Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Основы алгоритмизации и программирования

Отчёт

по учебной практике(ознакомительной)

на тему:

Программа составления расписания транспортных средств

Студент Н.А.Гиль

Руководитель А.М.Ковальчук

МИНСК 2022

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ ...........................................................................................................2

1. ВХОДНЫЕ ДАННЫЕ.........................................................................................3

2. ВЫХОДНЫЕ ДАННЫЕ......................................................................................5

3. РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ.....................................................6

3.1. БЛОК-СХЕМА ГОЛОВНОГО МОДУЛЯ..........................................................6

3.2. АЛГОРИТМ ПО ШАГАМ ФУНКЦИИ sort\_route()...........................................................................................................6

3.3. АЛГОРИТМ ПО ШАГАМ ФУНКЦИИ poisk\_routes().......................................................................................................6

4. КОД ПРОГРАММЫ..........................................................................................7

4.1 main.cpp.........................................................................................................7

4.2 functions.cpp..................................................................................................7

4.3 typical.cpp.....................................................................................................17

5. РЕЗУЛЬТАТЫ...................................................................................................20

ЗАКЛЮЧЕНИЕ .....................................................................................................22

СПИСОК ЛИТЕРАТУРЫ .......................................................................................24

**ПРИЛОЖЕНИЕ А**...................................................................................................

**ПРИЛОЖЕНИЕ Б**...................................................................................................

**ПРИЛОЖЕНИЕ В**...................................................................................................

**ВВЕДЕНИЕ**

C — один из самых старых и популярных языков программирования. Он «легкий» и быстрый, поэтому его используют там, где нужна высокая производительность. При этом C сложно изучить — многое приходится писать с нуля. Если сравнивать языки программирования с автомобилями, то C — гоночный болид, неудобный на городских дорогах, но очень быстрый.

***Основные особенности Си:***

1. *простая языковая база, из которой в стандартную библиотеку вынесены многие существенные возможности, вроде математических функций или функций работы с файлами;*
2. *ориентация на процедурное программирование;*
3. *доступ к памяти через использование указателей;*
4. *передача параметров в функцию по значению, а не по ссылке (передача по ссылке эмулируется с помощью указателей);*
5. *наличие указателей на функции и статические переменные;*
6. *области видимости имён;*
7. *структуры и объединения — определяемые пользователем собирательные типы данных, которыми можно манипулировать как одним целым.*

Для создания минималистичного редактора расписания транспортных средств будут применены данные и другие особенности языка.

**ВХОДНЫЕ ДАННЫЕ**

Для упрощения некоторые данные будут заданы константами (константы заданные макросами могут быть изменены перед запуском программы). Ниже приведены участвующее в программе данные и их структуры:

#define number 10 //количество остановок по маршруту

#define length 20 //длина названий остановок

//структура списка в формате стека

struct stack

{

int route; //номер маршрута для удаления

stack\* pointer; //указатель на предыдущий элемент

};

//структура остановки

struct station

{

int number\_of\_station; //номер остановки

int first\_bus\_tm\_hour; //часы прибытия первого автобуса

int first\_bus\_tm\_minute; //минуты прибытия первого автобуса

int last\_bus\_tm\_hour; //часы прибытия последнего автобуса

int last\_bus\_tm\_minute; //минуты прибытия последнего автобуса

};

//информация о маршруте

struct information

{

int number\_route; //номер маршрута

int number\_stations; //количество остановок

station stations[number]; //остановки

int time\_6\_12; //интервал движения с 6 по 12

int time\_12\_18; //интервал движения с 12 по 18

int time\_18\_24; //интервал движения с 18 по 24

};

//структура маршрута

struct route

{

information info; //информация

route\* next; //указатель на следующий элемент

route\* last; //указaтель на предыдущий элемент

};

//структура списка в формате двунаправленной очереди

struct queue

{

int count; //количество элементов в очереди

route\* head; //указатель на голову очереди

route\* tail; //указaтель на хвост очереди

};

char name[] = "file"; //файл для записи маршрутов

char name1[] = "file1"; //файл для записи найденого маршрута

char name2[] = "file2"; //файл для записи маршрутов через выбранную остановку

//названия остановок

char names[number][length] = {"Минск","Брест","Гродно","Витебск","Могилев","Несвиж","Гомель","Столбцы","Пинск","Ивенец" };

queue\* routes = nullptr; //указатель на расписание

Также приведен пример хранения этой информации рисунок 1.

