**Курсовая работа**

**по курсу “Программирование и основы алгоритмизации”**

**Динамические структуры данных**

Выполнил

Студент группы 23503/1 <\_\_\_\_\_\_ > Г. И. Гуцол

Преподаватель <\_\_\_\_\_\_> Л.В. Иванова

**Задание на выполнение курсовой работы**

Студент группы 23503/1 Г. И. Гуцол

Курс: программирование и основы алгоритмизации

1. Тема работы:

Динамические структуры данных. Расписание занятий в аудиториях, загрузка аудиторий. С учетом вместимости аудитории и количества реально присутствующих на дату и время.

1. Создать программу с оконным интерфейсом для добавления исходных данных и вывода результатов:
   1. Таблица, в которой отображены аудитории, теоретическая посещаемость, реальная посещаемость, загруженность (разница теоретической и реальной посещаемости)
   2. Сортировка пункта 2.1. по посещаемости или загруженности.
2. Исходные данные:

4 текстовых файла:

“Groups.txt” – список групп;

“Raspisanie.txt” - расписание на неделю;

“Auditoria.txt” – список аудиторий;

“Poseshaemost.txt” – посещаемость;

1. Требование к программе

Необходимо использовать динамические структуры (списки и деревья)

Ядро программы отделено от интерфейса.

1. Содержание работы:
   1. Описание предметной области.
   2. Техническое задание к программе.
   3. Разработка алгоритма и структуры данных. Описание функций.
      1. Разработка ядра
      2. Разработка интерфейса
   4. Тестирование
   5. Заключение
   6. Приложение. Код программы.



Задание выдано 10.11.16

Преподаватель \_\_\_\_\_\_\_\_Иванова Л.В.

Задание принято к исполнению 10.11.16 \_\_\_\_\_\_\_\_Гуцол Г.И.

**Описание предметной области**

Задача: расписание занятий в аудиториях, загрузка аудиторий. С учетом вместимости аудитории и количества реально присутствующих на дату и время.

Объекты предметной области: аудитория, группа, расписание и посещаемость.

Свойства объекта «аудитория»:

-Номер аудитории

-Вместимость аудитории

Свойства объекта «группа»:

-Номер группы

-Количество студентов в группе

Свойства объекта «расписание»:

-Код расписания

-Номер группы

-Номер аудитории

-День недели

-Время начала

Свойства объекта «посещаемость»:

-код расписания

-дата

-количество студентов

Ограничения на информацию:

Для объекта «аудитория»:

1. Номер аудитории состоит из числа, обозначающего корпус и числа, которое принимает значения от 100 до 599
2. Вместимость аудитории - целое число, принимает значения от 30 до 250

Для объекта «группа»:

1. Номер группы – шестизначное целое положительное число, где первая цифра – номер курса, 2 последующие номер института, 3 остальных задаются случайно и принимают значение от 0 до 5.
2. Количество студентов в группе – целое число, принимает значение от 15 до 30.

Для объекта «расписание»:

1. Код расписания – целое положительное число
2. Теоретическая посещаемость принимает сумму значений количества студентов, состоящих в группах, которые сегодня находятся в этой аудитории .
3. Время начала занятия принимает значения 8, 10, 12, 14, 16
4. День недели может принимать значения Понедельник, Вторник, Среда, Четверг, Пятница

Для объекта посещаемость:

1. Количество студентов – суммарное количество студентов на данную дату в данной аудитории по расписанию.

Пользователи должны иметь возможность решать с ее помощью следующие задачи:

1. Вносить изменения в информацию о посещаемости студентов.
2. Вносить изменения в расписание.
3. Узнавать загруженность аудитории.

**Техническое задание:**

Требуется разработать программу способную анализировать по информации о расписании, аудиториях и посещаемости, загруженность аудиторий на заданный период, выраженную в процентах.

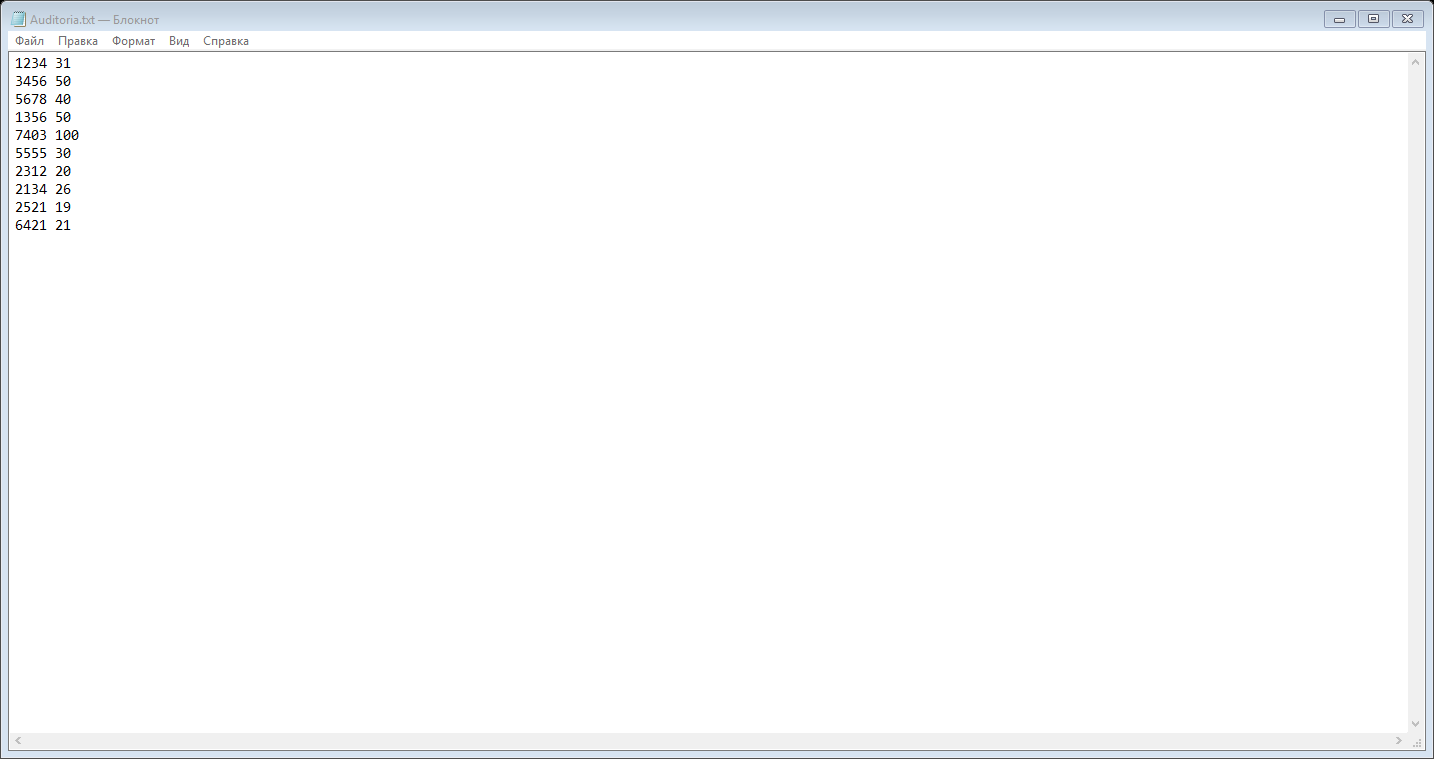
В программе предусмотреть вывод исходных данных из файла, добавления данных о расписании и посещаемости пользователем во время работы программы, вывод анализа в сводную таблицу, сортировку по значению загруженности.

**Требования к файлам исходных данных**

Файл с информацией об аудиториях (Auditoria.txt):

* первое число в строчке указывает номер аудитории
* второе число в строчке указывает вместимость аудитории
* номер аудитории специфичен и не должен повторяться в файле

Пример:



Файл с информацией о группах (Groups.txt):

* первое число в строчке указывает номер группы
* второе число в строчке указывает количество студентов в группе
* номер группы специфичен и не должен повторяться в файле

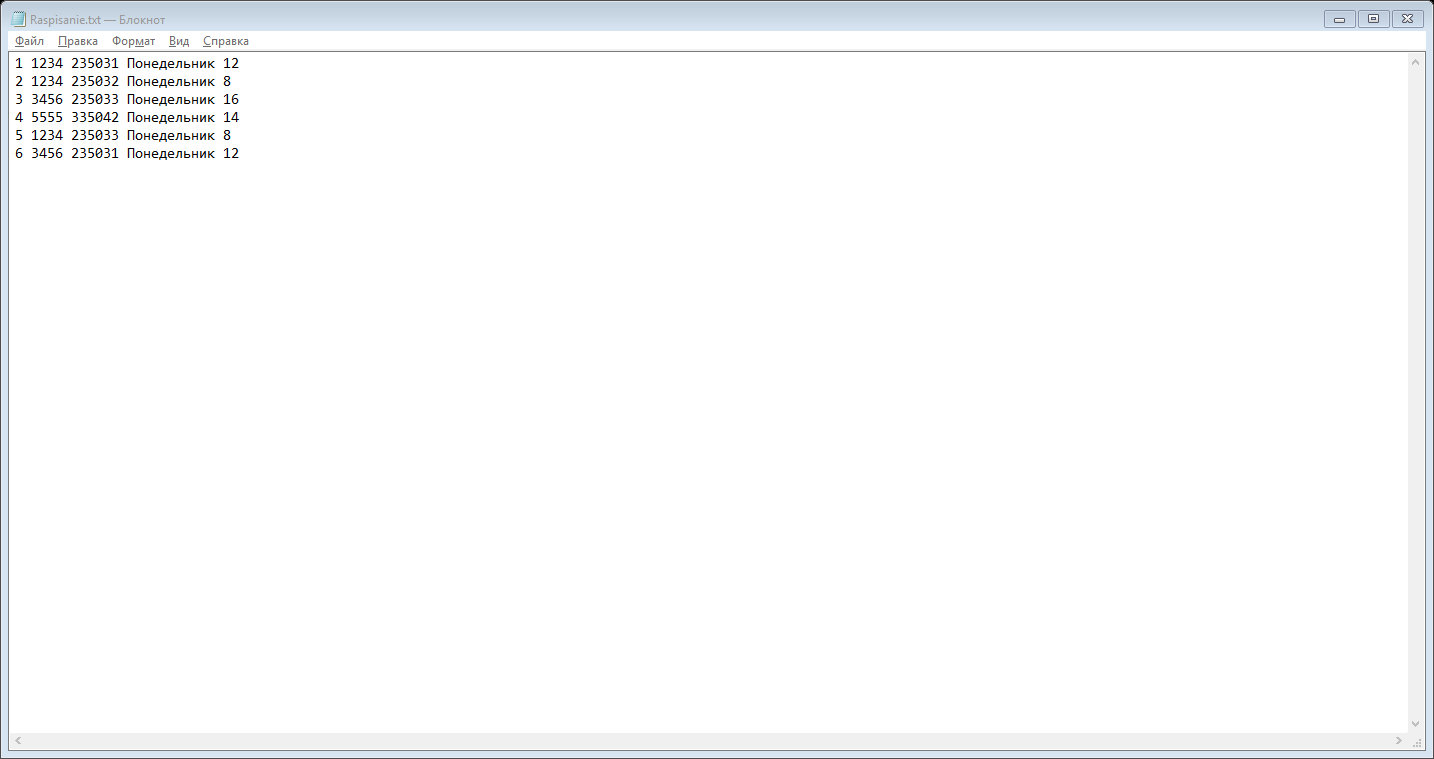
Пример:



Файл с информацией о расписании (Raspisanie.txt):

* первое число – номер расписания
* второе число – номер аудитории
* третье число – номер группы
* на четвертой позиции располагается информация о дне недели
* пятое число – время начала пар
* номер расписания специфичен и не должен повторяться в файле
* номера групп и аудиторий должны быть обозначены в файлах Groups.txt и Auditoria.txt соответственно
* день недели должен указываться с заглавной буквы

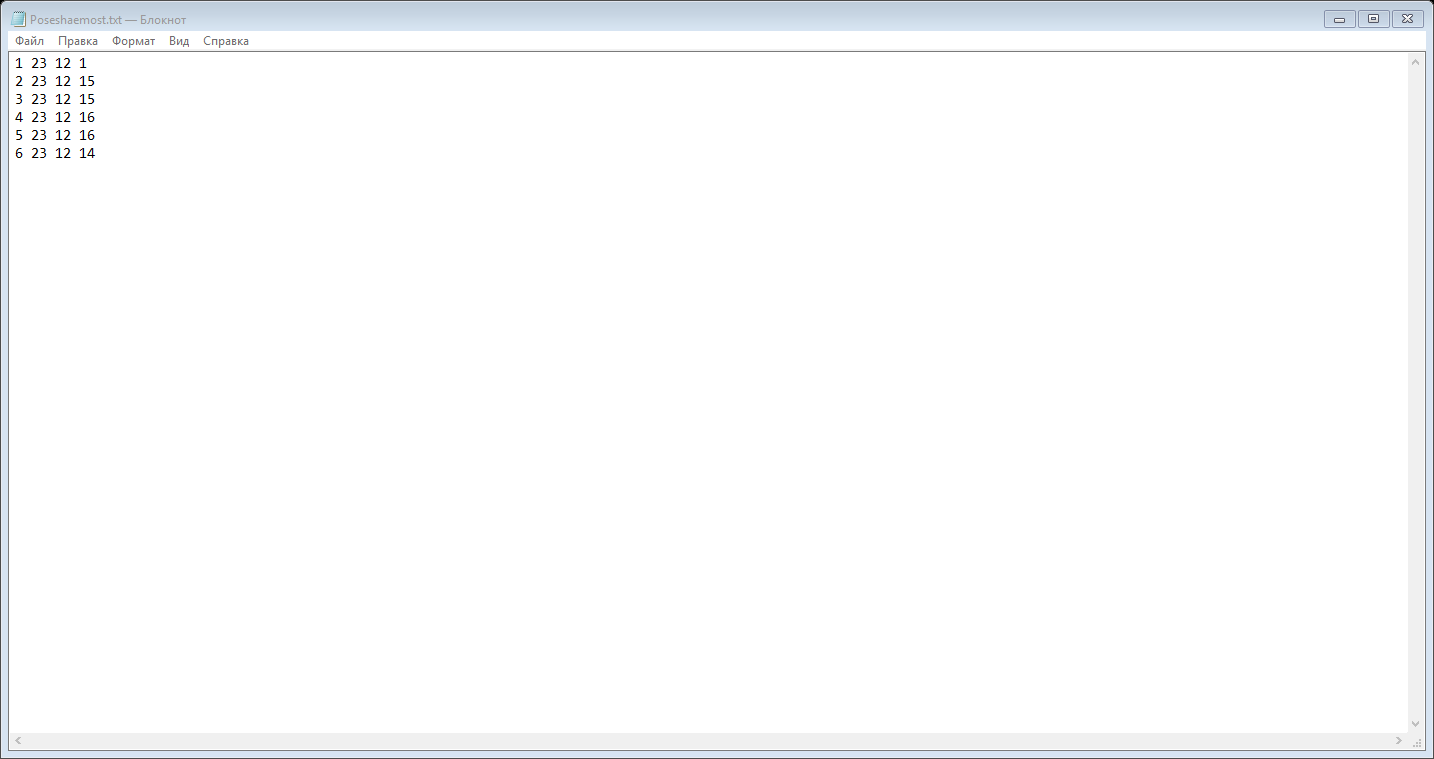
Пример:



Файл с информацией о посещении (Poseshaemost.txt):

* первое число – номер расписания, для которого составлена информация о посещаемости
* второе и третье число – дата посещаемости (число и месяц соответственно)
* четвертое число – количество студентов пришедших на занятие
* номер расписания должен присутствовать в списке расписаний

Пример:



**Требования к интерфейсу**

В интерфейсе должна присутствовать сводная таблица, куда будут добавляться данные о загруженности аудитории. Элементы позволяющие добавлять данные об информации и посещаемости и выполнять сортировку по значению загруженности.

**Ожидаемы результаты**

Входные данные:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Ожидаемые результаты:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Понедельник | |  |  |  |  |
| Аудитории\Время | 8 | 10 | 12 | 14 | 16 |
| 5678 |  |  | 72 |  |  |
| 1234 |  |  | 51 |  |  |
| 6421 |  |  | 119 |  |  |
| 2521 |  |  | 78 |  |  |
| 2134 |  |  | 65 |  |  |
| Вторник |  |  |  |  |  |
| 7403 |  |  |  | 45 |  |
| 5555 |  | 83 |  |  | 73 |
| Среда |  |  |  |  |  |
| 1234 | 35 |  |  |  |  |
| 3456 |  | 24 |  |  |  |
| 5678 |  |  | 37 |  |  |
| 1356 |  |  |  | 30 |  |
| 2312 |  |  |  |  | 50 |
| Четверг |  |  |  |  |  |
| 3456 | 39 |  |  |  |  |
| 1356 |  | 60 |  |  |  |
| 2312 |  |  | 130 |  |  |
| 2134 |  |  |  | 61 |  |
| 6421 |  |  |  |  | 71 |
| Пятница |  |  |  |  |  |
| 5555 |  | 346 |  |  |  |
|  |  |  |  |  |  |

**Разработка алгоритма и структуры данных**

**Алгоритм**

1. Ввод данных
2. Составление списков
3. Анализ данных
   1. Просмотр списка посещаемости
   2. Обращение к списку расписания
   3. Обращение к списку аудиторий
   4. Подсчет процента загруженности аудитории
4. Вывод полученного анализа.

**Структуры данных**

Данные

struct Auditory

{

int nomer\_a;

int vmestimost;

};

struct Gruppa

{

int nomer\_g;

int kolvo\_stud;

};

struct Raspisanie

{

int kod\_r;

int nomer\_g;

int nomer\_a;

char\* den=new char[50];

int time;

};

struct Posechaemost

{

int kod;

int dd;

int mm;

int kolvo;

};

struct Analyz //вспомогательная струткура

{

int nomer\_a;

int dd;

int mm;

char\* den = new char[50];

int time;

int zagr;

};

Списки:

struct s\_Auditory

{

s\_Auditory \*next;

Auditory data;

};

struct s\_Gruppa

{

s\_Gruppa \*next;

Gruppa data;

};

struct s\_Raspisanie

{

s\_Raspisanie \*next;

s\_Auditory\* link\_a;

s\_Gruppa\* link\_g;

Raspisanie data;

};

struct s\_Posechaemost

{

s\_Raspisanie\* link\_r;

s\_Posechaemost \*next;

Posechaemost data;

};

struct s\_Analyz //вспомогательный список

{

s\_Analyz\*next;

Analyz data;

};

struct Node //дерево

{

s\_Analyz\* x;

Node \*L, \*R;

};

**Разработка ядра**

**Функции инициализации списков:**

//Инициализация списков

s\_Auditory\* new\_Auditory(s\_Auditory\* Head)//список аудиторий

{

Head->next = Head;//указатель ставим на этот элемент

return Head;

}

s\_Gruppa\* new\_Gruppa(s\_Gruppa\* Head)//список групп

{

Head->next = Head;

return Head;

}

s\_Raspisanie\* new\_Raspisanie(s\_Raspisanie\* Head)//список расписаний

{

Head->next = Head;

return Head;

}

s\_Posechaemost\* new\_Posechaemost(s\_Posechaemost\* Head)//список посещаемости

{

Head->next = Head;

return Head ;

}

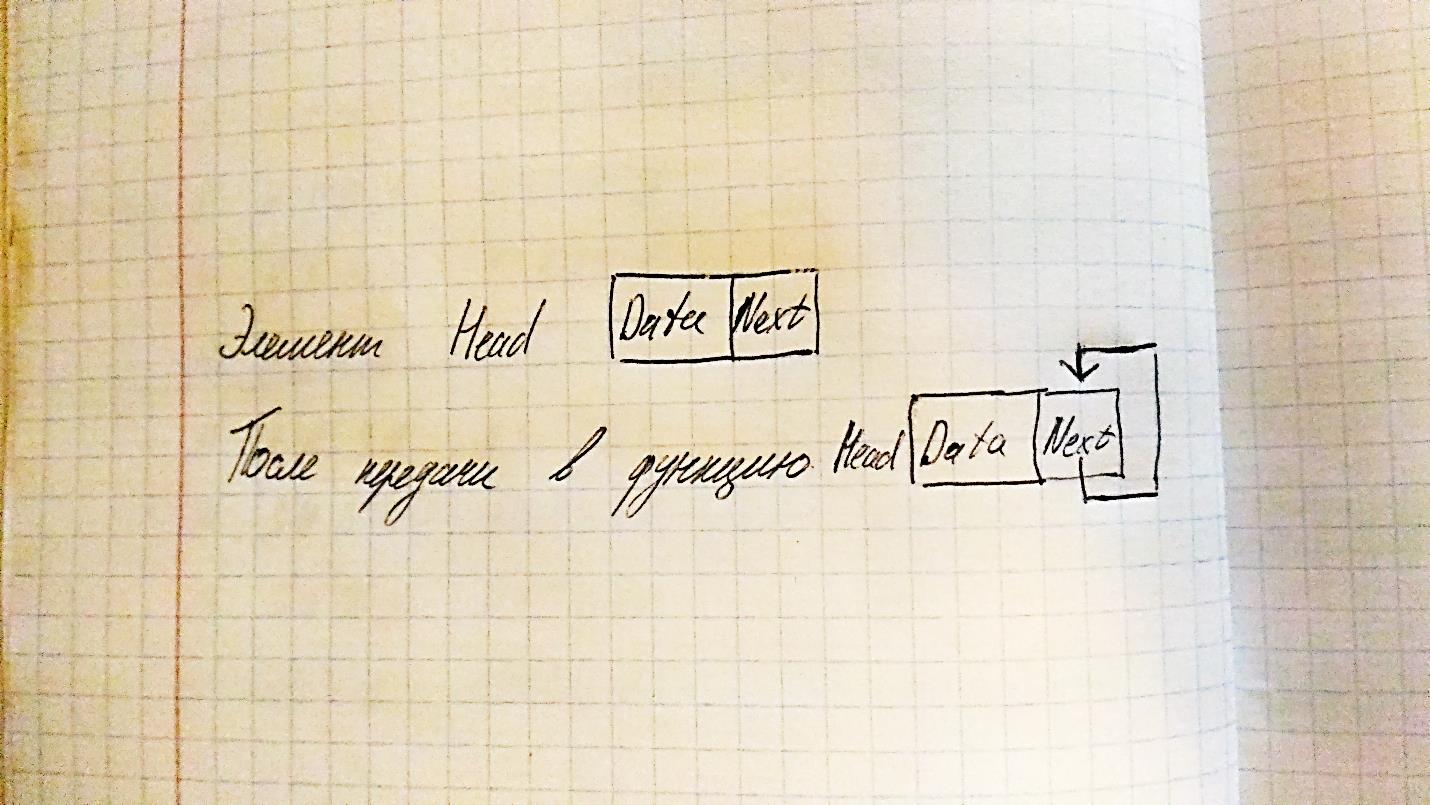
s\_Analyz\* new\_Analyz(s\_Analyz\* Head)//список анализа

{

Head->next = Head;

return Head;

}



Функции получают на вход головной элемент списка. В ходе его обработки указатель начинает указывать на самого себя. На выходе получаем головной элемент, позволяющий создать кольцевой односвязный список.

