МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра САПР

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Приложение для управления процессами операционной системы

|  |  |  |
| --- | --- | --- |
| Преподаватель |  | Калмычков В.А. |

Студент гр. 1335 Максимов Ю.Е.

Санкт-Петербург

20212

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Максимов Ю.Е.

Группа 1335

Тема работы: Приложение для управления процессами операционной системы.

Исходные данные: смоделировать образование очередей на остановках, посадку и высадку пассажиров.

Имеется информация о том, что от пункта А до пункта Б и обратно по разным маршрутам курсирует такси. На нескольких остановках, включая пункты А и Б, собираются очереди из жителей, желающих сесть в такси и доехать до пункта Б и А соответственно, в которых выходят все.

Необходимо обеспечить эффективную выдачу сведений:

* ФИО тех, кто в автобусе
* ФИО тех, кто вышел из автобуса
* ФИО тех, кто ждет автобус на остановках

Содержание пояснительной записки: Введение, теоретическая часть, реализация программы, использованное ПО, результаты работы программы, описание функций, заключение, список использованных источников, приложение 1 – блок-схема, приложение 2 – руководство пользователя, приложение 3 – исходный код.

Предполагаемый объем пояснительной записки:

Не менее 30 страниц.

|  |  |  |
| --- | --- | --- |
| Преподаватель |  | Калмычков В.А. |

Студент гр. 1335 Максимов Ю.Е.

АННОТАЦИЯ

Данная курсовая работа содержит реализацию модели образование очередей на остановках, посадку и высадку пассажиров на языке С++. Программа корректно обрабатывает ошибки, возникающие в процессе выполнения, и выводит сообщение о некорректных действиях пользователя на экран.

SUMMARY

This course work contains the implementation models for queuing at bus stops, boarding and disembarking passengers’ estate in C++. The program correctly handles errors that occur during execution, and displays a message about incorrect user actions on the screen.

СОДЕРЖАНИЕ

1. Введение 6
2. Теоретическая часть
3. Реализация программы
4. Использованное ПО
5. Описание структур, классов, функций, методов
6. Заключение
7. Список использованных источников
8. Приложение 1. Блок-схема
9. Приложение 2. Руководство пользователя
10. Приложение 3. Исходный код

ВВЕДЕНИЕ

В курсовой работе была поставлена задача смоделировать образование очередей на остановках, посадку и высадку пассажиров.

В рамках поставленной задачи была создана программа на языке С++, которая производит чтение заранее созданной базы ФИО и позволяет пользователю получать информацию о том, кто находится в автобусе, кто вышел из автобуса, кто ждет на остановке.

Подробнее об этом и не только можно прочесть далее в изложении теоретического материала.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ
   1. Модель такси

Модель такси подразумевает наличие ограниченной структурированной информации о людях, которые находятся в нем, выходят из него, ждут его, доступ к которой и предоставляет система. При всем при этом модель такси не подразумевает возможности его пополнения новыми данными со стороны пользователя.

Таким образом, программа должна иметь ограниченный набор функций, предоставляющий информацию в удобном виде, но не должна позволять пользователю вносить изменения в базу, чтобы не испортить систему.

* 1. L1-список

L1-список — линейный односвязный список отличается от L2-списка тем, что двигаться по нему можно только в одном направлении (календарь не разрешается листать назад), а вставлять и удалять элементы — только за указателем.

В рамках данной курсовой работы реализуем стандартный L1-список. Для этого нам потребуется структура данных Node1 для хранения данных одного узла и класс L1 для реализации самого списка. Их структура представлена на рисунках 1 и 2.

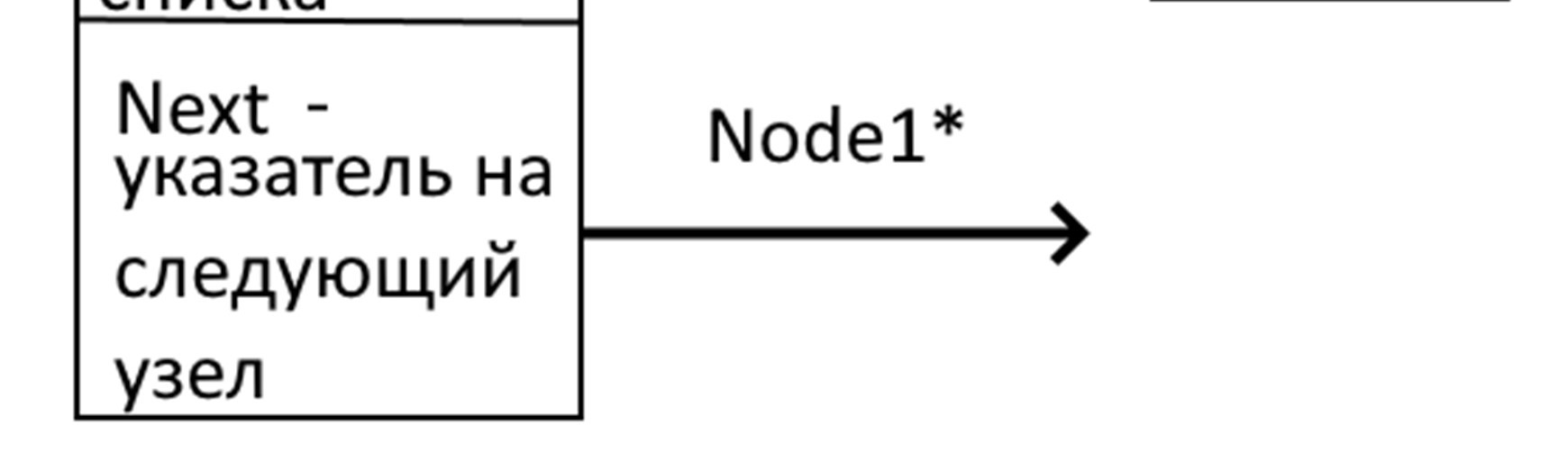
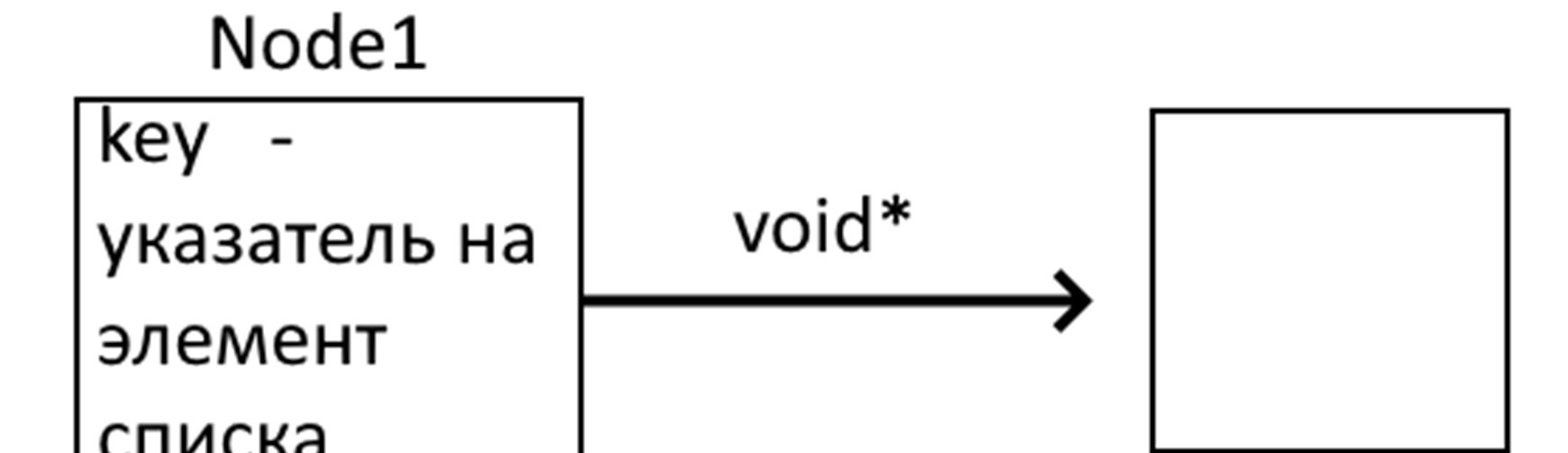


