**ОТЧЕТ**

по лабораторной работе №1

по дисциплине «Программирование»

Вариант №21

**Тема работы:** Добавление записи в массив структур и удаление записи из массива структур по запросу пользователя

**Цель работы:** получение студентами навыков работы с двумерными массивами в С++

**Задание:** Дан исходный массив записей, содержащие сведения о датах рождения студентов группы. Каждая запись содержит фамилию и инициалы, дату рождения (число, месяц, год). Найти студентов, родившихся в один и тот же месяц и год (выходной массив структур).

1. Сортировка исходного массива структур:
   1. По дню рождения в порядке убывания
   2. По дате рождения в порядке возрастания возраста.
2. Сортировка выходного массива структур:
   1. По фамилии и инициалам студента в алфавитном порядке.

**Выполнение задания:**

**Текст программы:**

#pragma once

**//Array.h**

template<typename T>

struct Array{

size\_t capasity,

size;

T data[30];

Array();

T& operator[](const int id);

int insert(T &node);

int earase(const int id);

void clear();

};

template<typename T>

Array<T>::Array() {

capasity = 30;

size = 0;

}

template<typename T>

T& Array<T>::operator[](const int id){

return data[id - 1];

}

template<typename T>

int Array<T>::insert(T &node) {

T tmp = node;

if (size < capasity) {

data[size] = tmp;

size++;

return 0;

}

return -1;

}

template<typename T>

int Array<T>::earase(const int id){

for (size\_t i = id - 1; i < size - 1; i++)

std::swap(data[i], data[i + 1]);

--size;

return 0;

}

template<typename T>

inline void Array<T>::clear(){

size = 0;

}

**//UserStructs.h**

#pragma once

#include <iostream>

#define eqYear 0xFFFF0000

#define eqMounth 0x0000FF00

#define eqDay 0x000000FF

struct Date {

int day,

mounth,

year;

bool isEqual(Date date, size\_t mask = 0xFFFFFFFF) {

return (getHash() & mask) == (date.getHash() & mask);

};

int getHash() {

return day | (mounth << 8) | (year << 16);

};

friend std::istream& operator>>(std::istream &in, Date &date);

friend std::ostream& operator<<(std::ostream &out, Date &date);

};

struct Student {

char name[36],

group[12];

Date birthday;

friend std::istream& operator>>(std::istream &in, Student &student);

friend std::ostream& operator<<(std::ostream &out, Student &student);

};

**//UserStructs.cpp**

#include <iostream>

#include <iomanip>

#include"UserStructs.h"

std::istream& operator>>(std::istream &in, Date &date) {

char p;

in >> date.day >> p >> date.mounth >> p >> date.year;

return in;

};

std::ostream & operator<<(std::ostream & out, Date & date) {

out << std::setw(3) << date.day << " |" << std::setw(3) << date.mounth << " |" << std::setw(5) << date.year << " |";

return out;

}

std::istream& operator>>(std::istream &in, Student &student) {

char inic[5];

in >> student.name >> inic >> student.group >> student.birthday;

strcat\_s(student.name, " ");

strcat\_s(student.name, inic);

return in;

};

std::ostream & operator<<(std::ostream & out, Student & student) {

out << std::setw(35) << student.name << " |" << std::setw(11) << student.group << " |" << student.birthday << std::endl;

return out;

}

**//Interface.h**

#pragma once

#include <iostream>

#include <fstream>

#include <string>

#include <iomanip>

#include "Array.h"

#include "UserStructs.h"

const int headerTableLen = 5;

const int maxStrLen = 100;

const int maxTextMenuLen = 20;

using namespace std;

struct Interface{

char headerTable[headerTableLen][maxStrLen],

textMenu[maxTextMenuLen][maxStrLen];

int lenMenu;

Interface();

void loadFromFile(Array<Student> &arr);

void printHeaderTable(ostream &out);

void printTable(Array<Student> &arr, ostream & out = cout);

void printTable(Student student, int id, ostream & out = cout);

void writeToFile(Array<Student> &arr);

void add(Array<Student> &arr);

void del(Array<Student> &arr);

int menu();

};

Interface::Interface(){

ifstream fHeader("header.txt");

ifstream fMenu("textmenu.txt");

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

fHeader.getline(headerTable[i], maxStrLen);

