber: " << student.groupNumber << "\n"

<< " groupListNumber: " << student.groupListNumber << "\n"

<< " grades: ";

for (int i = 0; i < (EXAMS\_COUNT + TESTS\_COUNT); i++) cout << student.grades[i] << " ";

cout << "avg: " << student.avgScore << "\n"

<< " timestamp: " << student.timestamp << "\n";

}

void bubbleSort(Student arr[], const int N) {

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

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

struct Student swap = arr[j];

if (arr[j].avgScore < arr[j + 1].avgScore) {

arr[j] = arr[j + 1];

arr[j + 1] = swap;

}

}

}

}

void showSortedStudents(Student students[], const int studentsCount) {

bubbleSort(students, studentsCount);

for (int i = 0; i < studentsCount; i++) showStudent(students[i]);

}

void readData(Student students[], const int studentsCount) {

char data[MAX\_DATA\_SIZE];

ifstream db("students.txt");

int idx = 0;

short unsigned int fieldType = 0;

while(db.getline(data, MAX\_DATA\_SIZE)) {

switch (fieldType)

{

case 0: {

students[idx].id = atoi(data);

break;

}

case 1: {

for (int i = 0; i < MAX\_NAME\_SIZE; i++) {

students[idx].name[i] = data[i];

}

break;

}

case 2: {

students[idx].gender = data[0];

break;

}

case 3: {

students[idx].groupNumber = atoi(data);

break;

}

case 4: {

students[idx].groupListNumber = atoi(data);

break;

}

case 5: {

for (int i = 0; i < (EXAMS\_COUNT + TESTS\_COUNT); i++) {

students[idx].grades[i] = static\_cast<int>(data[i] - '0');

students[idx].calculateAverageScore();

}

break;

}

case 6: {

for (int i = 0; i < TIMESTAMP\_SIZE; i++) {

students[idx].timestamp[i] = data[i];

}

break;

}

}

fieldType++;

if (fieldType == STUDENT\_SIZE) {

showStudent(students[idx]);

fieldType = 0;

idx++;

}

}

db.close();

}

void clearStudentsFile() {

ofstream db("students.txt");

if (!db.is\_open()) {

cerr << "Error opening students file for writing" << endl;

return;

}

db << "";

db.close();

}

void writeData(Student& student) {

ofstream database("students.txt", ios::app); // начинаем запись с последней строки файла

if (!database.is\_open()) {

cout << '\n' << "Saving error!";

} else {

char charGrades[EXAMS\_COUNT + TESTS\_COUNT];

char buffer[sizeof(unsigned short int)];

for (int i = 0; i < EXAMS\_COUNT + TESTS\_COUNT; i++) charGrades[i] = \*itoa(student.grades[i], buffer, 10);

student.calculateAverageScore();

database

<< student.id << '\n'

<< student.name << '\n'

<< student.gender << '\n'

<< student.groupNumber << '\n'

<< student.groupListNumber << '\n';

for (int i = 0; i < EXAMS\_COUNT + TESTS\_COUNT; i++) database << charGrades[i];

database << '\n'

<< student.timestamp

<< '\n';

database.close();

}

}

Student createStudent() {

struct Student student;

cout << "\nEnter a data:\n";

cout << "name (max " << MAX\_NAME\_SIZE << " symbols): ";

// cin >> student.name;

cin.get();

cin.getline(student.name, MAX\_NAME\_SIZE);

cin.clear(); // Clearing the input stream from possible errors

cin.sync();

cout << "gender (char): ";

cin >> student.gender;

cout << "groupNumber: ";

cin >> student.groupNumber;

cout << "groupListNumber: ";

cin >> student.groupListNumber;

cout << "grades (" << (EXAMS\_COUNT + TESTS\_COUNT) <<" numbers): ";

for (int i = 0; i < (EXAMS\_COUNT + TESTS\_COUNT); i++) cin >> student.grades[i];

student.id = getRandomValueFromRange(1000);

char currentTime[TIMESTAMP\_SIZE];

getCurrentTime(currentTime);

for (int i = 0; i < TIMESTAMP\_SIZE; i++) student.timestamp[i] = currentTime[i];

writeData(student);

return student;

}

int getIndexOfStudent(unsigned int id, Student \*students, const int N) {

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

if (students[i].id == id) return i;

}

return -1;

}

void updateStudent(struct Student\* student, const short unsigned int fieldType) {

switch (fieldType)

{

case 1: {

cin.get();

cin.getline(student->name, MAX\_NAME\_SIZE);

cin.clear(); // Clearing the input stream from possible errors

cin.sync();

break;

}

case 2: {

cin >> student->gender;

break;

}

case 3: {

cin >> student->groupNumber;

break;

}

case 4: {

cin >> student->groupListNumber;

break;