Рисунок 1 – схема устройства узла списка

Т.к. нам потребуется использовать несколько списков будем хранить указатель не на определённый класс или тип, а хранить просто void\* для того, чтобы можно было использовать список с разными данными.

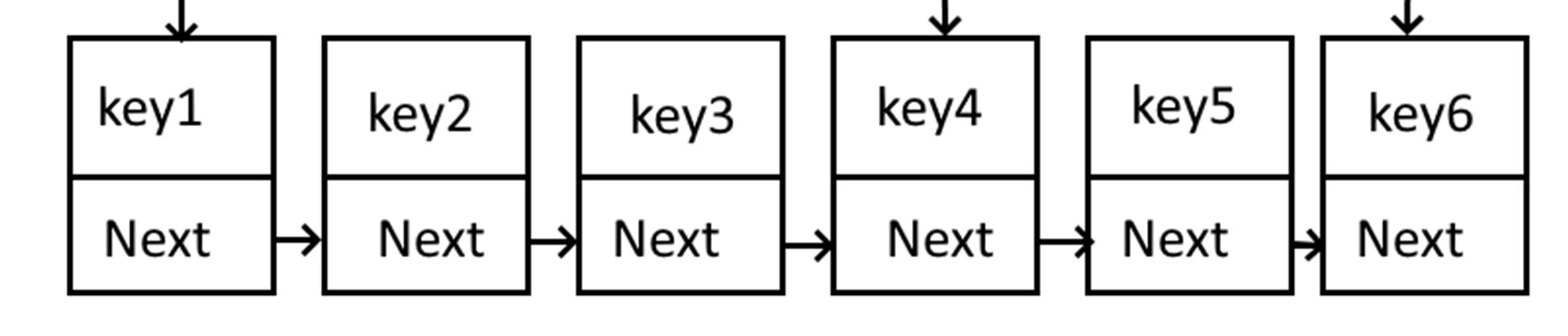
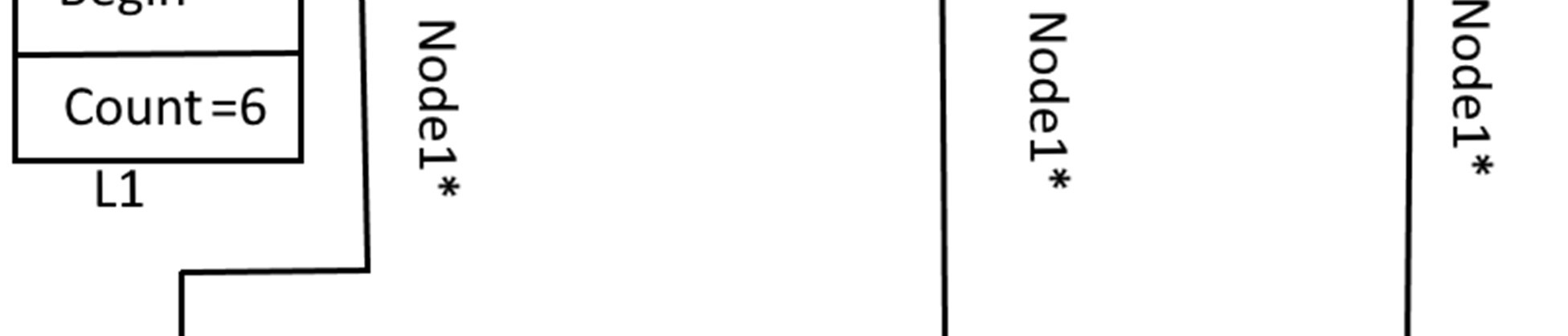
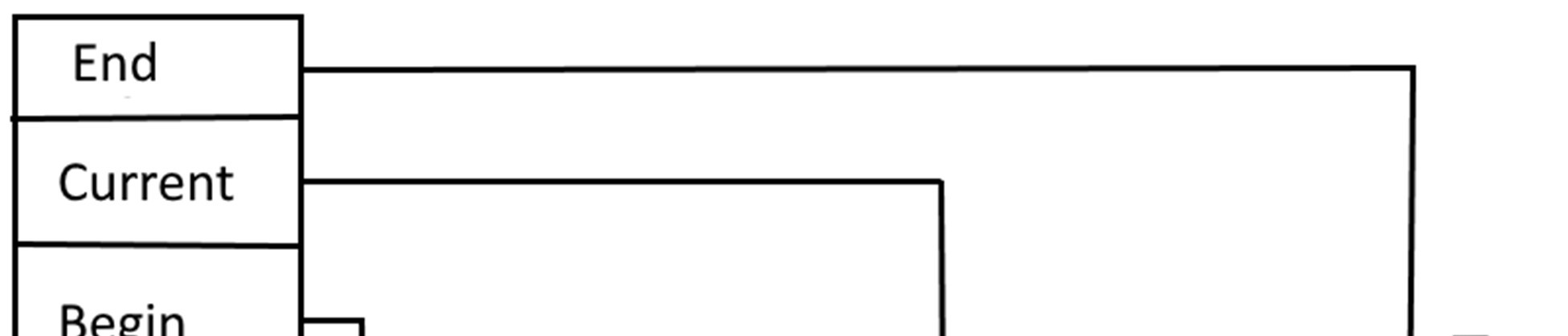


Рисунок 2 – Схема устройства списка, в котором хранится 6 элементов

1.3. Хранение данных в файле

Существует немало способов хранения информации в текстовых файлах. Приведем некоторые из них:

* Json - текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми. Формат JSON был разработан Дугласом Крокфордом. Несмотря на происхождение от JavaScript, формат считается независимым от языка и может использоваться практически с любым языком программирования. Для многих языков существует готовый код для создания и обработки данных в формате JSON. Для с++ также существует библиотека для работы с этим форматом, но т.к. по заданию нельзя использовать сторонние библиотеки, откажемся от этого формата.
* Txt с разделителем – табуляция. Т.к. в разных текстовых редакторах табуляция представляется разными способам.
* CSV (от англ. Comma-Separated Values — значения, разделённые запятыми) — текстовый формат, предназначенный для представления табличных данных. Строка таблицы соответствует строке текста, которая содержит одно или несколько полей, разделенных запятыми. При использовании такого формата можно забыть хранении полей, содержащих запятые. В нашей базе запятая встречается в адресе квартиры, что может привести к некорректному чтению строки. Именно поэтому можно заменить, на ; для решения этой проблемы.

