**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №1**

**по дисциплине «Программирование»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 2373 |  | Чишко В. О. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2023

**Цель работы.**

Изучение структур в c++, массивов структур, создание прототипа баз данный в c++ при помощи структур, добавление функционала.

**Основные теоретические положения.**

В языке c++, структура — композитный тип данных, инкапсулирующий без сокрытия набор значений различных типов. Порядок размещения значений в памяти задаётся при определении типа и сохраняется на протяжении времени жизни объектов, что даёт возможность косвенного доступа. Пример объявления структуры: struct str\_name

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.

Объединения также хранят набор элементов, но в отличие от структуры все элементы объединения имеют нулевое смещение. А это значит, что разные элементы занимают в памяти один и тот же участок.

Для определения объединений применяется ключевое слово union и следующий формальный синтаксис:

union имя\_объединения

{

    определения\_элементов

};

**Постановка задачи**

Необходимо создать программу, позволяющую:

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.

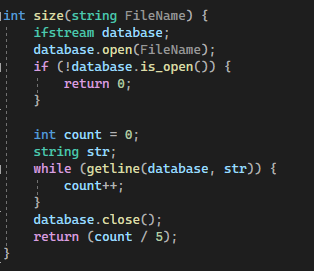
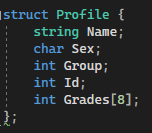
7.   Вывод данных о студентах, которые не получают стипендию; учатся только на «хорошо» и «отлично»; учатся только на «отлично»;

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

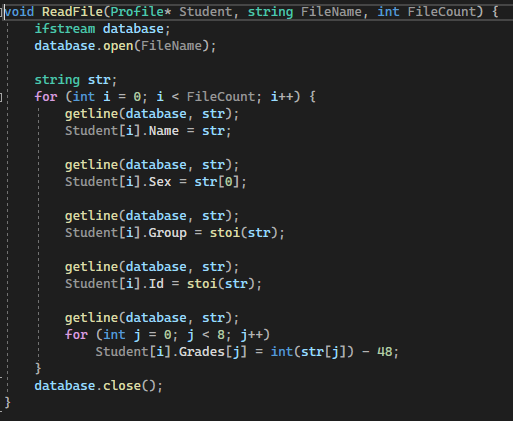
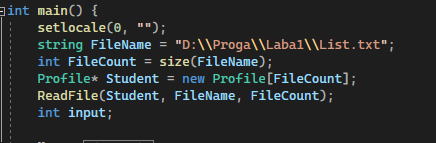
**Выполнение работы.**

Код программы представлен в приложении А.

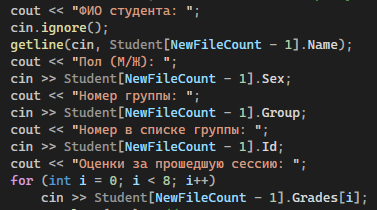
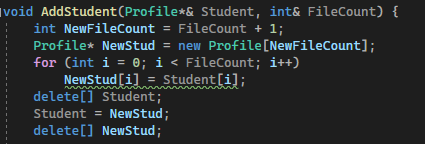
**Блок описания кода и использованных алгоритмов:**

Создаём динамический массив структур, размер которого узнаём считывая количество строк в изначальном файле List.txt.

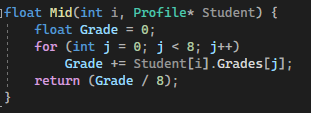
Далее просто считываем файл и полученные данные добавляем в массив структур.

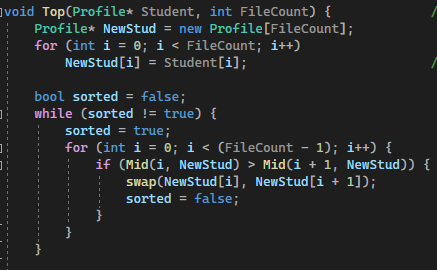


1. Сначала массив расширяется на единицу, а затем в появившуюся ячейку записываются данные о новом студенте.



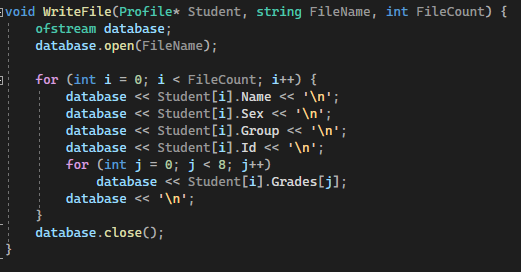
1. Пользователь вводит номер студента, данные которого хочет изменить, а затем меняет их.
2. Циклом for по очереди выводим всех студентов в списке.
3. Пользователь вводит номер группы и перебором выводятся студенты с соответствующим номером группы.
4. Сначала создаём функцию для высчитывания среднего значения оценок обучающегося.