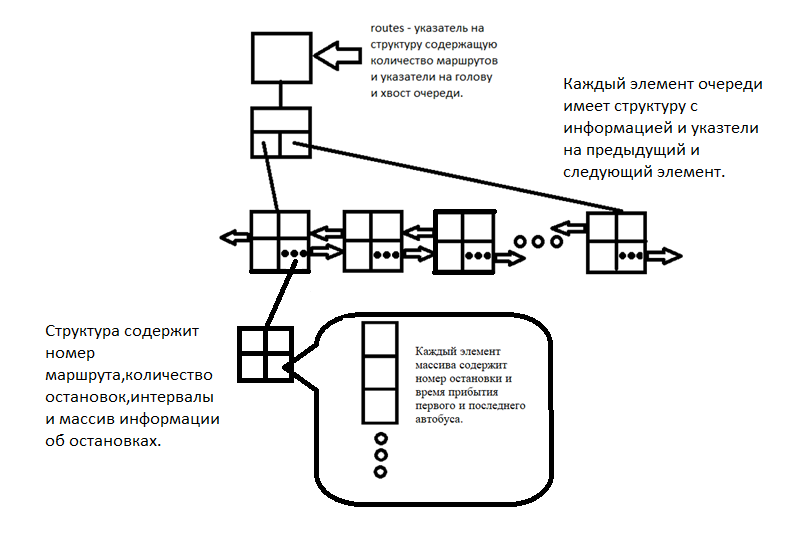


Рисунок 1.

**ВЫХОДНЫЕ ДАННЫЕ**

Выходными данными будут файлы:

char name[] = "file"; //имя файла для записи расписания

char name1[] = "file1"; //имя файла для записи найденного маршрута

char name2[] = "file2"; //имя файла для записи маршрутов через выбранную остановку

**Также стоит отметить**: файл с именем “file” будет хранить готовое расписание, он может быть авторизован при повторном запуске программы (Данные будут занесены в очередь, для дальнейшего использования).

**РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ**

1. **БЛОК-СХЕМА ГОЛОВНОГО МОДУЛЯ:**

Блок-схема головного модуля представлен в **ПРИЛОЖЕНИИ А.**

1. ***АЛГОРИТМ ПО ШАГАМ ФУНКЦИИ* sort\_route():**

Алгоритм по шагам представлен в **ПРИЛОЖЕНИИ Б.**

1. **АЛГОРИТМ ПО ШАГАМ ФУНКЦИИ poisk\_routes():**

Алгоритм по шагам представлен в **ПРИЛОЖЕНИИ В.**

**КОД ПРОГРАММЫ**

* 1. ***Код main.cpp***

#include "header.h"

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

queue\* routes = nullptr; //указатель на расписание

for (;;)

{

system("CLS"); //очистка экрана

printf("\033[32m");

puts("ВАС ПРИВЕТСТВУЕТ РЕДАКТОР ТРАНСПОРТНЫХ РАСПИСАНИЙ GILLTRANS");

printf("\033[0m");

routes = give\_memory(); //выделение памяти

menu(&routes); //вызов меню для выбора способа занесения информации

function\_menu(&routes); //вызов функционального меню

write\_file(routes, "file"); //функция резервной записи информации в файл

clear(&routes); //полная очистка очереди

free(routes);

ending(); //завершение работы

}

}

* 1. ***Код fuctions.cpp***

#include "header.h"

// функция добовления элемента в стек

void push\_front(stack \*\*point, int route)

{

stack\* buff = (stack\*)malloc(sizeof(stack)); //выделение памяти

if (!buff)error();

else

{

buff->route = route; //ввод данных

if (!(\*point))

{

(\*point) = buff;

(\*point)->pointer = nullptr;

}

else

{

buff->pointer = (\*point); //указываем на предыдущий элемент

(\*point) = buff; //меняем вершину стека

}

}

}

//функция удаления элемента из стека

void pop\_back(stack\*\* point)

{

if (!\*point)return;

if (!((\*point)->pointer))

{

free(\*point); //освобождение памяти

\*point = nullptr; //зануление указателя

}

else

{

stack\* buff = (\*point); //запоминаем вершину стека

(\*point) = (\*point)->pointer; //сдвигаем вершину стека

free(buff); //освобождение памяти

}

}

//функция добовления элемента очереди(в хвост)

void push\_front(queue\*\* queues)

{

route\* buff = (route\*)malloc(sizeof(route)); //выделение памяти

if (!buff)error();

else

{

if (!(\*queues)->head and !(\*queues)->tail)

{

(\*queues)->head = (\*queues)->tail = buff;

buff->last = nullptr; //зануления указателя на предыдущий элемент

buff->next = nullptr; //зануления указателя на следующий элемент

(\*queues)->count = 1; //количество маршрутов

}

else

{

buff->next = nullptr; //зануления указателя на следующий элемент

buff->last = (\*queues)->tail; //указание на предыдущий элемент

(\*queues)->tail->next = buff; //указание на следующий

(\*queues)->tail = buff; //новый хвост

(\*queues)->count++; //количество маршрутов

}

}

}

//функция удаления с хвоста

void pop\_back(queue\*\* queues)

