Министерство образование Республики Беларусь

Учреждение образования «Полоцкий государственный университет»

Информационно-технологический факультет

Кафедра технологий программирования

Лабораторная работа №6

По дисциплина «Основы алгоритмизации и программирования»

Выполнил: Крутилин С.Д.

Группа 19-ИТ-1.

Проверил: Адамовский Е.

Новополоцк

2019 г.

**Модульное программирование**

**Вариант 2**

**Цель работы**: изучить принципы построения модулей на языке С++, а также реализацию программ, включающих несколько модулей.

**Source.cpp:** подключение заголовочных файлов «Header.h» и «menu.h»; русификация ввода/вывод; установка размеров консоли; вызов **menu()**;

#include "Header.h"

#include "menu.h"

int main() {

setlocale(LC\_ALL, "ru"); // подключение русскоязычной библиотеки

system("chcp 1251> null"); //подключение кириллицы

system("mode con cols=124 lines=70");

menu();

return 0;

}

**ЗАДАНИЕ 1.1.** Создайте новый модуль для меню.

**Описание:** В «menu.h» подключает все необходимые заголовочные файлы; декларирует функцию menu(), которая выводит меню и вызывает все необходимые функции. menu() описана в «menu.cpp».

**menu.h:**

#pragma once

#ifndef MENU\_H

#define MENU\_H

#include "Header.h"

#include "files.h"

#include "ioput.h"

void menu();

#endif //!MENU\_H

**menu.cpp:**

#include "menu.h"

void menu() {

student\* base = NULL; // создаем ссылку на структуру

int size = 0;

char action; //объявление переменной ввода

while (true) {

cout << "Меню:" << endl;

cout << "1. Ввод данных." << endl;

cout << "2. Вывод данных в табличном виде." << endl;

cout << "3. Вывод данных в виде списка." << endl;

cout << "4. Создать базу данных." << endl;

cout << "5. Удалить базу даных." << endl;

cout << "6. Очистить динамическую память" << endl;

cout << "7. Сохранить базу данных." << endl;

cout << "8. Загрузить базу данных." << endl;

cout << "0. Выход из прогаммы." << endl;

action = \_getch();

system("cls");

if (action == '0') {

cout << "Вы вышли из программы" << endl;

exit(0);

}

else if (action == '1') {

resize(&base, size + 1);

input(base, size);

size++;

}

else if (action == '2') {

table(base, size);

}

else if (action == '3') {

list(base, size);

}

else if (action == '4') {

new\_DB();

}

else if (action == '5') {

remove\_DB();

}

else if (action == '6') {

clear(&base, &size);

}

else if (action == '7') {

save\_DB(base, size);

}

else if (action == '8') {

load\_DB(&base, &size);

}

}

}

**ЗАДАНИЕ 1.2.** Создать новый модуль для функций ввод/вывода данных.

**Описание:** в «ioput.h» подключаем все необходимые заголовочные файлы; декларируем функции; в «ioput.cpp» описываем функции необходимые для ввода вывода.

Здесь подключается заголовочный файл «check.h», который содержит функции для проверки корректности ввода; описание объявленный функций содержит файл «check.cpp».

**ioput.h:**

#pragma once

#ifndef IOPUT\_H

#define IOPUT\_H

#include "Header.h"

#include "check.h"

void input(student\* agr, int itr); // ввод данных в структуру

void table(student\* agr, int size); // вывод данных в виде таблицы

void list(student\* agr, int size); // вывод структуры в виде списка

void resize(student\*\* pointer, int newsize); // выделение динамической памяти для структуры

void getLine(char\* str); // считывание данных

#endif //!IOPUT\_H

**ioput.cpp:**

#include "ioput.h"

void input(student\* agr, int itr) {

char buff[256];

// ФИО

do {

cout << "Фамилия (не более 15 символов)" << endl;

getLine(buff);

} while (strlen(buff) > 16 || isContainEnLetter(buff)

|| isContainNumber(buff) || isContainOperationSign(buff)

|| !(isContainRuLetter(buff)));

strcpy(agr[itr].last, buff);

do {

cout << "Имя (не более 15 символов): " << endl;

getLine(buff);

} while (strlen(buff) > 16 || isContainEnLetter(buff)

|| isContainNumber(buff) || isContainOperationSign(buff)

|| !(isContainRuLetter(buff)));

strcpy(agr[itr].first, buff);

do {

cout << "Отчество (не более 15 символов)" << endl;

getLine(buff);

} while (strlen(buff) > 16 || isContainEnLetter(buff)

|| isContainNumber(buff) || isContainOperationSign(buff)

|| !(isContainRuLetter(buff)));

strcpy(agr[itr].middle, buff);