}

case 5: {

for (int i = 0; i < (EXAMS\_COUNT + TESTS\_COUNT); i++) cin >> student->grades[i];

student->calculateAverageScore();

break;

}

}

}

int main() {

// If when entering RU characters there are hieroglyphs, then write the command in the console: chcp 1251

SetConsoleCP(1251); // Setting the input encoding

SetConsoleOutputCP(1251); // Setting the output encoding

setlocale(LC\_ALL, "Russian");

srand(time(0));

cout << "Navigation:" << "\n"

<< "1) Create a new student record" << "\n"

<< "2) Making changes to an existing record" << "\n"

<< "3) Displaying all student data" << "\n"

<< "4) Displays info about all students in group <N>" << "\n"

<< "5) Finding top students with the highest rated average score for the last session" << "\n"

<< "6) Displaying the number of male and female students" << "\n"

<< "7) Displaying students who don't receive a scholarship; who have only 4 or 5 grades; who have only 5 grades" << "\n"

<< "8) Displaying data about students with list number <k>" << "\n";

int fileStudentsCount = countStudents();

short unsigned int newStudentsCount = 0; // те, кого ещё нет в файле, но хотим добавить

cout << "\nStudents count in file: "<< fileStudentsCount << "\n";

cout << "Enter number of new students: ";

cin >> newStudentsCount;

if (!cin.good()) {

cout << "\nYou entered an incorrect value\n";

}

int aviablePlaces = newStudentsCount; // тк работаем без векторов, то заранее указываем сколько новых

const int studentsCount = fileStudentsCount + newStudentsCount;

struct Student students[studentsCount];

readData(students, studentsCount);

cout << "\n^^^ Students data from file ^^^\n";

while(true) {

cin.clear(); // Clearing the input stream from possible errors

cin.sync();

short int workPoint;

cout << "Select point of work (number 1 to 8): ";

cin >> workPoint;

switch (workPoint)

{

case 1: {

if (newStudentsCount < 1 || aviablePlaces < 1) {

cout << "\nNo places for new students(\n";

break;

}

students[studentsCount - aviablePlaces] = createStudent();

showStudent(students[studentsCount - aviablePlaces]);

aviablePlaces--;

break;

}

case 2: {

unsigned int id;

cout << "\nEnter student ID: ";

cin >> id;

int idx = getIndexOfStudent(id, students, studentsCount);

if (idx != -1) {

showStudent(students[idx]);

short unsigned int fieldType;

cout << "\n" << "Enter field type (1-name;2-gender;3-groupNumber;4-groupListNumber;5-grades): ";

cin >> fieldType;

if (!cin.good()) {

cout << "\nYou entered an incorrect value";

break;

}

updateStudent(&students[idx], fieldType);

cout << "\nModified student\n";

showStudent(students[idx]);

clearStudentsFile();

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

writeData(students[i]);

}

} else {

cout << "\nNo such students found";

}

break;

}

case 3: {

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

showStudent(students[i]);

}

break;

}

case 4: {

int groupNumber;

cout << "\n" << "Enter group number: ";

cin >> groupNumber;

if (!cin.good()) {

cout << "\n" << "You entered an incorrect value";

break;

}

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

if (students[i].groupNumber == groupNumber) showStudent(students[i]);

}

break;

}

case 5: {

cout << "\nStudents TOP\n";

showSortedStudents(students, studentsCount);

break;

}

case 6: {

int maleCount = 0;

int femaleCount = 0;

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

if (students[i].gender == 'M') maleCount++;

if (students[i].gender == 'W') femaleCount++;

}

cout << "\nNumber of male students: " << maleCount;

cout << "\nNumber of female students: " << femaleCount;

break;

}

case 7:{

cout << "\nStudents who do not receive a scholarship:\n";

Sleep(800);

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

if (students[i].avgScore < 4) showStudent(students[i]), Sleep(500);

}

cout << "\nStudents who study at 4:\n";

Sleep(800);

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

if (students[i].avgScore >= 4 && students[i].avgScore < 5) showStudent(students[i]), Sleep(500);

}

cout << "\nSuper best students krasavchiki:\n";

Sleep(800);

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

if (students[i].avgScore == 5) showStudent(students[i]), Sleep(500);

}

break;

}

case 8: {

int groupListNumber;

cout << "\n" << "Enter a group list number: ";

cin >> groupListNumber;

if (!cin.good()) {

cout << "\n" << "You entered an incorrect value";

break;

}

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

if (students[i].groupListNumber == groupListNumber) showStudent(students[i]);

}

break;

}

default: {

cout << "\n" << "You did not enter a number in the range from 1 to 8";

break;

}

}

cin.clear(); // Clearing the input stream from possible errors

cin.sync();

char stopFlag;

cout << "\n" << "Continue the program? (Y/N) ";

cin >> stopFlag;

if (stopFlag != 'Y' && stopFlag != 'y') break;

}

return 0;

}