{

if (!(\*queues)->head and !(\*queues)->tail)

{

puts("Очередь пуста!");

return;

}

else

{

if ((\*queues)->tail->last == nullptr)

{

free((\*queues)->tail); //очистка

(\*queues)->tail = (\*queues)->head = nullptr; //зануление

(\*queues)->count = 0; //количество маршрутов

}

else

{

route\* buff = (\*queues)->tail;

(\*queues)->tail = (\*queues)->tail->last; //новый хвост

(\*queues)->tail->next = nullptr; //зануление

free(buff); //очистка

(\*queues)->count--; //количество маршрутов

}

}

}

//полная очистка очереди

void clear(queue\*\* queues)

{

while (((\*queues)->tail) or (\*queues)->head)pop\_back(queues);

}

//функция меню

void menu(queue\*\* queues)

{

printf("\033[33m");

puts("Выберите способ получения информации:");

printf("\033[0m");

printf("1 - ввод новой информации с клавиатуры.\n"

"2 - использование старой информации из файла.\n");

int choice = input(1, 2); //выбор

if (choice == 1)

{

input\_information(queues); //функция ввода информации

}

else

{

read\_file(queues); //функция чтения информации из файла

}

}

//функция ввода информации

void input\_information(queue\*\* queues)

{

for (;;)

{

push\_front(queues); //добавление элемента

puts("ВВЕДИТЕ НОМЕР МАРШРУТА:");

(\*queues)->tail->info.number\_route = input(1, 100);

puts("ВВЕДИТЕ КОЛИЧЕСТВО ОСТАНОВОК:");

(\*queues)->tail->info.number\_stations = input(2, 10);

puts("ВВЕДИТЕ ИНТЕРВАЛ С 6 ДО 12:");

(\*queues)->tail->info.time\_6\_12 = input(1, 10);

puts("ВВЕДИТЕ ИНТЕРВАЛ С 12 ДО 18:");

(\*queues)->tail->info.time\_12\_18 = input(1, 10);

puts("ВВЕДИТЕ ИНТЕРВАЛ С 18 ДО 24:");

(\*queues)->tail->info.time\_18\_24 = input(1, 10);

int flag = 0; //флаг для ввода времени начала движения

//цикл по остановкам

for (int g = 0; g < (\*queues)->tail->info.number\_stations; g++)

{

if (!flag)

{

puts("ВВЕДИТЕ ЧАСЫ ПРИБЫТИЯ ПЕРВОГО АВТОБУСА:");

(\*queues)->tail->info.stations[g].first\_bus\_tm\_hour = input(6, 23);

puts("ВВЕДИТЕ МИНУТЫ ПРИБЫТИЯ ПЕРВОГО АВТОБУСА:");

(\*queues)->tail->info.stations[g].first\_bus\_tm\_minute = input(0, 59);

puts("ВВЕДИТЕ ЧАСЫ ПРИБЫТИЯ ПОСЛЕДНЕГО АВТОБУСА:");

(\*queues)->tail->info.stations[g].last\_bus\_tm\_hour = input(6, 23);

puts("ВВЕДИТЕ МИНУТЫ ПРИБЫТИЯ ПОСЛЕДНЕГО АВТОБУСА:");

(\*queues)->tail->info.stations[g].last\_bus\_tm\_minute = input(0, 59);

//функция заполнения времени прибытия первого и последнего автобуса

input\_all\_time((\*queues)->tail, (\*queues)->tail->info.number\_stations);

system("CLS"); //очистка экрана

output\_stations(); //функция вывода остановок

flag = 1;

}

printf("ВВЕДИТЕ НОМЕР %d ОСТАНОВКИ:\n", g + 1);

(\*queues)->tail->info.stations[g].number\_of\_station = input(1, 10);

}

puts("Хотите продолжить?(Y/N)");

for (;;)

{

rewind(stdin); //очистка буфера

char choice = getchar(); //ввод

if (choice == 'N' or choice == 'n')return; //выход из программы

else if (choice == 'Y' or choice == 'y') break; //выход из функции

else

{

printf("\033[31m");

puts("--->ОШИБКА<---");

printf("\033[0m");

}

}

}

}

//функция вывода остановок

void output\_stations()

{

//вывод названий остановок

for (int i = 0; i < number; i++)printf("%d - %s\n", i + 1, names[i]);

}

//функция заполнения времени прибытия первого и последнего автобуса

void input\_all\_time(route\* routes, int numbers)

{

//часы прибытия первого автобуса на первую остановку

int timehour1 = routes->info.stations[0].first\_bus\_tm\_hour;

//минуты прибытия первого автобуса на первую остановку

int timeminute1 = routes->info.stations[0].first\_bus\_tm\_minute;