// определение пола

int gender;

while (true) {

cout << "Пол (мужской - 1, женский - 0)" << endl;

if (cin >> gender) {

if (gender == 0) {

strcpy(agr[itr].gender, "женский");

break;

}

else if (gender == 1) {

strcpy(agr[itr].gender, "мужской");

break;

}

}

cin.clear();

cin.ignore(250, '\n');

}

// дата рождения

char dott;

int day, month, years;

int days[12] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 }; // количество дней в месяцах

while (true) {

cout << "День рождения (ДД.ММ.ГГГГ)" << endl;

if (cin >> day >> dott >> month >> dott >> years) //проверка на правильный ввод данных, если данные соответствуют условию, выход из цикла и продолжается работа программы

{

if (month >= 1 && month <= 12 && day >= 1 && dott == '.' && years >= 0 && years <= 2019) {

if (!(years % 4 != 0 || years % 100 == 0 && years % 400 != 0) && day <= 29) {

break;

}

else if (day <= days[month - 1]) {

break;

}

}

}

cin.clear();//восстановление работы потока ввода

cin.ignore(250, '\n'); //очищение потока ввода

}

agr[itr].birthDay = day;

agr[itr].birthMonth = month;

agr[itr].birthYear = years;

// ввод паспортных данных

char serial[10][3] = { "AB", "BM", "HB", "KH", "MP", "MC", "KB", "PP", "SP", "DP" };

while (true) {

do {

cout << "Введите серию пасспорта (AB, BM, HB, KH, MP, MC, KB, PP, SP, DP)" << endl;

getLine(buff);

} while (strlen(buff) > 2 || !(isContainEnLetter(buff))

|| isContainNumber(buff) || isContainOperationSign(buff)

|| isContainRuLetter(buff));

bool isTrue = false;

for (int i = 0, j = 0; i < 10;) {

if (buff[j] == serial[i][j]) {

j++;

if (j == 2) {

isTrue = true;

break;

}

}

else {

i++;

j = 0;

}

}

if (isTrue) {

break;

}

}

strcpy(agr[itr].series, buff);

do {

cout << "Номер паспорта (не более 7 цифр)" << endl;

getLine(buff);

} while (strlen(buff) != 7 || !(isContainNumber(buff)) || isContainEnLetter(buff) || isContainRuLetter(buff) || isContainOperationSign(buff));

strcpy(agr[itr].idNum, buff);

// факультет

int action = -1;

char facl[7][4] = { "ГФ", "ИСФ", "МТФ", "РТФ", "ФИТ", "ФЭФ", "ЮФ" };

while (true) {

do {

cout << "Введите название факультета (";

for (int i = 0; i < sizeof(facl) / sizeof(\*facl); i++) {

if (i != 0 && i != sizeof(facl) / sizeof(\*facl)) {

cout << ", ";

}

else {

cout << " ";

}

cout << facl[i];

}

cout << ")" << endl;

getLine(buff);

} while (strlen(buff) > 3 || isContainEnLetter(buff)

|| isContainNumber(buff) || isContainOperationSign(buff)

|| !(isContainRuLetter(buff)));

for (int i = 0, j = 0; i < 7;) {

if (buff[j] == facl[i][j]) {

j++;

if (j == strlen(facl[i]) && j == strlen(buff)) {

action = i;

break;

}

}

else {

i++;

j = 0;

}

}

if (action >= 0) {

break;

}

}

strcpy(agr[itr].faculty, buff);

// специальность

char spec[7][10][4] = {

{ "ДО", "ФС", "ИС", "ИЯ", "ОТ", "ПП", "РГФ", "ТГ", "ТТ", NULL },

{ "АД", "АРХ", "ВВ", "ДИЗ", "ПГС", "ТВ", "УН", NULL, NULL, NULL },

{ "ТС", "АС", "ТМ", "ТЭА", "ТОМ", "ОДД", "ХТ", "ТНГ", "МА", "ПТТ" },

{ "ПЭЛ", "РК", "ЭС", "РТ", NULL, NULL, NULL, NULL, NULL, NULL },

{ "ВС", "ГЕО", "ГИС", "ИТ", "КБ", "ПГ", "МС", NULL, NULL, NULL },

{ "БК", "ЭП", "ФК", "ЛГ", "СК", "ЭМ", NULL, NULL, NULL, NULL },

{ "ПР", NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL }

};

int choise = -1;

while (true) {

do {

cout << "Введите название специальности (";

for (int i = 0; i < sizeof(spec[action]) / sizeof(\*spec[action]); i++) {

if (i != 0 && i != sizeof(spec[action]) / sizeof(\*spec[action]) && \*spec[action][i] != NULL) {

cout << ", ";

}

else if (\*spec[action][i] != NULL) {

cout << " ";

}

cout << spec[action][i];