**Методы добавления одного элемента в список**:

//Добавление одного элемента

void Dob\_Auditory(s\_Auditory \*audit, Auditory data)//добавление аудитории

{

s\_Auditory\* au = new s\_Auditory;//создаем указатель на элемент списка

au->next = audit->next;//указатель на первый элемент списка

audit->next = au;//указатель пердпоследнего на последний(новый) элемент списка

au->data = data;//сопоставляем данные

}

void Dob\_Gruppa(s\_Gruppa \*grupp, Gruppa data)//добавление группы

{

s\_Gruppa\* au = new s\_Gruppa;

au->next = grupp->next;

grupp->next = au;

au->data = data;

}

void Dob\_Analyz(s\_Analyz\* anal, Analyz data)

{

s\_Analyz\* an = new s\_Analyz;

an->next = anal->next;

anal->next = an;

an->data = data;

}

s\_Raspisanie\* Dob\_Raspisanie(s\_Raspisanie\* rasp, Raspisanie data, s\_Auditory\* audit, s\_Gruppa\* grup)//добавление товара

{

s\_Raspisanie\* au = new s\_Raspisanie;

au->next = rasp->next;

rasp->next = au;

au->data = data;

s\_Auditory\* result = new s\_Auditory;

Poisk\_Auditory(audit, au->data.nomer\_a, result);

au->link\_a = result;

s\_Gruppa\* res1 = new s\_Gruppa;

Poisk\_Grup(grup, au->data.nomer\_g, res1);

au->link\_g = res1;

return rasp;

}

s\_Posechaemost\* Dob\_Posechaemost(s\_Posechaemost \*pos, Posechaemost data,s\_Raspisanie\* rasp )//добавление товара

{

s\_Posechaemost\* au = new s\_Posechaemost;

au->next = pos->next;

pos->next = au;

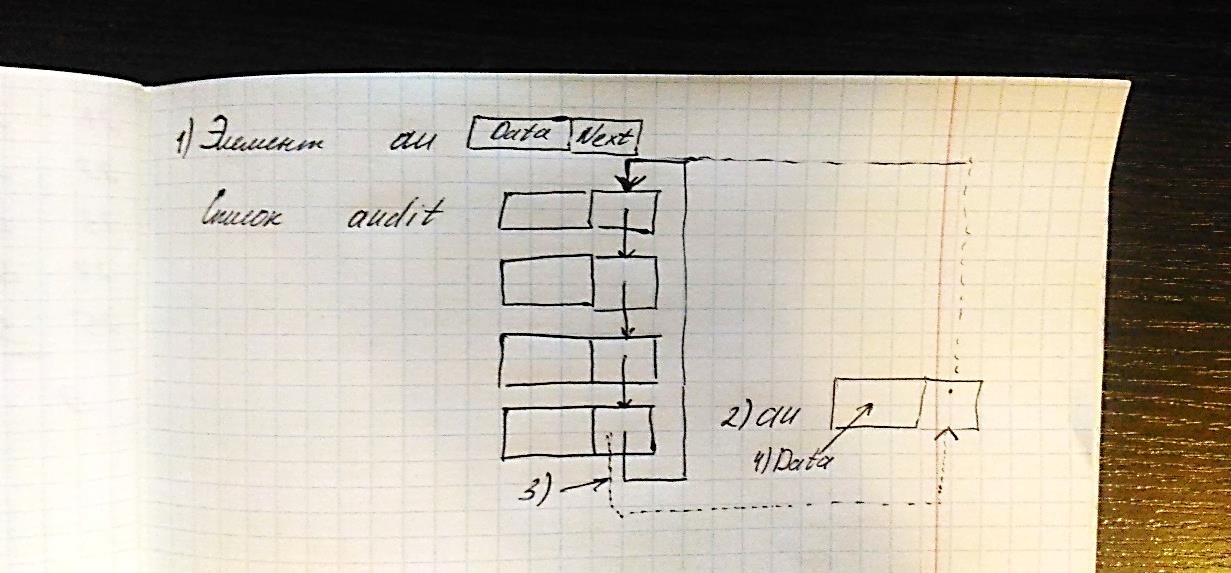
au->data = data;

s\_Raspisanie\* result = new s\_Raspisanie;

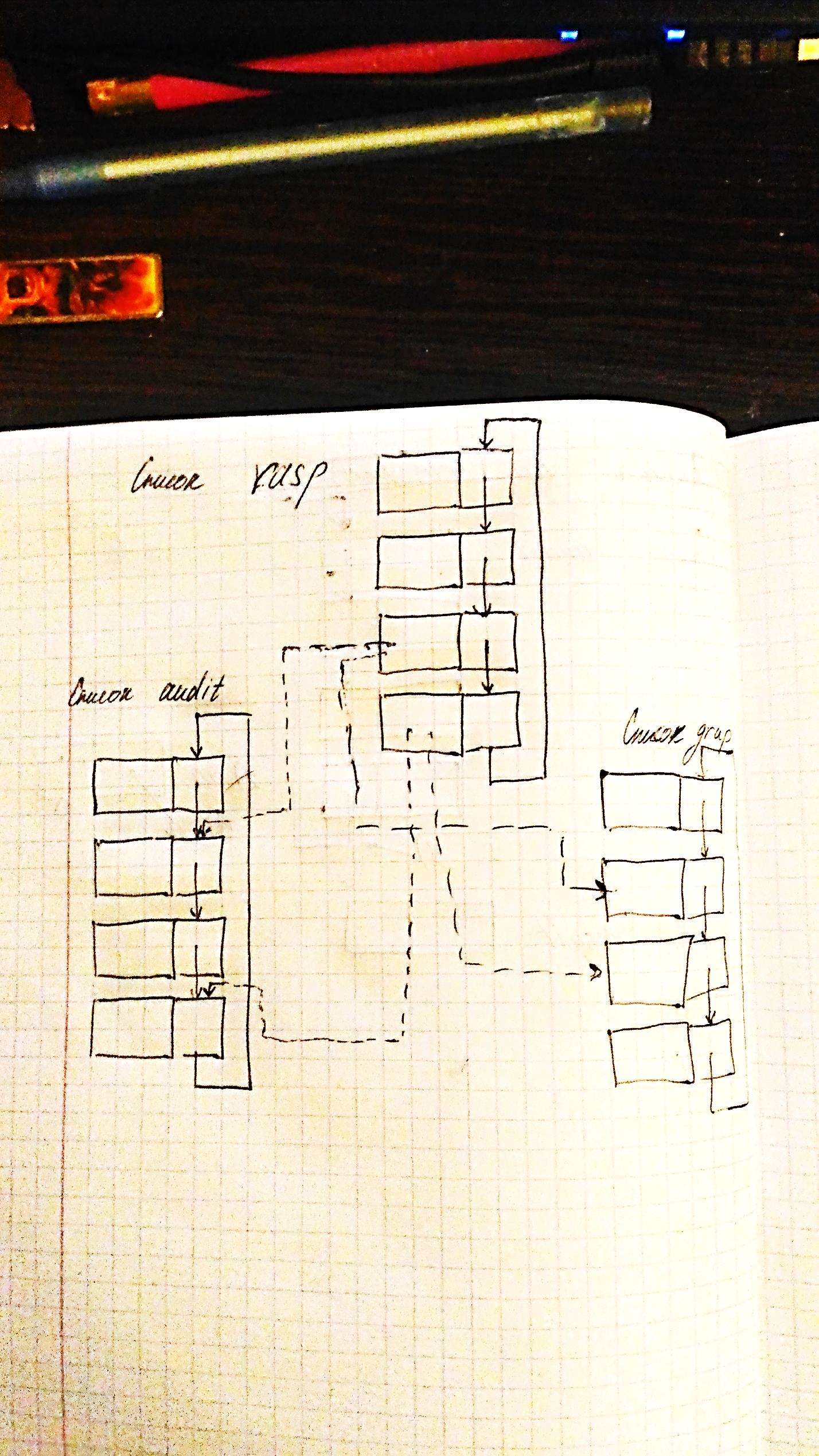
Poisk\_Raspisanie(rasp, au->data.kod, result);

au->link\_r = result;

return pos;

}

Методы добавляют в список один элемент, указатель которого задают на первый элемент, а указатель последнего элемента начинает указывать на новый. Таким образом, кольцевой односвязный список не нарушает своей целостности. Так же в двух последних методах происходит связывание списков друг с другом.



**Методы считывания из файла**

//Считывание из файла

int Read\_Auditory(s\_Auditory\* s\_au, char \*NameFile)//считывание списка аудиторий

{

FILE \*f;//создаем переменную для файла

fopen\_s(&f, NameFile, "rt");//открываем файл для чтения

while (!feof(f))//проверка на конец файла

{

Auditory p;//переменная для аудитории

FileAud(f, p);//считываем данные в переменную для аудитории

Dob\_Auditory(s\_au, p);//добавляем в список

}

fclose(f);//закрываем файл

return 0;

}

int Read\_Gruppa(s\_Gruppa\* s\_gr, char \*NameFile)//считывание списка групп

{

FILE \*f;

fopen\_s(&f, NameFile, "rt");

while (!feof(f))

{

Gruppa p;

FileGrup(f, p);

Dob\_Gruppa(s\_gr, p);

}

fclose(f);

return 0;

}

int Read\_Rasp(s\_Raspisanie\* s\_rasp, char \*NameFile, s\_Auditory\* audit, s\_Gruppa\* grup)//считывание списка расписания

{

FILE \*f;

fopen\_s(&f, NameFile, "rt");

while (!feof(f))

{

Raspisanie p;

FileRasp(f, p);

s\_rasp=Dob\_Raspisanie(s\_rasp, p,audit,grup);

}

fclose(f);

return 0;

}

int Read\_Posech(s\_Posechaemost\* s\_pos, char \*NameFile, s\_Raspisanie\* rasp)//считывания списка посещаемости

{

FILE \*f;

fopen\_s(&f, NameFile, "rt");

while (!feof(f))

{

Posechaemost p;

FilePos(f, p);

s\_pos=Dob\_Posechaemost(s\_pos, p, rasp);

}

fclose(f);

return 0;

}

//вспомогательное считывание

void FileAud(FILE \*f, Auditory &p)//аудитории

{

fscanf(f, "%i", &p.nomer\_a);

fscanf(f, "%i", &p.vmestimost);

}

void FileGrup(FILE \*f, Gruppa &p)//группы

{

fscanf(f, "%i", &p.nomer\_g);

fscanf(f, "%i", &p.kolvo\_stud);

}

void FileRasp(FILE \*f, Raspisanie &p)//расписания

{

fscanf(f, "%i", &p.kod\_r);

fscanf(f, "%i", &p.nomer\_a);

fscanf(f, "%i", &p.nomer\_g);

fscanf(f, "%s", p.den);

fscanf(f, "%i", &p.time);

}

void FilePos(FILE \*f, Posechaemost &p)//посещаемости

{

fscanf(f, "%i", &p.kod);

fscanf(f, "%i", &p.dd);

fscanf(f, "%i", &p.mm);

fscanf(f, "%i", &p.kolvo);

}

Методы считывают информацию из файлов и при помощи методов добавления, описанных выше, создают списки. Для корректной работы всех методом рекомендуется сначала считать данные об аудиториях и группах, затем о расписании и, наконец, о посещаемости.

**Методы поиска**

//Методы поиска

void Poisk\_Auditory(s\_Auditory\* ptr, int nomer\_auditory,s\_Auditory\* result)//поиск аудитории

{

s\_Auditory\* p = ptr;//создаем указатель приравниваем его на элемент текущий элемент

result->data.nomer\_a = -404;//номер аудитории равен -404

ptr = ptr->next;

while (ptr!= p)//пока не вернемся на входной элемент

{

if (ptr->data.nomer\_a == nomer\_auditory)//если номер аудитории совпадает с тем, что мы ищем

{

//результат равен найденному элементу

result->next = ptr->next;

result->data = ptr->data;

break;//выход

}

ptr = ptr->next;//идем к следующему элементу

}

}

void Poisk\_Grup(s\_Gruppa\* ptr, int nomer\_grup, s\_Gruppa\* result)//поиск группы

{

s\_Gruppa\* p = ptr;

result->data.nomer\_g = -404;

ptr = ptr->next;

while (ptr!= p)

{

if (ptr->data.nomer\_g == nomer\_grup)

{

result->next = ptr->next;

result->data = ptr->data;

break;

}

ptr = ptr->next;

}

}

void Poisk\_Raspisanie(s\_Raspisanie\* ptr, int kod, s\_Raspisanie\* result)//поиск расписания

{

s\_Raspisanie\* p = ptr;

result->data.kod\_r = -404;

ptr = ptr->next;

while (ptr!= p)

{

if (ptr->data.kod\_r == kod)

{

result->next = ptr->next;

result->link\_a = ptr->link\_a;

result->link\_g = ptr->link\_g;

result->data = ptr->data;

break;

}

ptr = ptr->next;

}

}

void Poisk\_Posechaemost(s\_Posechaemost\* ptr, int kod, s\_Posechaemost\* result)//поиска посещаемости

{

s\_Posechaemost\* p = ptr;

result->data.kod = -404;

ptr = ptr->next;

while (ptr!= p)

{

if (ptr->data.kod== kod)

{

result->next = ptr->next;

result->link\_r = ptr->link\_r;

result->data = ptr->data;

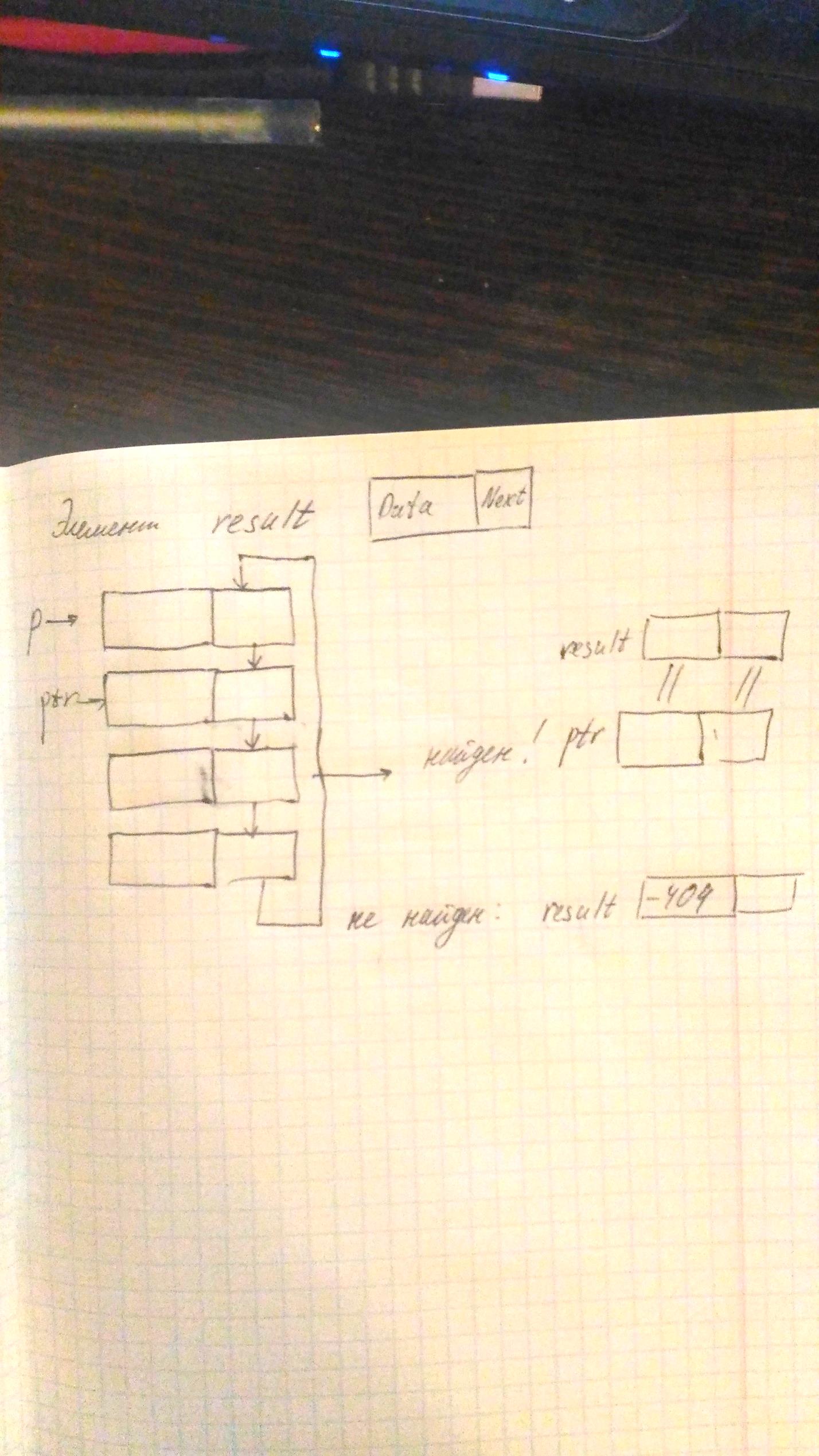
break;

}

ptr = ptr->next;

}

}



Методы осуществляют поиск элемента в списке по заданному номеру. В случае его отсутствия номер результата становиться равным -404.

**Дерево для сортировки:**

//Дерево

void showTree(Node\*&Tree, s\_Analyz\* a)////рекрусивный метод обхода дерева

{

if (Tree != NULL)//проверяем что ветвь существует

{

showTree(Tree->L,a);//идем в левую ветвь

Dob\_Analyz(a, Tree->x->data);//добавляем элемент в список

showTree(Tree->R,a);//идем в правую ветвь

}

}

void add\_node(s\_Analyz\* x, Node\*&MyTree)//добавить ветвь

{

if (NULL == MyTree)//если ветки не существует

{

MyTree = new Node;//создаем ветвь

MyTree->x = x;//записываем данные

MyTree->L = NULL;//описываем подветви

MyTree->R = NULL;//

}

if ((x->data.zagr - MyTree->x->data.zagr) <0)//если данные, которые хотим добавить меньше ...

