**Федеральное агентство по образованию**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Институт компьютерных наук и технологий**

**Кафедра систем и технологий управления**

**Курсовая работа**

**по теме**

**«Динамические структуры данных»**

**Курс «Программирование и основы алгоритмизации»**

**Студент гр. 23503/2 Петров И.В.**

**Преподаватель Ковалевский В.Э.**

**Санкт- Петербург**

**2016**

**Задание на выполнение курсовой работы**

1. Тема работы: Динамические структуры данных. Музыка.

2. Создать программу с консольным интерфейсом для добавления удаления, сортировки и вывода исходных данных. Программа сначала должна считывать исходные данные из файлов. Также реализовать построение списков песен и групп.

3. Исходные данные: 3 текстовых файла:

* “Звезды.txt” – список существующых оценок: номер оценки, название.
* “Альбомы.txt ” – список альбомов: номер оценки, номер альбома, название.
* “Композиции.txt” – список песен и групп: номер альбома, номер песни, название композиции, группа.

4. Требование к программе: Необходимо использовать динамические структуры

5. Содержание работы:

1. Техническое задание.
2. Описание структуры программы. Выбор типа динамической структуры и обоснование. Требования к файлам исходных данных. Требования к интерфейсу.
3. Описание функций. Тестирование.
4. Заключение (выводы о работе программы, ее универсальности и о возможностях ее развития и модификации)
5. Приложение. Код программы.

Задание выдано \_\_\_\_\_\_\_\_\_\_\_ преподаватель В.Э. Ковалевский

# Задание принято к исполнению \_\_\_\_\_\_\_\_\_\_\_ студент О.Г. Масликова

# Введение

Цель курсовой работы: знакомство с динамическими структурами данных. Поиск информации в связанных файлах. Создание простейшей плоской базы данных и системы управления. Моя тема: фильмотека.

Динамические структуры данных – это структуры данных, память под которые выделяется и освобождается по мере необходимости. Динамические структуры данных в процессе существования в памяти могут изменять не только число составляющих их элементов, но и характер связей между элементами. При этом не учитывается изменение содержимого самих элементов данных. Такая особенность динамических структур, как непостоянство их размера и характера отношений между элементами, приводит к тому, что на этапе создания машинного кода программа-компилятор не может выделить для всей структуры в целом участок памяти фиксированного размера, а также не может сопоставить с отдельными компонентами структуры конкретные адреса. Для решения проблемы адресации динамических структур данных используется метод, называемый динамическим распределением памяти, то есть память под отдельные элементы выделяется в момент, когда они "начинают существовать" в процессе выполнения программы, а не во время компиляции. Компилятор в этом случае выделяет фиксированный объем памяти для хранения адреса динамически размещаемого элемента, а не самого элемента.

**Описание структуры программы**

Программа включает в себя:

1. main.cpp – файл реализации интерфейса.
2. Constructors.cpp – файл содержит все конструкторы
3. input functions.cpp – функции считывания данных.
4. add\_functions.cрр – функции добавления элементов
5. output functions.срр – функции вывода данных на консоль.
6. other functions.срр – функции сортировки и удаления.
7. Sourse.cpp – функции ядра программы.
8. main information.h - заголовочный файл.
9. Functions.h – список всех функций
10. Три файла с данными.

Хранение записей реализовано в данной программе с помощью структур struct и односвязных списков. В курсовой работе задействованы только те инструменты, которые были пройдены на курсе «Программирование и основы алгоритмизации» за 3 семестр.

Интерфейс позволяет совершать добавление и удаление записей, сортировку списков по заданной области, а также позволяет менять группу указанной песни

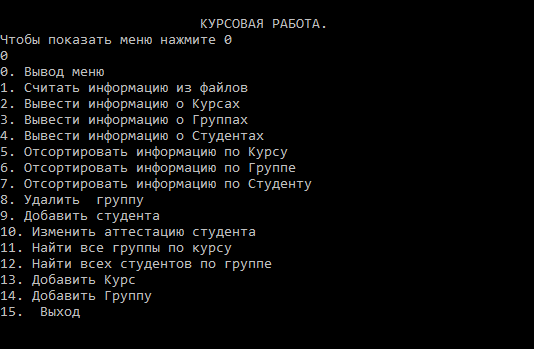
**Техническое задание. Описание работы меню программы**

Программа содержит главное меню, которое состоит из 16 пунктов.

