**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Курсовая работа по дисциплине

«Программирование»

Текстовый редактор для работы с двоичным с файлом, содержащим односвязный список строк переменной длины со счетчиком

Вариант 4.2

Группа: АВТ-610

Студент: Дунаев Н.Ю.

Преподаватель: Васюткина И. А.

Новосибирск 2017

Оглавление

[Описание задания 3](#_Toc501990908)

[Структурное описание разработки 3](#_Toc501990909)

[Конструктор с параметром 4](#_Toc501990910)

[Считывание из txt-файла 4](#_Toc501990911)

[Запись в txt файл 5](#_Toc501990912)

[Поиск строки в бинарном файле 5](#_Toc501990913)

[Удаление строки из бинарного файла 6](#_Toc501990914)

[Добавление строки 6](#_Toc501990915)

[Замена строки 7](#_Toc501990916)

[Функциональное описание разработки 7](#_Toc501990917)

[Комментарии к нетривиальным операциям 7](#_Toc501990918)

[Блок–схема для перемещения по блокам заданное количество раз 13](#_Toc501990919)

[Описание пользовательского интерфейса 14](#_Toc501990920)

[Меню программы 14](#_Toc501990921)

[Считывание с txt 14](#_Toc501990922)

[Поиск 15](#_Toc501990923)

[Удаление 15](#_Toc501990924)

[Вставка 15](#_Toc501990925)

[Замена 16](#_Toc501990926)

[Запись в txt 16](#_Toc501990927)

[Печать бинарника 16](#_Toc501990928)

[Выход 17](#_Toc501990929)

[Описание работы программы на контрольных примерах 17](#_Toc501990930)

[Вывод 18](#_Toc501990931)

# Описание задания

Класс - структура данных в двоичном файле

Класс двоичного файла, производный от fstream. Двоичный файл содержит заданную структуру данных. Программа должна представлять собой простой текстовый редактор, использующий структуру данных для промежуточного хранения редактируемого файла. Должны быть реализованы операции создания и заполнения двоичного файла из заданного текстового и сохранения содержимого двоичного файла в текстовый.

Класс – двоичный файл, производный от fstream. Файл содержит односвязный список строк в формате записей переменной длины со счетчиком. Формат файла: в начале файла – заголовок списка. Элемент списка содержит файловый указатель на следующий и саму строку в виде записи переменной длины.

# Структурное описание разработки

Для реализации поставленной задачи был разработан новый класс «binfile».

Он является классом-наследником от fstream. BinFile позволяет осуществлять работу со структурой данных односвязный список строк (переменной длины), используя двоичный файл как основное место хранения информации.

В следующей таблице приведён порядок заполнения бинарного файла

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Заголовок | Указатель | Длина | Строка | Указатель | Длина | Строка |

## Конструктор с параметром

binfile(char\* name)

Для записи заголовка в начало бинарного файла был создан конструктор с параметром класса binfile, принимающий на вход строку, содержащую название бинарного файла ( по умолчанию передаем строку “binary.bin”). Конструктор открывает файл “binary.bin” с параметром для записи ( ios::out), перемещает указатель для записи на начало файла и записывает туда заголовок – указатель на себя. После чего сохранят позицию указателя записи во вспомогательную переменную, и, перемещаясь к заголовку, записывает переменную в заголовок. Тем самым в начале бинарного файла мы имеем заголовок, указывающий на позицию, с которой позже мы будем начинать запись строк из txt файла.

Графическое представление алгоритма:



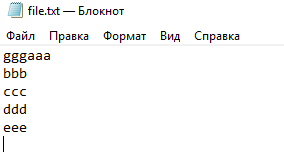
Рис.1 Запись заголовка в начало бинарного файла

## Считывание из txt-файла

void readtxt(char \*txt\_name);

Функция принимает на вход строку – название txt файла, из которого будет производиться построчное чтение.

Пример содержания текстового файла:



Функция открывает тестовый файл с параметром для чтения (ios::in), с помощью функции getline() считывает строки во вспомогательную строку, сохраняя длину строки в переменную len, после чего перемещается в конец бинарного файла, записывает указатель блока самого на себя, длину строки и саму строку. После чего считывает позицию указателя записи и переписывает ее на старую позицию указателя. Данный алгоритм выполняется, пока не считаются все строки из txt файла.

Графическое представление алгоритма:



Рис.2 Считывание из текстового файла

## Запись в txt файл

void writeout();

Функция не принимает на вход параметров. Открывает бинарный файл для чтения и текстовый для записи. Из бинарного файла считывает заголовок и перемещается на считанную позицию (начало следующего «блока»). Пока не достигли конца бинарного файла, поочередно считываем «блоки» и записываем строки, извлеченные из блоков в текстовый файл при помощи оператора <<.