}

for (lenMenu = 0; !fMenu.eof() && lenMenu < maxTextMenuLen; lenMenu++){

fMenu.getline(textMenu[lenMenu], maxStrLen);

}

};

inline int Interface::menu(){

int res;

for (size\_t i = 3; i < lenMenu; i++) {

cout << textMenu[i] << endl;

}

cout << textMenu[0]; cin >> res;

return res;

}

void Interface::loadFromFile(Array<Student> &arr) {

string fileName;

cout << "Введите имя файла: "; cin >> fileName; //C++11

ifstream file(fileName);

if (!file.is\_open())

cout << "\n";

Student tmp;

while (!file.eof()) {

file >> tmp;

if (arr.insert(tmp) == -1) {

cout << "Ошибка при загрузке записей из файла. Их слишком много?";

return;

}

};

cout << "Записи добавлены!" << endl;

};

void Interface::printHeaderTable(ostream & out){

for (size\_t i = 0; i<5; i++) {

out << headerTable[i] << endl;

}

};

void Interface::printTable(Array<Student> &arr, ostream & out) {

printHeaderTable(out);

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

out << "|" << setw(3) << i << " |" << arr[i];

}

out << headerTable[4]<<endl;

}

void Interface::printTable(Student student, int id, ostream & out){

printHeaderTable(out);

out << "|" << setw(3) << id << " |" << student;

out << headerTable[4] << endl;

}

void Interface::writeToFile(Array<Student> & arr){

string fileName;

cout << "Введите имя файла: "; cin >> fileName;

ofstream file(fileName);

this->printTable(arr, file);

cout << "Файл записан!" << endl;

}

void Interface::add(Array<Student> & arr){

Student node;

char inic[5];

cout << "Введите фамилю и инициалы студента: ";

cin >> node.name >> inic;

strcat\_s(node.name, " ");

strcat\_s(node.name, inic);

cout << "Введите группу студента: "; cin>>node.group;

cout << "Введите дату рождения студента в формате дд.мм.гггг: "; cin >> node.birthday;

if (arr.insert(node) == 0)

cout << "Запись добавлена." << endl;

else

cout << "Ошибка при добавлении записи!" << endl;

}

void Interface::del(Array<Student>& arr){

printTable(arr);

char ans;

int id;

cout << "Укажите номер записи для удаления: "; cin >> id;

printTable(arr[id],id);

cout << "Вы уверены, что нужно удалить запись? (д/н): "; cin >> ans;

if (ans == 'д') arr.earase(id);

};

Array<Student> FindStudents(Array<Student> &students) {

bool added[30] = { false };

Array<Student> result;

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

if (added[i])

continue;

for (size\_t j = i + 1 ; j < students.size; j++) {

if (!students[i].birthday.isEqual(students[j].birthday, eqMounth | eqYear))

continue;

if (!added[i]) {

result.insert(students[i]);

added[i] = true;

}

if (!added[j]) {

result.insert(students[j]);

added[j] = true;

}

};

};

return result;

}

**//Main.cpp**

#include<iostream>

#include<Windows.h>

#include <cstdlib>

#include "Interface.h"

using namespace std;

//Сортировка пузырьком на шаблонах.

template<typename T, typename Compare>

void sort(T &arr, Compare cmp) {

for (int i = arr.size; i > 0; i--)

for (int j = 1; j < i; j++)

if (cmp(arr[j], arr[j + 1]))

swap(arr[j], arr[j + 1]);

cout << "Записи отсортированы!" << endl;

};

bool ByDay(Student &a, Student &b) {

return a.birthday.day < b.birthday.day;

}

struct ByBirthday{

bool operator() (Student &a, Student &b) {

return a.birthday.getHash() > b.birthday.getHash();

};

}byBirthday;

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Interface UI;

Array<Student> inArr, outArr;

while (true){

system("cls");

int key = UI.menu();

switch (key){

case 1:

UI.loadFromFile(inArr); break;

case 2:

UI.printTable(inArr); break;

case 3:

UI.add(inArr); break;

case 4:

UI.del(inArr); break;

case 5:

sort(inArr, ByDay); //В качестве компаратора указатель на функцию

break;

case 6:

sort(inArr, byBirthday); break;//структура с перегруженным опер. ()

case 7:

UI.writeToFile(inArr); break;

case 8:

outArr = FindStudents(inArr); break;

case 9:

UI.printTable(outArr); break;

case 10:

sort(outArr, [](Student &a, Student &b) { //Лямба-функция

return a.birthday.year > b.birthday.year ? true :

a.birthday.year == b.birthday.year &&

a.birthday.mounth > b.birthday.mounth ? true :

a.birthday.mounth == b.birthday.mounth &&

strcmp(a.name, b.name) > 0 ? true : false;

});

break;

case 11:

UI.writeToFile(outArr); break;

case 12:

exit(0);

default:

cout << "Такого пункта не существует. Попробуйте еще раз." << endl;

}

system("pause");

}

return 0;

};

**Скриншоты программы:**

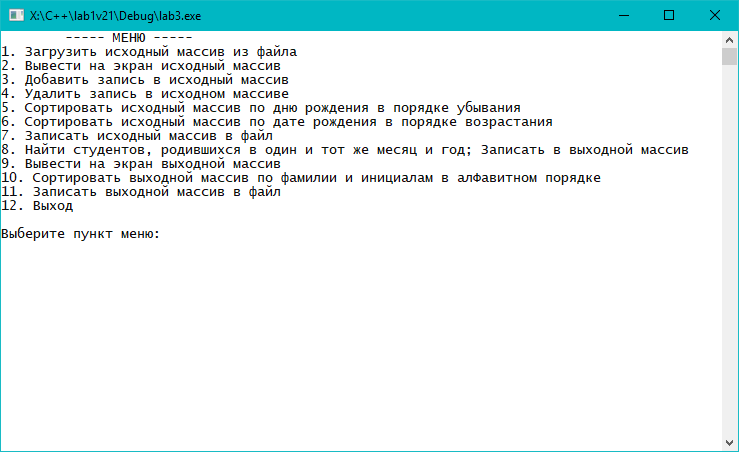


Рис. 1. Меню

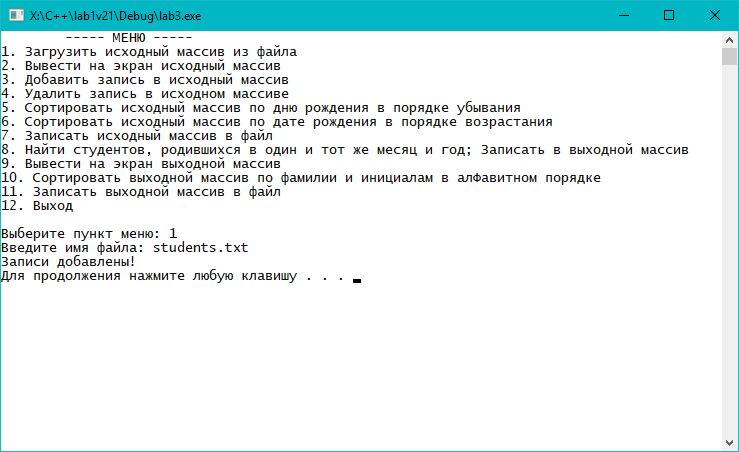


Рис. 2. Ввод данных с файла.