//часы прибытия последнего автобуса на первую остановку

int timehour2 = routes->info.stations[0].last\_bus\_tm\_hour;

//минуты прибытия последнего автобуса на первую остановку

int timeminute2 = routes->info.stations[0].last\_bus\_tm\_minute;

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

{

//добавление интервала

if (timehour1 < 12)timeminute1 += routes->info.time\_6\_12;

else if (timehour1 < 18)timeminute1 += routes->info.time\_12\_18;

else timeminute1 += routes->info.time\_18\_24;

//контроль минут

if (timeminute1 >= 60)

{

timeminute1 -= 60;

++timehour1;

if (timehour1 >= 24)timehour2 -= 24;

}

//добавление интервала

if (timehour2 < 12)timeminute2 += routes->info.time\_6\_12;

else if (timehour2 < 18)timeminute2 += routes->info.time\_12\_18;

else timeminute2 += routes->info.time\_18\_24;

//контроль минут

if (timeminute2 >= 60)

{

timeminute2 -= 60;

++timehour2;

if (timehour2 >= 24)timehour2 -= 24;

}

//часы прибытия первого автобуса на остановку

routes->info.stations[i].first\_bus\_tm\_hour = timehour1;

//минуты прибытия первого автобуса на остановку

routes->info.stations[i].first\_bus\_tm\_minute = timeminute1;

//часы прибытия последнего автобуса на остановку

routes->info.stations[i].last\_bus\_tm\_hour = timehour2;

//минуты прибытия последнего автобуса на остановку

routes->info.stations[i].last\_bus\_tm\_minute = timeminute2;

}

}

//функция вывода информации

void output\_information(queue\* queues)

{

printf("\033[32m");

puts("НОМЕР КОЛИЧЕСТВО 6-12 12-18 18-24 "

"Прибытие первого Прибытие последнего МАРШРУТ");

printf("\033[0m");

//цикл по маршрутам

route\* buff = queues->head; //буферная переменная для вывода

for (; buff; buff = buff->next)

{

printf("%-6d%-11d%-5d%-6d%-6d",

(buff>info.number\_route),(buff>info.number\_stations),(buff->info.time\_6\_12),

(buff->info.time\_12\_18), (buff->info.time\_18\_24));

printf("%2d:", (buff->info.stations[0].first\_bus\_tm\_hour));

if (buff->info.stations[0].first\_bus\_tm\_minute < 10)printf("%d", 0);

printf("%d%14d:",(buff->info.stations[0].first\_bus\_tm\_minute),

(buff->info.stations[0].last\_bus\_tm\_hour));

if (buff->info.stations[0].last\_bus\_tm\_minute < 10)printf("%d", 0);

printf("%d", (buff->info.stations[0].last\_bus\_tm\_minute));

printf(" ");

printf("%-8s-- ", names[(buff->info.stations[0].number\_of\_station) - 1]);

for (int g = 1; g < buff->info.number\_stations; g++)

{

printf("%-8s", names[(buff->info.stations[g].number\_of\_station) - 1]);

if (g != buff->info.number\_stations - 1)printf("-- ");

}

printf("\n");

}

}

//функция записи информации в файл

void write\_file(queue\* queues, const char\* name)

{

FILE\* f;

if (fopen\_s(&f, name, "wt") == 0)

{

fprintf(f, "НОМЕР КОЛИЧЕСТВО 6-12 12-18 18-24 "

"Прибытие первого Прибытие последнего МАРШРУТ\n");

route\* buff = queues->head; //буферная переменная для вывода

//цикл по маршрутам

for (; buff; buff = buff->next)

{

fprintf(f,"%-6d%-11d%-5d%-6d%-6d",(buff->info.number\_route),(buff->info.number\_stations), (buff->info.time\_6\_12), (buff->info.time\_12\_18), (buff->info.time\_18\_24));

fprintf(f, "%2d:", (buff->info.stations[0].first\_bus\_tm\_hour));

if (buff->info.stations[0].first\_bus\_tm\_minute < 10)fprintf(f, "%d", 0);

fprintf(f, "%d%14d:", (buff->info.stations[0].first\_bus\_tm\_minute),

(buff->info.stations[0].last\_bus\_tm\_hour));

if (buff->info.stations[0].last\_bus\_tm\_minute < 10)fprintf(f, "%d", 0);

fprintf(f, "%d", (buff->info.stations[0].last\_bus\_tm\_minute));

fprintf(f, "%18d", (buff->info.stations[0].number\_of\_station));

for (int g = 1; g < buff->info.number\_stations; g++)

{

fprintf(f, "%3d", (buff->info.stations[g].number\_of\_station));

}

fprintf(f,"\n");

}

fclose(f);