{

if (MyTree->L!=NULL) add\_node(x, MyTree->L);//если у подветвь пустая, создаем ее

else//иначе

{

MyTree->L = new Node;//создаем новую подветь

MyTree->L->L = NULL;//описываем подветви

MyTree->L->R = NULL;//

MyTree->L->x = x;//заполняем данные

}

}

if (x->data.zagr - MyTree->x->data.zagr>0)

{

if (MyTree->R != NULL)add\_node(x, MyTree->R);

else

{

MyTree->R= new Node;

MyTree->R->L = NULL;

MyTree->R->R = NULL;

MyTree->R->x = x;

}

}

}

void del(Node\*&Tree)//удаляем дерево

{

if (Tree != NULL)//если дерево не пустое

{

del(Tree->L);//удаляем левую ветвь

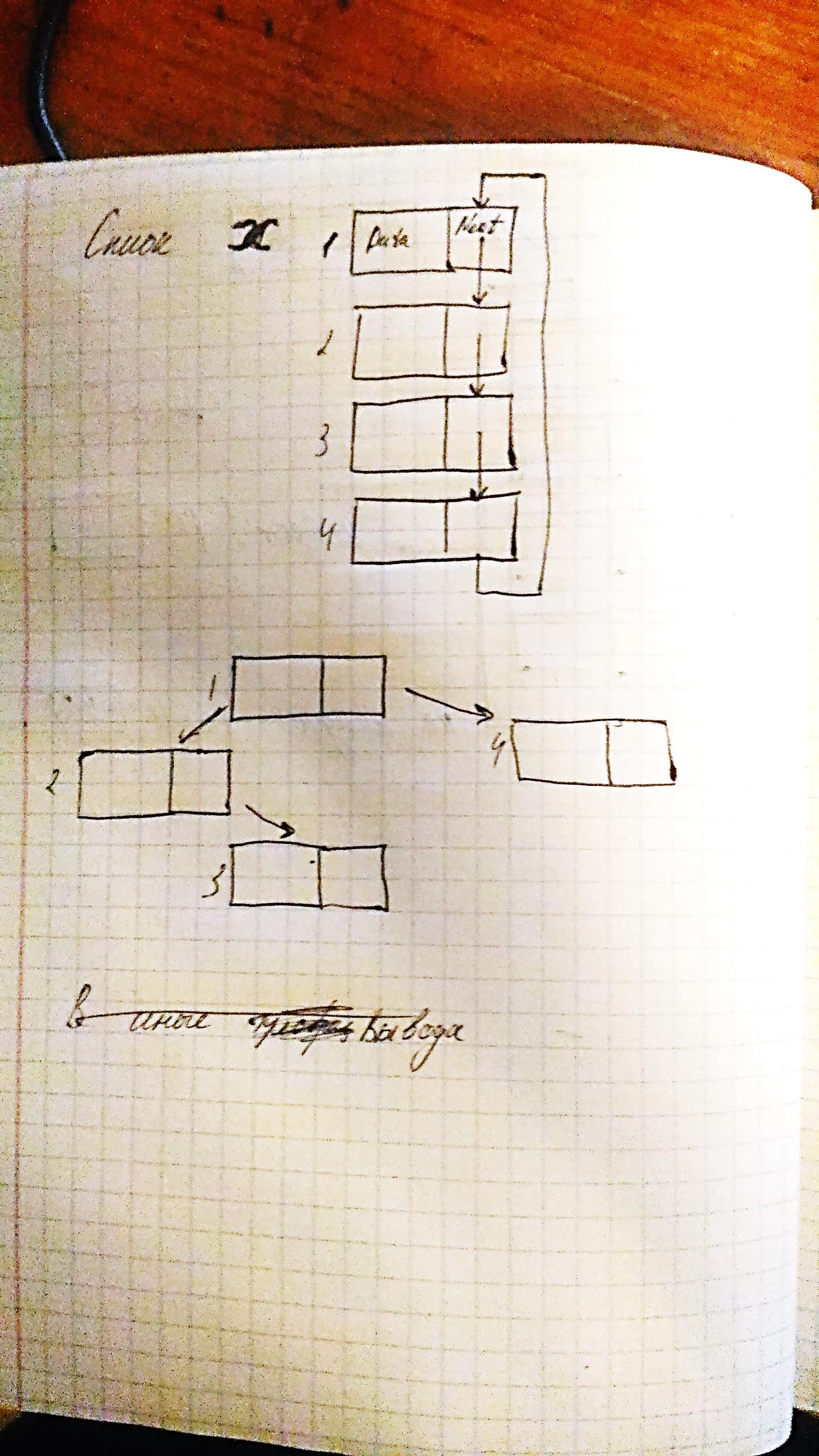
del(Tree->R);//правую

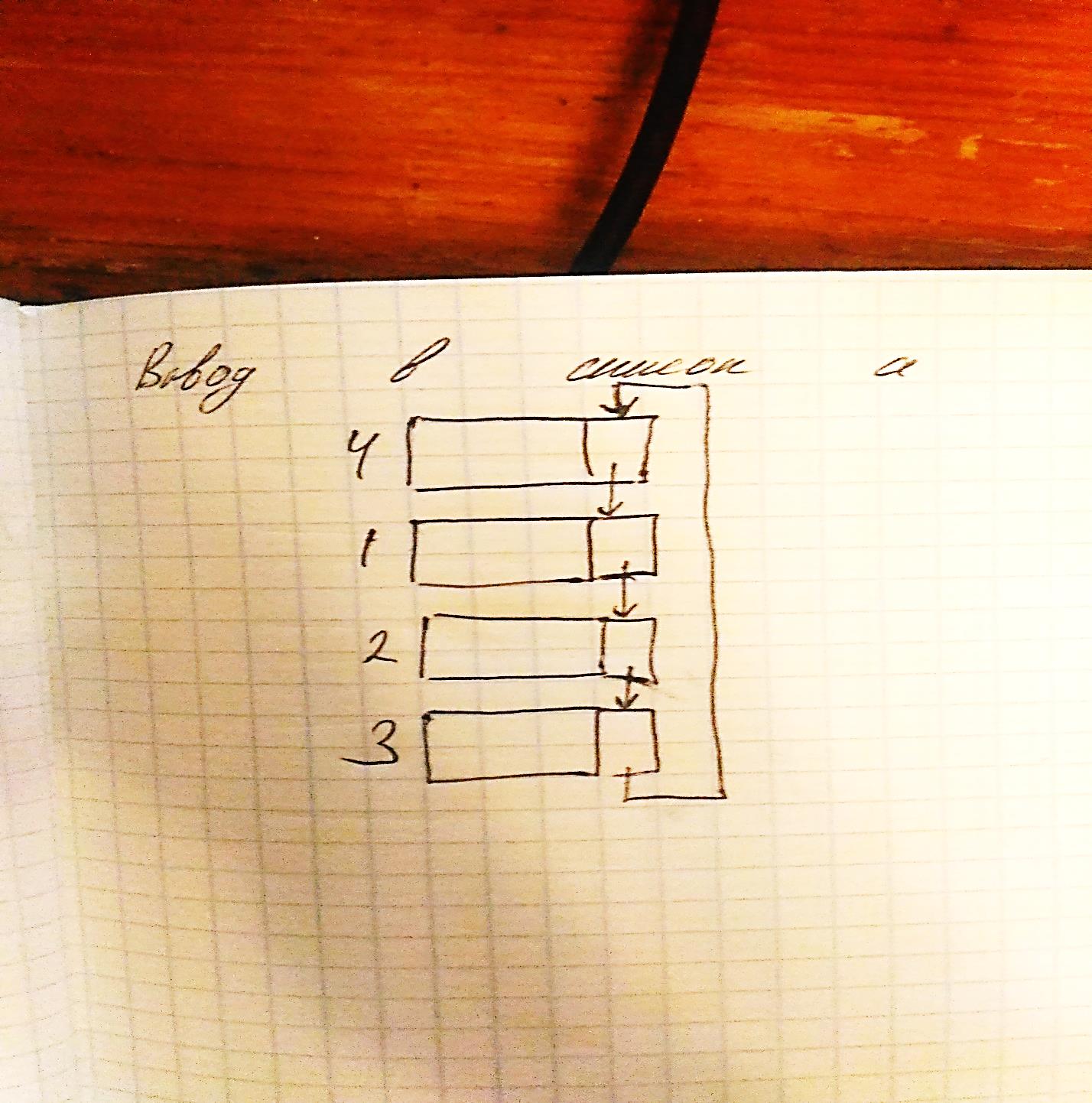
delete Tree;

Tree = NULL;

}

}





Сортировка загруженности выполняется при помощи бинарного дерева. При помощи метода add\_node дерево заполняется элементами списка, а при помощи рекурсивного метода обхода дерева showTree, элементы заполняют новый список, в котором они расположены упорядоченно, по убыванию значения загруженности.

**Метод для анализа**

void Analyzirovat(s\_Posechaemost\* pos, int audit,int dd, int mm, int time, s\_Analyz\* analyz)//метод для анализа

{

s\_Posechaemost\* p = pos;//новый указатель(необходим для цикла)

pos = pos->next;//смещаем "бегунок"

Analyz an;

an.dd = dd;//заполняем данные

an.mm = mm;//

an.nomer\_a = audit;//

an.time = time;//

int sum = 0;//

int teor = 0;//

bool check = false;

while (pos != p)//цикл

{

if (pos->data.dd == dd && pos->data.mm == mm && pos->link\_r->data.nomer\_a == audit && pos->link\_r->data.time == time)//ищем необходимые данные

{

an.den = pos->link\_r->data.den;//заполняем данные

sum +=pos->data.kolvo;//

teor = pos->link\_r->link\_a->data.vmestimost;//

check = true;

}

pos = pos->next;//смещаем бегунок

}

if (teor == 0) teor = 1;

an.zagr = (sum\*100) /teor;//ищем загруженность в процентах

if (check) Dob\_Analyz(analyz, an);

}

При помощи этого метода можно проанализировать загруженность аудитории на заданное число и время. В методе совершается обход по списку посещаемости, который связан с списком расписания, и если элемент списка удовлетворяет входным данным, то количество человек суммируется, а после завершения обхода вычисляется загруженность в процентах, после чего происходит добавление в список элемента с полученными данными.

Методы для добавления расписания и посещаемости «вручную»:

void Create\_Posech(s\_Posechaemost\* pos,s\_Raspisanie\* rasp, int kod, int dd, int mm, int kolvo)//метод для создания посещаемости "вручную"

{

Posechaemost p ;//элемент посещаемсть

p.kod = kod;//добавляем данные

p.dd = dd;//

p.mm = mm;//

p.kolvo = kolvo;//

Dob\_Posechaemost(pos, p, rasp);//добавляем в список

}

void Create\_Rasp(s\_Raspisanie\* rasp, s\_Gruppa\* gru, s\_Auditory\* s\_au, int kod, int audit, int gruppa, char\* den, int time)

{

Raspisanie r;

r.kod\_r = kod;

r.nomer\_a = audit;

r.nomer\_g = gruppa;

r.den = den;

r.time = time;

Dob\_Raspisanie(rasp, r, s\_au, gru);

}

При помощи этих методов появляется возможность добавлять в соответствующие списки элементы во время работы программы, не прибегая к считыванию из файлов.

**Методы удаления списков**

//Методы удаления списков

void Del\_El(s\_Analyz\* an)//удаление элемента

{

s\_Analyz\* p;

if (an != NULL)

{

p = an;

an=an->next;

delete(p);

}

}

void Del\_Spisok(s\_Analyz\* an, s\_Analyz\* Head)//удаление списка

{

an = Head;

an->next;

while (an!=Head)

{

Del\_El(an);

}

Del\_El(Head);

}

void Del\_El(s\_Auditory\* an)//удаление элемента

{

s\_Auditory\* p;

if (an != NULL)

{

p = an;

an = an->next;

delete(p);

}

}

void Del\_Spisok(s\_Auditory\* an, s\_Auditory\* Head)//удаление списка

{

an = Head;

an->next;

while (an != Head)

{

Del\_El(an);

}

Del\_El(Head);

}

void Del\_El(s\_Gruppa\* an)//удаление элемента

{

s\_Gruppa\* p;

if (an != NULL)

{

p = an;

an = an->next;

delete(p);

}

}

void Del\_Spisok(s\_Gruppa\* an, s\_Gruppa\* Head)//удаление списка

{

an = Head;

an->next;

while (an != Head)

{

Del\_El(an);

}

Del\_El(Head);

}

void Del\_El(s\_Raspisanie\* an)//удаление элемента

{

s\_Raspisanie\* p;

if (an != NULL)

{

p = an;

an = an->next;

delete(p);

}

}

void Del\_Spisok(s\_Raspisanie\* an, s\_Raspisanie\* Head)//удаление списка

{

an = Head;

an->next;

while (an != Head)

{

Del\_El(an);

}

Del\_El(Head);

}

void Del\_El(s\_Posechaemost\* an)//удаление элемента

{

s\_Posechaemost\* p;

if (an != NULL)

{

p = an;

an = an->next;

delete(p);

}

}

void Del\_Spisok(s\_Posechaemost\* an, s\_Posechaemost\* Head)//удаление списка

{

an = Head;

an->next;

while (an != Head)

{

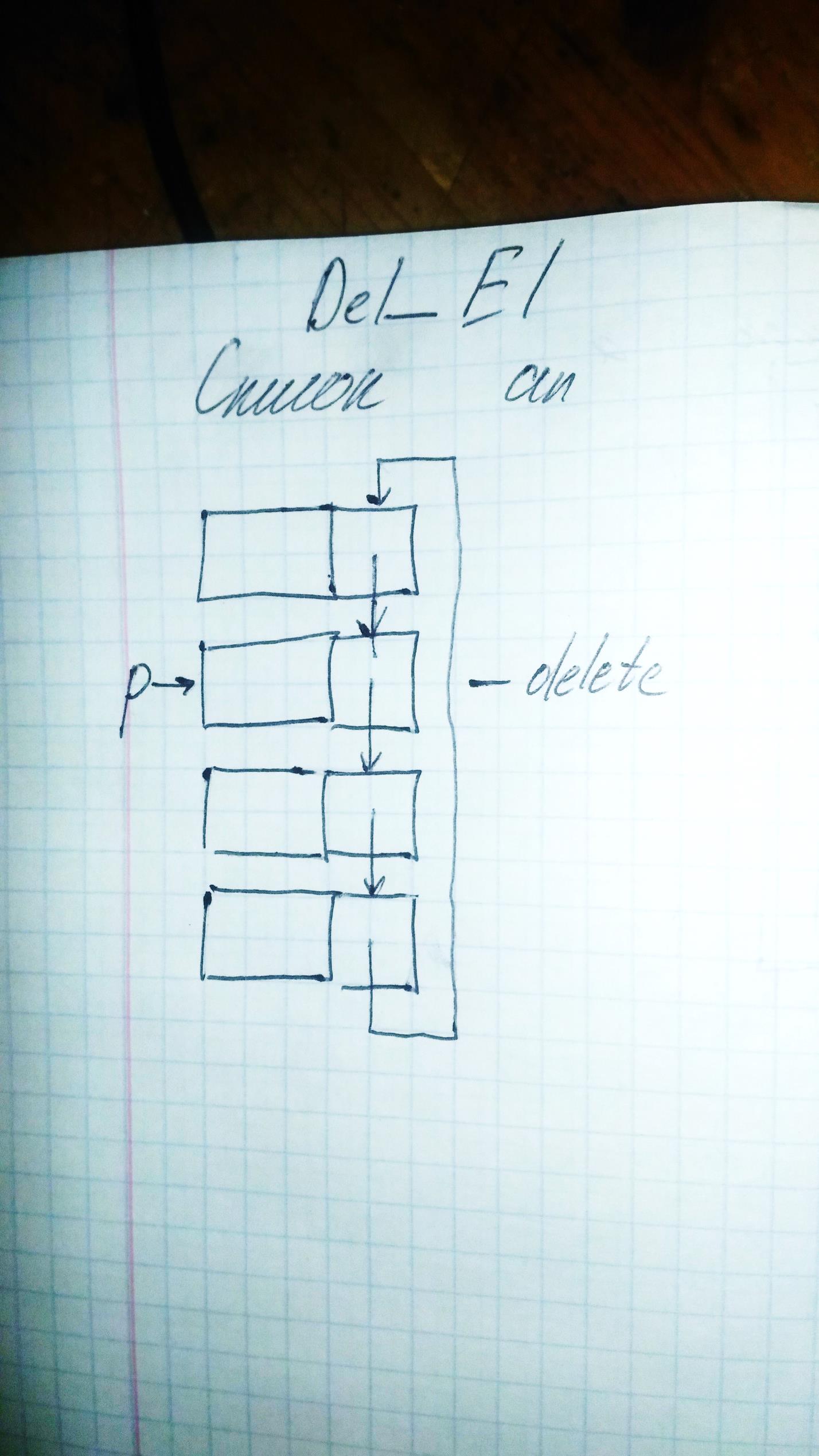
Del\_El(an);

}

Del\_El(Head);

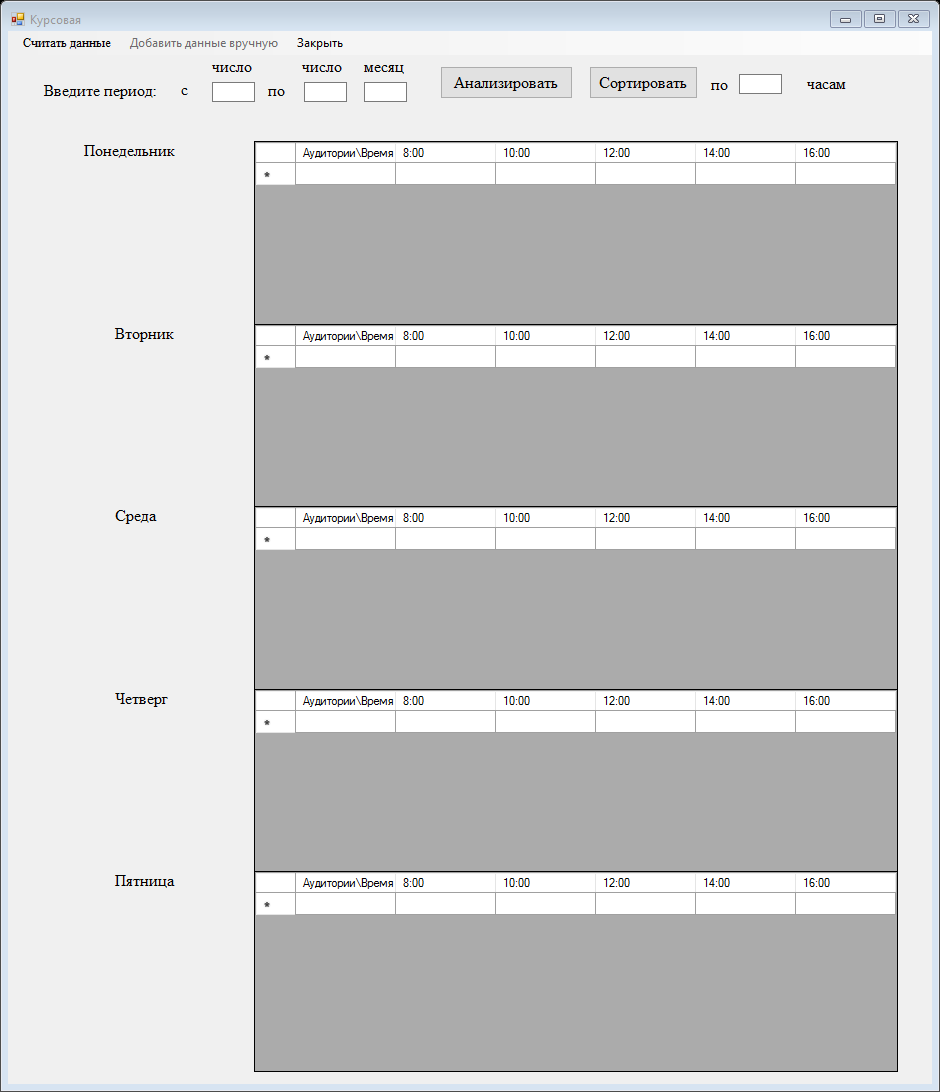
}

Данные методы удаляют списки и очищают память. Запускаются при завершении программы.



**Разработка интерфейса**

При запуске программы главная форма имеет следующий вид:



На форме присутствует меню, при помощи которого можно считать данные из файлов и добавить данные самостоятельно. Кнопки, при нажатии на которые происходит анализ данных или их сортировка.

При выборе пункта меню «добавить данные вручную» форма приобретает следующий вид.

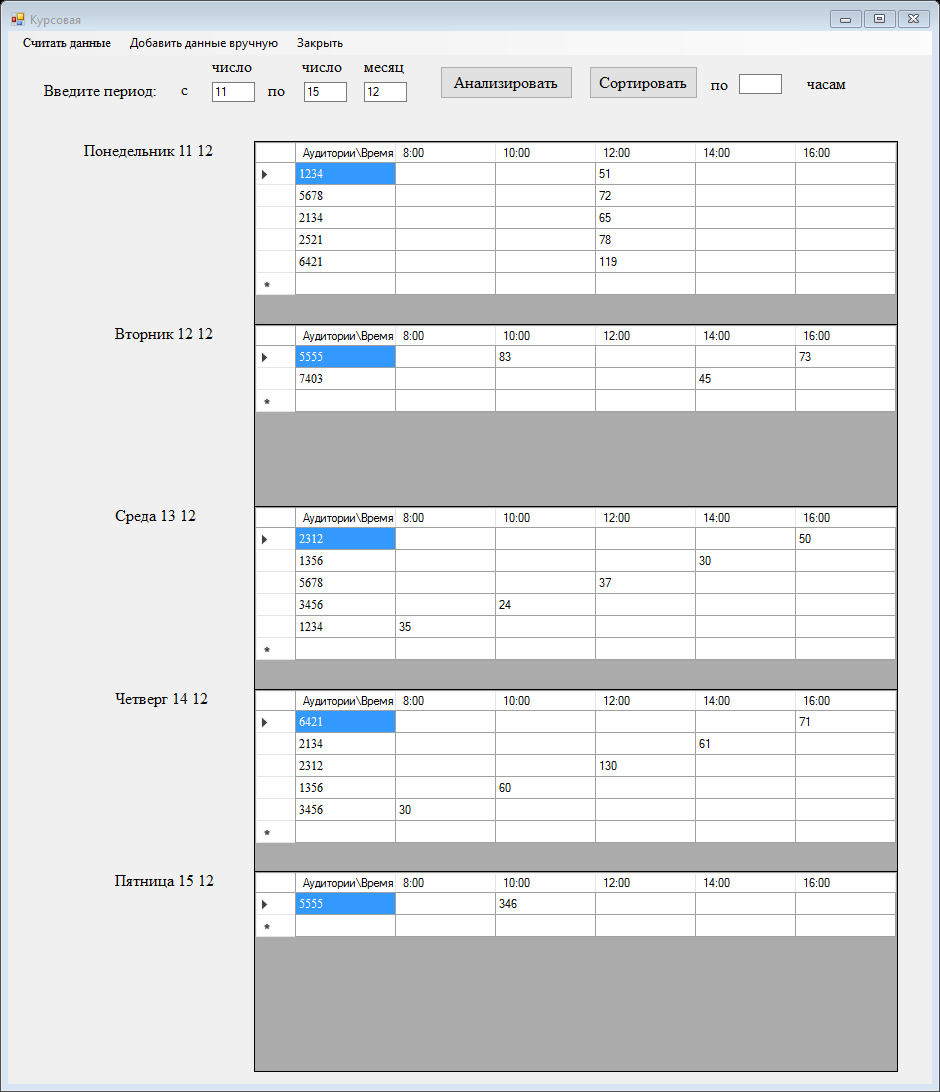
Появляются дополнительные элементы на форме, позволяющие добавлять в список элементы с данными, заданными пользователем. Эти элементы можно скрыть.