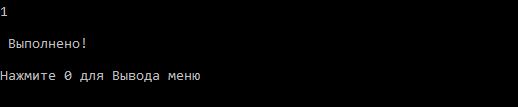
1. Вывод меню
2. Считать информацию из файлов
3. Вывести информацию о звездах
4. Вывести информацию о альбомах
5. Вывести информацию о песнях
6. Отсортировать информацию по звездам
7. Отсортировать информацию по альбомам
8. Отсортировать информацию по песням
9. Удалить альбом
10. Добавить песню
11. Изменить группу песни
12. Найти все альбомы по оценке
13. Найти все песни по альбому
14. Добавить Оценку
15. Добавить Альбом
16. Выход

**Отображение работы программы на консоль (примеры работы пунктов)**

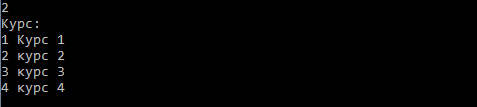
Пункт 0. «0. Вывод меню» - на консоли отображается меню.



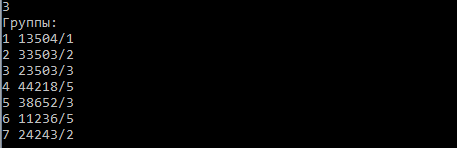
Пункт 1. «Считать информацию из файлов» - считывается информация из файла.



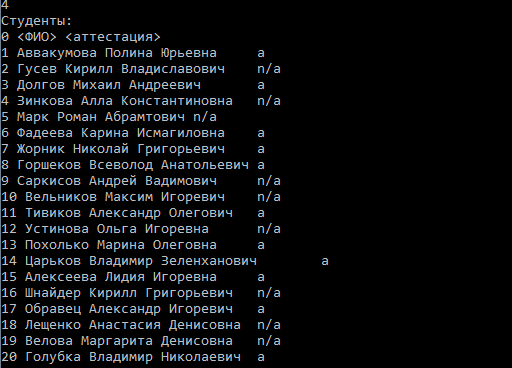
Пункт 2. «Вывести информацию о Курсах» - на консоли отображаются все курсы и их номера.



Пункт 3. «Вывести информацию о Группах» - на консоли отображаются группы, считанные с файла.

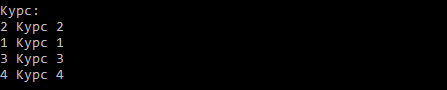


Пункт 4. «Вывести информацию о Студентах» на консоли отображается информация о студентах и аттестации

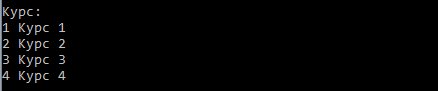


Пункт 5. «Отсортировать информацию по Курсу» - сортируются по алфавиту.

Было:

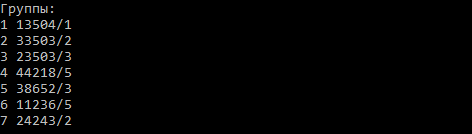


Стало:

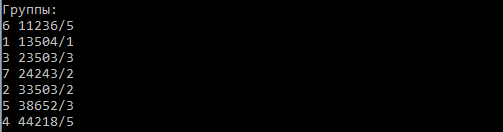


Пункт 6. «Отсортировать информацию по Группе» - сортируются по номеру

Было:

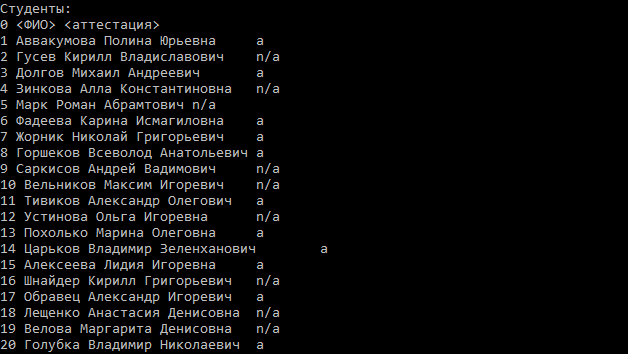


Стало:

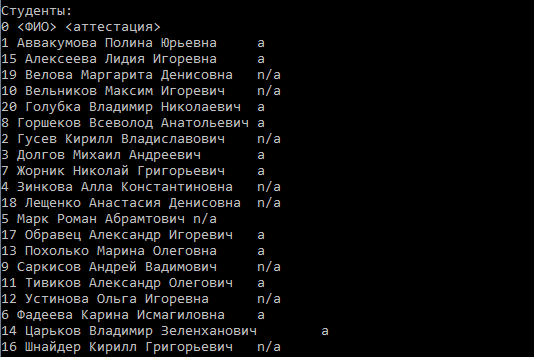


Пункт 7. «Отсортировать информацию по Студенту» - сортируются по алфавиту

Было:



Стало:



Пункт 8. «Удалить Группу» - пользователь вводит номер группы, которую хочет удалить, информация удаляется из списка групп и студентов

Происходит обращение в двум спискам: Группы и Студенты по обобщенному алгоритму:

Проверка группы:

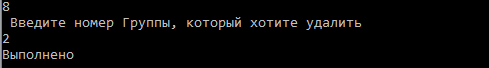
1) это введенный номер группы?