}

else error();

printf("\033[33m");

printf("--->резервное копирование завершено в файл %s<---\n", name);

printf("\033[0m");

}

//функция чтения информации из файла

void read\_file(queue\*\* queues)

{

FILE\* f; //указатель на файл

fpos\_t pos; //позиция утпф

if (fopen\_s(&f, name, "rt") == 0)

{

fseek(f, 80, SEEK\_CUR); //пропуск переноса строки и таблицы

//цикл по маршрутам

do

{

push\_front(queues); //добовление элемента

fscanf\_s(f, "%d%d%d%d%d%d", &((\*queues)->tail->info.number\_route),

&((\*queues)->tail->info.number\_stations), &((\*queues)->tail->info.time\_6\_12),

&((\*queues)->tail->info.time\_12\_18), &((\*queues)->tail->info.time\_18\_24),

&((\*queues)->tail->info.stations[0].first\_bus\_tm\_hour));

fseek(f, 1, SEEK\_CUR);

fscanf\_s(f, "%d%d", &((\*queues)->tail->info.stations[0].first\_bus\_tm\_minute),

&((\*queues)->tail->info.stations[0].last\_bus\_tm\_hour));

fseek(f, 1, SEEK\_CUR);

fscanf\_s(f, "%d", &((\*queues)->tail->info.stations[0].last\_bus\_tm\_minute));

input\_all\_time((\*queues)->tail, (\*queues)->tail->info.number\_stations);

//функция заполнения времени прибытия первового и последнего автобуса

for (int g = 0; g < (\*queues)->tail->info.number\_stations; g++)

{

fscanf\_s(f, "%d", &((\*queues)->tail->info.stations[g].number\_of\_station));

}

fseek(f, 2, SEEK\_CUR); //пропуск переноса строки

fgetpos(f, &pos); //получение позиции утпф

} while (\_filelength(\_fileno(f)) > pos);

fclose(f);

}

else error();

}

//функция функционального меню

void function\_menu(queue\*\* queues)

{

for (;;)

{

stack\* number\_route = nullptr; //стек с номерами маршрутов для поиска или удаления

printf("\033[33m");

puts("Выберете опцию:");

printf("\033[0m");

puts("1 - добавление маршрута.\n2 - вывод расписания. "

"\n3 - сортировка по номеру маршрута."

"\n4 - удаление последнего маршрута.\n5 - удаление любого маршрута."

"\n6 - поиск маршрута.\n7 - поиск маршрутов проходящих через выбранную остановку."

"\n8 - выход.");

int choice = input(1, 9); //выбор опции

switch (choice)

{

case 1:

input\_information(queues); //добaвление нового маршрута

break;

case 2:

output\_information(\*queues); //вывод информации

break;

case 3:

sort\_route(queues); //сортировка по номеру маршрута

break;

case 4:

printf("\033[31m");

printf("Маршрут %d удален.\n", (\*queues)->tail->info.number\_route);

printf("\033[0m");

pop\_back(queues); //функция удаления с конца очереди

break;

case 5:

input\_del(queues, &number\_route); //ввод маршшрутов для удаления

delete\_any(queues, &number\_route); //функция удаления любых маршрутов

break;

case 6:

input\_del(queues, &number\_route); //ввод маршрутов для поиска

poisk\_routes(queues, &number\_route); //функция поиска маршрутов

break;

case 7:

poisk\_ost(queues); //функция поиска маршрутов проходящее через остановку

break;

case 8:break;

}

if (choice == 8)break;

}

}

//сортировка по группе

void sort\_route(queue\*\* queues)

{

route\* buff1, \* buff2;

if (!(\*queues)->head)

{

puts("Очередь пуста!");

return;

}

else

{

//цикл от хвоста

for (buff1 = (\*queues)->tail;buff1!=(\*queues)->head;buff1 = buff1->last)

for (buff2 = (\*queues)->head; buff2 != buff1; buff2 = buff2->next)

{

if ((buff2->info.number\_route) > buff2->next->info.number\_route)

{

change(queues, buff2, buff2->next); //функция замены

buff2 = buff2->last; //чтобы не перешагнуть через элемент

if (buff2 == buff1)buff1 = buff1->next; //чтобы цикл завершался

}

}

}

printf("\033[32m");

puts("\*Сортировка выполнена\*");

printf("\033[0m");

}

//функция замены

void change(queue\*\* queues, route\* buff1, route\* buff2)

{

route\* next = buff2->next; //элемент идущий впереди

route\* last = buff1->last; //элемент идущий позади

if (next)next->last = buff1; //если есть элемент идущий впереди

else (\*queues)->tail = buff1;

if (last)last->next = buff2; //если есть элемент идущий позади

else (\*queues)->head = buff2;

buff1->next = next;

buff1->last = buff2;

buff2->next = buff1;

buff2->last = last;