**Тестирование**

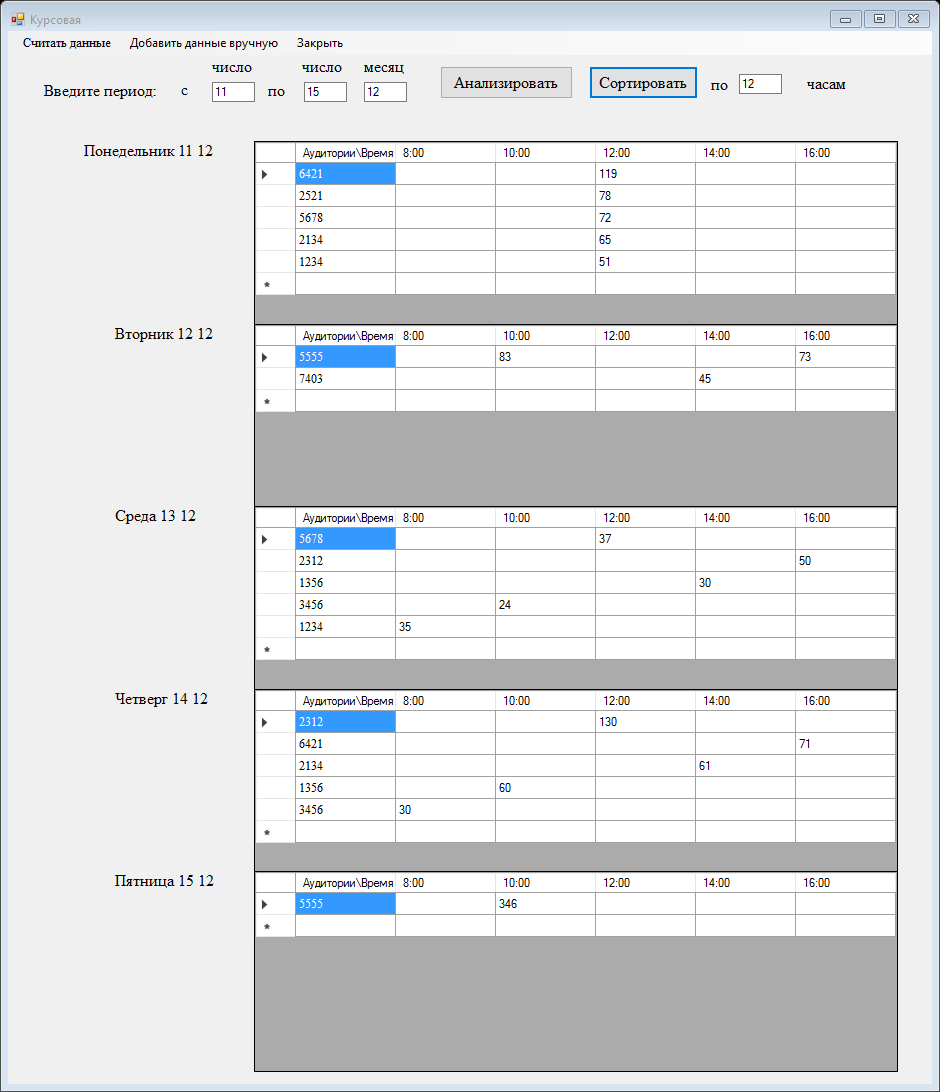
Входные данные:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Результат работы программы:



Как видно результат работы программы полностью совпадает с ожидаемыми результатами. Отсортируем таблицу по столбцу 12 часов.



**Заключение**

Разработка программы выполнялась в два этапа: разработка основных методов и разработка оконного интерфейса. При разработке программы были отточены навыки работы с динамическими структурами данных и разработки оконных интерфейсов.

Программу можно использовать для анализа загруженности аудиторий, а после получении данных выбирать оптимальный вариант для направления группы в ту или иную аудиторию. Программу можно модифицировать, добавляя новые поля в структуры данных, новые элементы на форму, что позволит использовать эту программу не только для анализа, но и составления расписания на неделю и не только.

**Приложение**

**Header.h**

#define Kurs

#ifdef Kurs

#include <iostream>

#include "io.h"

#include <fstream>

#include <conio.h>

#include <stdio.h>

#include <string>

#pragma warning(disable: 4996)

#define \_CRT\_SECURE\_NO\_WARNINGS

struct Auditory

{

int nomer\_a;

int vmestimost;

};

struct Gruppa

{

int nomer\_g;

int kolvo\_stud;

};

struct Raspisanie

{

int kod\_r;

int nomer\_g;

int nomer\_a;

char\* den=new char[50];

int time;

};

struct Posechaemost

{

int kod;

int dd;

int mm;

int kolvo;

};

struct s\_Auditory

{

s\_Auditory \*next;

Auditory data;

};

struct s\_Gruppa

{

s\_Gruppa \*next;

Gruppa data;

};

struct s\_Raspisanie

{

s\_Raspisanie \*next;

s\_Auditory\* link\_a;

s\_Gruppa\* link\_g;

Raspisanie data;

};

struct s\_Posechaemost

{

s\_Raspisanie\* link\_r;

s\_Posechaemost \*next;

Posechaemost data;

};

struct Analyz //вспомогательная струткура

{

int nomer\_a;

int dd;

int mm;

char\* den = new char[50];

int time;

int zagr;

};

struct s\_Analyz //вспомогательный список

{

s\_Analyz\*next;

Analyz data;

};

struct Node //дерево

{

s\_Analyz\* x;

Node \*L, \*R;

};

//Инициализация списков

s\_Auditory\* new\_Auditory(s\_Auditory\* Head);/\*список аудиторий\*/

s\_Gruppa\* new\_Gruppa(s\_Gruppa\* Head);//список групп

s\_Raspisanie\* new\_Raspisanie(s\_Raspisanie\* Head);//список расписаний

s\_Posechaemost\* new\_Posechaemost(s\_Posechaemost\* Head);//список посещаемости

s\_Analyz\* new\_Analyz(s\_Analyz\* Head);

//Добавление одного элемента

void Dob\_Auditory(s\_Auditory \*audit, Auditory data);//добавление аудитории

void Dob\_Gruppa(s\_Gruppa \*grupp, Gruppa data);//добавление группы

void Dob\_Analyz(s\_Analyz\* anal, Analyz data);

s\_Raspisanie\* Dob\_Raspisanie(s\_Raspisanie\* rasp, Raspisanie data, s\_Auditory\* audit, s\_Gruppa\* grup);//добавление товара

s\_Posechaemost\* Dob\_Posechaemost(s\_Posechaemost \*pos, Posechaemost data, s\_Raspisanie\* rasp);//добавление товара

//Считывание из файла

int Read\_Auditory(s\_Auditory\* s\_au, char \*NameFile);//считывание списка аудиторий

int Read\_Gruppa(s\_Gruppa\* s\_gr, char \*NameFile);//считывание списка групп

int Read\_Posech(s\_Posechaemost\* s\_pos, char \*NameFile, s\_Raspisanie\* rasp);//считывание списка расписания

int Read\_Rasp(s\_Raspisanie\* s\_rasp, char \*NameFile, s\_Auditory\* audit, s\_Gruppa\* grup);//считывания списка посещаемости

//вспомогательное считывание

void FileAud(FILE \*f, Auditory &p);//аудитории

void FileGrup(FILE \*f, Gruppa &p);//группы

void FileRasp(FILE \*f, Raspisanie &p);//расписания

void FilePos(FILE \*f, Posechaemost &p);//посещаемости

//Методы поиска

void Poisk\_Auditory(s\_Auditory\* ptr, int nomer\_auditory, s\_Auditory\* result);//поиск аудитории

void Poisk\_Grup(s\_Gruppa\* ptr, int nomer\_grup, s\_Gruppa\* result);//поиск группы

void Poisk\_Raspisanie(s\_Raspisanie\* ptr, int kod, s\_Raspisanie\* result);//поиск расписания

void Poisk\_Posechaemost(s\_Posechaemost\* ptr, int kod, s\_Posechaemost\* result);//поиска посещаемости

void Create\_Posech(s\_Posechaemost\* pos, s\_Raspisanie\* rasp, int kod, int dd, int mm, int kolvo);//Метод сзодания посещаемости "вручную"

void Create\_Rasp(s\_Raspisanie\* rasp, s\_Gruppa\* gru, s\_Auditory\* s\_au, int kod, int audit, int gruppa, char\* den, int time);

//Методы для дерева

void add\_node(s\_Analyz\* x, Node\*&MyTree);//создание ветвей

void showTree(Node\*&Tree, s\_Analyz\* a);//метод обхода

void showTreeVosr(Node\*&Tree, s\_Analyz\* a);//метод обхода 2

void del(Node\*&Tree);//удаление

void Analyzirovat(s\_Posechaemost\* pos, int audit, int dd, int mm, int time, s\_Analyz\* analyz);//метод для анализа

//Методы удаления

void Del\_El(s\_Analyz\* an);

void Del\_Spisok(s\_Analyz\* an, s\_Analyz\* Head);

void Del\_El(s\_Auditory\* an);

void Del\_Spisok(s\_Auditory\* an, s\_Auditory\* Head);

void Del\_El(s\_Gruppa\* an);

void Del\_Spisok(s\_Gruppa\* an, s\_Gruppa\* Head);

void Del\_El(s\_Raspisanie\* an);

void Del\_Spisok(s\_Raspisanie\* an, s\_Raspisanie\* Head);

void Del\_El(s\_Posechaemost\* an);

void Del\_Spisok(s\_Posechaemost\* an, s\_Posechaemost\* Head);

#endif // Kurs

**Core.cpp**

#include "Header.h"

#include <io.h>

#include <iostream>

#pragma warning(disable: 4996)

#define \_CRT\_SECURE\_NO\_WARNINGS

using namespace std;

//Инициализация списков

s\_Auditory\* new\_Auditory(s\_Auditory\* Head)//список аудиторий

{

Head->next = Head;//указатель ставим на этот элемент

return Head;

}

s\_Gruppa\* new\_Gruppa(s\_Gruppa\* Head)//список групп

{

Head->next = Head;

return Head;

}

s\_Raspisanie\* new\_Raspisanie(s\_Raspisanie\* Head)//список расписаний

{

Head->next = Head;

return Head;

}

s\_Posechaemost\* new\_Posechaemost(s\_Posechaemost\* Head)//список посещаемости

{

Head->next = Head;

return Head ;

}

s\_Analyz\* new\_Analyz(s\_Analyz\* Head)//список анализа

{

Head->next = Head;

return Head;

}

//Добавление одного элемента

void Dob\_Auditory(s\_Auditory \*audit, Auditory data)//добавление аудитории

{

s\_Auditory\* au = new s\_Auditory;//создаем указатель на элемент списка

au->next = audit->next;//указатель на первый элемент списка

audit->next = au;//указатель пердпоследнего на последний(новый) элемент списка

au->data = data;//сопоставляем данные

}

void Dob\_Gruppa(s\_Gruppa \*grupp, Gruppa data)//добавление группы

{

s\_Gruppa\* au = new s\_Gruppa;

au->next = grupp->next;

grupp->next = au;

au->data = data;

}

void Dob\_Analyz(s\_Analyz\* anal, Analyz data)

{

s\_Analyz\* an = new s\_Analyz;

an->next = anal->next;

anal->next = an;

an->data = data;

}

s\_Raspisanie\* Dob\_Raspisanie(s\_Raspisanie\* rasp, Raspisanie data, s\_Auditory\* audit, s\_Gruppa\* grup)//добавление товара

{

s\_Raspisanie\* au = new s\_Raspisanie;

au->next = rasp->next;

rasp->next = au;

au->data = data;

s\_Auditory\* result = new s\_Auditory;

Poisk\_Auditory(audit, au->data.nomer\_a, result);

au->link\_a = result;

s\_Gruppa\* res1 = new s\_Gruppa;

Poisk\_Grup(grup, au->data.nomer\_g, res1);

au->link\_g = res1;

return rasp;

}

s\_Posechaemost\* Dob\_Posechaemost(s\_Posechaemost \*pos, Posechaemost data,s\_Raspisanie\* rasp )//добавление товара

{

s\_Posechaemost\* au = new s\_Posechaemost;

au->next = pos->next;

pos->next = au;

au->data = data;

s\_Raspisanie\* result = new s\_Raspisanie;

Poisk\_Raspisanie(rasp, au->data.kod, result);

au->link\_r = result;

return pos;

}

//Считывание из файла

int Read\_Auditory(s\_Auditory\* s\_au, char \*NameFile)//считывание списка аудиторий

{

FILE \*f;//создаем переменную для файла

fopen\_s(&f, NameFile, "rt");//открываем файл для чтения

while (!feof(f))//проверка на конец файла

{

Auditory p;//переменная для аудитории

FileAud(f, p);//считываем данные в переменную для аудитории

Dob\_Auditory(s\_au, p);//добавляем в список

}

fclose(f);//закрываем файл

return 0;

}

int Read\_Gruppa(s\_Gruppa\* s\_gr, char \*NameFile)//считывание списка групп

{

FILE \*f;

fopen\_s(&f, NameFile, "rt");

while (!feof(f))

{

Gruppa p;

FileGrup(f, p);

Dob\_Gruppa(s\_gr, p);

}

fclose(f);

return 0;

}

int Read\_Rasp(s\_Raspisanie\* s\_rasp, char \*NameFile, s\_Auditory\* audit, s\_Gruppa\* grup)//считывание списка расписания

{

FILE \*f;

fopen\_s(&f, NameFile, "rt");

while (!feof(f))

{

Raspisanie p;

FileRasp(f, p);

s\_rasp=Dob\_Raspisanie(s\_rasp, p,audit,grup);

}

fclose(f);

return 0;

}

int Read\_Posech(s\_Posechaemost\* s\_pos, char \*NameFile, s\_Raspisanie\* rasp)//считывания списка посещаемости

{

FILE \*f;

fopen\_s(&f, NameFile, "rt");

while (!feof(f))

{

Posechaemost p;

FilePos(f, p);

s\_pos=Dob\_Posechaemost(s\_pos, p, rasp);

}

fclose(f);

return 0;

}

//вспомогательное считывание

void FileAud(FILE \*f, Auditory &p)//аудитории

{

fscanf(f, "%i", &p.nomer\_a);

fscanf(f, "%i", &p.vmestimost);

}

void FileGrup(FILE \*f, Gruppa &p)//группы

{

fscanf(f, "%i", &p.nomer\_g);

fscanf(f, "%i", &p.kolvo\_stud);

}

void FileRasp(FILE \*f, Raspisanie &p)//расписания

{

fscanf(f, "%i", &p.kod\_r);

fscanf(f, "%i", &p.nomer\_a);

fscanf(f, "%i", &p.nomer\_g);

fscanf(f, "%s", p.den);

fscanf(f, "%i", &p.time);

}

void FilePos(FILE \*f, Posechaemost &p)//посещаемости

{

fscanf(f, "%i", &p.kod);

fscanf(f, "%i", &p.dd);

fscanf(f, "%i", &p.mm);

fscanf(f, "%i", &p.kolvo);

}

//Методы поиска

void Poisk\_Auditory(s\_Auditory\* ptr, int nomer\_auditory,s\_Auditory\* result)//поиск аудитории

{

s\_Auditory\* p = ptr;//создаем указатель приравниваем его на элемент текущий элемент

result->data.nomer\_a = -404;//номер аудитории равен -404

ptr = ptr->next;

while (ptr!= p)//пока не вернемся на входной элемент

{

if (ptr->data.nomer\_a == nomer\_auditory)//если номер аудитории совпадает с тем, что мы ищем

{

//результат равен найденному элементу

result->next = ptr->next;

result->data = ptr->data;

break;//выход

}

ptr = ptr->next;//идем к следующему элементу

}

}

void Poisk\_Grup(s\_Gruppa\* ptr, int nomer\_grup, s\_Gruppa\* result)//поиск группы

{

s\_Gruppa\* p = ptr;

result->data.nomer\_g = -404;

ptr = ptr->next;

while (ptr!= p)

{

if (ptr->data.nomer\_g == nomer\_grup)

{

result->next = ptr->next;

result->data = ptr->data;

break;

}

ptr = ptr->next;

}

}

void Poisk\_Raspisanie(s\_Raspisanie\* ptr, int kod, s\_Raspisanie\* result)//поиск расписания

{

s\_Raspisanie\* p = ptr;

result->data.kod\_r = -404;

ptr = ptr->next;

while (ptr!= p)

{

if (ptr->data.kod\_r == kod)

{

result->next = ptr->next;

result->link\_a = ptr->link\_a;

result->link\_g = ptr->link\_g;

result->data = ptr->data;

break;

}

ptr = ptr->next;

}

}

void Poisk\_Posechaemost(s\_Posechaemost\* ptr, int kod, s\_Posechaemost\* result)//поиска посещаемости

{

s\_Posechaemost\* p = ptr;

result->data.kod = -404;

ptr = ptr->next;

while (ptr!= p)

{

if (ptr->data.kod== kod)

{

result->next = ptr->next;

result->link\_r = ptr->link\_r;

result->data = ptr->data;

break;

}

ptr = ptr->next;

}

}

//Дерево

void showTreeVosr(Node\*&Tree, s\_Analyz\* a)//рекрусивный метод обхода дерева

{

if (Tree != NULL)//проверяем что ветвь существует

{

showTree(Tree->R, a);//идем в правую ветвь

Dob\_Analyz(a, Tree->x->data);//добавляем элемент в список

showTree(Tree->L, a);//идем в левую ветвь

}

}

//Дерево

void showTree(Node\*&Tree, s\_Analyz\* a)////рекрусивный метод обхода дерева

{

if (Tree != NULL)//проверяем что ветвь существует

{

showTree(Tree->L,a);//идем в левую ветвь

Dob\_Analyz(a, Tree->x->data);//добавляем элемент в список

showTree(Tree->R,a);//идем в правую ветвь

}

}

void add\_node(s\_Analyz\* x, Node\*&MyTree)//добавить ветвь

{

if (NULL == MyTree)//если ветки не существует

{

MyTree = new Node;//создаем ветвь

MyTree->x = x;//записываем данные

MyTree->L = NULL;//описываем подветви

MyTree->R = NULL;//

}

if ((x->data.zagr - MyTree->x->data.zagr) <0)//если данные, которые хотим добавить меньше ...

{

if (MyTree->L!=NULL) add\_node(x, MyTree->L);//если у подветвь пустая, создаем ее

else//иначе

{

MyTree->L = new Node;//создаем новую подветь

MyTree->L->L = NULL;//описываем подветви

MyTree->L->R = NULL;//

MyTree->L->x = x;//заполняем данные

}

}

if (x->data.zagr - MyTree->x->data.zagr>0)

{

if (MyTree->R != NULL)add\_node(x, MyTree->R);

else

{

MyTree->R= new Node;

MyTree->R->L = NULL;

MyTree->R->R = NULL;

MyTree->R->x = x;

}

}

}

void del(Node\*&Tree)//удаляем дерево

{

if (Tree != NULL)//если дерево не пустое

{

del(Tree->L);//удаляем левую ветвь

del(Tree->R);//правую

delete Tree;

Tree = NULL;

}

}

void Analyzirovat(s\_Posechaemost\* pos, int audit,int dd, int mm, int time, s\_Analyz\* analyz)//метод для анализа

{

s\_Posechaemost\* p = pos;//новый указатель(необходим для цикла)

pos = pos->next;//смещаем "бегунок"

Analyz an;

an.dd = dd;//заполняем данные

an.mm = mm;//

an.nomer\_a = audit;//

an.time = time;//

int sum = 0;//

int teor = 0;//

bool check = false;

while (pos != p)//цикл

{

if (pos->data.dd == dd && pos->data.mm == mm && pos->link\_r->data.nomer\_a == audit && pos->link\_r->data.time == time)//ищем необходимые данные

{

an.den = pos->link\_r->data.den;//заполняем данные

sum +=pos->data.kolvo;//

teor = pos->link\_r->link\_a->data.vmestimost;//

check = true;

}

pos = pos->next;//смещаем бегунок

}

if (teor == 0) teor = 1;

an.zagr = (sum\*100) /teor;//ищем загруженность в процентах

if (check) Dob\_Analyz(analyz, an);

}

void Create\_Posech(s\_Posechaemost\* pos,s\_Raspisanie\* rasp, int kod, int dd, int mm, int kolvo)//метод для создания посещаемости "вручную"

{

Posechaemost p ;//элемент посещаемсть

p.kod = kod;//добавляем данные

p.dd = dd;//

p.mm = mm;//

p.kolvo = kolvo;//

Dob\_Posechaemost(pos, p, rasp);//добавляем в список

}

void Create\_Rasp(s\_Raspisanie\* rasp, s\_Gruppa\* gru, s\_Auditory\* s\_au, int kod, int audit, int gruppa, char\* den, int time)

{

Raspisanie r;

r.kod\_r = kod;

r.nomer\_a = audit;

r.nomer\_g = gruppa;

r.den = den;

r.time = time;

Dob\_Raspisanie(rasp, r, s\_au, gru);

}

//Методы удаления списков

void Del\_El(s\_Analyz\* an)//удаление элемента

{

s\_Analyz\* p;

if (an != NULL)

{

p = an;

an=an->next;

delete(p);

}

}

void Del\_Spisok(s\_Analyz\* an, s\_Analyz\* Head)//удаление списка

{

an = Head;

an->next;

while (an!=Head)

{

Del\_El(an);

}

Del\_El(Head);

}

void Del\_El(s\_Auditory\* an)//удаление элемента

{

s\_Auditory\* p;

if (an != NULL)

{

p = an;

an = an->next;

delete(p);

}

}

void Del\_Spisok(s\_Auditory\* an, s\_Auditory\* Head)//удаление списка

{

an = Head;

an->next;

while (an != Head)

{

Del\_El(an);

}

Del\_El(Head);

}

void Del\_El(s\_Gruppa\* an)//удаление элемента

{

s\_Gruppa\* p;

if (an != NULL)

{

p = an;

an = an->next;

delete(p);

}

}

void Del\_Spisok(s\_Gruppa\* an, s\_Gruppa\* Head)//удаление списка

{

an = Head;

an->next;

while (an != Head)

{

Del\_El(an);

}

Del\_El(Head);

}

void Del\_El(s\_Raspisanie\* an)//удаление элемента

{

s\_Raspisanie\* p;

if (an != NULL)

{

p = an;

an = an->next;

delete(p);

}

}

void Del\_Spisok(s\_Raspisanie\* an, s\_Raspisanie\* Head)//удаление списка

{

an = Head;

an->next;

while (an != Head)

{

Del\_El(an);

}

Del\_El(Head);

}

void Del\_El(s\_Posechaemost\* an)//удаление элемента

{

s\_Posechaemost\* p;

if (an != NULL)

{

p = an;

an = an->next;

delete(p);

}

}

void Del\_Spisok(s\_Posechaemost\* an, s\_Posechaemost\* Head)//удаление списка

{

an = Head;

an->next;

while (an != Head)

{

Del\_El(an);

}

Del\_El(Head);

}

**Form.h**

#pragma once

#include "Header.h"

namespace CppWinForm\_Kurs {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Summary for MyForm