На рисунке 3 пример хранения информации о недвижимости в файле:

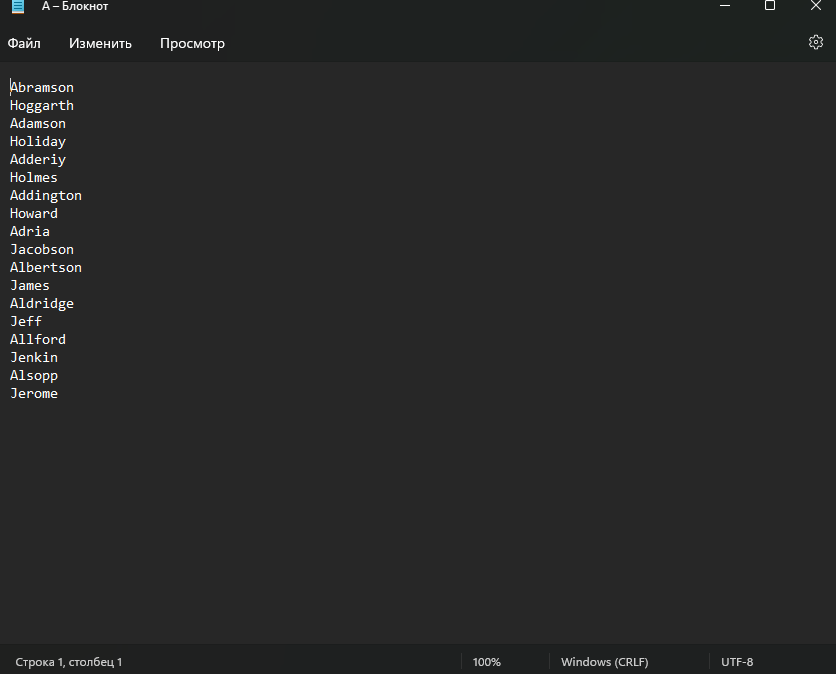


Рисунок 3 – текстовое представление данных в файле

2. РЕАЛИЗАЦИЯ ПРОГРАММЫ

* 1. Использованное ПО

Операционная система: Microsoft Windows 11.

Среда разработки: Microsoft Visual Studio 2022.

Компилятор:Microsoft Visual C++.

* 1. Описание структур, классов, функций, методов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя моду  ля | Имя структуры/класс а/метода | Назначение | Параметры функции | Возвращае мое значение |
| MyString  .h | Класс  MyString | Обрабатывает и хранит данные в виде строк |  |  |
| Конструктор MyString | Определяет размерность строки и выделяет под неё место | Const char\* str |  |
| Деструктор MyString | Удаляет объект и высвобождает динамическую память | Не принимает | Не возвращае  т |
| Lenght | Возвращает размер объекта | Не принимает | lenght |
| Erase | Удаляет | Не принимает | Не возвращае  т |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| list.h | Класс Node | Создает объекты класса MyString хранит информацию и адрес объекта |  |  |
| Конструктор Node | Обрабатывает информацию и указывает начальный адрес объекта | Принимает данные и адрес |  |
| Класс List | Список типа L1. Работает с не типизированными указателями, чтобы предоставить возможность хранить любые типы данных в этом списке. При извлечении элемента из списка потребуется явное приведение, поэтому нужно помнить, какие типы помещаются в каждый созданный список |  |  |
| Конструктор List | Обрабатывает информацию и указывает начальный адрес объекта, а также определяет размерностью | Не принимает |  |
| pop\_front | Удаляет объект Node, в начала списка List | Не принимает | Не возвращае  т |
| push\_back | Добавляет новый объект Node в конец списка List | Принимает объект класса MySrting | Не возвращае  т |
| clear | Удаляет все элементы | Не принимает | Не возвращае  т |
| Деструктор ~List | Корректно освобождает  захваченные ресурсы |  |  |

ЗАКЛЮЧЕНИЕ

В процессе написания данной курсовой работы я изучил устройство списков и некоторых других динамических структур данных, изучил различные способы хранения данных в файлах, а также улучшил свои навыки программирования на языке с++.

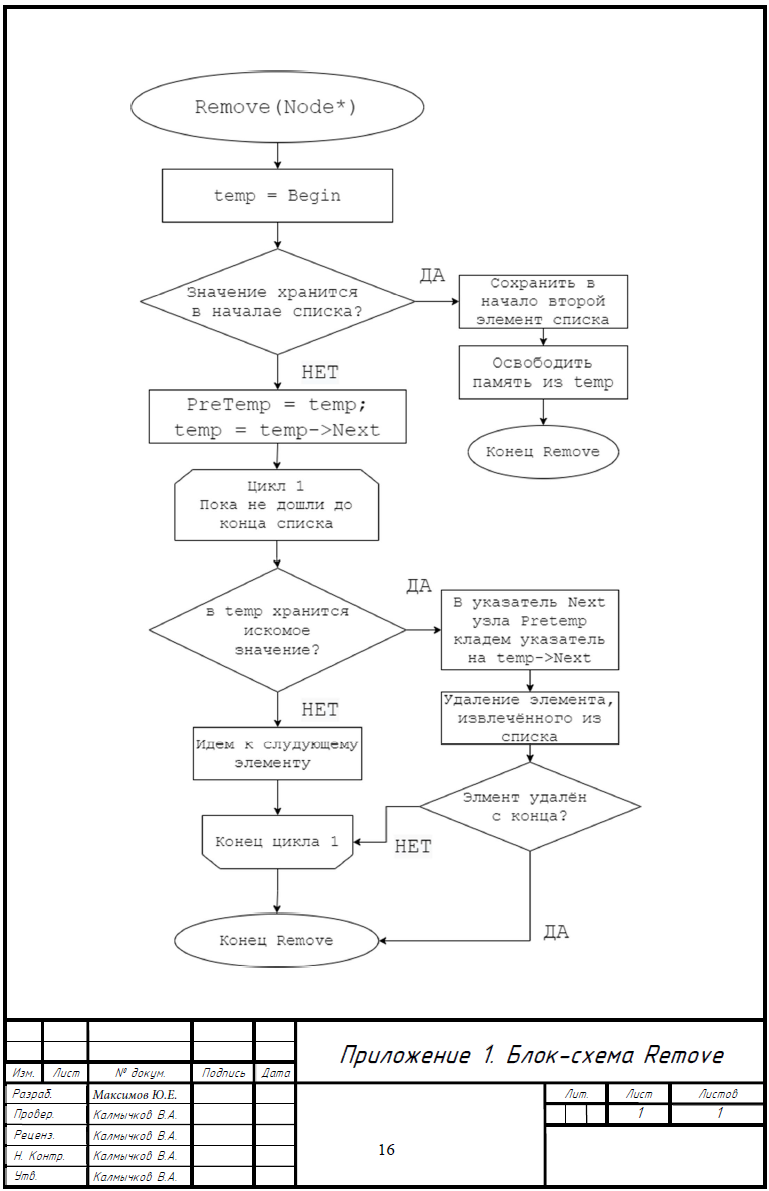
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