}

//функция удаления любых маршрутов

void delete\_any(queue\*\* queues,stack\*\* number\_route)

{

while (\*number\_route)

{

int flag = 0; //flag для удаления самого первого элемента

route\* buff = ((\*queues)->head); //дополнительная переменная для удаления

for (; buff; buff = buff->next)

{

if (flag)

{

if (!buff)buff = (\*queues)->head; //удалился первый элемент и этот стал последним

else buff = buff->last; //удалился первый элемент и он не последний

flag = 0;

}

if ((\*number\_route)->route == buff->info.number\_route)

{

if (!(buff->last) and !(buff->next))

{

pop\_back(queues); //удаление единственного элемента

buff = nullptr;

}

else

if (!(buff->last))

{

(\*queues)->head = (\*queues)->head->next; //сдвиг головы вперед

(\*queues)->head->last = nullptr; //предыдущего элемента уже нет

free(buff); //очиcтка элемента очереди

buff = (\*queues)->head;

flag = 1; //чтобы не перескочить

}

else

if (!(buff->next))

{

pop\_back(queues); //удаление последнего элемента

buff = nullptr;

}

else

{

route\* point = buff->last; //буферная переменная

buff->last->next = buff->next;

buff->last->next->last = buff->last;

free(buff); //очистка элемента очереди

buff = point;

}

}

if (!buff)break;

else

if (!flag and !(buff->next))break;

}

printf("\033[32m");

printf("%d маршрут удален.\n", (\*number\_route)->route);

printf("\033[0m");

pop\_back(number\_route);

}

}

//функция ввода номеров удаляемых маршрутов

void input\_del(queue\*\* queues, stack\*\* numbers)

{

if (!(\*queues)->head and !(\*queues)->tail)

{

puts("Очередь пуста!");

return;

}

else

{

for (;;)

{

puts("Ведите номер удаляемого(искаймого) маршрута");

int del = input(1, 100); //ввод номера удаляемого маршрута

if (check(\*queues, del))

{

push\_front(numbers, del); //добавление в стек для удаления

}

else

{

printf("\033[31m");

puts("Маршрут отсутствует");

printf("\033[0m");

}

puts("Продолжить ввод?(Y/N)");

for (;;)

{

rewind(stdin); //очистка буффера

char choice = getchar(); //ввод

if (choice == 'N' or choice == 'n')return; //выход из программы

else if (choice == 'Y' or choice == 'y') break; //выход из функции

else

{

printf("\033[31m");

puts("--->ОШИБКА<---");

printf("\033[0m");

}

}

}

}

}

//функция проверки на присутствие маршрута

int check(queue\* queues, int num)

{

route\* buff = queues->head; //переменная для поиска по очереди

for (; buff ; buff = buff->next)

{

if (buff->info.number\_route == num)return 1; //найден в очереди

}

return 0;

}

//функция поиска маршрутов

void poisk\_routes(queue\*\* queues,stack\*\* number\_route)

{

queue\* new\_routes = give\_memory(); //очередь для новых маршрутов

while (\*number\_route)

{

route\* buff = (\*queues)->head; //переменная для поиска

for (; buff; buff = buff->next)

{

if ((\*number\_route)->route == buff->info.number\_route)

{

push\_front(&new\_routes);

copy(new\_routes->tail, buff); //копирование информации

}

}

pop\_back(number\_route);

}

output\_information(new\_routes);

write\_file(new\_routes, name1); //сохранение в файл результата

clear(&new\_routes); //очистка результата

free(new\_routes);

}

//функция копирования информации

void copy(route\* buff1, route\* buff2)

{

//копирование

buff1->info = buff2->info;

}

//функция поиска маршрутов проходящее через остановку

void poisk\_ost(queue\*\* queues)

{

queue\* new\_routes = give\_memory(); //очередь для новых маршрутов

output\_stations(); //вывод остановок

puts("Введите номер остановки");

int number\_ost = input(1, 10); //ввод номер остановки

route\* buff = (\*queues)->head; //переменная для поиска

for (; buff; buff = buff->next)

{

if (check(\*buff, buff->info.number\_stations, number\_ost))

{

push\_front(&new\_routes);

copy(new\_routes->tail, buff); //копирование информации

}

}

output\_information(new\_routes);

write\_file(new\_routes, name2); //сохранение в файл результата

clear(&new\_routes); //очистка результата

free(new\_routes);

}

//функция проверяющая присутствует ли остановка по маршруту

int check(route mass, int count, int num)

{

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

{

if (mass.info.stations[i].number\_of\_station == num)return 1;

}

return 0;

}

* 1. ***Код typical.cpp***

#include "header.h"

//функция окончания программы

void ending()

{

puts("Хотите продолжить?(Y/N)");

for (;;)