/// </summary>

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: Add the constructor code here

//

}

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::TextBox^ textBox\_firstDD;

protected:

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::Label^ label5;

private: System::Windows::Forms::TextBox^ textBox\_SecondMM;

private: System::Windows::Forms::Label^ label6;

private: System::Windows::Forms::Label^ label7;

private: System::Windows::Forms::TextBox^ textBox\_SecondDD;

private: System::Windows::Forms::Button^ button\_Analiz;

private: System::Windows::Forms::DataGridView^ dataGridView\_Ponedel;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column1;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column2;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column3;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column4;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column5;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column6;

private: System::Windows::Forms::Label^ label\_Pon;

private: System::Windows::Forms::MenuStrip^ menuStrip1;

private: System::Windows::Forms::ToolStripMenuItem^ считатьДанныеToolStripMenuItem;

private: System::Windows::Forms::Label^ label\_Vt;

private: System::Windows::Forms::DataGridView^ dataGridView\_Vtorn;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn1;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn2;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn3;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn4;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn5;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn6;

private: System::Windows::Forms::Label^ label\_SR;

private: System::Windows::Forms::DataGridView^ dataGridView\_Sred;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn7;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn8;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn9;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn10;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn11;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn12;

private: System::Windows::Forms::Label^ label\_Чт;

private: System::Windows::Forms::DataGridView^ dataGridView\_Chetverg;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn13;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn14;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn15;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn16;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn17;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn18;

private: System::Windows::Forms::DataGridView^ dataGridView\_Pyatn;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn19;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn20;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn21;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn22;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn23;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn24;

private: System::Windows::Forms::Label^ label\_Pt;

private: System::Windows::Forms::ToolStripMenuItem^ добавитьДанныеВручнуюToolStripMenuItem;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::TextBox^ textBox\_kode;

private: System::Windows::Forms::ListBox^ listBox\_Grups;

private: System::Windows::Forms::Button^ button\_ChoiceGroup;

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::ListBox^ listBox\_audit;

private: System::Windows::Forms::Button^ button\_den;

private: System::Windows::Forms::ListBox^ listBox\_den;

private: System::Windows::Forms::Button^ button\_time;

private: System::Windows::Forms::ListBox^ listBox\_time;

private: System::Windows::Forms::Label^ label8;

private: System::Windows::Forms::TextBox^ textBox\_Time;

private: System::Windows::Forms::TextBox^ textBox\_Den;

private: System::Windows::Forms::TextBox^ textBox\_audit;

private: System::Windows::Forms::Button^ button\_addRasp;

private: System::Windows::Forms::Label^ label9;

private: System::Windows::Forms::Button^ button\_Rasp;

private: System::Windows::Forms::ListBox^ listBox\_Rasp;

private: System::Windows::Forms::ListBox^ listBox2;

private: System::Windows::Forms::TextBox^ textBox\_Month;

private: System::Windows::Forms::Label^ label10;

private: System::Windows::Forms::Label^ label11;

private: System::Windows::Forms::TextBox^ textBox\_Chis;

private: System::Windows::Forms::TextBox^ textBox\_Kolvo;

private: System::Windows::Forms::Label^ label12;

private: System::Windows::Forms::TextBox^ textBox\_Gr;

private: System::Windows::Forms::TextBox^ textBox\_Rasp;

private: System::Windows::Forms::Button^ button\_AddPos;

private: System::Windows::Forms::Button^ button\_Сортировать;

private: System::Windows::Forms::TextBox^ textBox\_Sort;

private: System::Windows::Forms::Label^ label13;

private: System::Windows::Forms::Label^ label14;

private: System::Windows::Forms::Button^ button2;

private: System::Windows::Forms::ToolStripMenuItem^ закрытьToolStripMenuItem;

private:

/// <summary>

/// Required designer variable.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

void InitializeComponent(void)

{

System::Windows::Forms::DataGridViewCellStyle^ dataGridViewCellStyle1 = (gcnew System::Windows::Forms::DataGridViewCellStyle());

System::Windows::Forms::DataGridViewCellStyle^ dataGridViewCellStyle2 = (gcnew System::Windows::Forms::DataGridViewCellStyle());

System::Windows::Forms::DataGridViewCellStyle^ dataGridViewCellStyle3 = (gcnew System::Windows::Forms::DataGridViewCellStyle());

System::Windows::Forms::DataGridViewCellStyle^ dataGridViewCellStyle4 = (gcnew System::Windows::Forms::DataGridViewCellStyle());

System::Windows::Forms::DataGridViewCellStyle^ dataGridViewCellStyle5 = (gcnew System::Windows::Forms::DataGridViewCellStyle());

this->textBox\_firstDD = (gcnew System::Windows::Forms::TextBox());

this->label1 = (gcnew System::Windows::Forms::Label());

this->label2 = (gcnew System::Windows::Forms::Label());

this->label3 = (gcnew System::Windows::Forms::Label());

this->label5 = (gcnew System::Windows::Forms::Label());

this->textBox\_SecondMM = (gcnew System::Windows::Forms::TextBox());

this->label6 = (gcnew System::Windows::Forms::Label());

this->label7 = (gcnew System::Windows::Forms::Label());

this->textBox\_SecondDD = (gcnew System::Windows::Forms::TextBox());

this->button\_Analiz = (gcnew System::Windows::Forms::Button());

this->dataGridView\_Ponedel = (gcnew System::Windows::Forms::DataGridView());

this->Column1 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column2 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column3 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column4 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column5 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column6 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->label\_Pon = (gcnew System::Windows::Forms::Label());

this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());

this->считатьДанныеToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->добавитьДанныеВручнуюToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->закрытьToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->label\_Vt = (gcnew System::Windows::Forms::Label());

this->dataGridView\_Vtorn = (gcnew System::Windows::Forms::DataGridView());

this->dataGridViewTextBoxColumn1 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn2 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn3 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn4 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn5 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn6 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->label\_SR = (gcnew System::Windows::Forms::Label());

this->dataGridView\_Sred = (gcnew System::Windows::Forms::DataGridView());

this->dataGridViewTextBoxColumn7 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn8 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn9 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn10 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn11 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn12 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->label\_Чт = (gcnew System::Windows::Forms::Label());

this->dataGridView\_Chetverg = (gcnew System::Windows::Forms::DataGridView());

this->dataGridViewTextBoxColumn13 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn14 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn15 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn16 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn17 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn18 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridView\_Pyatn = (gcnew System::Windows::Forms::DataGridView());

this->dataGridViewTextBoxColumn19 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn20 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn21 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn22 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn23 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn24 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->label\_Pt = (gcnew System::Windows::Forms::Label());

this->label4 = (gcnew System::Windows::Forms::Label());

this->textBox\_kode = (gcnew System::Windows::Forms::TextBox());

this->listBox\_Grups = (gcnew System::Windows::Forms::ListBox());

this->button\_ChoiceGroup = (gcnew System::Windows::Forms::Button());

this->button1 = (gcnew System::Windows::Forms::Button());

this->listBox\_audit = (gcnew System::Windows::Forms::ListBox());

this->button\_den = (gcnew System::Windows::Forms::Button());

this->listBox\_den = (gcnew System::Windows::Forms::ListBox());

this->button\_time = (gcnew System::Windows::Forms::Button());

this->listBox\_time = (gcnew System::Windows::Forms::ListBox());

this->label8 = (gcnew System::Windows::Forms::Label());

this->textBox\_Time = (gcnew System::Windows::Forms::TextBox());

this->textBox\_Den = (gcnew System::Windows::Forms::TextBox());

this->textBox\_audit = (gcnew System::Windows::Forms::TextBox());

this->button\_addRasp = (gcnew System::Windows::Forms::Button());

this->label9 = (gcnew System::Windows::Forms::Label());

this->button\_Rasp = (gcnew System::Windows::Forms::Button());

this->listBox\_Rasp = (gcnew System::Windows::Forms::ListBox());

this->listBox2 = (gcnew System::Windows::Forms::ListBox());

this->textBox\_Month = (gcnew System::Windows::Forms::TextBox());

this->label10 = (gcnew System::Windows::Forms::Label());

this->label11 = (gcnew System::Windows::Forms::Label());

this->textBox\_Chis = (gcnew System::Windows::Forms::TextBox());

this->textBox\_Kolvo = (gcnew System::Windows::Forms::TextBox());

this->label12 = (gcnew System::Windows::Forms::Label());

this->textBox\_Gr = (gcnew System::Windows::Forms::TextBox());

this->textBox\_Rasp = (gcnew System::Windows::Forms::TextBox());

this->button\_AddPos = (gcnew System::Windows::Forms::Button());

this->button\_Сортировать = (gcnew System::Windows::Forms::Button());

this->textBox\_Sort = (gcnew System::Windows::Forms::TextBox());

this->label13 = (gcnew System::Windows::Forms::Label());

this->label14 = (gcnew System::Windows::Forms::Label());

this->button2 = (gcnew System::Windows::Forms::Button());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView\_Ponedel))->BeginInit();

this->menuStrip1->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView\_Vtorn))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView\_Sred))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView\_Chetverg))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView\_Pyatn))->BeginInit();

this->SuspendLayout();

//

// textBox\_firstDD

//

this->textBox\_firstDD->Location = System::Drawing::Point(204, 51);

this->textBox\_firstDD->Name = L"textBox\_firstDD";

this->textBox\_firstDD->Size = System::Drawing::Size(43, 20);

this->textBox\_firstDD->TabIndex = 0;

this->textBox\_firstDD->TextChanged += gcnew System::EventHandler(this, &MyForm::textBox\_firstDD\_TextChanged);

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label1->Location = System::Drawing::Point(32, 50);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(121, 19);

this->label1->TabIndex = 1;

this->label1->Text = L"Введите период:";

//

// label2

//

this->label2->AutoSize = true;

this->label2->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label2->Location = System::Drawing::Point(200, 26);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(49, 19);

this->label2->TabIndex = 2;

this->label2->Text = L"число";

//

// label3

//

this->label3->AutoSize = true;

this->label3->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label3->Location = System::Drawing::Point(169, 49);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(16, 19);

this->label3->TabIndex = 3;

this->label3->Text = L"с";

//

// label5

//

this->label5->AutoSize = true;

this->label5->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label5->Location = System::Drawing::Point(256, 50);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(26, 19);

this->label5->TabIndex = 6;

this->label5->Text = L"по";

//

// textBox\_SecondMM

//

this->textBox\_SecondMM->Location = System::Drawing::Point(356, 51);

this->textBox\_SecondMM->Name = L"textBox\_SecondMM";

this->textBox\_SecondMM->Size = System::Drawing::Size(43, 20);

this->textBox\_SecondMM->TabIndex = 10;

this->textBox\_SecondMM->TextChanged += gcnew System::EventHandler(this, &MyForm::textBox\_SecondMM\_TextChanged);

//

// label6

//

this->label6->AutoSize = true;

this->label6->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label6->Location = System::Drawing::Point(352, 26);

this->label6->Name = L"label6";

this->label6->Size = System::Drawing::Size(49, 19);

this->label6->TabIndex = 9;

this->label6->Text = L"месяц";

//

// label7

//

this->label7->AutoSize = true;

this->label7->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label7->Location = System::Drawing::Point(290, 26);

this->label7->Name = L"label7";

this->label7->Size = System::Drawing::Size(49, 19);

this->label7->TabIndex = 8;

this->label7->Text = L"число";

//

// textBox\_SecondDD

//

this->textBox\_SecondDD->Location = System::Drawing::Point(296, 51);

this->textBox\_SecondDD->Name = L"textBox\_SecondDD";

this->textBox\_SecondDD->Size = System::Drawing::Size(43, 20);

this->textBox\_SecondDD->TabIndex = 7;

this->textBox\_SecondDD->TextChanged += gcnew System::EventHandler(this, &MyForm::textBox\_SecondDD\_TextChanged);

//

// button\_Analiz

//

this->button\_Analiz->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button\_Analiz->Location = System::Drawing::Point(432, 35);

this->button\_Analiz->Name = L"button\_Analiz";

this->button\_Analiz->Size = System::Drawing::Size(133, 33);

this->button\_Analiz->TabIndex = 11;

this->button\_Analiz->Text = L"Анализировать";

this->button\_Analiz->UseVisualStyleBackColor = true;

this->button\_Analiz->Click += gcnew System::EventHandler(this, &MyForm::button\_Analiz\_Click);

//

// dataGridView\_Ponedel

//

this->dataGridView\_Ponedel->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView\_Ponedel->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(6) {

this->Column1,

this->Column2, this->Column3, this->Column4, this->Column5, this->Column6

});

this->dataGridView\_Ponedel->Location = System::Drawing::Point(246, 110);

this->dataGridView\_Ponedel->Name = L"dataGridView\_Ponedel";

this->dataGridView\_Ponedel->ReadOnly = true;

this->dataGridView\_Ponedel->Size = System::Drawing::Size(644, 186);

this->dataGridView\_Ponedel->TabIndex = 12;

//

// Column1

//

dataGridViewCellStyle1->Font = (gcnew System::Drawing::Font(L"Times New Roman", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->Column1->DefaultCellStyle = dataGridViewCellStyle1;

this->Column1->HeaderText = L"Аудитории\\Время";

this->Column1->Name = L"Column1";

this->Column1->ReadOnly = true;

//

// Column2

//

this->Column2->HeaderText = L"8:00";

this->Column2->Name = L"Column2";

this->Column2->ReadOnly = true;

//

// Column3

//

this->Column3->HeaderText = L"10:00";

this->Column3->Name = L"Column3";

this->Column3->ReadOnly = true;

//

// Column4

//

this->Column4->HeaderText = L"12:00";

this->Column4->Name = L"Column4";

this->Column4->ReadOnly = true;

//

// Column5

//

this->Column5->HeaderText = L"14:00";

this->Column5->Name = L"Column5";

this->Column5->ReadOnly = true;

//

// Column6

//

this->Column6->HeaderText = L"16:00";

this->Column6->Name = L"Column6";

this->Column6->ReadOnly = true;

//

// label\_Pon

//

this->label\_Pon->AutoSize = true;

this->label\_Pon->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label\_Pon->Location = System::Drawing::Point(72, 110);

this->label\_Pon->Name = L"label\_Pon";

this->label\_Pon->Size = System::Drawing::Size(99, 19);

this->label\_Pon->TabIndex = 13;

this->label\_Pon->Text = L"Понедельник";

//

// menuStrip1

//

this->menuStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(3) {

this->считатьДанныеToolStripMenuItem,

this->добавитьДанныеВручнуюToolStripMenuItem, this->закрытьToolStripMenuItem

});

this->menuStrip1->Location = System::Drawing::Point(0, 0);

this->menuStrip1->Name = L"menuStrip1";

this->menuStrip1->Size = System::Drawing::Size(924, 24);

this->menuStrip1->TabIndex = 14;

this->menuStrip1->Text = L"menuStrip1";

//

// считатьДанныеToolStripMenuItem

//

this->считатьДанныеToolStripMenuItem->Font = (gcnew System::Drawing::Font(L"Times New Roman", 9.75F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(0)));

this->считатьДанныеToolStripMenuItem->Name = L"считатьДанныеToolStripMenuItem";

this->считатьДанныеToolStripMenuItem->Size = System::Drawing::Size(107, 20);

this->считатьДанныеToolStripMenuItem->Text = L"Считать данные";

this->считатьДанныеToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::считатьДанныеToolStripMenuItem\_Click);

//

// добавитьДанныеВручнуюToolStripMenuItem

//

this->добавитьДанныеВручнуюToolStripMenuItem->Enabled = false;

this->добавитьДанныеВручнуюToolStripMenuItem->Name = L"добавитьДанныеВручнуюToolStripMenuItem";

this->добавитьДанныеВручнуюToolStripMenuItem->Size = System::Drawing::Size(167, 20);

this->добавитьДанныеВручнуюToolStripMenuItem->Text = L"Добавить данные вручную";

this->добавитьДанныеВручнуюToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::добавитьДанныеВручнуюToolStripMenuItem\_Click);

//

// закрытьToolStripMenuItem

//

this->закрытьToolStripMenuItem->Name = L"закрытьToolStripMenuItem";

this->закрытьToolStripMenuItem->Size = System::Drawing::Size(65, 20);

this->закрытьToolStripMenuItem->Text = L"Закрыть";

this->закрытьToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::закрытьToolStripMenuItem\_Click);

//

// label\_Vt

//

this->label\_Vt->AutoSize = true;

this->label\_Vt->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label\_Vt->Location = System::Drawing::Point(103, 293);

this->label\_Vt->Name = L"label\_Vt";

this->label\_Vt->Size = System::Drawing::Size(67, 19);

this->label\_Vt->TabIndex = 15;

this->label\_Vt->Text = L"Вторник";

//

// dataGridView\_Vtorn

//

this->dataGridView\_Vtorn->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView\_Vtorn->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(6) {

this->dataGridViewTextBoxColumn1,

this->dataGridViewTextBoxColumn2, this->dataGridViewTextBoxColumn3, this->dataGridViewTextBoxColumn4, this->dataGridViewTextBoxColumn5,

this->dataGridViewTextBoxColumn6

});

this->dataGridView\_Vtorn->Location = System::Drawing::Point(246, 293);

this->dataGridView\_Vtorn->Name = L"dataGridView\_Vtorn";

this->dataGridView\_Vtorn->ReadOnly = true;

this->dataGridView\_Vtorn->Size = System::Drawing::Size(644, 186);

this->dataGridView\_Vtorn->TabIndex = 16;

//

// dataGridViewTextBoxColumn1

//

dataGridViewCellStyle2->Font = (gcnew System::Drawing::Font(L"Times New Roman", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->dataGridViewTextBoxColumn1->DefaultCellStyle = dataGridViewCellStyle2;

this->dataGridViewTextBoxColumn1->HeaderText = L"Аудитории\\Время";

this->dataGridViewTextBoxColumn1->Name = L"dataGridViewTextBoxColumn1";

this->dataGridViewTextBoxColumn1->ReadOnly = true;

//

// dataGridViewTextBoxColumn2

//

this->dataGridViewTextBoxColumn2->HeaderText = L"8:00";

this->dataGridViewTextBoxColumn2->Name = L"dataGridViewTextBoxColumn2";

this->dataGridViewTextBoxColumn2->ReadOnly = true;

//

// dataGridViewTextBoxColumn3

//

this->dataGridViewTextBoxColumn3->HeaderText = L"10:00";

this->dataGridViewTextBoxColumn3->Name = L"dataGridViewTextBoxColumn3";

this->dataGridViewTextBoxColumn3->ReadOnly = true;

//

// dataGridViewTextBoxColumn4

//

this->dataGridViewTextBoxColumn4->HeaderText = L"12:00";

this->dataGridViewTextBoxColumn4->Name = L"dataGridViewTextBoxColumn4";

this->dataGridViewTextBoxColumn4->ReadOnly = true;

//

// dataGridViewTextBoxColumn5

//

this->dataGridViewTextBoxColumn5->HeaderText = L"14:00";

this->dataGridViewTextBoxColumn5->Name = L"dataGridViewTextBoxColumn5";

this->dataGridViewTextBoxColumn5->ReadOnly = true;

//

// dataGridViewTextBoxColumn6

//

this->dataGridViewTextBoxColumn6->HeaderText = L"16:00";

this->dataGridViewTextBoxColumn6->Name = L"dataGridViewTextBoxColumn6";

this->dataGridViewTextBoxColumn6->ReadOnly = true;

//

// label\_SR

//

this->label\_SR->AutoSize = true;

this->label\_SR->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label\_SR->Location = System::Drawing::Point(103, 475);

this->label\_SR->Name = L"label\_SR";

this->label\_SR->Size = System::Drawing::Size(50, 19);

this->label\_SR->TabIndex = 17;

this->label\_SR->Text = L"Среда";

//

// dataGridView\_Sred

//

this->dataGridView\_Sred->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView\_Sred->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(6) {

this->dataGridViewTextBoxColumn7,

this->dataGridViewTextBoxColumn8, this->dataGridViewTextBoxColumn9, this->dataGridViewTextBoxColumn10, this->dataGridViewTextBoxColumn11,

this->dataGridViewTextBoxColumn12

});

this->dataGridView\_Sred->Location = System::Drawing::Point(246, 475);

this->dataGridView\_Sred->Name = L"dataGridView\_Sred";

this->dataGridView\_Sred->ReadOnly = true;

this->dataGridView\_Sred->Size = System::Drawing::Size(644, 186);

this->dataGridView\_Sred->TabIndex = 18;

//

// dataGridViewTextBoxColumn7

//

dataGridViewCellStyle3->Font = (gcnew System::Drawing::Font(L"Times New Roman", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->dataGridViewTextBoxColumn7->DefaultCellStyle = dataGridViewCellStyle3;

this->dataGridViewTextBoxColumn7->HeaderText = L"Аудитории\\Время";

this->dataGridViewTextBoxColumn7->Name = L"dataGridViewTextBoxColumn7";

this->dataGridViewTextBoxColumn7->ReadOnly = true;

//

// dataGridViewTextBoxColumn8

//

this->dataGridViewTextBoxColumn8->HeaderText = L"8:00";

this->dataGridViewTextBoxColumn8->Name = L"dataGridViewTextBoxColumn8";

this->dataGridViewTextBoxColumn8->ReadOnly = true;

//

// dataGridViewTextBoxColumn9

//

this->dataGridViewTextBoxColumn9->HeaderText = L"10:00";

this->dataGridViewTextBoxColumn9->Name = L"dataGridViewTextBoxColumn9";

this->dataGridViewTextBoxColumn9->ReadOnly = true;

//

// dataGridViewTextBoxColumn10

//

this->dataGridViewTextBoxColumn10->HeaderText = L"12:00";

this->dataGridViewTextBoxColumn10->Name = L"dataGridViewTextBoxColumn10";

this->dataGridViewTextBoxColumn10->ReadOnly = true;

//

// dataGridViewTextBoxColumn11

//

this->dataGridViewTextBoxColumn11->HeaderText = L"14:00";

this->dataGridViewTextBoxColumn11->Name = L"dataGridViewTextBoxColumn11";

this->dataGridViewTextBoxColumn11->ReadOnly = true;