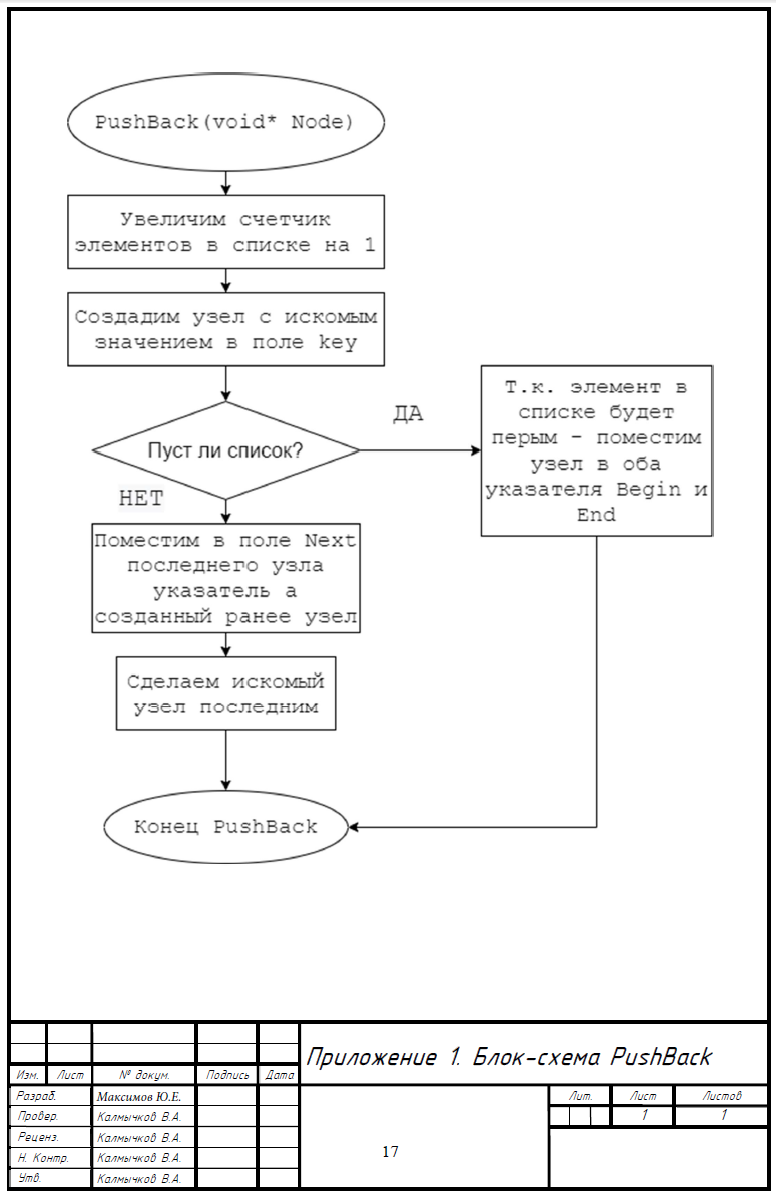
1. Брайан Керниган, Деннис Ритчи. Язык программирования Си. – Москва: Финансы и статистика, 1992. — 272 с.
2. База данных фамилий

https://nsportal.ru/blog/obshche/hunya/obrazovatelnaya-tematika/all/2012/11/20/ (дата посещения 15.04.2022)

1. Лекция 30: Динамические структуры данных: однонаправленные и двунаправленные списки https://intuit.ru/studies/courses/648/504/lecture/11456 (дата посещения 19.04.2022)
2. Лекция 10: Основы объектно-ориентированного программирования https://intuit.ru/studies/courses/105/105/lecture/3077?page=6

(дата посещения 19.04.2022)





ПРИЛОЖЕНИЕ 2. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

1. Минимальные системные требования

Операционная система: Microsoft Windows 10 Pro. 64 – разрядная операционная система.

Процессор: Как минимум 1 ГГц или SoC.

ОЗУ: 2 ГБ.

Место на жестком диске: 20 ГБ.

Дисплей: 800 x 600.

1. Процесс установки

Скопируйте файл Course\_work.exe в выбранную директорию.

1. Процесс запуска программы и работы с ней

Запустите Course\_work.exe из выбранной директории. Появится окно, которое показано на рисунке 4:

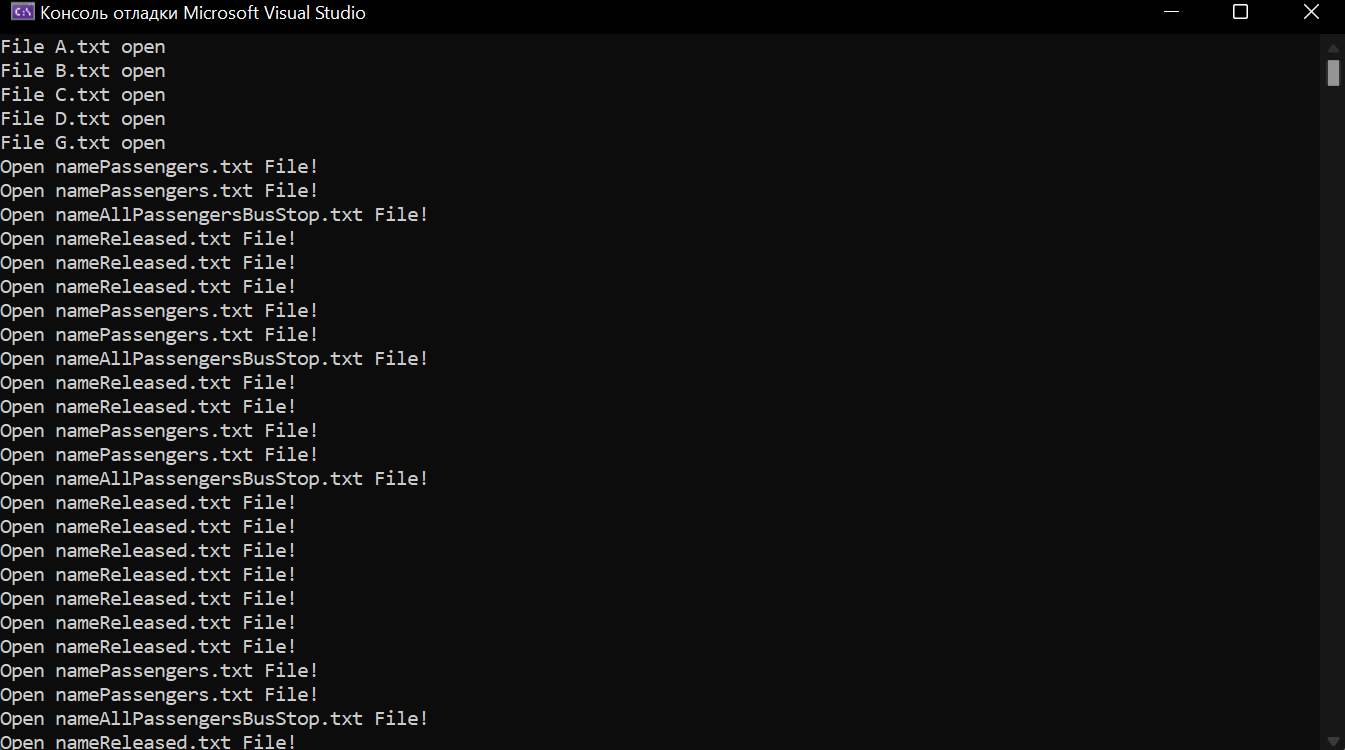


Рисунок 4 – Начальное окно программы

.