2) если да, удалить, если нет следующий ->1

Проверка студентов:

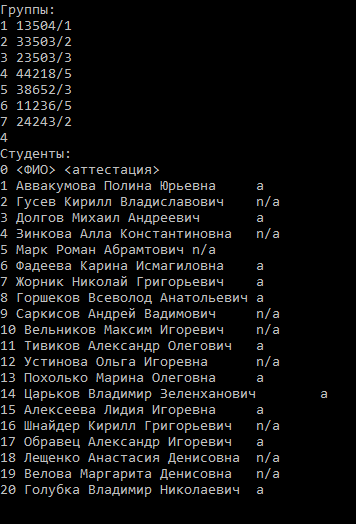
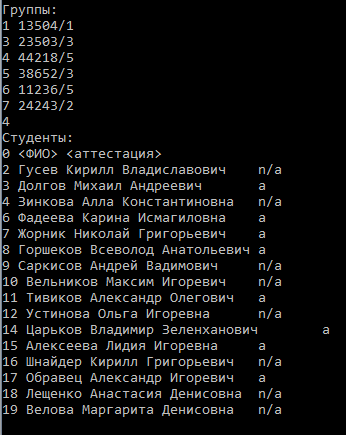
1) это введенный номер группы?

2) если да, удалить, если нет следующий ->1

.

Пример:

Было: Стало:



Пункт 9. «Добавить студента» - пользователь вводит № группы, куда необходимо добавить студента затем ФИО студент, после он добавляется в конец списка студентов.

if (list)

{addСтудент(list->next, data, list)} // считавание данных

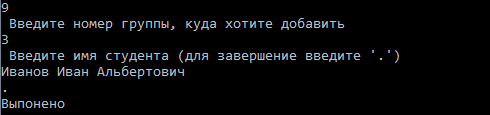
else if ((prevData != NULL) && (list == NULL))//проверка

{ data.номер\_студента = prevData->data.номер\_студента + 1; //новый номер=последний+1

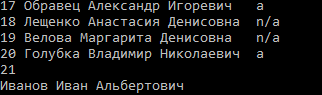
list = new List\_of\_студентов\_и\_аттест(data, NULL, prevData)

}

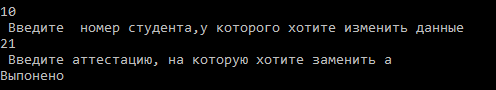
else list = new List\_of\_студентов\_и\_аттест(data, NULL, prevData);



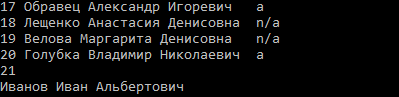
В списке студентов:



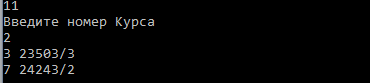
Пункт 10. «Изменить аттестацию студента» - пользователь вводит номер студента и новую аттестацию.



В списке студентов:



Пункт 11. «Найти все группы по курсу» - пользователь вводит номер курса, затем выводятся все группы, которые есть на этом курсе.



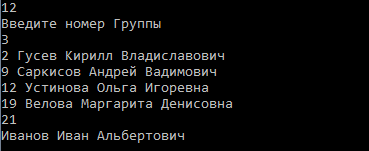
{if ((currentStore1->data.номер\_Курс == номер\_курса) && (номер\_курса == currentStore2->data.номер\_курса)) //находим введенный курс

cout << currentStore2->data.номер\_группы << currentStore2->data.группа << endl;

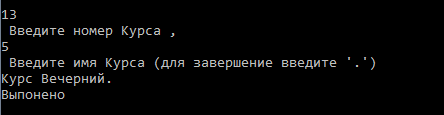
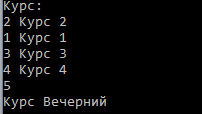
// вывод на консоль всех групп этого курса

}

Пункт 12. «Найти всех студентов по группе» - пользователь вводит номер группы, затем выводятся все студенты, которые есть в этой группе.