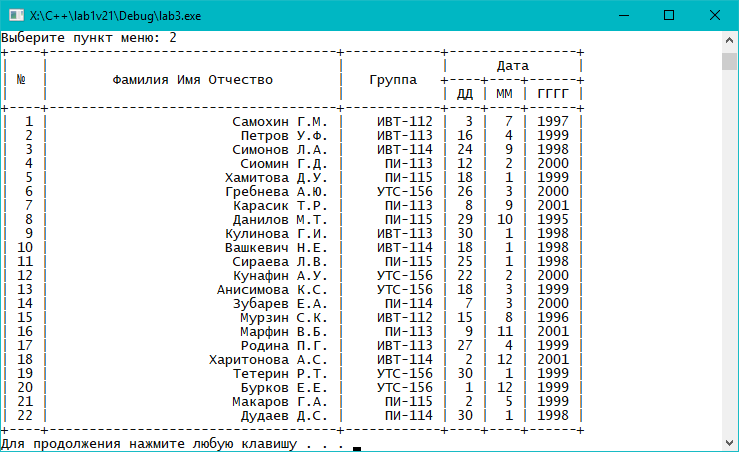


Рис. 3. Вывод массива на экран

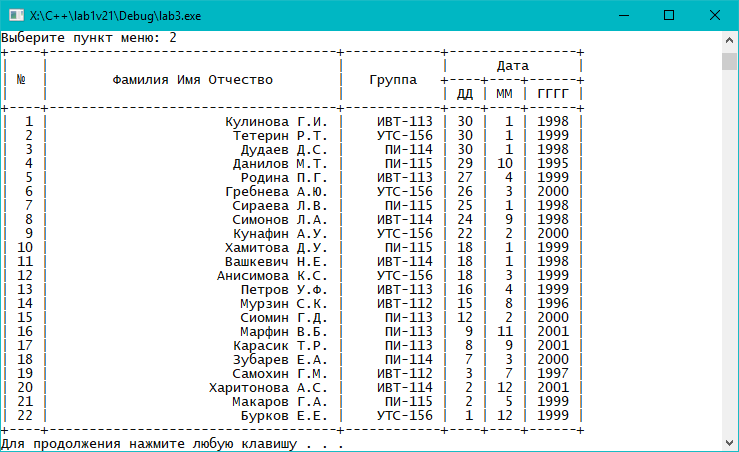


Рис. 4. Сортировка исходного массива по дню рождения в порядке убывания

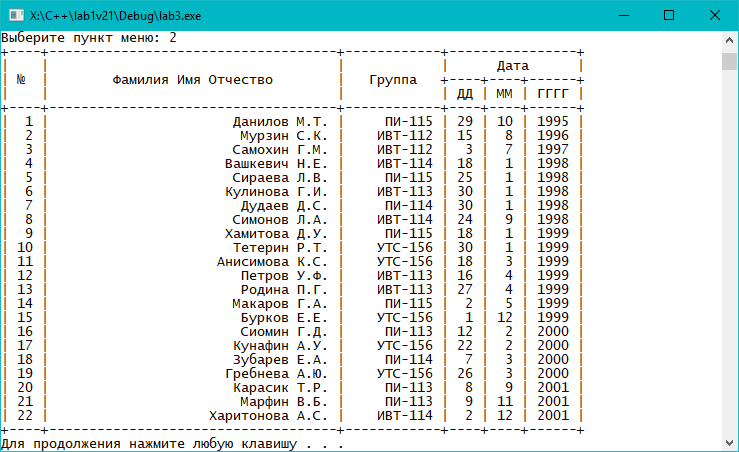


Рис. 5. Сортировка исходного массива по дате рождения в порядке возрастания

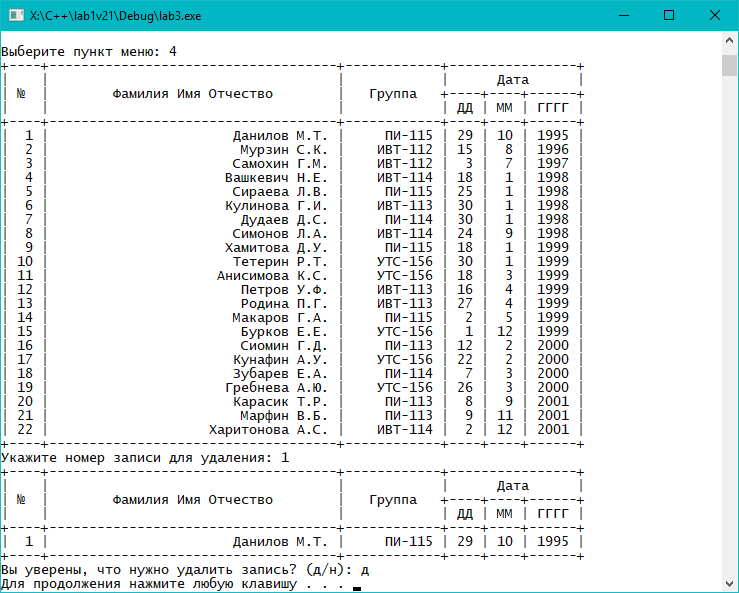