//

// dataGridViewTextBoxColumn12

//

this->dataGridViewTextBoxColumn12->HeaderText = L"16:00";

this->dataGridViewTextBoxColumn12->Name = L"dataGridViewTextBoxColumn12";

this->dataGridViewTextBoxColumn12->ReadOnly = true;

//

// label\_Чт

//

this->label\_Чт->AutoSize = true;

this->label\_Чт->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label\_Чт->Location = System::Drawing::Point(103, 658);

this->label\_Чт->Name = L"label\_Чт";

this->label\_Чт->Size = System::Drawing::Size(62, 19);

this->label\_Чт->TabIndex = 19;

this->label\_Чт->Text = L"Четверг";

//

// dataGridView\_Chetverg

//

this->dataGridView\_Chetverg->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView\_Chetverg->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(6) {

this->dataGridViewTextBoxColumn13,

this->dataGridViewTextBoxColumn14, this->dataGridViewTextBoxColumn15, this->dataGridViewTextBoxColumn16, this->dataGridViewTextBoxColumn17,

this->dataGridViewTextBoxColumn18

});

this->dataGridView\_Chetverg->Location = System::Drawing::Point(246, 658);

this->dataGridView\_Chetverg->Name = L"dataGridView\_Chetverg";

this->dataGridView\_Chetverg->ReadOnly = true;

this->dataGridView\_Chetverg->Size = System::Drawing::Size(644, 186);

this->dataGridView\_Chetverg->TabIndex = 20;

//

// dataGridViewTextBoxColumn13

//

dataGridViewCellStyle4->Font = (gcnew System::Drawing::Font(L"Times New Roman", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->dataGridViewTextBoxColumn13->DefaultCellStyle = dataGridViewCellStyle4;

this->dataGridViewTextBoxColumn13->HeaderText = L"Аудитории\\Время";

this->dataGridViewTextBoxColumn13->Name = L"dataGridViewTextBoxColumn13";

this->dataGridViewTextBoxColumn13->ReadOnly = true;

//

// dataGridViewTextBoxColumn14

//

this->dataGridViewTextBoxColumn14->HeaderText = L"8:00";

this->dataGridViewTextBoxColumn14->Name = L"dataGridViewTextBoxColumn14";

this->dataGridViewTextBoxColumn14->ReadOnly = true;

//

// dataGridViewTextBoxColumn15

//

this->dataGridViewTextBoxColumn15->HeaderText = L"10:00";

this->dataGridViewTextBoxColumn15->Name = L"dataGridViewTextBoxColumn15";

this->dataGridViewTextBoxColumn15->ReadOnly = true;

//

// dataGridViewTextBoxColumn16

//

this->dataGridViewTextBoxColumn16->HeaderText = L"12:00";

this->dataGridViewTextBoxColumn16->Name = L"dataGridViewTextBoxColumn16";

this->dataGridViewTextBoxColumn16->ReadOnly = true;

//

// dataGridViewTextBoxColumn17

//

this->dataGridViewTextBoxColumn17->HeaderText = L"14:00";

this->dataGridViewTextBoxColumn17->Name = L"dataGridViewTextBoxColumn17";

this->dataGridViewTextBoxColumn17->ReadOnly = true;

//

// dataGridViewTextBoxColumn18

//

this->dataGridViewTextBoxColumn18->HeaderText = L"16:00";

this->dataGridViewTextBoxColumn18->Name = L"dataGridViewTextBoxColumn18";

this->dataGridViewTextBoxColumn18->ReadOnly = true;

//

// dataGridView\_Pyatn

//

this->dataGridView\_Pyatn->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView\_Pyatn->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(6) {

this->dataGridViewTextBoxColumn19,

this->dataGridViewTextBoxColumn20, this->dataGridViewTextBoxColumn21, this->dataGridViewTextBoxColumn22, this->dataGridViewTextBoxColumn23,

this->dataGridViewTextBoxColumn24

});

this->dataGridView\_Pyatn->Location = System::Drawing::Point(246, 840);

this->dataGridView\_Pyatn->Name = L"dataGridView\_Pyatn";

this->dataGridView\_Pyatn->ReadOnly = true;

this->dataGridView\_Pyatn->Size = System::Drawing::Size(644, 201);

this->dataGridView\_Pyatn->TabIndex = 21;

//

// dataGridViewTextBoxColumn19

//

dataGridViewCellStyle5->Font = (gcnew System::Drawing::Font(L"Times New Roman", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->dataGridViewTextBoxColumn19->DefaultCellStyle = dataGridViewCellStyle5;

this->dataGridViewTextBoxColumn19->HeaderText = L"Аудитории\\Время";

this->dataGridViewTextBoxColumn19->Name = L"dataGridViewTextBoxColumn19";

this->dataGridViewTextBoxColumn19->ReadOnly = true;

//

// dataGridViewTextBoxColumn20

//

this->dataGridViewTextBoxColumn20->HeaderText = L"8:00";

this->dataGridViewTextBoxColumn20->Name = L"dataGridViewTextBoxColumn20";

this->dataGridViewTextBoxColumn20->ReadOnly = true;

//

// dataGridViewTextBoxColumn21

//

this->dataGridViewTextBoxColumn21->HeaderText = L"10:00";

this->dataGridViewTextBoxColumn21->Name = L"dataGridViewTextBoxColumn21";

this->dataGridViewTextBoxColumn21->ReadOnly = true;

//

// dataGridViewTextBoxColumn22

//

this->dataGridViewTextBoxColumn22->HeaderText = L"12:00";

this->dataGridViewTextBoxColumn22->Name = L"dataGridViewTextBoxColumn22";

this->dataGridViewTextBoxColumn22->ReadOnly = true;

//

// dataGridViewTextBoxColumn23

//

this->dataGridViewTextBoxColumn23->HeaderText = L"14:00";

this->dataGridViewTextBoxColumn23->Name = L"dataGridViewTextBoxColumn23";

this->dataGridViewTextBoxColumn23->ReadOnly = true;

//

// dataGridViewTextBoxColumn24

//

this->dataGridViewTextBoxColumn24->HeaderText = L"16:00";

this->dataGridViewTextBoxColumn24->Name = L"dataGridViewTextBoxColumn24";

this->dataGridViewTextBoxColumn24->ReadOnly = true;

//

// label\_Pt

//

this->label\_Pt->AutoSize = true;

this->label\_Pt->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label\_Pt->Location = System::Drawing::Point(103, 840);

this->label\_Pt->Name = L"label\_Pt";

this->label\_Pt->Size = System::Drawing::Size(68, 19);

this->label\_Pt->TabIndex = 22;

this->label\_Pt->Text = L"Пятница";

//

// label4

//

this->label4->AutoSize = true;

this->label4->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label4->Location = System::Drawing::Point(959, 52);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(238, 19);

this->label4->TabIndex = 23;

this->label4->Text = L"Добавить расписание для группы";

//

// textBox\_kode

//

this->textBox\_kode->Location = System::Drawing::Point(963, 99);

this->textBox\_kode->Name = L"textBox\_kode";

this->textBox\_kode->Size = System::Drawing::Size(65, 20);

this->textBox\_kode->TabIndex = 24;

this->textBox\_kode->TextChanged += gcnew System::EventHandler(this, &MyForm::textBox\_kode\_TextChanged);

//

// listBox\_Grups

//

this->listBox\_Grups->Enabled = false;

this->listBox\_Grups->FormattingEnabled = true;

this->listBox\_Grups->Location = System::Drawing::Point(1046, 123);

this->listBox\_Grups->Name = L"listBox\_Grups";

this->listBox\_Grups->Size = System::Drawing::Size(107, 121);

this->listBox\_Grups->TabIndex = 25;

this->listBox\_Grups->Visible = false;

this->listBox\_Grups->MouseDoubleClick += gcnew System::Windows::Forms::MouseEventHandler(this, &MyForm::listBox\_Grups\_MouseDoubleClick);

//

// button\_ChoiceGroup

//

this->button\_ChoiceGroup->Location = System::Drawing::Point(1046, 82);

this->button\_ChoiceGroup->Name = L"button\_ChoiceGroup";

this->button\_ChoiceGroup->Size = System::Drawing::Size(107, 37);

this->button\_ChoiceGroup->TabIndex = 26;

this->button\_ChoiceGroup->Text = L"Выбрать группу";

this->button\_ChoiceGroup->UseVisualStyleBackColor = true;

this->button\_ChoiceGroup->Click += gcnew System::EventHandler(this, &MyForm::button\_ChoiceGroup\_Click);

//

// button1

//

this->button1->Location = System::Drawing::Point(1168, 82);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(107, 37);

this->button1->TabIndex = 27;

this->button1->Text = L"Выбрать аудиторию";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &MyForm::button1\_Click);

//

// listBox\_audit

//

this->listBox\_audit->Enabled = false;

this->listBox\_audit->FormattingEnabled = true;

this->listBox\_audit->Location = System::Drawing::Point(1168, 123);

this->listBox\_audit->Name = L"listBox\_audit";

this->listBox\_audit->Size = System::Drawing::Size(107, 121);

this->listBox\_audit->TabIndex = 28;

this->listBox\_audit->Visible = false;

this->listBox\_audit->MouseDoubleClick += gcnew System::Windows::Forms::MouseEventHandler(this, &MyForm::listBox\_audit\_MouseDoubleClick);

//

// button\_den

//

this->button\_den->Location = System::Drawing::Point(1290, 82);

this->button\_den->Name = L"button\_den";

this->button\_den->Size = System::Drawing::Size(107, 37);

this->button\_den->TabIndex = 29;

this->button\_den->Text = L"День недели";

this->button\_den->UseVisualStyleBackColor = true;

this->button\_den->Click += gcnew System::EventHandler(this, &MyForm::button\_den\_Click);

//

// listBox\_den

//

this->listBox\_den->Enabled = false;

this->listBox\_den->FormattingEnabled = true;

this->listBox\_den->Location = System::Drawing::Point(1290, 125);

this->listBox\_den->Name = L"listBox\_den";

this->listBox\_den->Size = System::Drawing::Size(107, 121);

this->listBox\_den->TabIndex = 30;

this->listBox\_den->Visible = false;

this->listBox\_den->MouseDoubleClick += gcnew System::Windows::Forms::MouseEventHandler(this, &MyForm::listBox\_den\_MouseDoubleClick);

//

// button\_time

//

this->button\_time->Location = System::Drawing::Point(1412, 82);

this->button\_time->Name = L"button\_time";

this->button\_time->Size = System::Drawing::Size(107, 37);

this->button\_time->TabIndex = 31;

this->button\_time->Text = L"Время";

this->button\_time->UseVisualStyleBackColor = true;

this->button\_time->Click += gcnew System::EventHandler(this, &MyForm::button\_time\_Click);

//

// listBox\_time

//

this->listBox\_time->Enabled = false;

this->listBox\_time->FormattingEnabled = true;

this->listBox\_time->Location = System::Drawing::Point(1412, 123);

this->listBox\_time->Name = L"listBox\_time";

this->listBox\_time->Size = System::Drawing::Size(107, 121);

this->listBox\_time->TabIndex = 32;

this->listBox\_time->Visible = false;

this->listBox\_time->MouseDoubleClick += gcnew System::Windows::Forms::MouseEventHandler(this, &MyForm::listBox\_time\_MouseDoubleClick);

//

// label8

//

this->label8->AutoSize = true;

this->label8->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label8->Location = System::Drawing::Point(970, 77);

this->label8->Name = L"label8";

this->label8->Size = System::Drawing::Size(35, 19);

this->label8->TabIndex = 33;

this->label8->Text = L"Код";

//

// textBox\_Time

//

this->textBox\_Time->Enabled = false;

this->textBox\_Time->Location = System::Drawing::Point(1412, 125);

this->textBox\_Time->Name = L"textBox\_Time";

this->textBox\_Time->Size = System::Drawing::Size(107, 20);

this->textBox\_Time->TabIndex = 36;

this->textBox\_Time->Visible = false;

this->textBox\_Time->TextChanged += gcnew System::EventHandler(this, &MyForm::textBox\_Time\_TextChanged);

//

// textBox\_Den

//

this->textBox\_Den->Enabled = false;

this->textBox\_Den->Location = System::Drawing::Point(1290, 125);

this->textBox\_Den->Name = L"textBox\_Den";

this->textBox\_Den->Size = System::Drawing::Size(107, 20);

this->textBox\_Den->TabIndex = 37;

this->textBox\_Den->Visible = false;

this->textBox\_Den->TextChanged += gcnew System::EventHandler(this, &MyForm::textBox\_Den\_TextChanged);

//

// textBox\_audit

//

this->textBox\_audit->Enabled = false;

this->textBox\_audit->Location = System::Drawing::Point(1168, 125);

this->textBox\_audit->Name = L"textBox\_audit";

this->textBox\_audit->Size = System::Drawing::Size(107, 20);

this->textBox\_audit->TabIndex = 35;

this->textBox\_audit->Visible = false;

this->textBox\_audit->TextChanged += gcnew System::EventHandler(this, &MyForm::textBox\_audit\_TextChanged);

//

// button\_addRasp

//

this->button\_addRasp->Enabled = false;

this->button\_addRasp->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button\_addRasp->Location = System::Drawing::Point(1214, 38);

this->button\_addRasp->Name = L"button\_addRasp";

this->button\_addRasp->Size = System::Drawing::Size(133, 33);

this->button\_addRasp->TabIndex = 38;

this->button\_addRasp->Text = L"Добавить";

this->button\_addRasp->UseVisualStyleBackColor = true;

this->button\_addRasp->Click += gcnew System::EventHandler(this, &MyForm::button2\_Click);

//

// label9

//

this->label9->AutoSize = true;

this->label9->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label9->Location = System::Drawing::Point(959, 277);

this->label9->Name = L"label9";

this->label9->Size = System::Drawing::Size(310, 19);

this->label9->TabIndex = 39;

this->label9->Text = L"Добавить посещаемость по коду расписания";

//

// button\_Rasp

//

this->button\_Rasp->Location = System::Drawing::Point(963, 309);

this->button\_Rasp->Name = L"button\_Rasp";

this->button\_Rasp->Size = System::Drawing::Size(107, 37);

this->button\_Rasp->TabIndex = 40;

this->button\_Rasp->Text = L"Выбрать расписание";

this->button\_Rasp->UseVisualStyleBackColor = true;

this->button\_Rasp->Click += gcnew System::EventHandler(this, &MyForm::button\_Rasp\_Click);

//

// listBox\_Rasp

//

this->listBox\_Rasp->Enabled = false;

this->listBox\_Rasp->FormattingEnabled = true;

this->listBox\_Rasp->Location = System::Drawing::Point(963, 358);

this->listBox\_Rasp->Name = L"listBox\_Rasp";

this->listBox\_Rasp->Size = System::Drawing::Size(107, 121);

this->listBox\_Rasp->TabIndex = 41;

this->listBox\_Rasp->Visible = false;

this->listBox\_Rasp->MouseDoubleClick += gcnew System::Windows::Forms::MouseEventHandler(this, &MyForm::listBox\_Rasp\_MouseDoubleClick);

//

// listBox2

//

this->listBox2->Enabled = false;

this->listBox2->FormattingEnabled = true;

this->listBox2->Location = System::Drawing::Point(1844, 633);

this->listBox2->Name = L"listBox2";

this->listBox2->Size = System::Drawing::Size(107, 121);

this->listBox2->TabIndex = 46;

this->listBox2->Visible = false;

//

// textBox\_Month

//

this->textBox\_Month->Location = System::Drawing::Point(1154, 326);

this->textBox\_Month->Name = L"textBox\_Month";

this->textBox\_Month->Size = System::Drawing::Size(43, 20);

this->textBox\_Month->TabIndex = 45;

this->textBox\_Month->TextChanged += gcnew System::EventHandler(this, &MyForm::textBox\_Month\_TextChanged);

//

// label10

//

this->label10->AutoSize = true;

this->label10->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label10->Location = System::Drawing::Point(1148, 301);

this->label10->Name = L"label10";

this->label10->Size = System::Drawing::Size(49, 19);

this->label10->TabIndex = 44;

this->label10->Text = L"месяц";

//

// label11

//

this->label11->AutoSize = true;

this->label11->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label11->Location = System::Drawing::Point(1081, 301);

this->label11->Name = L"label11";

this->label11->Size = System::Drawing::Size(49, 19);

this->label11->TabIndex = 43;

this->label11->Text = L"число";

//

// textBox\_Chis

//

this->textBox\_Chis->Location = System::Drawing::Point(1085, 326);

this->textBox\_Chis->Name = L"textBox\_Chis";

this->textBox\_Chis->Size = System::Drawing::Size(43, 20);

this->textBox\_Chis->TabIndex = 42;

this->textBox\_Chis->TextChanged += gcnew System::EventHandler(this, &MyForm::textBox\_Chis\_TextChanged);

//

// textBox\_Kolvo

//

this->textBox\_Kolvo->Location = System::Drawing::Point(1224, 326);

this->textBox\_Kolvo->Name = L"textBox\_Kolvo";

this->textBox\_Kolvo->Size = System::Drawing::Size(75, 20);

this->textBox\_Kolvo->TabIndex = 47;

this->textBox\_Kolvo->TextChanged += gcnew System::EventHandler(this, &MyForm::textBox\_Kolvo\_TextChanged);

//

// label12

//

this->label12->AutoSize = true;

this->label12->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label12->Location = System::Drawing::Point(1220, 301);

this->label12->Name = L"label12";

this->label12->Size = System::Drawing::Size(89, 19);

this->label12->TabIndex = 48;

this->label12->Text = L"Количество";

//

// textBox\_Gr

//

this->textBox\_Gr->Enabled = false;

this->textBox\_Gr->Location = System::Drawing::Point(1046, 125);

this->textBox\_Gr->Name = L"textBox\_Gr";

this->textBox\_Gr->Size = System::Drawing::Size(107, 20);

this->textBox\_Gr->TabIndex = 34;

this->textBox\_Gr->Visible = false;

this->textBox\_Gr->TextChanged += gcnew System::EventHandler(this, &MyForm::textBox\_Gr\_TextChanged);

//

// textBox\_Rasp

//

this->textBox\_Rasp->Enabled = false;

this->textBox\_Rasp->Location = System::Drawing::Point(963, 358);

this->textBox\_Rasp->Name = L"textBox\_Rasp";

this->textBox\_Rasp->Size = System::Drawing::Size(107, 20);

this->textBox\_Rasp->TabIndex = 49;

this->textBox\_Rasp->Visible = false;

this->textBox\_Rasp->TextChanged += gcnew System::EventHandler(this, &MyForm::textBox\_Rasp\_TextChanged);

//

// button\_AddPos

//

this->button\_AddPos->Enabled = false;

this->button\_AddPos->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button\_AddPos->Location = System::Drawing::Point(1290, 265);

this->button\_AddPos->Name = L"button\_AddPos";

this->button\_AddPos->Size = System::Drawing::Size(133, 33);

this->button\_AddPos->TabIndex = 50;

this->button\_AddPos->Text = L"Добавить";

this->button\_AddPos->UseVisualStyleBackColor = true;

this->button\_AddPos->Click += gcnew System::EventHandler(this, &MyForm::button\_AddPos\_Click);

//

// button\_Сортировать

//

this->button\_Сортировать->Enabled = false;

this->button\_Сортировать->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button\_Сортировать->Location = System::Drawing::Point(581, 35);

this->button\_Сортировать->Name = L"button\_Сортировать";

this->button\_Сортировать->Size = System::Drawing::Size(109, 33);

this->button\_Сортировать->TabIndex = 51;

this->button\_Сортировать->Text = L"Сортировать";

this->button\_Сортировать->UseVisualStyleBackColor = true;

this->button\_Сортировать->Click += gcnew System::EventHandler(this, &MyForm::button\_Сортировать\_Click);

//

// textBox\_Sort

//

this->textBox\_Sort->Location = System::Drawing::Point(731, 43);

this->textBox\_Sort->Name = L"textBox\_Sort";

this->textBox\_Sort->Size = System::Drawing::Size(43, 20);

this->textBox\_Sort->TabIndex = 52;

this->textBox\_Sort->TextChanged += gcnew System::EventHandler(this, &MyForm::textBox1\_TextChanged);

//

// label13

//

this->label13->AutoSize = true;

this->label13->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label13->Location = System::Drawing::Point(699, 44);

this->label13->Name = L"label13";

this->label13->Size = System::Drawing::Size(26, 19);

this->label13->TabIndex = 53;

this->label13->Text = L"по";

//

// label14

//

this->label14->AutoSize = true;

this->label14->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label14->Location = System::Drawing::Point(795, 43);

this->label14->Name = L"label14";

this->label14->Size = System::Drawing::Size(48, 19);

this->label14->TabIndex = 54;

this->label14->Text = L"часам";

//

// button2

//

this->button2->Font = (gcnew System::Drawing::Font(L"Times New Roman", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button2->Location = System::Drawing::Point(1290, 437);

this->button2->Name = L"button2";

this->button2->Size = System::Drawing::Size(133, 33);

this->button2->TabIndex = 55;

this->button2->Text = L"<<Скрыть";

this->button2->UseVisualStyleBackColor = true;

this->button2->Click += gcnew System::EventHandler(this, &MyForm::button2\_Click\_1);