Далее создаём копию исходного массива и сортируем его алгоритмом Bubble sort.



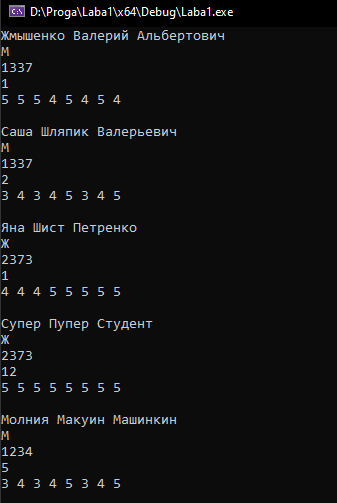
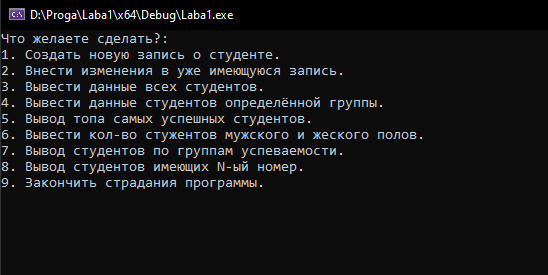
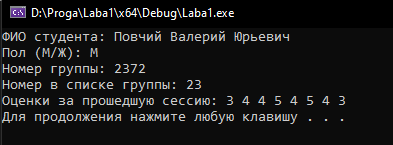
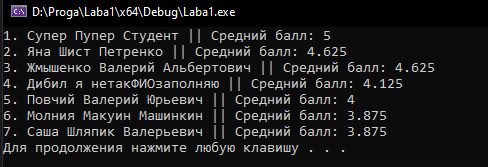
1. Перебором проходимся по всему массиву и увеличиваем счётчики обеих полов.
2. Делаем проверку оценок студентов. Если находим соответствие, то выводим в консоль.
3. Пользователь вводит номер и перебором выводятся студенты с соответствующим номеров в группе.

По завершению работы программы записываем получившиеся данные обратно в файл.



**Выводы.**

Мы научились использовать структуры c++, написали прототип баз данный c++, научились использовать функционал структур.

**Блок скриншотов работы программы**

Приложение А

рабочий код

#include <iostream>

#include <string>

#include <fstream>

#include <Windows.h>

using namespace std;

struct Profile {

string Name;

char Sex;

int Group;

int Id;

int Grades[8];

};

void ReadFile(Profile\* Student, string FileName, int FileCount) { //Чтение файла

ifstream database;

database.open(FileName);

string str;

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

getline(database, str);

Student[i].Name = str;

getline(database, str);

Student[i].Sex = str[0];

getline(database, str);

Student[i].Group = stoi(str);

getline(database, str);

Student[i].Id = stoi(str);

getline(database, str);

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

Student[i].Grades[j] = int(str[j]) - 48;

}

database.close();

}

void WriteFile(Profile\* Student, string FileName, int FileCount) { // Запись в файл

ofstream database;

database.open(FileName);

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

database << Student[i].Name << '\n';

database << Student[i].Sex << '\n';

database << Student[i].Group << '\n';

database << Student[i].Id << '\n';

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

database << Student[i].Grades[j];

database << '\n';

}

database.close();

}

void ShowAll(Profile\* Student, int FileCount) { // Вывод всех студентов

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

cout << Student[i].Name << "\n";

cout << Student[i].Sex << "\n";

cout << Student[i].Group << "\n";

cout << Student[i].Id << "\n";

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

cout << Student[i].Grades[j] << " ";

cout << "\n\n";

}

}

void AddStudent(Profile\*& Student, int& FileCount) { // Добавление студента в базу данных

int NewFileCount = FileCount + 1;

Profile\* NewStud = new Profile[NewFileCount];

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

NewStud[i] = Student[i];

delete[] Student;

Student = NewStud; // Расширяем массив на (+1) и тут же заполняем новую ячейку

SetConsoleCP(1251); // <--- Меняем кодировку консоли, чтоб она воспринимала кириллицу

cout << "ФИО студента: ";

cin.ignore();

getline(cin, Student[NewFileCount - 1].Name);

cout << "Пол (М/Ж): ";

cin >> Student[NewFileCount - 1].Sex;

cout << "Номер группы: ";

cin >> Student[NewFileCount - 1].Group;

cout << "Номер в списке группы: ";

cin >> Student[NewFileCount - 1].Id;

cout << "Оценки за прошедшую сессию: ";

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

cin >> Student[NewFileCount - 1].Grades[i];

SetConsoleCP(866); // <--- Меняем кодировку консоли обратно, чтоб не возникало непредвиденных проблем

FileCount = NewFileCount;