Графическое представление алгоритма:



Рис.3 Запись в текстовый файл

## Поиск строки в бинарном файле

void find(int num);

Функция принимает на вход номер строки, которую нужно найти. Открывает бинарный файл, считывает заголовок и перемещается на первый блок. Пока не достигнут конец бинарного файла, начинает выполняться цикл, считывающий указатель на следующий блок и перемещающий указатель чтения на соответствующую позицию (см. Рис.3). Каждый раз при перемещении по указателю к счетчику поиска прибавляется единица. Внутри цикла выполняется проверка, и если счетчик равен введенному нами номеру строки, которую требуется найти, считывает из данного блока строку и выводит ее на экран, после чего завершаются все циклы.

## Удаление строки из бинарного файла

void deleteStr(int num);

Функция принимает на вход номер строки, которую требуется удалить. Открывает бинарный файл, считывает заголовок и перемещается на первый блок. Пока не достигнут конец бинарного файла, начинает выполняться цикл, считывающий указатель на следующий блок и перемещающий указатель чтения на соответствующую позицию (см. Рис.3). Цикл доходит до блока(1), после которого находится строка, которую нужно удалить и сохраняет позицию – указатель на начало данного блока. После чего перемещается к блоку, строку которого нужно удалить, считывает указатель текущего блока(в котором хранится позиция следующего блока) и переписывает его в указатель блока (1).

## Добавление строки

void eject(char\* streject, int num);

Функция принимает на вход строку, которую требуется добавить и номер строки, после которой требуется добавление. Открывает бинарный файл, считывает заголовок и перемещается на первый блок. Пока не достигнут конец бинарного файла, начинает выполняться цикл, считывающий указатель на следующий блок и перемещающий указатель чтения на соответствующую позицию (см. Рис.3). Цикл доходит до блока(1), после которого требуется добавить, сохраняя его позицию. Перемещается к следующему блоку(2), также сохраняя его позицию. После чего переходит к концу файла и записывает новый блок(3), вписывая в его указатель позицию на блока(2). Следующим шагом переходит к блоку(1) и вписывает в него указатель на блок, который добавили в конец.

Графическое описание алгоритма



Рис.4 Добавление строки в бинарный файл

## Замена строки

void change(char\* change, int num);

Функция принимает на вход строку, на которую требуется заменить и номер строки, которую требуется заменить.

При вызове данной функции, она вызывает две другие функции: удаления строки и добавления новой строки, принцип работы которых описан в предыдущих пунктах.

deleteStr(num);

eject(str, num);

# Функциональное описание разработки

## Комментарии к нетривиальным операциям

binfile(char\* name) // конструктор с параметрами

{

ios\_base::seekdir ptr1, ptr2;

//Выделяем память под название бинарного файла и копируем переданную строку

char\* binname = new char[strlen(name) + 1];

strcpy(binname, name);

//Открываем для записи бинарник с переданной строкой в названии

open(binname, ios::out);

//Перемещаемся на начало бинарника

seekp(0, ios::beg);

//Сохраняем позицию курсора для записи

ptr1 = tellp();

//Записываем позицию курсора

write((char\*)&ptr1, sizeof(ptr1));

seekg(0, ios::end);

//Сохраняем новый конец файла и записываем его в заголовок

ptr2 = tellg();

seekp(ptr1);

write((char\*)&ptr2, sizeof(ptr2));

}

void binfile::print() //печать бинарника

{

ios\_base::seekdir next, end;

//Открываем бинарник с параметром для чтения

int len;

open("binary.bin", ios::in);

//Сохраняем указатель на конец

seekp(0, ios::end);

end = tellp();

//Сохраняем указатель на начало

seekp(0, ios::beg);

next = tellp();

//Читаем заголовок

read((char\*)&next, sizeof(next));

//Перемещаемся к первому блоку

seekg(next);

cout << "Содержимое bin файла: " << endl;

//Цикл – пока не дошли до конца бинарника

while (next != end)

{

char\* tmp;

//Считываем указатель на следующий блок

read((char\*)&next, sizeof(next));

//Считываем длину

read((char\*)&len, sizeof(int));

//Выделяем память под строку для вывода

tmp = new char[len + 1];

//Считываем и выводим строку

read(tmp, len + 1);

cout << tmp << endl;

//Очищаем память и перемещаемся к след. блоку

delete []tmp;

seekg(next);

}

cout << endl;

close();

}

void binfile::change(char\* str, int num) // замена строки

{

//Вызываем функцию удаления

deleteStr(num);

//Вызываем функцию добавления

eject(str, num);

}

void binfile::deleteStr(int num) //удаление строки

{

//Проверка, не допускающая удаление нулевой строки

if (num == 0)

{

cout << "Введите номер > 1!!!" << endl;

return;

}

//Открываем бинарник с параметром для записи и для чтения

open("binary.bin", ios::in | ios::out);

int count = 1; //счетчик удаления

ios\_base::seekdir tmp1, tmp2, next, end;

//Сохраняем конец файла

seekp(0, ios::end);

end = tellp();

//Сохранаяем начало файла

seekp(0, ios::beg);

next = tellp();

while (count < num)

{

// Пока не достигли номера строки, которую хотим удалить

count++;

//Считываем указатель на след блок

read((char\*)&next, sizeof(next));

//Перемещаемся к след блоку

seekp(next);

}

//Сохраняем позицию блока

tmp1 = next;