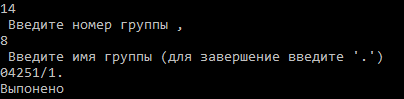
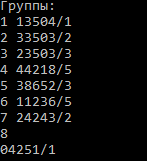


Пункт 13. «Добавить курс» - пользователем вводится номер курса, затем требуется ввести имя курса.

Пункт 14. «Добавить группу» - пользователем вводится номер курса, затем требуется ввести имя новой группы.

Выполняется аналогично Добавлению студента

Пункт 15. «Завершить работу программы» - завершение работы программы.

**Заключение**

На выполнение курсовой работы было потрачено 4 недели. Все поставленные условия и задачи реализованы. Программа отлажена. В ходе выполнения данной курсовой работы была написана программа для работы с простейшей базой данных. Реализована поддержка трех файлов с данными. Разработаны собственные методы и функции для решения поставленных задач. Была изучена вся учебная литература из списка литературы, а также дополнительная литература, необходимая для решения поставленных задач.

Изучены основы работы с динамическими структурами и простейшими базами данных. Были более подробно изучены типы динамических структур. Опробованы и использованы новые, ранее незнакомые библиотеки и функции.

По факту выполненной работы, были получены фундаментальные знания в области разработки программ для работы с базами данных и динамическими структурами.

1. **Приложение**
   1. **Приложение main.cpp**

#include <locale.h>

include <string>

#include "main information.h"

#include "functions.h"

#include <windows.h>

using namespace std;

void Menu()

{

cout << "0. Вывод меню\n";

cout << "1. Считать информацию из файлов\n";

cout << "2. Вывести информацию о Курсах\n";

cout << "3. Вывести информацию о Группах\n";

cout << "4. Вывести информацию о Студентах\n";

cout << "5. Отсортировать информацию по Курсу\n";

cout << "6. Отсортировать информацию по Группе\n";

cout << "7. Отсортировать информацию по Студенту\n";

cout << "8. Удалить группу\n";

cout << "9. Добавить студента\n";

cout << "10. Изменить аттестацию студента\n";

cout << "11. Найти все группы по курсу\n";

cout << "12. Найти всех студентов по группе\n";

cout << "13. Добавить Курс\n";

cout << "14. Добавить Группу\n";

cout << "15. Выход\n";

}

int main() {

setlocale(LC\_ALL, "Russian");

setlocale(LC\_ALL, "rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

string newАттестация = "";

cout <<"\n\t\t\t КУРСОВАЯ РАБОТА. \t\t\t\n";

cout << "Чтобы показать меню нажмите 0\n";

void Menu();

List\_of\_курс\* курс = NULL;

List\_of\_групп\* группы = NULL;

List\_of\_студентов\_и\_аттест\* студенты = NULL;

информация\_о\_студеите\_и\_аттест new\_студенты;

информация\_о\_Курсе new\_Nкурса;

информация\_о\_группе new\_Nгруппы;

int номер\_студента;

int номер\_группы;

int номер\_курс;

int номер\_изм\_студента;

int del\_группу;

int s\_punkt = 1;

int nomer\_menu;

char c = ' ';

while (s\_punkt != 0)

{

cin >> nomer\_menu;

switch (nomer\_menu)

{

case 0:

Menu();

break;

case 1:

курс = getКурсListFile("Курсы.txt");

группы = getГруппыListFile("Группы.txt");

студенты = getСтудентыListFile("Студенты.txt");

cout << " \n Выполнено! \n" << endl;

cout << "Нажмите 0 для Вывода меню" << endl;

break;

case 2:

cout << "Курс:" << endl;

printКурс(курс);

break;

case 3:

cout << "Группы:" << endl;

printГруппы(группы);

break;

case 4:

cout << "Студенты:" << endl;

printСтуденты(студенты);

break;

case 5:

sortКурс(курс);

cout << "Выполнено" << endl;

break;

case 6:

sortГруппы(группы);

cout << "Выполнено" << endl;

break;

case 7:

sortСтуденты(студенты);

cout << "Выполнено" << endl;

break;

case 8:

cout << " Введите номер Группы, который хотите удалить" << endl;

scanf\_s("%d", &del\_группу);

deleteГруппу(группы,студенты, del\_группу);

cout << "Выполнено " << endl;

break;

case 9:

cout << " Введите номер группы, куда хотите добавить" << endl;

cin >> new\_студенты.номергруппы;