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(924, 1053);

this->Controls->Add(this->button2);

this->Controls->Add(this->label14);

this->Controls->Add(this->label13);

this->Controls->Add(this->textBox\_Sort);

this->Controls->Add(this->button\_Сортировать);

this->Controls->Add(this->button\_AddPos);

this->Controls->Add(this->textBox\_Rasp);

this->Controls->Add(this->label12);

this->Controls->Add(this->textBox\_Kolvo);

this->Controls->Add(this->listBox2);

this->Controls->Add(this->textBox\_Month);

this->Controls->Add(this->label10);

this->Controls->Add(this->label11);

this->Controls->Add(this->textBox\_Chis);

this->Controls->Add(this->listBox\_Rasp);

this->Controls->Add(this->button\_Rasp);

this->Controls->Add(this->label9);

this->Controls->Add(this->button\_addRasp);

this->Controls->Add(this->textBox\_Den);

this->Controls->Add(this->textBox\_Time);

this->Controls->Add(this->textBox\_audit);

this->Controls->Add(this->textBox\_Gr);

this->Controls->Add(this->label8);

this->Controls->Add(this->listBox\_time);

this->Controls->Add(this->button\_time);

this->Controls->Add(this->listBox\_den);

this->Controls->Add(this->button\_den);

this->Controls->Add(this->listBox\_audit);

this->Controls->Add(this->button1);

this->Controls->Add(this->button\_ChoiceGroup);

this->Controls->Add(this->listBox\_Grups);

this->Controls->Add(this->textBox\_kode);

this->Controls->Add(this->label4);

this->Controls->Add(this->label\_Pt);

this->Controls->Add(this->dataGridView\_Pyatn);

this->Controls->Add(this->dataGridView\_Chetverg);

this->Controls->Add(this->label\_Чт);

this->Controls->Add(this->dataGridView\_Sred);

this->Controls->Add(this->label\_SR);

this->Controls->Add(this->dataGridView\_Vtorn);

this->Controls->Add(this->label\_Vt);

this->Controls->Add(this->label\_Pon);

this->Controls->Add(this->dataGridView\_Ponedel);

this->Controls->Add(this->button\_Analiz);

this->Controls->Add(this->textBox\_SecondMM);

this->Controls->Add(this->label6);

this->Controls->Add(this->label7);

this->Controls->Add(this->textBox\_SecondDD);

this->Controls->Add(this->label5);

this->Controls->Add(this->label3);

this->Controls->Add(this->label2);

this->Controls->Add(this->label1);

this->Controls->Add(this->textBox\_firstDD);

this->Controls->Add(this->menuStrip1);

this->MainMenuStrip = this->menuStrip1;

this->Name = L"MyForm";

this->Text = L"Курсовая";

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView\_Ponedel))->EndInit();

this->menuStrip1->ResumeLayout(false);

this->menuStrip1->PerformLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView\_Vtorn))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView\_Sred))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView\_Chetverg))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView\_Pyatn))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

s\_Auditory\* s\_auHead = new s\_Auditory;

s\_Auditory\* s\_au = new s\_Auditory;

s\_Gruppa\* s\_grHead = new s\_Gruppa;

s\_Gruppa\* s\_gr = new s\_Gruppa;

s\_Raspisanie\* s\_raspHead = new s\_Raspisanie;

s\_Raspisanie\* s\_rasp = new s\_Raspisanie;

s\_Posechaemost\* s\_posHead = new s\_Posechaemost;

s\_Posechaemost\* s\_pos = new s\_Posechaemost;

s\_Analyz\* analyzHead = new s\_Analyz;

s\_Analyz\* an = new s\_Analyz;

int selGr, selAu, selTime, kod\_r;

bool check6 = false, check7 = false, check8 = false, check9 = false, check10 = false, check11 = false, check12 = false, check13 = false, check14 = false;

bool check1 = false, check2 = false, check3 = false, check4 = false, check5 = false, check15 = false;

private: System::Void button\_Analiz\_Click(System::Object^ sender, System::EventArgs^ e) {

dataGridView\_Ponedel->Rows->Clear();

dataGridView\_Vtorn->Rows->Clear();

dataGridView\_Sred->Rows->Clear();

dataGridView\_Chetverg->Rows->Clear();

dataGridView\_Pyatn->Rows->Clear();

int dd=0, dd1=0;

int mm = 0;

if (check5)

{

if (check3&&check1&&check4)

{

dd = Convert::ToInt32(textBox\_firstDD->Text);

dd1 = Convert::ToInt32(textBox\_SecondDD->Text);

mm = Convert::ToInt32(textBox\_SecondMM->Text);

if (dd == dd1)

{

for (int i = 8; i <= 16; i = i + 2)

{

s\_au = s\_auHead;

s\_au = s\_au->next;

while (s\_au != s\_auHead)

{

s\_pos = s\_posHead;

Analyzirovat(s\_pos, s\_au->data.nomer\_a, dd, mm, i, an);

s\_au = s\_au->next;

}

}

check2 = true;

}

else for (dd; dd <= dd1; dd++)

{

for (int i = 8; i <= 16; i = i + 2)

{

s\_au = s\_auHead;

s\_au = s\_au->next;

while (s\_au != s\_auHead)

{

s\_pos = s\_posHead;

Analyzirovat(s\_pos, s\_au->data.nomer\_a, dd, mm, i, an);

s\_au = s\_au->next;

}

}

}

check2 = true;

}

else MessageBox::Show("Вы не указали период. ВВедите данные.");

if (check2)

{

an = analyzHead;

an = an->next;

int t = 0;

int v = 0;

int c = 0;

int h = 0;

int g = 0;

while (an != analyzHead)

{

if (an->data.den[0] == 'П' && an->data.den[1] == 'о' && an->data.den[2] == 'н')

{

label\_Pon->Text = "Понедельник " + an->data.dd + " " + an->data.mm;

int k = 0;

bool l = false;

for (int i = 0; i < dataGridView\_Ponedel->RowCount; i++)

{

if (an->data.nomer\_a == Convert::ToInt32(dataGridView\_Ponedel->Rows[i]->Cells[0]->Value))

{

k = i;

l = true;

}

}

if (l)

{

if (an->data.time == 8)

{

dataGridView\_Ponedel->Rows[k]->Cells[1]->Value = an->data.zagr;

}

if (an->data.time == 10)

{

dataGridView\_Ponedel->Rows[k]->Cells[2]->Value = an->data.zagr;

}

if (an->data.time == 12)

{

dataGridView\_Ponedel->Rows[k]->Cells[3]->Value = an->data.zagr;

}

if (an->data.time == 14)

{

dataGridView\_Ponedel->Rows[k]->Cells[4]->Value = an->data.zagr;

}

if (an->data.time == 16)

{

dataGridView\_Ponedel->Rows[k]->Cells[5]->Value = an->data.zagr;

}

}

else

{

dataGridView\_Ponedel->Rows->Add();

dataGridView\_Ponedel->Rows[t]->Cells[0]->Value = an->data.nomer\_a;

if (an->data.time == 8)

{

dataGridView\_Ponedel->Rows[t]->Cells[1]->Value = an->data.zagr;

}

if (an->data.time == 10)

{

dataGridView\_Ponedel->Rows[t]->Cells[2]->Value = an->data.zagr;

}

if (an->data.time == 12)

{

dataGridView\_Ponedel->Rows[t]->Cells[3]->Value = an->data.zagr;

}

if (an->data.time == 14)

{

dataGridView\_Ponedel->Rows[t]->Cells[4]->Value = an->data.zagr;

}

if (an->data.time == 16)

{

dataGridView\_Ponedel->Rows[t]->Cells[5]->Value = an->data.zagr;

}

t++;

}

}

if (an->data.den[0] == 'В' && an->data.den[1] == 'т' && an->data.den[2] == 'о')

{

label\_Vt->Text = "Вторник " + an->data.dd + " " + an->data.mm;

int k = 0;

bool l = false;

for (int i = 0; i < dataGridView\_Vtorn->RowCount; i++)

{

if (an->data.nomer\_a == Convert::ToInt32(dataGridView\_Vtorn->Rows[i]->Cells[0]->Value))

{

k = i;

l = true;

}

}

if (l)

{

if (an->data.time == 8)

{

dataGridView\_Vtorn->Rows[k]->Cells[1]->Value = an->data.zagr;

}

if (an->data.time == 10)

{

dataGridView\_Vtorn->Rows[k]->Cells[2]->Value = an->data.zagr;

}

if (an->data.time == 12)

{

dataGridView\_Vtorn->Rows[k]->Cells[3]->Value = an->data.zagr;

}

if (an->data.time == 14)

{

dataGridView\_Vtorn->Rows[k]->Cells[4]->Value = an->data.zagr;

}

if (an->data.time == 16)

{

dataGridView\_Vtorn->Rows[k]->Cells[5]->Value = an->data.zagr;

}

}

else

{

dataGridView\_Vtorn->Rows->Add();

dataGridView\_Vtorn->Rows[v]->Cells[0]->Value = an->data.nomer\_a;

if (an->data.time == 8)

{

dataGridView\_Vtorn->Rows[v]->Cells[1]->Value = an->data.zagr;

}

if (an->data.time == 10)

{

dataGridView\_Vtorn->Rows[v]->Cells[2]->Value = an->data.zagr;

}

if (an->data.time == 12)

{

dataGridView\_Vtorn->Rows[v]->Cells[3]->Value = an->data.zagr;

}

if (an->data.time == 14)

{

dataGridView\_Vtorn->Rows[v]->Cells[4]->Value = an->data.zagr;

}

if (an->data.time == 16)

{

dataGridView\_Vtorn->Rows[v]->Cells[5]->Value = an->data.zagr;

}

v++;

}

}

if (an->data.den[0] == 'С' && an->data.den[1] == 'р' && an->data.den[2] == 'е')

{

label\_SR->Text = "Среда " + an->data.dd + " " + an->data.mm;

int k = 0;

bool l = false;

for (int i = 0; i < dataGridView\_Sred->RowCount; i++)

{

if (an->data.nomer\_a == Convert::ToInt32(dataGridView\_Sred->Rows[i]->Cells[0]->Value))

{

k = i;

l = true;

}

}

if (l)

{

if (an->data.time == 8)

{

dataGridView\_Sred->Rows[k]->Cells[1]->Value = an->data.zagr;

}

if (an->data.time == 10)

{

dataGridView\_Sred->Rows[k]->Cells[2]->Value = an->data.zagr;

}

if (an->data.time == 12)

{

dataGridView\_Sred->Rows[k]->Cells[3]->Value = an->data.zagr;

}

if (an->data.time == 14)

{

dataGridView\_Sred->Rows[k]->Cells[4]->Value = an->data.zagr;

}

if (an->data.time == 16)

{

dataGridView\_Sred->Rows[k]->Cells[5]->Value = an->data.zagr;

}

}

else

{

dataGridView\_Sred->Rows->Add();

dataGridView\_Sred->Rows[c]->Cells[0]->Value = an->data.nomer\_a;

if (an->data.time == 8)

{

dataGridView\_Sred->Rows[c]->Cells[1]->Value = an->data.zagr;

}

if (an->data.time == 10)

{

dataGridView\_Sred->Rows[c]->Cells[2]->Value = an->data.zagr;

}

if (an->data.time == 12)

{

dataGridView\_Sred->Rows[c]->Cells[3]->Value = an->data.zagr;

}

if (an->data.time == 14)

{

dataGridView\_Sred->Rows[c]->Cells[4]->Value = an->data.zagr;

}

if (an->data.time == 16)

{

dataGridView\_Sred->Rows[c]->Cells[5]->Value = an->data.zagr;

}

c++;

}

}

if (an->data.den[0] == 'Ч' && an->data.den[1] == 'е' && an->data.den[2] == 'т')

{

label\_Чт->Text = "Четверг " + an->data.dd + " " + an->data.mm;

int k = 0;

bool l = false;

for (int i = 0; i < dataGridView\_Chetverg->RowCount; i++)

{

if (an->data.nomer\_a == Convert::ToInt32(dataGridView\_Chetverg->Rows[i]->Cells[0]->Value))

{

k = i;

l = true;

}

}

if (l)

{

if (an->data.time == 8)

{

dataGridView\_Chetverg->Rows[k]->Cells[1]->Value = an->data.zagr;

}

if (an->data.time == 10)

{

dataGridView\_Chetverg->Rows[k]->Cells[2]->Value = an->data.zagr;

}

if (an->data.time == 12)

{

dataGridView\_Chetverg->Rows[k]->Cells[3]->Value = an->data.zagr;

}

if (an->data.time == 14)

{

dataGridView\_Chetverg->Rows[k]->Cells[4]->Value = an->data.zagr;

}

if (an->data.time == 16)

{

dataGridView\_Chetverg->Rows[k]->Cells[5]->Value = an->data.zagr;

}

}

else

{

dataGridView\_Chetverg->Rows->Add();

dataGridView\_Chetverg->Rows[h]->Cells[0]->Value = an->data.nomer\_a;

if (an->data.time == 8)

{

dataGridView\_Chetverg->Rows[h]->Cells[1]->Value = an->data.zagr;

}

if (an->data.time == 10)

{

dataGridView\_Chetverg->Rows[h]->Cells[2]->Value = an->data.zagr;

}

if (an->data.time == 12)

{

dataGridView\_Chetverg->Rows[h]->Cells[3]->Value = an->data.zagr;

}

if (an->data.time == 14)

{

dataGridView\_Chetverg->Rows[h]->Cells[4]->Value = an->data.zagr;

}

if (an->data.time == 16)

{

dataGridView\_Chetverg->Rows[h]->Cells[5]->Value = an->data.zagr;

}

h++;

}

}

if (an->data.den[0] == 'П' && an->data.den[1] == 'я' && an->data.den[2] == 'т')

{

label\_Pt->Text = "Пятница " + an->data.dd + " " + an->data.mm;

int k = 0;

bool l = false;

for (int i = 0; i < dataGridView\_Pyatn->RowCount; i++)

{

if (an->data.nomer\_a == Convert::ToInt32(dataGridView\_Pyatn->Rows[i]->Cells[0]->Value))

{

k = i;

l = true;

}

}

if (l)

{

if (an->data.time == 8)

{

dataGridView\_Pyatn->Rows[k]->Cells[1]->Value = an->data.zagr;

}

if (an->data.time == 10)

{

dataGridView\_Pyatn->Rows[k]->Cells[2]->Value = an->data.zagr;

}

if (an->data.time == 12)

{

dataGridView\_Pyatn->Rows[k]->Cells[3]->Value = an->data.zagr;

}

if (an->data.time == 14)

{

dataGridView\_Pyatn->Rows[k]->Cells[4]->Value = an->data.zagr;

}

if (an->data.time == 16)

{

dataGridView\_Pyatn->Rows[k]->Cells[5]->Value = an->data.zagr;

}

}

else

{

dataGridView\_Pyatn->Rows->Add();

dataGridView\_Pyatn->Rows[g]->Cells[0]->Value = an->data.nomer\_a;

if (an->data.time == 8)

{

dataGridView\_Pyatn->Rows[g]->Cells[1]->Value = an->data.zagr;

}

if (an->data.time == 10)

{

dataGridView\_Pyatn->Rows[g]->Cells[2]->Value = an->data.zagr;

}

if (an->data.time == 12)

{

dataGridView\_Pyatn->Rows[g]->Cells[3]->Value = an->data.zagr;

}

if (an->data.time == 14)

{

dataGridView\_Pyatn->Rows[g]->Cells[4]->Value = an->data.zagr;

}

if (an->data.time == 16)

{

dataGridView\_Pyatn->Rows[g]->Cells[5]->Value = an->data.zagr;

}

g++;

}

}

an = an->next;

}

}

button\_Сортировать->Enabled = true;

}

else MessageBox::Show("Вы не считали данные");

}

private: System::Void считатьДанныеToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

s\_au = new\_Auditory(s\_auHead);

s\_gr = new\_Gruppa(s\_grHead);

s\_rasp = new\_Raspisanie(s\_raspHead);

s\_pos = new\_Posechaemost(s\_posHead);

an = new\_Analyz(analyzHead);

Read\_Auditory(s\_au, "Auditoria.txt");

Read\_Gruppa(s\_gr, "Groups.txt");

Read\_Rasp(s\_rasp, "Raspisanie.txt", s\_au, s\_gr);

Read\_Posech(s\_pos, "Poseshaemost.txt", s\_rasp);

MessageBox::Show("Данные успешно считаны");

добавитьДанныеВручнуюToolStripMenuItem->Enabled = true;

check5 = true;

}

private: System::Void textBox\_firstDD\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

check1 = true;

}

private: System::Void textBox\_SecondDD\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

check3 = true;

}

private: System::Void textBox\_SecondMM\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

check4 = true;

}

private: System::Void добавитьДанныеВручнуюToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

button\_AddPos->Enabled = true;

button\_addRasp->Enabled = true;

Width = 1600;

}

private: System::Void button\_ChoiceGroup\_Click(System::Object^ sender, System::EventArgs^ e) {

listBox\_Grups->Items->Clear();

s\_gr = s\_grHead;

s\_gr = s\_gr->next;

while (s\_gr != s\_grHead)

{

listBox\_Grups->Items->Add(s\_gr->data.nomer\_g);

s\_gr = s\_gr->next;

}

textBox\_Gr->Visible = false;

listBox\_Grups->Enabled = true;

listBox\_Grups->Visible = true;

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

listBox\_audit->Items->Clear();

s\_au = s\_auHead;

s\_au = s\_au->next;

while (s\_au != s\_auHead)

{

listBox\_audit->Items->Add(s\_au->data.nomer\_a);

s\_au = s\_au->next;

}

textBox\_audit->Visible = false;

listBox\_audit->Enabled = true;

listBox\_audit->Visible = true;

}

private: System::Void button\_den\_Click(System::Object^ sender, System::EventArgs^ e) {

listBox\_den->Items->Clear();

listBox\_den->Items->Add("Понедельник");

listBox\_den->Items->Add("Вторник");

listBox\_den->Items->Add("Среда");

listBox\_den->Items->Add("Четверг");

listBox\_den->Items->Add("Пятница");

textBox\_Den->Visible = false;

listBox\_den->Enabled = true;

listBox\_den->Visible = true;

}

private: System::Void button\_time\_Click(System::Object^ sender, System::EventArgs^ e) {

listBox\_time->Items->Clear();

for (int i = 8; i <= 16; i = i + 2)

{

listBox\_time->Items->Add(i);

}

textBox\_Time->Visible = false;

listBox\_time->Enabled = true;

listBox\_time->Visible = true;

}

private: System::Void listBox\_Grups\_MouseDoubleClick(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e) {

textBox\_Gr->Text = listBox\_Grups->SelectedItem->ToString();

selGr = Convert::ToInt32(textBox\_Gr->Text);

textBox\_Gr->Visible = true;

listBox\_Grups->Enabled = false;

listBox\_Grups->Visible = false;

}

private: System::Void listBox\_audit\_MouseDoubleClick(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e) {

textBox\_audit->Text = listBox\_audit->SelectedItem->ToString();

selAu = Convert::ToInt32(textBox\_audit->Text);

textBox\_audit->Visible = true;

listBox\_audit->Enabled = false;

listBox\_audit->Visible = false;

}

private: System::Void listBox\_den\_MouseDoubleClick(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e) {

textBox\_Den->Text = listBox\_den->SelectedItem->ToString();

textBox\_Den->Visible = true;

listBox\_den->Enabled = false;

listBox\_den->Visible = false;

}

private: System::Void listBox\_time\_MouseDoubleClick(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e) {

textBox\_Time->Text = listBox\_time->SelectedItem->ToString();

selTime = Convert::ToInt32(textBox\_Time->Text);

textBox\_Time->Visible = true;

listBox\_time->Enabled = false;

listBox\_time->Visible = false;

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

if (check10&&check6&&check7&&check8&&check9)

{

bool g = true;

s\_rasp = s\_raspHead;

s\_rasp = s\_rasp->next;

kod\_r = Convert::ToInt32(textBox\_kode->Text);

while (s\_rasp != s\_raspHead)

{

if (s\_rasp->data.kod\_r == kod\_r)

{

g = false;

MessageBox::Show("Расписание с таким кодом уже существует, введите другой код");

}

s\_rasp = s\_rasp->next;

}

if (g)

{

char\* den = new char[50];

if (textBox\_Den-> Text == "Понедельник")

{

den = "Понедельник";

}

if (textBox\_Den->Text == "Вторник")

{

den = "Вторник";

}

if (textBox\_Den->Text == "Среда")

{

den = "Среда";

}

if (textBox\_Den->Text == "Четверг")

{

den = "Четверг";

}

if (textBox\_Den->Text == "Четверг")

{

den = "Пятница";

}

Create\_Rasp(s\_rasp, s\_gr, s\_au, kod\_r, selAu, selGr, den,selTime);

MessageBox::Show("Данные о расписании добавлены");

}

}

else MessageBox::Show("Вы ввели не все данные ниже");

}

private: System::Void textBox\_kode\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

check6 = true;

}

private: System::Void textBox\_Gr\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

check7 = true;

}

private: System::Void textBox\_audit\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

check8 = true;

}

private: System::Void textBox\_Den\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

check9 = true;

}

private: System::Void textBox\_Time\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

check10 = true;

}

private: System::Void button\_Rasp\_Click(System::Object^ sender, System::EventArgs^ e) {

textBox\_Rasp->Visible = false;

s\_rasp = s\_raspHead;

s\_rasp = s\_rasp->next;

s\_Posechaemost\* result = new s\_Posechaemost;

bool a = false;

while (s\_rasp != s\_raspHead)

{

s\_pos = s\_posHead;

Poisk\_Posechaemost(s\_pos, s\_rasp->data.kod\_r, result);

if (result->data.kod == -404)

{

listBox\_Rasp->Items->Add(s\_rasp->data.kod\_r);

a = true;

listBox\_Rasp->Visible = true;

listBox\_Rasp->Enabled = true;

}

s\_rasp = s\_rasp->next;

}

if (!a) MessageBox::Show("Информация о посещаемости полностью заполнена");

}

private: System::Void listBox\_Rasp\_MouseDoubleClick(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e) {

textBox\_Rasp->Text = listBox\_Rasp->SelectedItem->ToString();

listBox\_Rasp->Visible = false;

listBox\_Rasp->Enabled = false;

textBox\_Rasp->Visible = true;

}

private: System::Void textBox\_Rasp\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

check11 = true;

}

private: System::Void textBox\_Chis\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

check13 = true;

}

private: System::Void textBox\_Month\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

check14 = true;

}

private: System::Void textBox\_Kolvo\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

check12 = true;

}

private: System::Void button\_AddPos\_Click(System::Object^ sender, System::EventArgs^ e) {

if (check11 && check12 && check13 && check14)

{

bool g = true;

s\_pos = s\_posHead;

s\_pos = s\_pos->next;

int kod = Convert::ToInt32(textBox\_Rasp->Text);

while (s\_pos != s\_posHead)

{

if (s\_pos->data.kod== kod)

{

g = false;

MessageBox::Show("Расписание с таким кодом уже существует, введите другой код");