Рис. 6. Удаление записи из массива

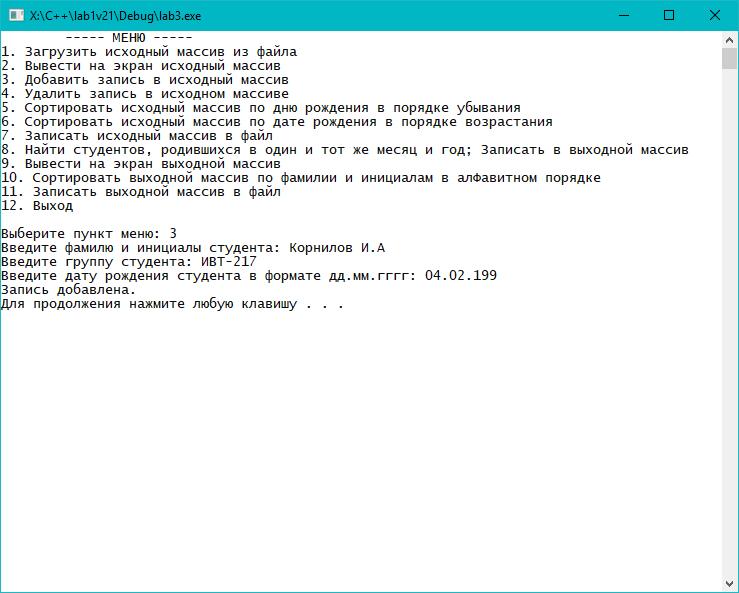


Рис. 7. Добавление новой записи в массив

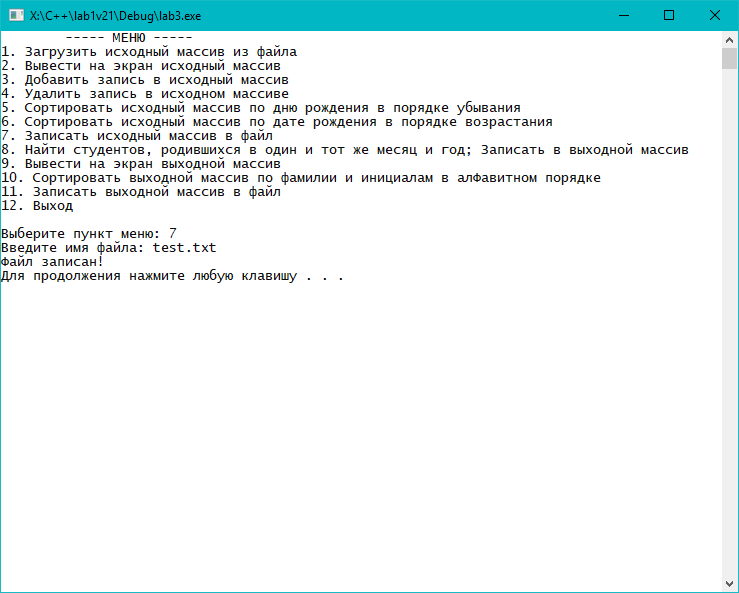


Рис. 8. Вывод массива в файл

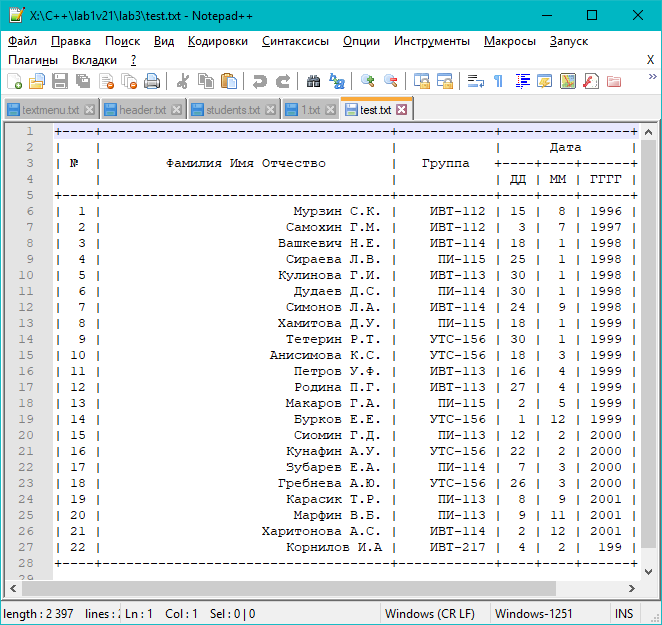


Рис. 9. Результат вывода массива в файл

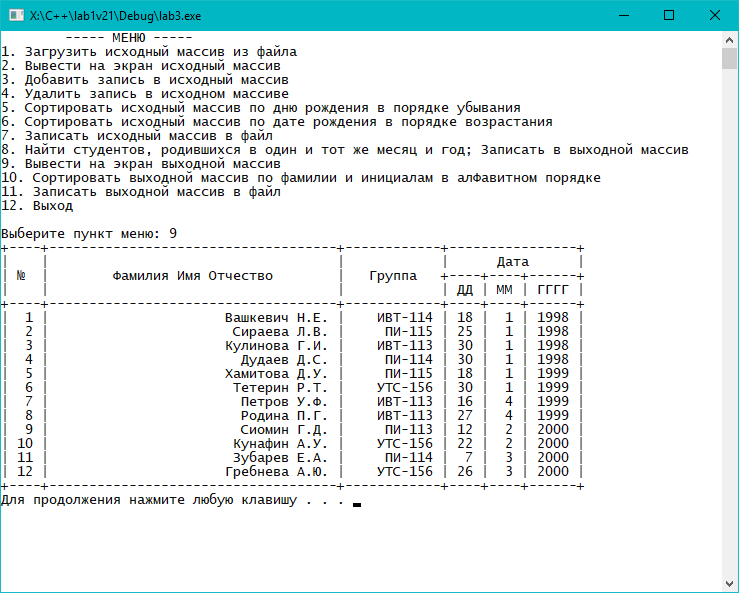


Рис. 10. Поиск студентов, родившихся в один и тот же месяц и год