cout << " Введите имя студента (для завершение введите '.')" << endl;

getline(cin, new\_студенты.студент,'.');

addСтудент(студенты, new\_студенты);

cout << "Выполнено" << endl;

break;

case 10:

cout << " Введитеномер студента,у которого хотите изменить данные"<< endl;

cin >> номер\_изм\_студента;

cout << " Введите аттестацию, на которую хотите заменить" ;

cin >> newАттестация;

changeАТТСтудента(группы, студенты, номер\_изм\_студента,newАттестация);

cout << "Выпонено" << endl;

break;

case 11:

cout << "Введите номер Курса " << endl;

cin >> номер\_курс;

printГруппыFromGroup(курс, группы, номер\_курс);

break;

case 12:

cout << "Введите номер Группы " << endl;

cin >> номер\_студента;

printСтудентыFromAlbum(группы, студенты, номер\_студента);

break;

case 13:

cout << " Введите номер Курса ," << endl;

cin >> new\_Nкурса.номер\_Курс;

cout << " Введите имя Курса (для завершение введите '.')" << endl;

getline(cin, new\_Nкурса.курс, '.');

addКурс(курс, new\_Nкурса);

cout << "Выпонено" << endl;

break;

case 14:

cout << " Введите номер группы ," << endl;

cin >> new\_Nгруппы.номер\_группы;

cout << " Введите имя группы (для завершение введите '.')" << endl;

getline(cin, new\_Nгруппы.группа, '.');

addГруппы(группы, new\_Nгруппы);

cout << "Выпонено" << endl;

break;

case 15:

s\_punkt = 0;

break;

default: cout << "Ошибка! Правильно введите пункт меню";

break;

}

}

return 0;

}

* 1. **Приложение Constructors.cpp**

#include "main information.h"

//Конструктор информация\_о\_курсе без параметров

информация\_о\_Курсе::информация\_о\_Курсе()

{

курс = "";

номер\_Курс = 0;

}

//Конструктор информация\_о\_курсе

информация\_о\_Курсе::информация\_о\_Курсе(string курсNew, int номер\_курсNew)

{

курс = курсNew;

номер\_Курс = номер\_курсNew;

}

информация\_о\_группе::информация\_о\_группе()

{

группа = "";

номер\_группы = 0;

номер\_курса = 0;

}

информация\_о\_группе::информация\_о\_группе(string группаNew, int номер\_группыNew, intномер\_курсNew)

группа = группаNew;

номер\_группы = номер\_группыNew;

номер\_курса = номер\_курсNew;

}

информация\_о\_студеите\_и\_аттест::информация\_о\_студеите\_и\_аттест()

{

студент = "";

аттестация = "";

номер\_студента = 0;

номергруппы = 0;

}

информация\_о\_студеите\_и\_аттест::информация\_о\_студеите\_и\_аттест(string студентNew, string аттестацияNew, int номер\_студентаNew, int номергруппыNew)

{

аттестация = аттестацияNew;

студент = студентNew;

номер\_студента = номер\_студентаNew;

номергруппы = номергруппыNew;

};

//Конструктор без параметров

List\_of\_курс::List\_of\_курс()

{

data = информация\_о\_Курсе();

next = NULL;

prev = NULL;

}

List\_of\_курс::List\_of\_курс(информация\_о\_Курсе dataNew, List\_of\_курс\* nextNew,List\_of\_курс\* prevNew)

{

data = dataNew;

next = nextNew;

prev = prevNew;

}

List\_of\_групп::List\_of\_групп()

{

data = информация\_о\_группе();

next = NULL;

prev = NULL;

}

List\_of\_групп::List\_of\_групп(информация\_о\_группе dataNew, List\_of\_групп\* nextNew,List\_of\_групп\* prevNew)

{

data = dataNew;

next = nextNew;

prev = prevNew;

}

List\_of\_студентов\_и\_аттест::List\_of\_студентов\_и\_аттест()

{

data = информация\_о\_студеите\_и\_аттест();

next = NULL;

prev = NULL;

}

List\_of\_студентов\_и\_аттест::List\_of\_студентов\_и\_аттест(информация\_о\_студеите\_и\_аттест dataNew, List\_of\_студентов\_и\_аттест\* nextNew, List\_of\_студентов\_и\_аттест\* prevNew)

{

data = dataNew;

next = nextNew;

prev = prevNew;

}

* 1. **Приложение input functions.cpp**

#include "main information.h"

#include "functions.h"