}

void ChangeStudent(Profile\*& Student, int FileCount) { // Изменение записи определённого студента

int FindGroup, FindId;

cout << "Введите номер группы: ";

cin >> FindGroup;

cout << "Ведите номер студента в группе: "; // Поиск производится по номеру группы и номеру в группе. Можно было бы сделать поиск по номеру в файле,

cin >> FindId; // но так мне кажется более логично + не хотел вводить лишнюю переменную

int flag = -1;

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

if ((FindGroup == Student[i].Group) && (FindId == Student[i].Id)) {

flag = i;

break;

}

}

SetConsoleCP(1251);

if (flag == -1) {

cout << "\nТакого студента не существует.\n\n";

}

else {

int input;

CMenu: {

system("cls");

cout << Student[flag].Name << "\n";

cout << Student[flag].Sex << "\n";

cout << Student[flag].Group << "\n";

cout << Student[flag].Id << "\n";

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

cout << Student[flag].Grades[i] << " ";

cout << "\n\nЧто желаете изменить?\n" <<

"1. ФИО\n" <<

"2. Пол (М/Ж)\n" <<

"3. Номер группы\n" <<

"4. Номер в списке группы\n" <<

"5. Оценки за прошедшую сессию\n" <<

"6. Закончить редактирование\n";

}

cin >> input;

switch (input) {

case 1: {

cout << "ФИО студента: ";

cin.ignore();

getline(cin, Student[flag].Name);

goto CMenu;

}

case 2: {

cout << "Пол (М/Ж): ";

cin >> Student[flag].Sex;

goto CMenu;

}

case 3: {

cout << "Номер группы: ";

cin >> Student[flag].Group;

goto CMenu;

}

case 4: {

cout << "Номер в списке группы: ";

cin >> Student[flag].Id;

goto CMenu;

}

case 5: {

cout << "Оценки за прошедшую сессию: ";

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

cin >> Student[flag].Grades[i];

goto CMenu;

}

case 6: {

break;

}

default: {

cout << "\nНеверный ввод.\n\n";

cin.clear();

while (cin.get() != '\n');

system("pause");

goto CMenu;

}

}

}

SetConsoleCP(866);

}

void FindGroup(Profile\* Student, int FileCount) { // Поиск по группе

int N;

cout << "Введите номер группы: ";

cin >> N;

system("cls");

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

if (Student[i].Group == N) {

cout << Student[i].Name << "\n";

cout << Student[i].Sex << "\n";

cout << Student[i].Group << "\n";

cout << Student[i].Id << "\n";

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

cout << Student[i].Grades[j] << " ";

cout << "\n\n";

}

}

}

void SexCount(Profile\* Student, int FileCount) { // Счётчик мальчиков и девочек

int MaleCount = 0, FemaleCount = 0;

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

if (Student[i].Sex == 'М')

MaleCount++;

else

FemaleCount++;

}

cout << "Кол-во студентов мужского пола - " << MaleCount <<

"\nКол-во студентов женского пола - " << FemaleCount << "\n";

}

void FindNumber(Profile\* Student, int FileCount) { // Поиск по номеру в группе

int k;

cout << "Введите номер студентов в группе: ";

cin >> k;

system("cls");

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

if (Student[i].Id == k) {

cout << Student[i].Name << "\n";

cout << Student[i].Sex << "\n";

cout << Student[i].Group << "\n";

cout << Student[i].Id << "\n";

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

cout << Student[i].Grades[j] << " ";

cout << "\n\n";

}

}

}

float Mid(int i, Profile\* Student) { // Поиск среднего значения оценок. Нужно для упрощения следующей функции(Top)

float Grade = 0;

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

Grade += Student[i].Grades[j];

return (Grade / 8);

}

void Top(Profile\* Student, int FileCount) { // Вывод топа по оценкам. (Просто сортируем список и выводим его)

Profile\* NewStud = new Profile[FileCount];

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

NewStud[i] = Student[i]; // Создаём копию оригинальной структуры, чтобы не менять оригинальную

bool sorted = false;

while (sorted != true) {

sorted = true;

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

if (Mid(i, NewStud) > Mid(i + 1, NewStud)) {

swap(NewStud[i], NewStud[i + 1]);

sorted = false;

}

}

}

for (int i = FileCount - 1; i >= 0; i--) {

cout << FileCount - i << ". " << NewStud[i].Name << " || Средний балл: " << Mid(i, NewStud) << "\n";

}

delete[] NewStud;

}

void GiveMoney(Profile\* Student, int FileCount) { // Вывод группы студентов по успеваемости

int input;

GMenu: {

system("cls");

cout << "Данные каких студентов вы хотите получить?\n" <<

"1. Студенты НЕ получающие степендию.\n" <<

"2. Студенты 'Хорошиты'.\n" <<

"3. Студенты 'Отличники'.\n" <<

"4. Выход.\n";

