МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

**КАФЕДРА ИИТ**

ОТЧЁТ

по лабораторной работе №13

**«Модульное программирование»**

Выполнил:

студент 1 курса

группы ПО-9

Харитонович Захар Сергеевич

Проверила:

Хацкевич М. В.

Брест 2022

**Цель работы:** Изучить принципы модульного программирования в Си; ознакомиться с основными возможностями межмодульного взаимодействия

**Порядок выполнения работы**

**Исходный код**

*main.cpp*

#include <iostream>

#include <string.h>

#include "database.h"

#include "file.h"

#include "index.h"

using namespace std;

int main() {

students \*studs = new students[1];

size\_t n = 0;

import\_file("file.txt", studs, n);

int mrot = 400;

cout << "Welcome to students database.\n";

short mode = 0;

while (mode != 6) {

cout << "Choose mode:\n";

cout << "1 - add a record. 2 - list records. 3 - sort by.\n";

cout << "4 - delete by. 5 - list records with small income. 6 - exit.\n> ";

cin >> mode;

switch (mode) {

case 1:

add\_record(studs, n, "file.txt");

break;

case 2:

list(studs, n);

break;

case 3:

sort\_by(studs, n);

rewrite\_file("file.txt", studs, n);

break;

case 4:

delete\_by(studs, n);

rewrite\_file("file.txt", studs, n);

break;

case 5:

list\_by\_income(studs, n, mrot);

case 6:

cout << "Exit\n";

break;

default:

cout << "Unknown.\n";

break;

}

}

delete[] studs;

return 0;

}

*database.h*

#ifndef TASK\_DATABASE\_H

#define TASK\_DATABASE\_H

#include <iostream>

#include <string.h>

using namespace std;

struct students {

char surname[256];

char name[256];

char patronymic[256];

enum groups {

po8, po9, ii21, ii22, unknown

} grp;

double avrMark;

int income;

short bit: 4;

union {

int integer;

double real;

};

};

void display\_record(students);

void add\_record(students \*&, size\_t &, const char \*);

void add\_to\_array(students \*&, size\_t &, students);

void list(students \*, size\_t);

void sort\_by(students \*, size\_t);

void delete\_by(students \*&, size\_t &);

void delete\_by\_id(students \*&, size\_t &, size\_t);

void list\_by\_income(students \*, size\_t, int);

#endif

*database.cpp*

#include "database.h"

#include "index.h"

extern char currentIndexFileName[32];

void add\_record(students \*&studs, size\_t &n, const char \*filename) {

cout << "Add a record. Fields:\nSurname. Name. Patronymic. Group. Average mark. Income per family member.\n";

char group[256];

students current;

cin >> current.surname >> current.name >> current.patronymic >> group >> current.avrMark >> current.income;

if (!strcmp(group, "po-8") || !strcmp(group, "PO-8")) {

current.grp = students::po8;

} else if (!strcmp(group, "po-9") || !strcmp(group, "PO-9")) {

current.grp = students::po9;

} else if (!strcmp(group, "ii-21") || !strcmp(group, "II-21")) {

current.grp = students::ii21;

} else if (!strcmp(group, "ii-22") || !strcmp(group, "II-22")) {

current.grp = students::ii22;

} else {

current.grp = students::unknown;

}

add\_to\_array(studs, n, current);

FILE \*f = fopen(filename, "ab");

fwrite(&current, sizeof(students), 1, f);

fclose(f);

add\_to\_index(n - 1, current);

}

void add\_to\_array(students \*&studs, size\_t &n, students toAdd) {

students \*temp = new students[n + 1];

for (size\_t i = 0; i < n; i++) {

temp[i] = studs[i];

}

delete[] studs;

studs = temp;

studs[n] = toAdd;

n++;

}

void display\_record(students cur) {

cout << cur.surname << " " << cur.name << " " << cur.patronymic;

switch (cur.grp) {

case students::po8:

cout << " PO-8 ";

break;

case students::po9:

cout << " PO-9 ";

break;

case students::ii21:

cout << " II-21 ";

break;

case students::ii22:

cout << " II-22 ";

break;

default:

cout << " Unknown group ";

break;

}

cout << cur.avrMark << " " << cur.income << endl;

}

void list(students \*studs, size\_t n) {

cout << "Students list. Fields:\nSurname. Name. Patronymic. Group. Average mark. Income per family member.\n";

int ind[n], values[n];

extern char currentIndexFileName[32];

if (!strcmp(currentIndexFileName, "")) strcpy(currentIndexFileName, "group\_index.txt");

read\_index\_file(currentIndexFileName, ind, values, n);

for (size\_t i = 0; i < n; i++) {

cout << i + 1 << ") ";

display\_record(studs[ind[i]]);

}

}

void sort\_by(students \*studs, size\_t n) {

cout << "Sort by:\n";

cout << "1 - group. 2 - income.\n> ";

short mode;

cin >> mode;

switch (mode) {

case 1:

strcpy(currentIndexFileName, "group\_index.txt");

break;

case 2:

strcpy(currentIndexFileName, "income\_index.txt");

break;

default:

cout << "Unknown.\n";

return;

}

int ind[n], values[n];

read\_index\_file(currentIndexFileName, ind, values, n);

for (size\_t i = 1; i < n; i++) {

for (size\_t j = i; j > 0 && values[j - 1] > values[j]; j--) {

swap(values[j], values[j - 1]);

swap(ind[j], ind[j - 1]);

