Министерство образования РФ

Пермский национальный исследовательский политехнический университет

Кафедра ИТАС

ЛАБОРАТОРНАЯ РАБОТА № 8

ПО ИНФОРМАТИКЕ ЗА I СЕМЕСТР

Вариант 5

|  |
| --- |
| Выполнил студент:  Отинов Иван Юрьевич  Группа РИС-20-1бз  Шифр 20-ЭТФ-635  Кафедра ИТАС:  Полякова Ольга Андреевна |

ПЕРМЬ 2020

# Лабораторная работа №8 "Блоковый ввод-вывод"

**Вариант №5**

**Цель:**

Работа с двоичными файлами, организация ввода-вывода структурированной информации и ее хранение на внешних носителях.

**Задача:**

Сформировать двоичный файл из элементов, заданной в варианте структуры, распечатать его содержимое, выполнить удаление и добавление элементов в соответствии со своим вариантом, используя для поиска удаляемых или добавляемых элементов функцию. Формирование, печать, добавление и удаление элементов оформить в виде функций. Предусмотреть сообщения об ошибках при открытии файла и выполнении операций ввода/вывода.

**Вариант задания:**

Структура "Человек":

* фамилия, имя, отчество;
* год рождения;
* рост;
* вес.

Удалить все элемент с указанным ростом и весом, добавить элемент после элемента с указанной фамилией.

**Текст программы.**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include <Windows.h>

#include <cstdlib>

using namespace std;

struct HUMAN {

// поля структуры

// фамилия, имя, отчество

char surname[50];

char name[50];

char midname[50];

// год рождения

int yearOfBirth;

int grew;// рост

int weight;// вес

public:

// деструктор

~HUMAN()

{

// очистка полей

clear();

}

// очистка полей

void clear()

{

this->yearOfBirth = 0;

this->grew = 0;

this->weight = 0;

strcpy(this->surname, "");

strcpy(this->name, "");

strcpy(this->midname, "");

}

// запись данных в поля

void add(int yearOfBirth, int grew, int weight,

char surname[50], char name[50], char midname[50])

{

this->yearOfBirth = yearOfBirth;

this->grew = grew;

this->weight = weight;

strcpy(this->surname, surname);

strcpy(this->name, name);

strcpy(this->midname, midname);

}

// вывод записей экран

void display()

{

cout << this->surname << " " << this->name << " " << this->midname << endl;

cout << "Год рождения: " << this->yearOfBirth << endl;

cout << "Рост: " << this->grew << endl;

cout << "Вест: " << this->weight << endl;

cout << endl;

}

// формирует строку для файла

const char\* getLine()

{

char str[255];

strcpy(str, this->surname);

strcat(str, " ");

strcat(str, this->name);

strcat(str, " ");

strcat(str, this->midname);

strcat(str, " ");

char string[6] = "";

// год в строку

\_itoa(this->yearOfBirth, string, 10);

strcat(str, string);

strcat(str, " ");

// рост в строку

\_itoa(this->grew, string, 10);

strcat(str, string);

strcat(str, " ");

// вес в строку

\_itoa(this->weight, string, 10);

strcat(str, string);

strcat(str, "\n");

return str;

}

const char\* getSurname() { return this->surname; }

const int getGrew() { return this->grew; }

const int getWeight() { return this->weight; }

};

// является ли данная строка действительным или вещественным числом

bool isdigit(char\* s)

{

int i = 0;

int len = strlen(s);

while (i < len) {

if (s[i] < '0' || s[i] > '9') return false;

i++;

}

return true;

}

// вставка объекта в заданную позицию массива

void insert(const int &index, int &totalCount, HUMAN \*obj,

int yearOfBirth, int grew, int weight,

char surname[50], char name[50], char midname[50])

{

// копируем данные в новый массив, чтобы вставить новый элемент

int pos = -1;

int t = 0;

HUMAN objTemp[1000];// новый массив с добавляемым элементом

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

// запомним и пропустим позицию, куда нужно вставить новый элемент

if (i == index) {

pos = i;

t = 1;

}

// копируем элементы массива в другой

objTemp[i + t] = obj[i];

}// for

// вставка нового элемента в указанную позицию

objTemp[pos].add(yearOfBirth, grew, weight, surname, name, midname);

totalCount++;

// перемещение записей в старый массив

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

obj[i] = objTemp[i];

}

}

// ввод чисел с проверкой на ввод

void correctingInsert(const char msg[100], int &value)

{// msg - сообщение, value - корректное значение

// колво попыток ввода

int countTry = 0;

char sValue[10];

do {

if (countTry > 0) {

printf("Некорректный ввод числа!\n");

system("pause");

system("CLS");

}

cout << msg;

cin >> sValue;

} while (!isdigit(sValue));

value = atoi(sValue);

}

void create(HUMAN\* obj, int& totalCount, int inxInsert = -1)

{// bool inxInsert = -1 вставка / >= 0 - индекс добавление записи

char surname[50];

char name[50];

char midname[50];

// год рождения

int yearOfBirth;

int grew;// рост

int weight;// вес

cout << "ДОБАВЛЕНИЕ ДАННЫХ\n";

cout << "Фамилия имя отчество: ";

cin >> surname;

cin >> name;

cin >> midname;

// ввод чисел с проверкой на ввод

correctingInsert("Год рожденя: ", yearOfBirth);

correctingInsert("Рост: ", grew);

correctingInsert("Вес: ", weight);

// добавление данных в объект

if (inxInsert == -1)

obj[totalCount++].add(yearOfBirth, grew, weight, surname, name, midname);