}

cout << ")" << endl;

getLine(buff);

} while (strlen(buff) > 3 || isContainEnLetter(buff)

|| isContainNumber(buff) || isContainOperationSign(buff)

|| !(isContainRuLetter(buff)));

for (int i = 0, j = 0; i < 7;) {

if (buff[j] == spec[action][i][j]) {

j++;

if (j == strlen(spec[action][i]) && j == strlen(buff)) {

choise = i + 1;

break;

}

}

else {

i++;

j = 0;

}

}

if (choise >= 0) {

break;

}

}

strcpy(agr[itr].speciality, buff);

system("cls");

}

void list(student\* agr, int size) {

if (size == 0) {

cout << "Данные отсутствуют.";

}

else {

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

cout << "Имя: " << agr[i].first << endl;

cout << "Фамилия: " << agr[i].last << endl;

cout << "Отчество: " << agr[i].middle << endl;

cout << "Пол: " << agr[i].gender << endl;

cout << "День рождения: " << setfill('0') << setw(2) << agr[i].birthDay << "." << setw(2) << agr[i].birthMonth << "." << agr[i].birthYear << setfill(' ') << endl;

cout << "Серия: " << agr[i].series << endl;

cout << "Номер паспорта: " << agr[i].idNum << endl;

cout << "Факультет: " << agr[i].faculty << endl;

cout << "Специальнгость: " << agr[i].speciality << endl;

cout << endl;

}

}

cout << endl;

system("pause");

system("cls");

}

void table(student\* agr, int size) {

if (size == 0) {

cout << "Данные отсутствуют.";

}

else {

cout << "+---+---------------+---------------+---------------+-------+--------------+---------+-------------+-----------------------+" << endl;

cout << "| # | Фамилия | Имя | Отчество | Пол |Дата рождения|Факультет|Специальность|Серия и номер пасспорта|" << endl;

cout << "+---+---------------+---------------+---------------+-------+--------------+---------+-------------+-----------------------+" << endl;

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

if (i >= 0 && i < 10) {

cout << "| " << i << " |";

}

else {

cout << "| " << i << "|";

}

cout << setw(15) << agr[i].last << "|";

cout << setw(15) << agr[i].first << "|";

cout << setw(15) << agr[i].middle << "|";

cout << setw(7) << agr[i].gender << "|";

cout << setw(4) << " " << setfill('0') << setw(2) << agr[i].birthDay << "." << setw(2) << agr[i].birthMonth << "." << setw(4) << agr[i].birthYear << setfill(' ') << "|";

cout << setw(9) << agr[i].faculty << "|";

cout << setw(13) << agr[i].speciality << "|";

cout << setw(15) << agr[i].series << " ";

cout << agr[i].idNum << "|";

cout << endl;

cout << "+---+---------------+---------------+---------------+-------+--------------+---------+-------------+-----------------------+" << endl;

}

}

cout << endl;

system("pause");

system("cls");

}

void resize(student\*\* pointer, int newsize) {

student\* buff = (student\*)realloc(\*pointer, sizeof(student) \* newsize);

if (buff == NULL) {

cout << "Не удалось выделить память!" << endl;

exit(-1);

}

\*pointer = buff;

}

void getLine(char\* str)

{

do

{

fgets(str, 255, stdin);

} while (str[0] == '\n');

int length = strlen(str);

str[length - 1] = '\0';

}

**check.h:**

#pragma once

#ifndef CHECK\_H

#define CHECK\_H

#include "Header.h"

bool isContainNumber(char\* str); // проверка есть ли числа в строке

bool isContainRuLetter(char\* str); // проверяет есть ли русские буквы в строке

bool isContainEnLetter(char\* str); // проверяет есть ли английские буквы в строке

bool isContainOperationSign(char\* str); // поверяет есть ли

#endif

**check.cpp:**

#include "check.h"

bool isContainNumber(char\* str) {

int length = strlen(str);

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

if (str[i] >= '0' && str[i] <= '9') {

return true;

}

}

return false;

}

bool isContainEnLetter(char\* str) {

int length = strlen(str);

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

if ((str[i] >= 'a' && str[i] <= 'z') || (str[i] >= 'A' && str[i] <= 'Z')) {

return true;

}

}

return false;

}

bool isContainRuLetter(char\* str) {

int length = strlen(str);

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

if ((str[i] >= 'а' && str[i] <= 'я') || (str[i] >= 'А' && str[i] <= 'Я')) {

return true;

}

}

return false;

}

bool isContainOperationSign(char\* str) {

int length = strlen(str);

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

if (!((str[i] >= 'а' && str[i] <= 'я') || (str[i] >= 'А' && str[i] <= 'Я'))

&& !((str[i] >= 'a' && str[i] <= 'z') || (str[i] >= 'A' && str[i] <= 'Z'))

&& !(str[i] >= '0' && str[i] <= '9')) {

return true;

}

}

return false;

}

**ЗАДАНИЕ 1.3.** Создать новый модуль для функций работы с файлами.

**Описание:** декларируем функции в заголовочном файле «files.h»; описываем функции для работы с файлами в файле «files.cpp».