//Считывание информации о курсе из файла

информация\_о\_Курсе getКурсInformationFile(FILE\* file)

{

информация\_о\_Курсе info\_курса;

fscanf\_s(file, "%d", &info\_курса.номер\_Курс);

readToChar(file, '"');

info\_курса.курс = readToChar(file, '"');

fgetc(file);

return info\_курса;

}

//Считывание информации о групе из файла

информация\_о\_группе getГруппаInformationFile(FILE\* file)

{

информация\_о\_группе info\_группы;

fscanf\_s(file, "%d", &info\_группы.номер\_курса);

readToChar(file, '.');

fscanf\_s(file, "%d", &info\_группы.номер\_группы);

readToChar(file, '.');

info\_группы.группа = readToChar(file, '.');

fgetc(file);

return info\_группы;

}

информация\_о\_студеите\_и\_аттест getСтудентInformationFile(FILE\* file)

{

информация\_о\_студеите\_и\_аттест info\_студент;

fscanf\_s(file, "%d", &info\_студент.номергруппы);

readToChar(file, '.');

fscanf\_s(file, "%d", &info\_студент.номер\_студента);

readToChar(file, '.');

info\_студент.студент = readToChar(file, '.');

fgetc(file);

info\_студент.аттестация = readToChar(file, '.');

fgetc(file);

return info\_студент;

}

//Считывает из файла все до указанного символа, пихает в строку и возвращает

string readToChar(FILE\* file, char endChar)

{

string str = "";

if (file == NULL) return NULL;

while (true)

{

char c;

fscanf\_s(file, "%c", &c);

if (c == endChar) break;

str += c;

}

return str;

}

//Считывание списка групп из файла

List\_of\_курс\* getКурсListFile(char\* fileName)

{

FILE\* file = NULL;

fopen\_s(&file, fileName, "r");

if (file == NULL) return NULL;

List\_of\_курс\* list = NULL;

fgetc(file);

while (!feof(file))

{

fseek(file, -1, SEEK\_CUR);

addКурс(list,getКурсInformationFile(file));

fgetc(file);

}

fclose(file);

return list;

}

List\_of\_групп\* getГруппыListFile(char\* fileName)

{

FILE\* file = NULL;

fopen\_s(&file, fileName, "r");

if (file == NULL) return NULL;

List\_of\_групп\* list = NULL;

fgetc(file);

while (!feof(file))

{

fseek(file, -1, SEEK\_CUR);

addГруппы(list, getГруппаInformationFile(file));

fgetc(file);

}

fclose(file);

return list;

}

List\_of\_студентов\_и\_аттест\* getСтудентыListFile(char\* fileName)

{

FILE\* file = NULL;

fopen\_s(&file, fileName, "r");

if (file == NULL) return NULL;

List\_of\_студентов\_и\_аттест\* list = NULL;

fgetc(file);

while (!feof(file))

{

fseek(file, -1, SEEK\_CUR);

addСтудент(list, getСтудентInformationFile(file));

fgetc(file);

}

fclose(file);

return list;}

* 1. **Приложение add\_functions.cрр**

#include "main information.h"

// Добавляет в конец списка курсов

void addКурс(List\_of\_курс\* &list, информация\_о\_Курсе &data, List\_of\_курс\* prevData)

{

if (list)

{

addКурс(list->next, data, list);

}

else

list = new List\_of\_курс(data, NULL, prevData);

}

// Добавляет в конец списка группы

void addГруппы(List\_of\_групп\* &list, информация\_о\_группе &data, List\_of\_групп\*prevData)

{

if (list)

{

addГруппы(list->next, data, list);

}

else

list = new List\_of\_групп(data, NULL, prevData);

}

// Добавляет в конец списка Студента!!!!!!!!!!!!!!!!!!!!!!!!!!

void addСтудент(List\_of\_студентов\_и\_аттест\* &list, информация\_о\_студеите\_и\_аттест &data, List\_of\_студентов\_и\_аттест\* prevData)

{

if (list)

{

addСтудент(list->next, data, list);

}

else if ((prevData != NULL) && (list == NULL))

{

data.номер\_студента = prevData->data.номер\_студента + 1;

list = new List\_of\_студентов\_и\_аттест(data, NULL, prevData);

}

else list = new List\_of\_студентов\_и\_аттест(data, NULL, prevData);

}

* 1. **Приложение output functions.срр**

#include "main information.h"

//Вывод списка курсов)

void printКурс(List\_of\_курс\* курсы)

{

if (курсы)

{ cout << курсы->data.номер\_Курс << курсы->data.курс << endl;

printКурс(курсы->next);