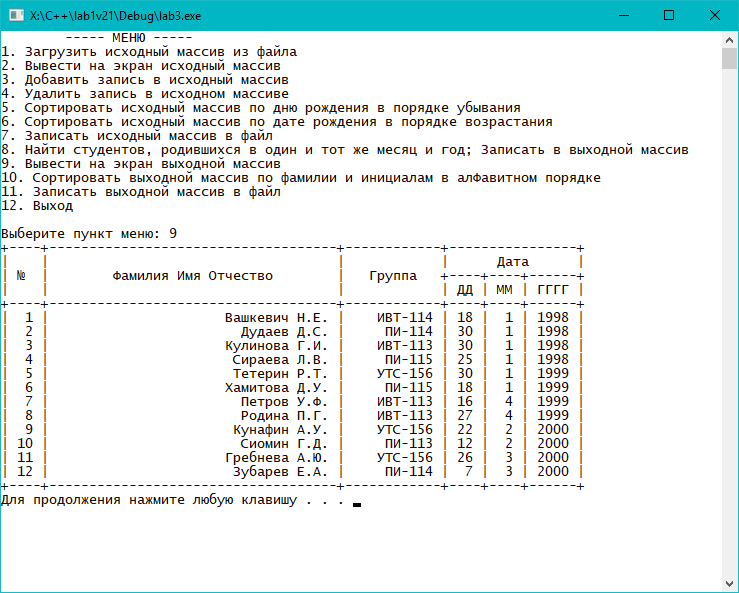


Рис. 11. Сортировка выходной массив по фамилии и инициалам в алфавитном порядке (по группам)

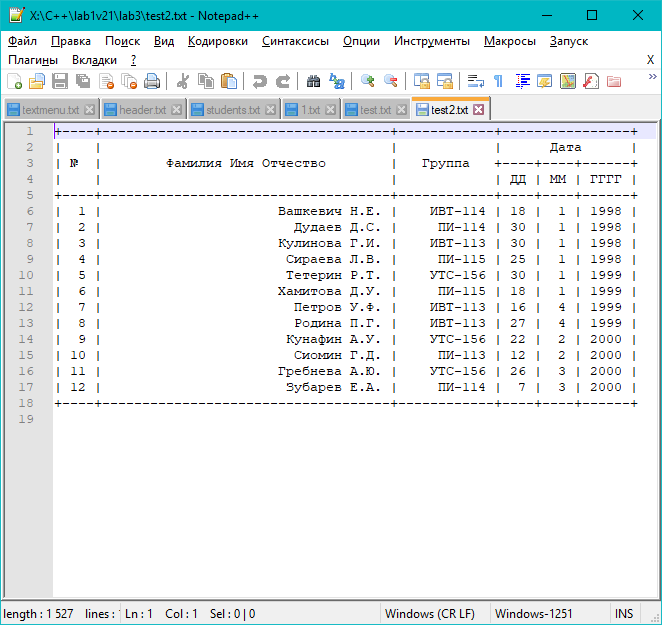


Рис. . Результат вывода массива в файл

**ВЫВОД:** получили навыки работы с массивами структур, добавления и удаления записей в этих структурах, поиск необходимых данных по заданным параметрам.