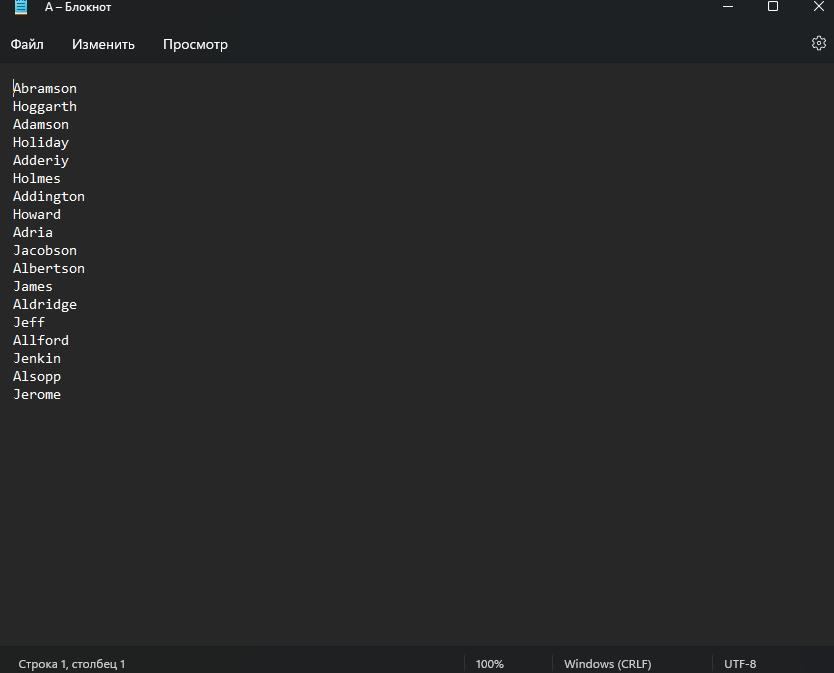


Рисунок 5 – файл с ФИО на остановке А

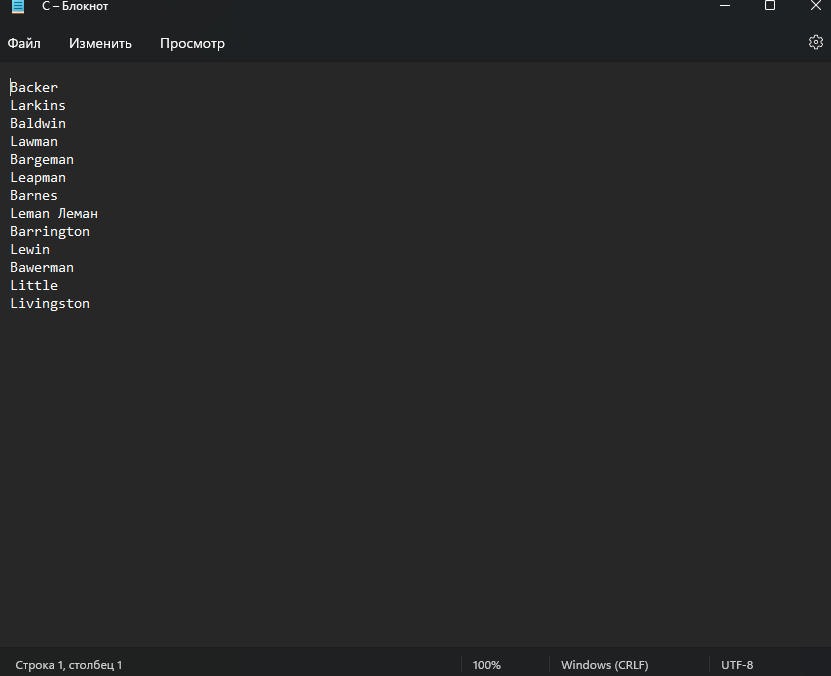


Рисунок 6 – файл с ФИО на остановке С

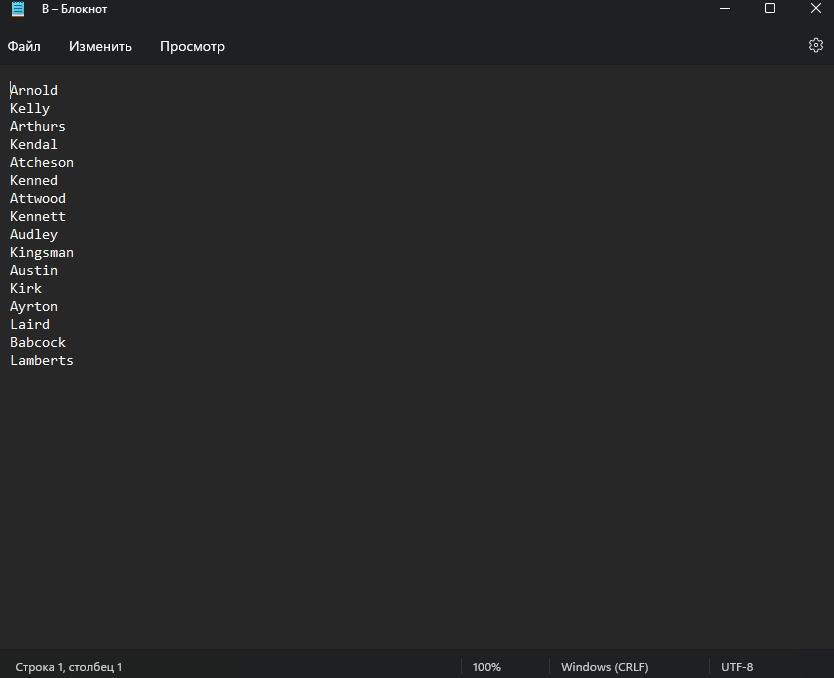


Рисунок 7 – файл с ФИО на остановке B

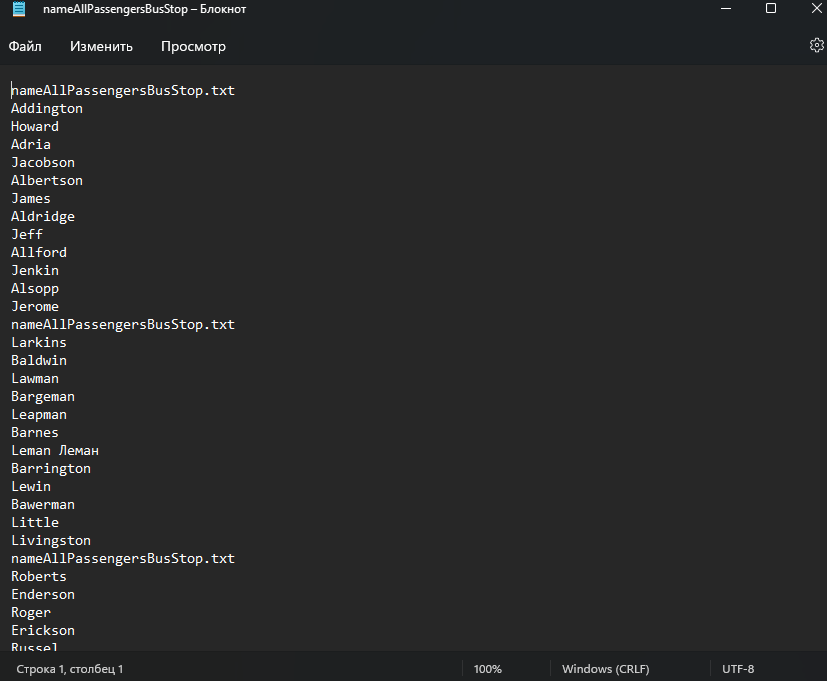


Рисунок 8 – файл ФИО тех, кто стоит на остановке

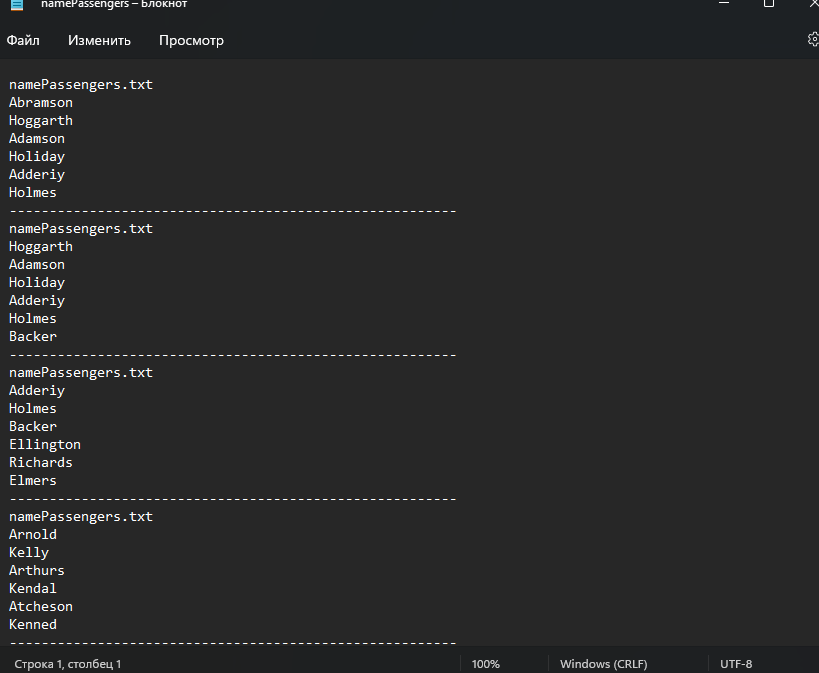


Рисунок 9 – файл ФИО тех, кто в автобусе

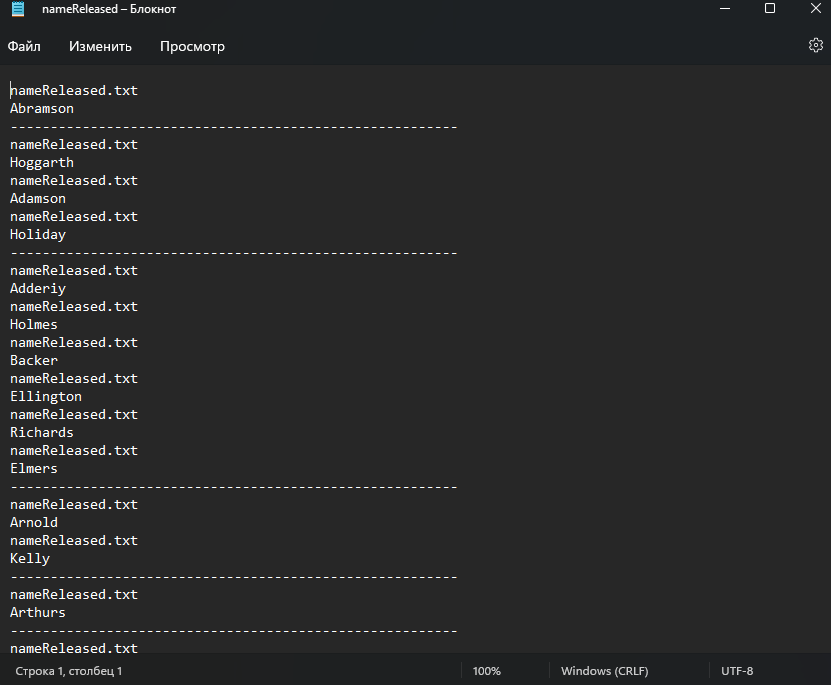


Рисунок 10 – Файл ФИО тех, кто вышел из автобуса.