{

rewind(stdin); //очистка буфера

char choice = getchar(); //ввод

if (choice == 'N' or choice == 'n')

{

printf("\033[31m");

puts("ЗАВЕРШЕНИЕ");

printf("\033[32m");

puts("СПАСИБО ЧТО ВЫБРАЛИ GILLTRANS");

printf("\033[0m");

exit(0); //выход из программы

}

else

if (choice == 'Y' or choice == 'y') return; //выход из функции

else

{

printf("\033[31m");

puts("--->ОШИБКА<---");

printf("\033[0m");

}

}

}

//экстренное завершение в случае ошибки

void error()

{

puts("ERROR"); //сообщение об ошибке

Sleep(3000); //задержка на 3 секунды

exit(-1); //экстренное завершение

}

//функция ввода типа инт с дополнительной проверкой

int input(int test1, int test2)

{

int number1;

for (;;)

{

rewind(stdin); //очистка буффера

int check = scanf\_s("%d", &number1); //ввод числа

if (check and number1 >= test1 and number1 <= test2) return number1; //возвращение аргумента

else

{

printf("\033[31m");

puts("--->ОШИБКА<---");

printf("\033[0m");

}

}

}

//функция выделения памяти

queue\* give\_memory()

{

queue\* pointer = nullptr;

for (int i = 0; i < 3; i++)

{

pointer = (queue\*)malloc(sizeof(queue)); //выделение памяти

if (!pointer)puts("-->ОШИБКА<--"); //проверка

else

{

pointer->head = nullptr;

pointer->tail = nullptr;

break;

}

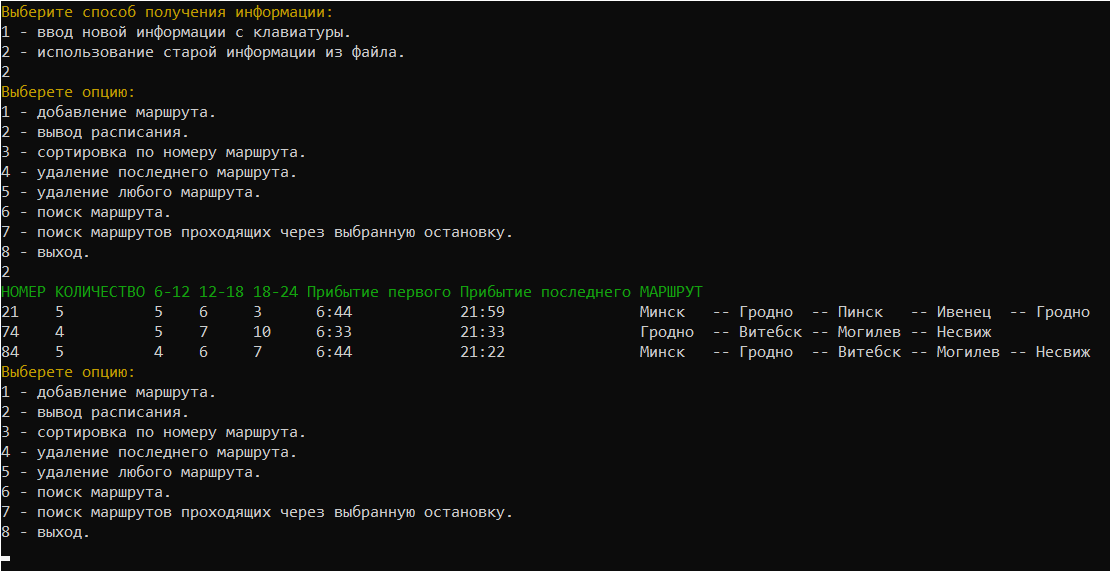
}

return pointer;