// вставка объекта в заданный индекс

else insert(inxInsert, totalCount, obj,

yearOfBirth, grew, weight,

surname, name, midname);

cout << "Данные успешно доавлены!\n";

}

// поиск индекса записи по фамилии

int searchBySurname(HUMAN\* obj, int& totalCount)

{

// ввод фамилии дляпоиска

char surname[100];// фамилия для ввода

printf("Введите фамилию для вставки: ");

scanf("%s", surname);

// поиск

int inxFound = -1;// индекс найденной записи в массиве

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

if (strcmp(surname, obj[i].getSurname()) == 0) {

inxFound = i;// запись найдена

break;

}

}

if (inxFound < 0) printf("Запись не найдена!\n");

return inxFound;

}

// вставка данных в массив объектов после заданного объекта по фамилии

void insertData(HUMAN\* obj, int& totalCount)

{

int inxFound = searchBySurname(obj, totalCount);

inxFound++;// переход к следующей записи

if (inxFound < 0) printf("Запись найти не удалось!\n");

else create(obj, totalCount, inxFound); // вставка данных в список

}

// вывод всех предприятий списка

void displayAll(HUMAN\* obj, int& totalCount)

{

cout << "ИНФОРМАЦИЯ О ВСЕХ ЗАПИСЯХ:\n\n";

if (totalCount > 0) {

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

obj[i].display();

}

else cout << "Данных нет...\n";

}

// удаление заданного элемента из массива по индексу

bool removeElementByInx(HUMAN\* obj, int& totalCount, int index)

{

bool nResult = false;// результат удаления

int t = 0;

HUMAN objTemp[1000];// новый массив без удаляемого элемента

// копируем данные в новый массив без удаляемого элемента

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

// копируем элементы массива в другой исключая удаляемый

if (i == index) {

// элемент удален

nResult = true;

t = 1;

}

objTemp[i] = obj[i+t];

}

if (nResult) {

// перемещение записей в старый массив

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

obj[i] = objTemp[i];

// очищаем последний элемент

if (i + 1 == totalCount) obj[i].clear();

}

totalCount--;

}

return nResult;

}

// удаление элементов с указанным ростом и весом

void removeData(HUMAN\* obj, int& totalCount)

{

int grew;// рост

int weight;// вес

// ввод данных с проверкой корреткности

correctingInsert("Рост: ", grew);

correctingInsert("Вес: ", weight);

// поиск записей для удаления,

// где значения совпадают указанному росту и весу

int countRemoved = 0;

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

if (obj[i].getGrew() == grew && obj[i].getWeight() == weight) {

countRemoved += removeElementByInx(obj, totalCount, i);

i--;

}

}

if (countRemoved > 0) printf("Данные удалены!\n");

else printf("Данные с указанными значениями не найдены!\n");

}

// запись объекта структуры в бинарный файл

bool writeArrObjToBinatyFile(const char\* pathToFile, HUMAN\* obj, unsigned size)

{

// unsigned size// колво цифр в файле

FILE\* iofile = NULL;// указатель на файловый поток

// создаем файл, если все корректно продолжаем работу

iofile = fopen(pathToFile, "wb");

if (iofile == NULL) {

printf("Ошибка открытия файла");

system("pause");

return false;

}

// запись объекта в бинарный файл

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

HUMAN tmp = obj[i];

fwrite(&tmp, sizeof(HUMAN), 1, iofile);

}

// закрываем файл

fclose(iofile);

return true;

}

// чтение объекта структуры из бинарного файла

bool readArrObjFromBinatyFile(const char\* pathToFile, HUMAN\* obj, int &size)

{

FILE \*iofile = fopen(pathToFile, "rb");

if (iofile == NULL)

{

printf("Ошибка чтения файла...\n");

return false;

}

while (!feof(iofile) && size <= 1000)

{

fread(&obj[size], sizeof(HUMAN), 1, iofile);

size++;

}

size--;

fclose(iofile);

return true;

}

// реализация меню программы

int menu()

{// реализация меню программы

system("CLS");//очистка окна консоли

cout << "1. Добавление записей\n";

cout << "2. Список всех записей\n";

cout << "3. Добавить запись после указанной фамилии\n";

cout << "4. Удаление записей\n";

cout << "0. Выход\n";

cout << "Ввод: ";

int choose;

cin >> choose;

//возврат выбранного пункта меню

return choose;

}

int main()

{

setlocale(LC\_ALL, "RU");

SetConsoleCP(1251);// установка кодовой страницы win-cp 1251 в поток ввода

SetConsoleOutputCP(1251); // установка кодовой страницы win-cp 1251 в поток вывода

HUMAN obj[1000];// объекта

int totalCount = 0;// заполненное колво записей

readArrObjFromBinatyFile("myFile.dat", obj, totalCount);

bool loop = true;

while (loop) {

int choose = menu();

system("CLS");

switch (choose)

{

case 1: create(obj, totalCount); break;

case 2: displayAll(obj, totalCount); break;

case 3: insertData(obj, totalCount); break;

case 4: removeData(obj, totalCount); break;

case 0: loop = false; break;

default: cout << "Неверный выбор...\n"; break;

}//switch

if (loop) system("pause");

}//while

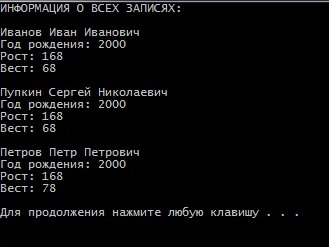
// запись даных в бинарный файл

writeArrObjToBinatyFile("myFile.dat", obj, totalCount);

return 0;

}

**Результаты работы программы:**

****