}

}

//Вывод списка групп)

void printГруппы(List\_of\_групп\* группы)

{

if (группы)

{ cout <<группы->data.номер\_группы << группы->data.группа << endl;

printГруппы(группы->next);

}

}

//Вывод списка студентов)

void printСтуденты(List\_of\_студентов\_и\_аттест\* студент)

{

if (студент)

{ cout << студент->data.номер\_студента << студент->data.студент << "\t" << студент->data.аттестация << endl;

printСтуденты(студент->next);

}

}

//Вывод списка студентов по группе)

void printСтудентыFromAlbum(List\_of\_групп\* группы, List\_of\_студентов\_и\_аттест\* студент,int номергруппы)

{for (List\_of\_групп\* currentStore1 = группы; currentStore1 != NULL; currentStore1 = currentStore1->next)

{

for (List\_of\_студентов\_и\_аттест\* currentStore2 = студент; currentStore2!= NULL; currentStore2 = currentStore2->next)

{if ((currentStore1->data.номер\_группы == номергруппы) && (номергруппы ==currentStore2->data.номергруппы ))

cout << currentStore2->data.номер\_студента << currentStore2->data.студент << endl;

}

}}

//Вывод списка групп по курсу

void printГруппыFromGroup(List\_of\_курс\* курс, List\_of\_групп\* группы, int номер\_курса)

{

for (List\_of\_курс\* currentStore1 = курс; currentStore1 != NULL; currentStore1 = currentStore1->next)

{

for (List\_of\_групп\* currentStore2 = группы; currentStore2 != NULL; currentStore2 = currentStore2->next)

{if ((currentStore1->data.номер\_Курс == номер\_курса) && (номер\_курса == currentStore2->data.номер\_курса))

cout << currentStore2->data.номер\_группы<<currentStore2->data.группа<<endl;

}

}}

* 1. **Приложение other functions.срр**

#include "main information.h"

// Сортируем курсы)

void sortКурс(List\_of\_курс\* курсы)

{

if (курсы)

{

List\_of\_курс\* next = курсы->next;

while (next)

{ if (next->data.курс < курсы->data.курс)

{

информация\_о\_Курсе tempData = next->data;

next->data = курсы->data;

курсы->data = tempData;

}

next = next->next;

}

sortКурс(курсы->next);

}

}

// Сортируем группы

void sortГруппы(List\_of\_групп\* группы)

{

if (группы)

{

List\_of\_групп\* next = группы->next;

while (next)

{

if (next->data.группа < группы->data.группа)

{

информация\_о\_группе tempData = next->data;

next->data = группы->data;

группы->data = tempData;

}

next = next->next;

}

sortГруппы(группы->next);

}

}

// Сортируем студентв

void sortСтуденты(List\_of\_студентов\_и\_аттест\* студенты)

{

if (студенты)

{

List\_of\_студентов\_и\_аттест\* next = студенты->next;

while (next)

{

if (next->data.студент < студенты->data.студент)

{

информация\_о\_студеите\_и\_аттест tempData = next->data;

next->data = студенты->data;

студенты->data = tempData;

}

next = next->next;

}

sortСтуденты(студенты->next);

}

}

// Изменить аттестацию студента)

void changeАТТСтудента(List\_of\_групп\* группы,List\_of\_студентов\_и\_аттест\* студент, int номер\_изм\_студента, string &аттестацияNew)

{

if (студент)

if (студент->data.номер\_студента == номер\_изм\_студента)

студент->data.аттестация = аттестацияNew;

else

changeАТТСтудента(группы,студент->next, номер\_изм\_студента, аттестацияNew);

}

// функция удаления группы)

void deleteГруппу(List\_of\_групп\* группы, List\_of\_студентов\_и\_аттест\* студенты, int &delete\_группу)

{

if (группы) {

if (группы->data.номер\_группы == delete\_группу)

{

if (группы->prev)

{

группы->prev->next = группы->next;

}

if (группы->next)

{

группы->next->prev = группы->prev;

}

}

else deleteГруппу(группы->next, студенты, delete\_группу);

}

if (студенты)

{

if (студенты->data.номергруппы == delete\_группу)/////!!!!!!!!!!!!!!!!