ПРИЛОЖЕНИЕ 3. ИСХОДНЫЙ КОД

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

class MyString {

public:

MyString() {

str = nullptr;

length = 0;

}

MyString(const char\* str) {

length = strlen(str);

this->str = new char[length + 1];

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

{

this->str[i] = str[i];

}

this->str[length] = '\0';

}

MyString(string& str) {

length = str.length();

this->str = new char[length + 1];

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

{

this->str[i] = str[i];

}

this->str[length] = '\0';

}

~MyString() {

delete[] this->str;

}

MyString(const MyString& other) {

length = strlen(other.str);

this->str = new char[length + 1];

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

{

this->str[i] = other.str[i];

}

this->str[length] = '\0';

}

MyString& operator =(const MyString& other) {

if (this->str != nullptr)

{

delete[] str;

}

length = strlen(other.str);

this->str = new char[length + 1];

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

{

this->str[i] = other.str[i];

}

this->str[length] = '\0';

return \*this;

}

MyString operator+(const MyString& other) {

MyString newStr;

int thisLength = strlen(this->str);

int otherLength = strlen(other.str);

newStr.length = thisLength + otherLength;

newStr.str = new char[thisLength + otherLength + 1];

int i = 0;

for (; i < thisLength; i++)

{

newStr.str[i] = this->str[i];

}

for (int j = 0; j < otherLength; j++, i++)

{

newStr.str[i] = other.str[j];

}

newStr.str[thisLength + otherLength] = '\0';

return newStr;

}

int Length() {

return length;

}

bool operator ==(const MyString& other) {

if (this->length != other.length)

{

return false;

}

for (int i = 0; i < this->length; i++)

{

if (this->str[i] != other.str[i])

{

return false;

}

}

return true;

}

bool operator !=(const MyString& other) {

return !(this->operator==(other));

}

char& operator [](int index) {

return this->str[index];

}

MyString(MyString&& other) noexcept {

this->length = other.length;

this->str = other.str;

other.str = nullptr;

}

friend std::ostream& operator<< (std::ostream& out, const MyString& other) {

for (int a = 0; a < other.length; a++) {

out << other.str[a];

}

return out;

}

private:

char\* str;

int length;

};

class List

{

public:

List();

~List();

void pop\_front();

void push\_back(MyString data);

void clear();

int getSize() { return Size; }

MyString& operator[](const int index);;

private:

class Node{

public:

Node\* pNext;

MyString data;

Node(MyString data = MyString(), Node\* pNext = nullptr){

this->data = data;

this->pNext = pNext;

}

};

int Size;

Node\* head;

};

List::List(){

Size = 0;

head = nullptr;

}

List::~List(){

clear();

}

void List::pop\_front(){

Node\* temp = head;

head = head->pNext;

delete temp;

Size--;

}

void List::push\_back(MyString data){

if (head == nullptr){

head = new Node(data);

}

else{

Node\* current = this->head;

while (current->pNext != nullptr){

current = current->pNext;

}

current->pNext = new Node(data);

}

Size++;

}

void List::clear(){

while (Size){

pop\_front();

}

}

MyString& List::operator[](const int index){

int counter = 0;

Node\* current = this->head;

while (current != nullptr){

if (counter == index){

return current->data;

}

current = current->pNext;

counter++;

}

}

void readTxt(const char \*nameTxt, fstream& fs, List& other) {

fs.open(nameTxt, fstream::out | fstream::in | fstream::app);

if (!fs.is\_open()) {

cout << "File " << nameTxt << " opening error!" << endl;

}

else {

cout << "File " << nameTxt << " open" << endl;

string z;

while (!fs.eof()) {

z = "";

getline(fs, z);

other.push\_back(z);

}

}

fs.close();

}

void writeTxtSeparator(const char\* nameTxt, fstream& fs) {

fs.open(nameTxt, fstream::out | fstream::in | fstream::app);

if (!fs.is\_open()) {

cout << "Error open File!";

}

else {

cout << "Open " << nameTxt << " File!" << endl;

fs << "--------------------------------------------------------" << '\n';

}

fs.close();

}

void writeTxt(const char \*nameTxt, fstream& fs, List& other, int size = 0) {

fs.open(nameTxt, fstream::out | fstream::in | fstream::app);

if (!fs.is\_open()) {

cout << "Error open File!";

}

else {

cout << "Open " << nameTxt<< " File!" << endl;

fs << nameTxt << '\n';

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

fs << other[i] << '\n';

}

}

fs.close();

}

void writeBusPassengers(const char \*nameTxt, fstream& fs, List& other, int&& size) {

writeTxt(nameTxt,fs,other,size);

writeTxtSeparator(nameTxt, fs);

}

void busLoading(List& otherOne, List& otherTwo, int size) {

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

otherOne.push\_back(otherTwo[0]);

otherTwo.pop\_front();

}

}

void busUnloading(List& other, int size,const char\* nameTxt,fstream& fs) {

while (size) {

writeTxt(nameTxt, fs, other);

other.pop\_front();

size--;

}

writeTxtSeparator(nameTxt, fs);

}

int getRandomNumber(int min, int max) {

static const double fraction = 1.0 / (static\_cast<double>(RAND\_MAX) + 1.0);

return static\_cast<int>(rand() \* fraction \* (max - min + 1) + min);

}

int main() {

srand(static\_cast<unsigned int>(time(0)));

rand();

List busStopA;

List busStopC;

List busStopB;

List busStopD;

List busStopG;

List bus;

fstream fs;

const char\* nameTxtA = "A.txt";

const char\* nameTxtB = "B.txt";

const char\* nameTxtC = "C.txt";

const char\* nameTxtD = "D.txt";

const char\* nameTxtG = "G.txt";

const char\* nameReleased = "nameReleased.txt";

const char\* namePassengers = "namePassengers.txt";

const char\* nameAllPassengersBusStop = "nameAllPassengersBusStop.txt";

readTxt(nameTxtA, fs, busStopA);

readTxt(nameTxtB, fs, busStopB);

readTxt(nameTxtC, fs, busStopC);

readTxt(nameTxtD, fs, busStopD);

readTxt(nameTxtG, fs, busStopG);

int sizeBusMax = 6;

int sizeBusMin = 1;

int sizeBusMidlle = 3;

int sizeBusOther;

int variableOne;

int variableTwo;

int max = 10;

int on = 0;

busLoading(bus, busStopA, sizeBusMax); // passengers get on the bus

while (true) {

// bus point A

if (busStopA.getSize() != 0) {

sizeBusMax= 6;

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

if (bus.getSize() < sizeBusMax) {

sizeBusMax--;

}

else { break; }

}

if (on == true) {

busUnloading(bus, sizeBusMax, nameReleased, fs);

sizeBusMax = 6;

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

if (busStopA.getSize() < sizeBusMax) {

sizeBusMax--;

}

else { break; }

}

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

sizeBusOther = 6 - bus.getSize();

if (sizeBusOther < sizeBusMax) {

sizeBusMax = sizeBusOther;