**files.h:**

#pragma once

#ifndef file\_h

#define file\_h

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "Header.h"

ofstream new\_DB();

void remove\_DB();

void clear(student\*\* pointer, int\* size);

void save\_DB(student\* agr, int size);

void load\_DB(student\*\* agr, int\* size);

void resize\_z(student\*\* pointer, int newsize);

#endif //!file\_H

**files.cpp:**

#include "files.h"

ofstream new\_DB() {

char file[256];

cout << "Введите название файла(с указанием формата): ";

cin >> file;

ofstream ofile(file, ios\_base::app);

if (ofile.is\_open()) {

cout << "Файл создан." << endl;

}

system("pause");

system("cls");

return ofile;

}

void remove\_DB() {

char file[256];

cout << "Введите название файла(с указанием формата): ";

cin >> file;

if (remove(file) != 0) {

cout << "Ошибка удаления файла" << endl;

}

else {

cout << "Файл успешно удалён" << endl;

}

system("pause");

system("cls");

}

void clear(student\*\* pointer, int\* size) {

student\* buff = (student\*)realloc(\*pointer, 0);

\*size = 0;

\*pointer = buff;

cout << "Данные успешно очищены" << endl;

system("pause");

system("cls");

}

void save\_DB(student\* agr, int size) {

char file[256];

cout << "Введите название файла(с указанием формата): ";

cin >> file;

ofstream fout(file, ofstream::trunc);

fout << size << endl;

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

fout << agr[i].last << endl;

fout << agr[i].first << endl;

fout << agr[i].middle << endl;

fout << agr[i].gender << endl;

//fout << setfill('0') << setw(2) << agr[i].birthDay << "." << setw(2) << agr[i].birthMonth << "." << setw(4) << agr[i].birthYear << setfill(' ') << endl;

fout << agr[i].birthDay << endl;

fout << agr[i].birthMonth << endl;

fout << agr[i].birthYear << endl;

fout << agr[i].faculty << endl;

fout << agr[i].speciality << endl;

fout << agr[i].series << endl;

fout << agr[i].idNum << endl;

}

fout.close();

cout << "Запись в файл прошла успешно!" << endl;

system("pause");

system("cls");

}

void load\_DB(student\*\* agr, int\* size) {

char file[50];

cout << "Введите название файла(с указанием формата): ";

cin >> file;

ifstream fin(file);

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

cout << "Ошибка загрузки файла." << endl;

}

else {

char ch[51];

fin.getline(ch, 51);

(\*size) = atoi(ch);

resize\_z(agr, \*size);

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

fin.getline((\*agr)[i].last, 16);

fin.getline((\*agr)[i].first, 16);

fin.getline((\*agr)[i].middle, 16);

fin.getline((\*agr)[i].gender, 8);

fin.getline(ch, 11);

(\*agr)[i].birthDay = atof(ch);

fin.getline(ch, 11);

(\*agr)[i].birthMonth = atof(ch);

fin.getline(ch, 11);

(\*agr)[i].birthYear = atof(ch);

fin.getline((\*agr)[i].faculty, 4);

fin.getline((\*agr)[i].speciality, 4);

fin.getline((\*agr)[i].series, 3);

fin.getline((\*agr)[i].idNum, 8);

}

fin.close();

cout << "Файл считан\n";

}

system("pause");

system("cls");

}

void resize\_z(student\*\* pointer, int newsize) {

student\* buff = (student\*)realloc(\*pointer, sizeof(student) \* newsize);

if (buff == NULL) {

cout << "Не удалось выделить память!" << endl;

exit(-1);

}

\*pointer = buff;

}

**ЗАДАНИЕ 2.** Перенесите пользовательские функции, реализованные в предыдущих лабораторных работа, в соответствующие модули.

**Описание:** смотрите в **ЗАДАНЯХ 1.1, 1.2, 1.3**;

**ЗАДАНИЕ 3.** Оформите для всех модулей.

**Описание:** смотрите в **ЗАДАНЯХ 1.1, 1.2, 1.3**;

**ЗАДАНИЕ 4.** Подключите модули друг к другу в соответствии с тем, каким образом они будут взаимодействовать.

**Описание:** смотрите в **ЗАДАНЯХ 1.1, 1.2, 1.3**;

**ЗАДАНИЕ 5.** Нарисуйте модульную схему вашего приложения.

main

Header

ioput

files

check