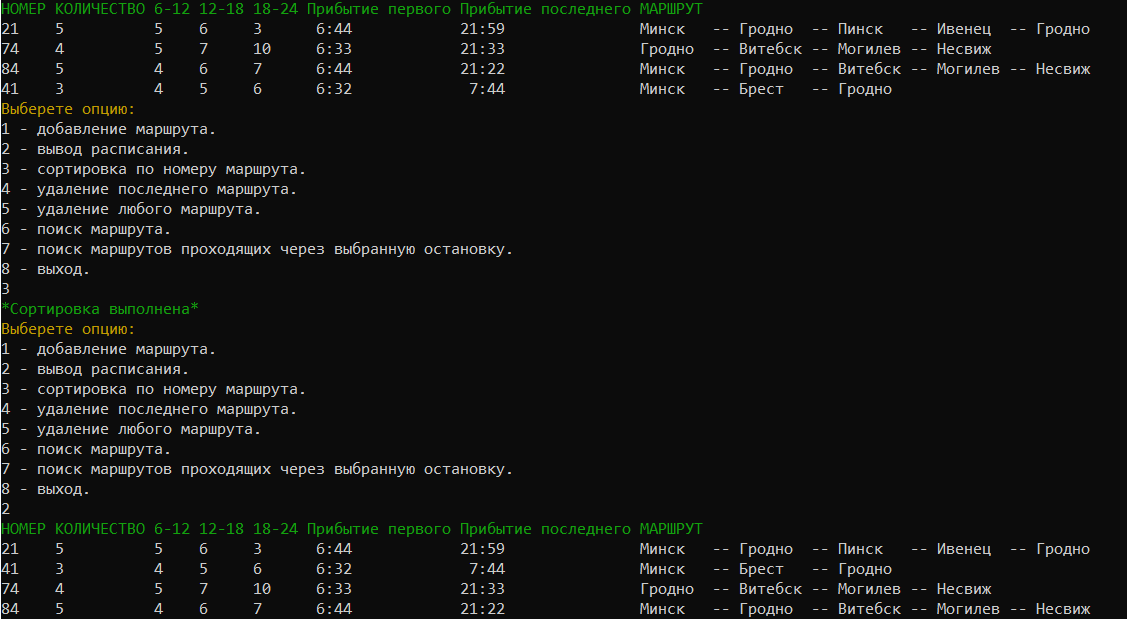
}

**РЕЗУЛЬТАТЫ**

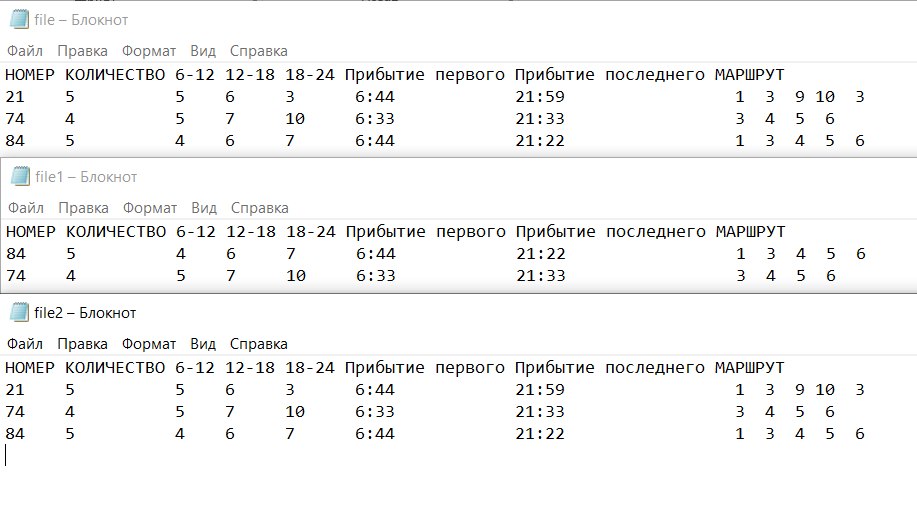
Меню и выведенное расписание:



Сортировка по номеру маршрута:



Результаты в файлах (file – полное расписание, file1 – маршруты выбранные пользователем, file2 – маршруты проходящие через 3 остановку).



**ЗАКЛЮЧЕНИЕ**

**Условия и ограничения для запуска программы:**

* Программа имеет константное значение остановок (их количество задано макросом).
* Номера маршрутов допускаются от 1 до 100.
* Чтобы изменить количество остановок и их названия, делать это нужно перед запуском программы.
* Время работы транспорта ограничено с 6:00 до 24:00.
* Интервалы движения ограничены 1 и 10 минутами.
* Названия остановок в файл не заносятся, а заносятся только их соответствующие номера.
* Также используя изменение маршрутов через текстовые редакторы в файлах информация должна быть корректна, иначе могут возникнуть ошибки в программе.
* Программа может быть скомпилирована с помощью любого Windows-совместимого компилятора и запущена на любом компьютере под управлением ОС Microsoft Windows.

**Применение**

Общественный транспорт играет важную роль в жизни населенных пунктов. И каждому человеку хотелось бы мониторить расписание как минимум в интернете, как максимум на табло около автобусной или других остановках. Если касаться крупных городов, то там уже развивается данное направление. Но для менее масштабных городов и районных центров это редкость. Поэтому мною было принято решение разработать минималистичный редактор расписаний, которым может воспользоваться каждый, чтобы составить нужное расписание.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

Основы алгоритмизации и программирования: язык С: учеб. метод. пособие / Ю. А. Луцик, А. М. Ковальчук, Е. А. Сасин. – Минск :БГУИР, 2015. − 170с. : ил. 1

**ПРИЛОЖЕНИЕ А.**

**ПРИЛОЖЕНИЕ Б.**

**ПРИЛОЖЕНИЕ В.**