}

cin >> input;

switch (input) {

case 1: { // Троешники

system("cls");

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

bool flag = false;

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

if (Student[i].Grades[j] == 3)

flag = true;

if (flag == true) {

cout << Student[i].Name << "\n";

cout << Student[i].Sex << "\n";

cout << Student[i].Group << "\n";

cout << Student[i].Id << "\n";

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

cout << Student[i].Grades[j] << " ";

cout << "\n\n";

}

}

system("pause");

goto GMenu;

}

case 2: { // Хорошисты

system("cls");

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

bool flag = false;

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

if (Student[i].Grades[j] == 3)

flag = true;

if (flag == false) {

cout << Student[i].Name << "\n";

cout << Student[i].Sex << "\n";

cout << Student[i].Group << "\n";

cout << Student[i].Id << "\n";

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

cout << Student[i].Grades[j] << " ";

cout << "\n\n";

}

}

system("pause");

goto GMenu;

}

case 3: { // Роботы

system("cls");

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

bool flag = false;

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

if (Student[i].Grades[j] == 3 || Student[i].Grades[j] == 4)

flag = true;

if (flag == false) {

cout << Student[i].Name << "\n";

cout << Student[i].Sex << "\n";

cout << Student[i].Group << "\n";

cout << Student[i].Id << "\n";

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

cout << Student[i].Grades[j] << " ";

cout << "\n\n";

}

}

system("pause");

goto GMenu;

}

case 4: {

break;

}

default: {

cout << "\nНеверный ввод.\n\n";

cin.clear();

while (cin.get() != '\n');

system("pause");

goto GMenu;

}

}

}

int size(string FileName) { // Штука для считывания строк в файле. По итогу мы получим точное количество студентов, а это нужно для создания динам. массива

ifstream database;

database.open(FileName);

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

return 0;

}

int count = 0;

string str;

while (getline(database, str)) {

count++;

}

database.close();

return (count / 5);

}

int main() {

setlocale(0, "");

string FileName = "D:\\Proga\\Laba1\\List.txt";

int FileCount = size(FileName);

Profile\* Student = new Profile[FileCount];

ReadFile(Student, FileName, FileCount);

int input;

Menu: {

system("cls");

cout << "Что желаете сделать?:\n" <<

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

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

"3. Вывести данные всех студентов.\n" <<

"4. Вывести данные студентов определённой группы.\n" <<

"5. Вывод топа самых успешных студентов.\n" <<

"6. Вывести кол-во стужентов мужского и жеского полов.\n" <<

"7. Вывод студентов по группам успеваемости.\n" <<

"8. Вывод студентов имеющих N-ый номер.\n" <<

"9. Закончить страдания программы.\n";

}

cin >> input;

switch (input) {

case 1: { /\* Создание новой записи о студенте. \*/

system("cls");

AddStudent(Student, FileCount);

system("pause");

goto Menu;

}

case 2: { /\* Внесение изменений в уже имеющуюся запись. \*/

system("cls");

ChangeStudent(Student, FileCount);

system("pause");

goto Menu;

}

case 3: { /\* Вывод всех данных о студентах. \*/

system("cls");

ShowAll(Student, FileCount);

system("pause");

goto Menu;

}

case 4: { /\* Вывод информации обо всех студентах группы N.N – инициализируется пользователем. \*/

system("cls");

FindGroup(Student, FileCount);

system("pause");

goto Menu;

}

case 5: { /\* Вывод топа самых успешных студентов с наивысшим по рейтингу средним баллом за прошедшую сессию. \*/

system("cls");

Top(Student, FileCount);

system("pause");

goto Menu;

}

case 6: { /\* Вывод количества студентов мужского и женского пола. \*/

system("cls");

SexCount(Student, FileCount);

system("pause");

goto Menu;

}

case 7: { /\* Вывод данных о студентах, которые не получают стипендию; учатся только на «хорошо» и «отлично»; учатся только на «отлично»; \*/

system("cls");

GiveMoney(Student, FileCount);

system("pause");

goto Menu;

}

case 8: { /\* Вывод данных о студентах, имеющих номер в списке – k. \*/

system("cls");

FindNumber(Student, FileCount);

system("pause");

goto Menu;

}

case 9: { /\* Вывод данных в файл \*/

WriteFile(Student, FileName, FileCount);

cout << "\n\*Смэрть\*\n";

break;

}

default: {

cout << "\nНеверный ввод.\n\n";

cin.clear(); //<-- Эта штука нужна для чистки потока cin. В противном случае, при вводе кучи символов (Например: fgsfgsfg; или 12412413),

while (cin.get() != '\n'); //<-- произойдёт троллинг вселенского масштаба и прога либо уйдет в бесконечный цикл, либо забьёт данные введённым мусором

system("pause");

goto Menu;

}

}

}