{

if (студенты->prev)

{

студенты->prev->next = студенты->next;

}

if (студенты->next)

{

студенты->next->prev = студенты->prev;

}

}

else deleteГруппу(группы, студенты->next, delete\_группу);

}

}

* 1. **Приложение main information.h**

#include <string>

#include <iostream>

#pragma once

using namespace std;

// Информация о курсе

struct информация\_о\_Курсе

{

string курс;

int номер\_Курс;

информация\_о\_Курсе();

информация\_о\_Курсе(string курсNew, int номер\_курсаNew);

};

//// Информация о группе

struct информация\_о\_группе

{

int номер\_курса;

int номер\_группы;

string группа;

информация\_о\_группе();

информация\_о\_группе(string группаNew, int номер\_группыNew, int номер\_курсаNew);

};

// Информация о студенте

struct информация\_о\_студеите\_и\_аттест

{

string студент;

int номергруппы;

int номер\_студента;

string аттестация;

информация\_о\_студеите\_и\_аттест();

информация\_о\_студеите\_и\_аттест(string студентNew, string аттестацияNew, intномер\_студентаNew, int номергруппыNew);

}

//// Список курсов

struct List\_of\_курс

{

информация\_о\_Курсе data;

List\_of\_курс\* next;

List\_of\_курс\* prev;

List\_of\_курс();

List\_of\_курс(информация\_о\_Курсе dataNew, List\_of\_курс\* nextNew, List\_of\_курс\* prevNew);

};

//Список групп

struct List\_of\_групп

{

информация\_о\_группе data;

List\_of\_групп\* next;

List\_of\_групп\* prev;

List\_of\_групп();

List\_of\_групп(информация\_о\_группе dataNew, List\_of\_групп\* nextNew, List\_of\_групп\* prevNew);

};

//Список студентов

struct List\_of\_студентов\_и\_аттест

{

информация\_о\_студеите\_и\_аттест data;

List\_of\_студентов\_и\_аттест\* next;

List\_of\_студентов\_и\_аттест\* prev;

List\_of\_студентов\_и\_аттест();

List\_of\_студентов\_и\_аттест(информация\_о\_студеите\_и\_аттест dataNew,List\_of\_студентов\_и\_аттест\* nextNew, List\_of\_студентов\_и\_аттест\* prevNew);

};

* 1. **Приложение Functions.h**

#pragma once

void addКурс(List\_of\_курс\* &list, информация\_о\_Курсе &data, List\_of\_курс\* prev =NULL); //добавляем курс

void addГруппы(List\_of\_групп\* &list, информация\_о\_группе &data, List\_of\_групп\* prevData = NULL); //группу

void addСтудент(List\_of\_студентов\_и\_аттест\* &list, информация\_о\_студеите\_и\_аттест &data, List\_of\_студентов\_и\_аттест\* prevData = NULL); //студента

информация\_о\_Курсе getКурсInformationFile(FILE\* file); // Информация о курсах

информация\_о\_группе getГруппаInformationFile(FILE\* file); // о группе

информация\_о\_студеите\_и\_аттест getСтудентInformationFile(FILE\* file);// о студентах

List\_of\_курс\* getКурсListFile(char\* fileName); // список курсов

List\_of\_групп\* getГруппыListFile(char\* fileName); // групп

List\_of\_студентов\_и\_аттест\* getСтудентыListFile(char\* fileName); // студентов

void printКурс(List\_of\_курс\* курсы); //функция печати на консоль курса

void printГруппы(List\_of\_групп\* группы); // группы

void printСтуденты(List\_of\_студентов\_и\_аттест\* студенты); // студентов

void sortКурс(List\_of\_курс\* курс); // сортировка Курсов по алфавиту

void sortГруппы(List\_of\_групп\* группы); // по группам

void sortСтуденты(List\_of\_студентов\_и\_аттест\* студенты); // по студенту

void printСтудентыFromAlbum(List\_of\_групп\* группа, List\_of\_студентов\_и\_аттест\* студенты, int номергруппы); // печать студентов по группе

void printГруппыFromGroup(List\_of\_курс\* курс, List\_of\_групп\* группы, int номер\_курса/\*, int номергруппы\*/); // печать групп по курсу

//void printСтудентыАТТ(List\_of\_групп\* группы, List\_of\_студентов\_и\_аттест\* студенты, int номер\_группы); // печать студентов

string readToChar(FILE\* file, char endChar); // считываение до указанного символа в строку

void changeАТТСтудента(List\_of\_групп\* группы, List\_of\_студентов\_и\_аттест\* студент, int номер\_изм\_студента, string &аттестацияNew); // изменить аттестацию

void deleteГруппу(List\_of\_групп\* группы, List\_of\_студентов\_и\_аттест\* студенты, int &delete\_группу); // удалить группу