}

}

rewrite\_index\_file(currentIndexFileName, ind, values, n);

}

void delete\_by\_id(students \*&studs, size\_t &n, size\_t id) {

n--;

int ind[n], groups[n], incomes[n];

for (size\_t j = id; j < n; j++) {

studs[j] = studs[j + 1];

}

for (size\_t i = 0; i < n; i++) {

ind[i] = i;

groups[i] = studs[i].grp;

incomes[i] = studs[i].income;

}

rewrite\_index\_file("group\_index.txt", ind, groups, n);

rewrite\_index\_file("income\_index.txt", ind, incomes, n);

}

void delete\_by(students \*&studs, size\_t &n) {

cout << "Delete by:\n";

cout << "1 - surname. 2 - group. 3 - mark. 4 - income.\n> ";

short mode;

cin >> mode;

switch (mode) {

case 1:

cout << "Surname: ";

char toDel[256];

cin >> toDel;

for (size\_t i = 0; i < n; i++) {

if (!strcmp(studs[i].surname, toDel)) {

delete\_by\_id(studs, n, i);

}

}

break;

case 2:

cout << "Group: ";

char group[256];

cin >> group;

short tempGrp;

if (!strcmp(group, "po-8") || !strcmp(group, "PO-8")) {

tempGrp = students::po8;

} else if (!strcmp(group, "po-9") || !strcmp(group, "PO-9")) {

tempGrp = students::po9;

} else if (!strcmp(group, "ii-21") || !strcmp(group, "II-21")) {

tempGrp = students::ii21;

} else if (!strcmp(group, "ii-22") || !strcmp(group, "II-22")) {

tempGrp = students::ii22;

} else {

tempGrp = students::unknown;

}

for (size\_t i = 0; i < n; i++) {

if (studs[i].grp == tempGrp) {

delete\_by\_id(studs, n, i);

}

}

break;

case 3:

cout << "Average mark: ";

double mark;

cin >> mark;

for (size\_t i = 0; i < n; i++) {

if (studs[i].avrMark == mark) {

delete\_by\_id(studs, n, i);

}

}

break;

case 4:

cout << "Income: ";

int income;

cin >> income;

for (size\_t i = 0; i < n; i++) {

if (studs[i].income == income) {

delete\_by\_id(studs, n, i);

}

}

break;

default:

cout << "Unknown.\n";

return;

break;

}

students \*temp = new students[n];

for (size\_t i = 0; i < n; i++) {

temp[i] = studs[i];

}

delete[] studs; studs = temp;

}

void list\_by\_income(students \*studs, size\_t n, int mrot) {

mrot \*= 2;

size\_t j = 1;

int ind[n], values[n];

read\_index\_file("income\_index.txt", ind, values, n);

for (size\_t i = 0; i < n; i++)

if (values[i] < mrot) {

cout << j << ") ";

display\_record(studs[ind[i]]);

j++;

}

}

*file.h*

#ifndef TASK\_FILE\_H

#define TASK\_FILE\_H

#include "database.h"

void import\_file(const char \*, students \*&, size\_t &);

void rewrite\_file(const char \*, students \*, size\_t);

#endif

*file.cpp*

#include "file.h"

void import\_file(const char \*filename, students \*&studs, size\_t &n) {

FILE \*f;

if ((f = fopen(filename, "rb")) == NULL) {

f = fopen(filename, "wb");

fclose(f);

return;

}

for (;;) {

struct students input;

if (fread(&input, sizeof(students), 1, f) != 1) break;

add\_to\_array(studs, n, input);

}

fclose(f);

}

void rewrite\_file(const char \*filename, students \*studs, size\_t n) {

FILE \*f = fopen(filename, "wb");

for (size\_t i = 0; i < n; i++) {

fwrite(&studs[i], sizeof(students), 1, f);

}

fclose(f);

}

*index.h*

#ifndef TASK\_INDEX\_H

#define TASK\_INDEX\_H

#include "database.h"

void read\_index\_file(const char \*, int \*, int \*, size\_t);

void rewrite\_index\_file(const char \*, int \*, int \*, size\_t);

void add\_to\_index(size\_t, students);

static char currentIndexFileName[32];

#endif

*index.cpp*

#include "index.h"

#include <string.h>

void read\_index\_file(const char \*filename, int \*ind, int \*values, size\_t n) {

FILE \*f;

if ((f = fopen(filename, "r")) == NULL) {

cout << "Failed to open index file.\n";

return;

}

for (size\_t i = 0; i < n; i++) {

fscanf(f, "%d %d", ind+i, values+i);

}

fclose(f);

}

void rewrite\_index\_file(const char \*filename, int \*ind, int \*values, size\_t n) {

FILE \*f;

if ((f = fopen(filename, "w")) == NULL) {

cout << "Failed to open index file.\n";

return;

}

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

fprintf(f, "%d %d\n", ind[i], values[i]);

}

fclose(f);

}

void add\_to\_index(size\_t id, students cur) {

FILE \*f;

f = fopen("group\_index.txt", "a");

fprintf(f, "%d %d\n", id, cur.grp);

fclose(f);

f = fopen("income\_index.txt", "a");

fprintf(f, "%d %d\n", id, cur.income);

fclose(f);

}

**Вывод:** были изучены принципы модульного программирования и проведено разделение уже существующей программы на модули.