}

else { break; }

} // bus point A, write neamRealeased

busLoading(bus, busStopA, sizeBusMax); // passengers get on the bus

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

{

if (bus.getSize() != 6) {

if (busStopA.getSize() != 0) {

busLoading(bus, busStopA, 1);

if (bus.getSize() == 6 || busStopA.getSize() == 0) {

break;

}

}

else { break; }

}

else { break; }

}

}

writeBusPassengers(namePassengers, fs, bus, bus.getSize() - 1); // bus passengers

writeTxt(nameAllPassengersBusStop, fs, busStopA, busStopA.getSize() - 1); // passengers point A

on=1;

}

//bus point C

if (busStopC.getSize() != 0) {

variableOne = getRandomNumber(sizeBusMin, sizeBusMidlle);

variableTwo = variableOne;

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

if (bus.getSize() <= variableOne) {

variableOne--;

}

else { break; }

}

busUnloading(bus, variableOne, nameReleased, fs); // passengers get off the bus, write neamRealeased

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

if (busStopC.getSize() < variableTwo) {

variableTwo--;

}

else { break; }

}

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

sizeBusOther = 6 - bus.getSize();

if (sizeBusOther < variableTwo) {

variableTwo = sizeBusOther;

}

else { break; }

}

busLoading(bus, busStopC, variableTwo);// passengers get on the bus

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

{

if (bus.getSize() != 6) {

if (busStopC.getSize() != 0) {

busLoading(bus, busStopC, 1);

if (bus.getSize() == 6 || busStopC.getSize() == 0) {

break;

}

}

else { break; }

}

else { break; }

}

writeBusPassengers(namePassengers, fs, bus, bus.getSize() - 1); // bus passengers

writeTxt(nameAllPassengersBusStop, fs, busStopC, busStopC.getSize() - 1); // passengers point C

}

//bus point D

if (busStopD.getSize() != 0) {

variableOne = getRandomNumber(sizeBusMin, sizeBusMidlle);

variableTwo = variableOne;

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

if (bus.getSize() < variableOne) {

variableOne--;

}

else { break; }

}

busUnloading(bus, variableOne, nameReleased, fs); // passengers get off the bus, write neamRealeased

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

if (busStopD.getSize() < variableTwo) {

variableTwo--;

}

else { break; }

}

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

sizeBusOther = 6 - bus.getSize();

if (sizeBusOther < variableTwo) {

variableTwo = sizeBusOther;

}

else { break; }

}

busLoading(bus, busStopD, variableTwo); // passengers get on the bus

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

{

if (bus.getSize() != 6) {

if (busStopD.getSize() != 0) {

busLoading(bus, busStopD, 1);

if (bus.getSize() == 6 || busStopD.getSize() == 0) {

break;

}

}

else { break; }

}

else { break; }

}

writeBusPassengers(namePassengers, fs, bus, bus.getSize() - 1); // bus passengers

writeTxt(nameAllPassengersBusStop, fs, busStopD, busStopD.getSize() - 1); // passengers point C

}

//bus point B

if (busStopB.getSize() != 0) {

sizeBusMax = 6;

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

if (bus.getSize() < sizeBusMax) {

sizeBusMax--;

}

else { break; }

}

busUnloading(bus, sizeBusMax, nameReleased, fs); // bus point B, write neamRealeased

sizeBusMax = 6;

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

if (busStopB.getSize() < sizeBusMax) {

sizeBusMax--;

}

else { break; }

}

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

sizeBusOther = 6 - bus.getSize();

if (sizeBusOther < sizeBusMax) {

sizeBusMax = sizeBusOther;

}

else {break;}

}

busLoading(bus, busStopB, sizeBusMax); // passengers get on the bus

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

{

if (bus.getSize() != 6) {

if (busStopB.getSize() != 0) {

busLoading(bus, busStopB, 1);

if (bus.getSize() == 6 || busStopB.getSize() == 0) {

break;

}

}

else { break; }

}

else { break; }

}

writeBusPassengers(namePassengers, fs, bus, bus.getSize() - 1); // bus passengers

writeTxt(nameAllPassengersBusStop, fs, busStopB, busStopB.getSize() - 1); // passengers point B

}

//bus point D

if (busStopD.getSize() != 0) {

variableOne = getRandomNumber(sizeBusMin, sizeBusMidlle);

variableTwo = variableOne;

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

if (bus.getSize() < variableOne) {

variableOne--;

}

else { break; }

}

busUnloading(bus, variableOne, nameReleased, fs); // passengers get off the bus, write neamRealeased

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

if (busStopD.getSize() < variableTwo) {

variableTwo--;

}

else { break; }

}

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

sizeBusOther = 6 - bus.getSize();

if (sizeBusOther < variableTwo) {

variableTwo = sizeBusOther;

}

else { break; }

}

busLoading(bus, busStopD, variableTwo); // passengers get on the bus

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

{

if (bus.getSize() != 6) {

if (busStopD.getSize() != 0) {

busLoading(bus, busStopD, 1);

if (bus.getSize() == 6 || busStopD.getSize() == 0) {

break;

}

}

else { break; }

}

else { break; }

}

writeBusPassengers(namePassengers, fs, bus, bus.getSize() - 1); // bus passengers

writeTxt(nameAllPassengersBusStop, fs, busStopD, busStopD.getSize() - 1); // passengers point C

}

//bus point C

if (busStopC.getSize() != 0) {

variableOne = getRandomNumber(sizeBusMin, sizeBusMidlle);

variableTwo = variableOne;

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

if (bus.getSize() <= variableOne) {

variableOne--;

}

else { break; }

}

busUnloading(bus, variableOne, nameReleased, fs); // passengers get off the bus, write neamRealeased

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

if (busStopC.getSize() < variableTwo) {

variableTwo--;

}

else { break; }

}

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

sizeBusOther = 6 - bus.getSize();

if (sizeBusOther < variableTwo) {

variableTwo = sizeBusOther;

}

else { break; }

}

busLoading(bus, busStopC, variableTwo);// passengers get on the bus

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

{

if (bus.getSize() != 6) {

if (busStopC.getSize() != 0) {

busLoading(bus, busStopC, 1);

if (bus.getSize() == 6 || busStopC.getSize() == 0) {

break;

}

}

else { break; }

}

else { break; }

}

writeBusPassengers(namePassengers, fs, bus, bus.getSize() - 1); // bus passengers

writeTxt(nameAllPassengersBusStop, fs, busStopC, busStopC.getSize() - 1); // passengers point C

}

//bus not point G

writeTxt(nameAllPassengersBusStop, fs, busStopG, busStopG.getSize() - 1); // passengers point A

if (busStopA.getSize() == 0 && busStopB.getSize() == 0 && busStopC.getSize() == 0 && busStopD.getSize() == 0) {

busUnloading(bus, bus.getSize(), nameReleased, fs);

break;

}

}

return 0;

}