}

s\_pos = s\_pos->next;

}

int chislo = Convert::ToInt32(textBox\_Chis->Text);

int month = Convert::ToInt32(textBox\_Month->Text);

int kolvo = Convert::ToInt32(textBox\_Kolvo->Text);

if (g) Create\_Posech(s\_pos, s\_rasp, kod, chislo, month, kolvo);

MessageBox::Show("Данные о посещаемости добавлены");

}

else MessageBox::Show("Введите все данные ниже");

}

private: System::Void button\_Сортировать\_Click(System::Object^ sender, System::EventArgs^ e) {

if (check15)

{

Node\* Tree = NULL;

dataGridView\_Ponedel->Rows->Clear();

dataGridView\_Vtorn->Rows->Clear();

dataGridView\_Sred->Rows->Clear();

dataGridView\_Chetverg->Rows->Clear();

dataGridView\_Pyatn->Rows->Clear();

s\_Analyz\* s\_anal = new s\_Analyz;

s\_Analyz\* headAnal = new s\_Analyz;

s\_anal = new\_Analyz(headAnal);

an = analyzHead;

an = an->next;

int sort = Convert::ToInt32(textBox\_Sort->Text);

while (an != analyzHead)

{

if (an->data.time == sort)

{

Dob\_Analyz(s\_anal, an->data);

}

an = an->next;

}

s\_Analyz\* sa = new s\_Analyz;

s\_Analyz\* aH = new s\_Analyz;

sa = new\_Analyz(aH);

s\_anal = headAnal;

s\_anal = s\_anal->next;

while (s\_anal != headAnal)

{

add\_node(s\_anal, Tree);

s\_anal = s\_anal->next;

}

showTree (Tree, sa);

sa = aH;

sa = sa->next;

int t = 0;

int v = 0;

int c = 0;

int h = 0;

int g = 0;

while (sa != aH)

{

if (sa->data.den[0] == 'П' && sa->data.den[1] == 'о' && sa->data.den[2] == 'н')

{

int k = 0;

bool l = false;

for (int i = 0; i < dataGridView\_Ponedel->RowCount; i++)

{

if (sa->data.nomer\_a == Convert::ToInt32(dataGridView\_Ponedel->Rows[i]->Cells[0]->Value))

{

k = i;

l = true;

}

}

if (l)

{

if (sa->data.time == 8)

{

dataGridView\_Ponedel->Rows[k]->Cells[1]->Value = sa->data.zagr;

}

if (sa->data.time == 10)

{

dataGridView\_Ponedel->Rows[k]->Cells[2]->Value = sa->data.zagr;

}

if (sa->data.time == 12)

{

dataGridView\_Ponedel->Rows[k]->Cells[3]->Value = sa->data.zagr;

}

if (sa->data.time == 14)

{

dataGridView\_Ponedel->Rows[k]->Cells[4]->Value = sa->data.zagr;

}

if (sa->data.time == 16)

{

dataGridView\_Ponedel->Rows[k]->Cells[5]->Value = sa->data.zagr;

}

}

else

{

dataGridView\_Ponedel->Rows->Add();

dataGridView\_Ponedel->Rows[t]->Cells[0]->Value = sa->data.nomer\_a;

if (sa->data.time == 8)

{

dataGridView\_Ponedel->Rows[t]->Cells[1]->Value = sa->data.zagr;

}

if (sa->data.time == 10)

{

dataGridView\_Ponedel->Rows[t]->Cells[2]->Value = sa->data.zagr;

}

if (sa->data.time == 12)

{

dataGridView\_Ponedel->Rows[t]->Cells[3]->Value = sa->data.zagr;

}

if (sa->data.time == 14)

{

dataGridView\_Ponedel->Rows[t]->Cells[4]->Value = sa->data.zagr;

}

if (sa->data.time == 16)

{

dataGridView\_Ponedel->Rows[t]->Cells[5]->Value = sa->data.zagr;

}

t++;

}

}

if (sa->data.den[0] == 'В' && sa->data.den[1] == 'т' && sa->data.den[2] == 'о')

{

int k = 0;

bool l = false;

for (int i = 0; i < dataGridView\_Vtorn->RowCount; i++)

{

if (sa->data.nomer\_a == Convert::ToInt32(dataGridView\_Vtorn->Rows[i]->Cells[0]->Value))

{

k = i;

l = true;

}

}

if (l)

{

if (sa->data.time == 8)

{

dataGridView\_Vtorn->Rows[k]->Cells[1]->Value = sa->data.zagr;

}

if (sa->data.time == 10)

{

dataGridView\_Vtorn->Rows[k]->Cells[2]->Value = sa->data.zagr;

}

if (sa->data.time == 12)

{

dataGridView\_Vtorn->Rows[k]->Cells[3]->Value = sa->data.zagr;

}

if (sa->data.time == 14)

{

dataGridView\_Vtorn->Rows[k]->Cells[4]->Value = sa->data.zagr;

}

if (sa->data.time == 16)

{

dataGridView\_Vtorn->Rows[k]->Cells[5]->Value = sa->data.zagr;

}

}

else

{

dataGridView\_Vtorn->Rows->Add();

dataGridView\_Vtorn->Rows[v]->Cells[0]->Value = sa->data.nomer\_a;

if (sa->data.time == 8)

{

dataGridView\_Vtorn->Rows[v]->Cells[1]->Value = sa->data.zagr;

}

if (sa->data.time == 10)

{

dataGridView\_Vtorn->Rows[v]->Cells[2]->Value = sa->data.zagr;

}

if (sa->data.time == 12)

{

dataGridView\_Vtorn->Rows[v]->Cells[3]->Value = sa->data.zagr;

}

if (sa->data.time == 14)

{

dataGridView\_Vtorn->Rows[v]->Cells[4]->Value = sa->data.zagr;

}

if (sa->data.time == 16)

{

dataGridView\_Vtorn->Rows[v]->Cells[5]->Value = sa->data.zagr;

}

v++;

}

}

if (sa->data.den[0] == 'С' && sa->data.den[1] == 'р' && sa->data.den[2] == 'е')

{

int k = 0;

bool l = false;

for (int i = 0; i < dataGridView\_Sred->RowCount; i++)

{

if (sa->data.nomer\_a == Convert::ToInt32(dataGridView\_Sred->Rows[i]->Cells[0]->Value))

{

k = i;

l = true;

}

}

if (l)

{

if (sa->data.time == 8)

{

dataGridView\_Sred->Rows[k]->Cells[1]->Value = sa->data.zagr;

}

if (sa->data.time == 10)

{

dataGridView\_Sred->Rows[k]->Cells[2]->Value = sa->data.zagr;

}

if (sa->data.time == 12)

{

dataGridView\_Sred->Rows[k]->Cells[3]->Value = sa->data.zagr;

}

if (sa->data.time == 14)

{

dataGridView\_Sred->Rows[k]->Cells[4]->Value = sa->data.zagr;

}

if (sa->data.time == 16)

{

dataGridView\_Sred->Rows[k]->Cells[5]->Value = sa->data.zagr;

}

}

else

{

dataGridView\_Sred->Rows->Add();

dataGridView\_Sred->Rows[c]->Cells[0]->Value = sa->data.nomer\_a;

if (sa->data.time == 8)

{

dataGridView\_Sred->Rows[c]->Cells[1]->Value = sa->data.zagr;

}

if (sa->data.time == 10)

{

dataGridView\_Sred->Rows[c]->Cells[2]->Value = sa->data.zagr;

}

if (sa->data.time == 12)

{

dataGridView\_Sred->Rows[c]->Cells[3]->Value = sa->data.zagr;

}

if (sa->data.time == 14)

{

dataGridView\_Sred->Rows[c]->Cells[4]->Value = sa->data.zagr;

}

if (sa->data.time == 16)

{

dataGridView\_Sred->Rows[c]->Cells[5]->Value = sa->data.zagr;

}

c++;

}

}

if (sa->data.den[0] == 'Ч' && sa->data.den[1] == 'е' && sa->data.den[2] == 'т')

{

int k = 0;

bool l = false;

for (int i = 0; i < dataGridView\_Chetverg->RowCount; i++)

{

if (sa->data.nomer\_a == Convert::ToInt32(dataGridView\_Chetverg->Rows[i]->Cells[0]->Value))

{

k = i;

l = true;

}

}

if (l)

{

if (sa->data.time == 8)

{

dataGridView\_Chetverg->Rows[k]->Cells[1]->Value = sa->data.zagr;

}

if (sa->data.time == 10)

{

dataGridView\_Chetverg->Rows[k]->Cells[2]->Value = sa->data.zagr;

}

if (sa->data.time == 12)

{

dataGridView\_Chetverg->Rows[k]->Cells[3]->Value = sa->data.zagr;

}

if (sa->data.time == 14)

{

dataGridView\_Chetverg->Rows[k]->Cells[4]->Value = sa->data.zagr;

}

if (sa->data.time == 16)

{

dataGridView\_Chetverg->Rows[k]->Cells[5]->Value = sa->data.zagr;

}

}

else

{

dataGridView\_Chetverg->Rows->Add();

dataGridView\_Chetverg->Rows[h]->Cells[0]->Value = sa->data.nomer\_a;

if (sa->data.time == 8)

{

dataGridView\_Chetverg->Rows[h]->Cells[1]->Value = sa->data.zagr;

}

if (sa->data.time == 10)

{

dataGridView\_Chetverg->Rows[h]->Cells[2]->Value = sa->data.zagr;

}

if (sa->data.time == 12)

{

dataGridView\_Chetverg->Rows[h]->Cells[3]->Value = sa->data.zagr;

}

if (sa->data.time == 14)

{

dataGridView\_Chetverg->Rows[h]->Cells[4]->Value = sa->data.zagr;

}

if (sa->data.time == 16)

{

dataGridView\_Chetverg->Rows[h]->Cells[5]->Value = sa->data.zagr;

}

h++;

}

}

if (sa->data.den[0] == 'П' && sa->data.den[1] == 'я' && sa->data.den[2] == 'т')

{

int k = 0;

bool l = false;

for (int i = 0; i < dataGridView\_Pyatn->RowCount; i++)

{

if (sa->data.nomer\_a == Convert::ToInt32(dataGridView\_Pyatn->Rows[i]->Cells[0]->Value))

{

k = i;

l = true;

}

}

if (l)

{

if (sa->data.time == 8)

{

dataGridView\_Pyatn->Rows[k]->Cells[1]->Value = sa->data.zagr;

}

if (sa->data.time == 10)

{

dataGridView\_Pyatn->Rows[k]->Cells[2]->Value = sa->data.zagr;

}

if (sa->data.time == 12)

{

dataGridView\_Pyatn->Rows[k]->Cells[3]->Value = sa->data.zagr;

}

if (sa->data.time == 14)

{

dataGridView\_Pyatn->Rows[k]->Cells[4]->Value = sa->data.zagr;

}

if (sa->data.time == 16)

{

dataGridView\_Pyatn->Rows[k]->Cells[5]->Value = sa->data.zagr;

}

}

else

{

dataGridView\_Pyatn->Rows->Add();

dataGridView\_Pyatn->Rows[g]->Cells[0]->Value = sa->data.nomer\_a;

if (sa->data.time == 8)

{

dataGridView\_Pyatn->Rows[g]->Cells[1]->Value = sa->data.zagr;

}

if (sa->data.time == 10)

{

dataGridView\_Pyatn->Rows[g]->Cells[2]->Value = sa->data.zagr;

}

if (sa->data.time == 12)

{

dataGridView\_Pyatn->Rows[g]->Cells[3]->Value = sa->data.zagr;

}

if (sa->data.time == 14)

{

dataGridView\_Pyatn->Rows[g]->Cells[4]->Value = sa->data.zagr;

}

if (sa->data.time == 16)

{

dataGridView\_Pyatn->Rows[g]->Cells[5]->Value = sa->data.zagr;

}

g++;

}

}

sa = sa->next;

}

an = analyzHead;

an = an->next;

while (an != analyzHead)

{

if (an->data.den[0] == 'П' && an->data.den[1] == 'о' && an->data.den[2] == 'н')

{

int k = 0;

bool l = false;

for (int i = 0; i < dataGridView\_Ponedel->RowCount; i++)

{

if (an->data.nomer\_a == Convert::ToInt32(dataGridView\_Ponedel->Rows[i]->Cells[0]->Value))

{

k = i;

l = true;

}

}

if (l)

{

if (an->data.time == 8)

{

dataGridView\_Ponedel->Rows[k]->Cells[1]->Value = an->data.zagr;

}

if (an->data.time == 10)

{

dataGridView\_Ponedel->Rows[k]->Cells[2]->Value = an->data.zagr;

}

if (an->data.time == 12)

{

dataGridView\_Ponedel->Rows[k]->Cells[3]->Value = an->data.zagr;

}

if (an->data.time == 14)

{

dataGridView\_Ponedel->Rows[k]->Cells[4]->Value = an->data.zagr;

}

if (an->data.time == 16)

{

dataGridView\_Ponedel->Rows[k]->Cells[5]->Value = an->data.zagr;

}

}

else

{

dataGridView\_Ponedel->Rows->Add();

dataGridView\_Ponedel->Rows[t]->Cells[0]->Value = an->data.nomer\_a;

if (an->data.time == 8)

{

dataGridView\_Ponedel->Rows[t]->Cells[1]->Value = an->data.zagr;

}

if (an->data.time == 10)

{

dataGridView\_Ponedel->Rows[t]->Cells[2]->Value = an->data.zagr;

}

if (an->data.time == 12)

{

dataGridView\_Ponedel->Rows[t]->Cells[3]->Value = an->data.zagr;

}

if (an->data.time == 14)

{

dataGridView\_Ponedel->Rows[t]->Cells[4]->Value = an->data.zagr;

}

if (an->data.time == 16)

{

dataGridView\_Ponedel->Rows[t]->Cells[5]->Value = an->data.zagr;

}

t++;

}

}

if (an->data.den[0] == 'В' && an->data.den[1] == 'т' && an->data.den[2] == 'о')

{

int k = 0;

bool l = false;

for (int i = 0; i < dataGridView\_Vtorn->RowCount; i++)

{

if (an->data.nomer\_a == Convert::ToInt32(dataGridView\_Vtorn->Rows[i]->Cells[0]->Value))

{

k = i;

l = true;

}

}

if (l)

{

if (an->data.time == 8)

{

dataGridView\_Vtorn->Rows[k]->Cells[1]->Value = an->data.zagr;

}

if (an->data.time == 10)

{

dataGridView\_Vtorn->Rows[k]->Cells[2]->Value = an->data.zagr;

}

if (an->data.time == 12)

{

dataGridView\_Vtorn->Rows[k]->Cells[3]->Value = an->data.zagr;

}

if (an->data.time == 14)

{

dataGridView\_Vtorn->Rows[k]->Cells[4]->Value = an->data.zagr;

}

if (an->data.time == 16)

{

dataGridView\_Vtorn->Rows[k]->Cells[5]->Value = an->data.zagr;

}

}

else

{

dataGridView\_Vtorn->Rows->Add();

dataGridView\_Vtorn->Rows[v]->Cells[0]->Value = an->data.nomer\_a;

if (an->data.time == 8)

{

dataGridView\_Vtorn->Rows[v]->Cells[1]->Value = an->data.zagr;

}

if (an->data.time == 10)

{

dataGridView\_Vtorn->Rows[v]->Cells[2]->Value = an->data.zagr;

}

if (an->data.time == 12)

{

dataGridView\_Vtorn->Rows[v]->Cells[3]->Value = an->data.zagr;

}

if (an->data.time == 14)

{

dataGridView\_Vtorn->Rows[v]->Cells[4]->Value = an->data.zagr;

}

if (an->data.time == 16)

{

dataGridView\_Vtorn->Rows[v]->Cells[5]->Value = an->data.zagr;

}

v++;

}

}

if (an->data.den[0] == 'С' && an->data.den[1] == 'р' && an->data.den[2] == 'е')

{

int k = 0;

bool l = false;

for (int i = 0; i < dataGridView\_Sred->RowCount; i++)

{

if (an->data.nomer\_a == Convert::ToInt32(dataGridView\_Sred->Rows[i]->Cells[0]->Value))

{

k = i;

l = true;

}

}

if (l)

{

if (an->data.time == 8)

{

dataGridView\_Sred->Rows[k]->Cells[1]->Value = an->data.zagr;

}

if (an->data.time == 10)

{

dataGridView\_Sred->Rows[k]->Cells[2]->Value = an->data.zagr;

}

if (an->data.time == 12)

{

dataGridView\_Sred->Rows[k]->Cells[3]->Value = an->data.zagr;

}

if (an->data.time == 14)

{

dataGridView\_Sred->Rows[k]->Cells[4]->Value = an->data.zagr;

}

if (an->data.time == 16)

{

dataGridView\_Sred->Rows[k]->Cells[5]->Value = an->data.zagr;

}

}

else

{

dataGridView\_Sred->Rows->Add();

dataGridView\_Sred->Rows[c]->Cells[0]->Value = an->data.nomer\_a;

if (an->data.time == 8)

{

dataGridView\_Sred->Rows[c]->Cells[1]->Value = an->data.zagr;

}

if (an->data.time == 10)

{

dataGridView\_Sred->Rows[c]->Cells[2]->Value = an->data.zagr;

}

if (an->data.time == 12)

{

dataGridView\_Sred->Rows[c]->Cells[3]->Value = an->data.zagr;

}

if (an->data.time == 14)

{

dataGridView\_Sred->Rows[c]->Cells[4]->Value = an->data.zagr;

}

if (an->data.time == 16)

{

dataGridView\_Sred->Rows[c]->Cells[5]->Value = an->data.zagr;

}

c++;

}

}

if (an->data.den[0] == 'Ч' && an->data.den[1] == 'е' && an->data.den[2] == 'т')

{

int k = 0;

bool l = false;

for (int i = 0; i < dataGridView\_Chetverg->RowCount; i++)

{

if (an->data.nomer\_a == Convert::ToInt32(dataGridView\_Chetverg->Rows[i]->Cells[0]->Value))

{

k = i;

l = true;

}

}

if (l)

{

if (an->data.time == 8)

{

dataGridView\_Chetverg->Rows[k]->Cells[1]->Value = an->data.zagr;

}

if (an->data.time == 10)

{

dataGridView\_Chetverg->Rows[k]->Cells[2]->Value = an->data.zagr;

}

if (an->data.time == 12)

{

dataGridView\_Chetverg->Rows[k]->Cells[3]->Value = an->data.zagr;

}

if (an->data.time == 14)

{

dataGridView\_Chetverg->Rows[k]->Cells[4]->Value = an->data.zagr;

}

if (an->data.time == 16)

{

dataGridView\_Chetverg->Rows[k]->Cells[5]->Value = an->data.zagr;

}

}

else

{

dataGridView\_Chetverg->Rows->Add();

dataGridView\_Chetverg->Rows[h]->Cells[0]->Value = an->data.nomer\_a;

if (an->data.time == 8)

{

dataGridView\_Chetverg->Rows[h]->Cells[1]->Value = an->data.zagr;

}

if (an->data.time == 10)

{

dataGridView\_Chetverg->Rows[h]->Cells[2]->Value = an->data.zagr;

}

if (an->data.time == 12)

{

dataGridView\_Chetverg->Rows[h]->Cells[3]->Value = an->data.zagr;

}

if (an->data.time == 14)

{

dataGridView\_Chetverg->Rows[h]->Cells[4]->Value = an->data.zagr;

}

if (an->data.time == 16)

{

dataGridView\_Chetverg->Rows[h]->Cells[5]->Value = an->data.zagr;

}

h++;

}

}

if (an->data.den[0] == 'П' && an->data.den[1] == 'я' && an->data.den[2] == 'т')

{

int k = 0;

bool l = false;

for (int i = 0; i < dataGridView\_Pyatn->RowCount; i++)

{

if (an->data.nomer\_a == Convert::ToInt32(dataGridView\_Pyatn->Rows[i]->Cells[0]->Value))

{

k = i;

l = true;

}

}

if (l)

{

if (an->data.time == 8)

{

dataGridView\_Pyatn->Rows[k]->Cells[1]->Value = an->data.zagr;

}

if (an->data.time == 10)

{

dataGridView\_Pyatn->Rows[k]->Cells[2]->Value = an->data.zagr;

}

if (an->data.time == 12)

{

dataGridView\_Pyatn->Rows[k]->Cells[3]->Value = an->data.zagr;

}

if (an->data.time == 14)

{

dataGridView\_Pyatn->Rows[k]->Cells[4]->Value = an->data.zagr;

}

if (an->data.time == 16)

{

dataGridView\_Pyatn->Rows[k]->Cells[5]->Value = an->data.zagr;

}

}

else

{

dataGridView\_Pyatn->Rows->Add();

dataGridView\_Pyatn->Rows[g]->Cells[0]->Value = an->data.nomer\_a;

if (an->data.time == 8)

{

dataGridView\_Pyatn->Rows[g]->Cells[1]->Value = an->data.zagr;

}

if (an->data.time == 10)

{

dataGridView\_Pyatn->Rows[g]->Cells[2]->Value = an->data.zagr;

}

if (an->data.time == 12)

{

dataGridView\_Pyatn->Rows[g]->Cells[3]->Value = an->data.zagr;

}

if (an->data.time == 14)

{

dataGridView\_Pyatn->Rows[g]->Cells[4]->Value = an->data.zagr;

}

if (an->data.time == 16)

{

dataGridView\_Pyatn->Rows[g]->Cells[5]->Value = an->data.zagr;

}

g++;

}

}

an = an->next;

}

del(Tree);

}

else MessageBox::Show("Введите по какому столбцу сортировать");

}

private: System::Void textBox1\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

check15 = true;

}

private: System::Void button2\_Click\_1(System::Object^ sender, System::EventArgs^ e) {

Width = 940;

}

private: System::Void закрытьToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

Del\_Spisok(s\_au, s\_auHead);

Del\_Spisok(s\_gr, s\_grHead);

Del\_Spisok(an, analyzHead);

Del\_Spisok(s\_pos, s\_posHead);

Del\_Spisok(s\_rasp, s\_raspHead);

Close();

}

};

}