//Перемещаемся к след блоку

read((char\*)&next, sizeof(next));

seekp(next);

//Считываем указатель на блок, перед которым находится блок для удаления

read((char\*)&tmp2, sizeof(tmp2));

//Переходим к блоку, после которого нужно удалить

seekg(tmp1);

//Переписываем указатель через один блок

write((char\*)&tmp2, sizeof(tmp2));

close();

}

void binfile::eject(char\* str, int num) // добавление строки

{

open("binary.bin", ios::in | ios::out);

int len = strlen(str);

int count = 0;

ios\_base::seekdir tmp1, tmp2, next, end, endblock, newend, oldend;

seekg(0, ios::end); // ук считывания на конец

end = tellg(); //сохранил ук считывания

seekg(0, ios::beg); // ищем последний блок, указывающий на конец

next = tellg();

read((char\*)&next, sizeof(next));

seekg(next);

while (next != end)

{

oldend = next; // блок, указывающий на конец

read((char\*)&next, sizeof(next));

seekg(next);// перемещение по указателям

}

seekg(0, ios::beg); // ук счит на начало

next = tellg(); // сохранил в некст

if (num != 1){ // проверка на смену головы

read((char\*)&next, sizeof(next)); // считал след ук

seekg(next); // переместился на ук

}

count += 2;

while (count < num)

{

count++;

read((char\*)&next, sizeof(next));

seekg(next);// перемещение по указателям

}

tmp1 = tellg(); // сохранил на тот, после которого записать

read((char\*)&next, sizeof(next));

seekg(next); // переместился на след

tmp2 = tellg(); // сохранил след

seekp(0, ios::end); // для записи в конец

endblock = tellp(); // сохранил указатель на последний блок

write((char\*)&tmp2, sizeof(tmp2)); //пишем указатель на следующий, после которого надо записать

write((char\*)&len, sizeof(len)); //пишем длину

write(str, len + 1); // пишем строку

newend = tellp();

seekp(tmp1); // переместился на блок, после которого записать

write((char\*)&endblock, sizeof(endblock)); // в блок, после которого записать пишем указатель на последний блок

seekp(oldend);

write((char\*)&newend, sizeof(newend));

close();

}

void binfile::writeout() // запись в тхт

{

int len;

int kol = 0;

//Открываем бинарник с параметром для чтения

open("binary.bin", ios::in);

//Открываем тхт с параметром для записи

ofstream os("newfile.txt", ios::out);

if (!os)

{

cout << "chto-to poshlo ne tak..." << endl;

return;

}

else

{

ios\_base::seekdir next, end;

seekg(0, ios::end);

end = tellg();

seekg(0, ios::beg);

next = tellg();

read((char\*)&next, sizeof(next));

seekg(next);

//Пока не достигли конца бинарника

while (next != end)

{

//Считываем указатель на след блок, длину строки

char \*tmp;

read((char\*)&next, sizeof(next));

read((char\*)&len, sizeof(int));

//Выделяем память под строку

tmp = new char[len + 1];

//Считываем строку

read(tmp, len + 1);

//Записываем в текстовый файл строку и символ новой строки

os << tmp;

os << "\n";

//Переходим к след блоку

seekg(next);

delete[]tmp;

kol++;

}

}

cout << "Кол-во строк записали:: " << kol << endl;

close();

}

void binfile::readtxt(char \*name) //чтение из тхт

{

int kol = 0; // счетчик считанных строк

ios\_base::seekdir ptr, helptr;

//Открываем текстовый файл для чтения

ifstream is(name, ios::in);

if (!is)

{

cout << "can't open txt file!" << endl;

close();

return;

}

else

{

//Пока не достигнем конца текстового файла

while (!is.eof()){

char strm[100];

//Считываем строку из тхт

is.getline(strm, 100);

//Перемещаемся на конец бинарника

seekp(0, ios::end);

ptr = tellp();

//Записываем блок в бинарник

write((char\*)&ptr, sizeof(ptr));

int len = strlen(strm);

write((char\*)&len, sizeof(len));

write(strm, len + 1);

//Запоминаем указатель на новый конец и переписываем его в записанный выше блок

helptr = tellp();

seekp(ptr);

write((char\*)&helptr, sizeof(helptr));

kol++;

}

is.close();

close();

cout << "кол-во строк считали: " << kol << endl;

}

}

void binfile::find(int num) //поиск

{

//Открываем бинарник с параметром для чтения

open("binary.bin", ios::in);

int ln;

int findednumber = 0;

seekg(0, ios::end);

ios\_base::seekdir end, next;

//Запоминаем конец и начало

end = tellg();

seekg(0, ios::beg);

next = tellg();

read((char\*)&next, sizeof(next));

seekg(next);

while (next != end)

{

//Пока не конец файла, перемещаемся по блокам

read((char\*)&next, sizeof(next));

findednumber += 1;

//Если счетчик равен введенному нами номеру строки

if (findednumber == num)

{

//Считываем из блока строку и печатаем ее, после чего заканчиваем цикл

char \*readedStr;

read((char\*)&ln, sizeof(int));

readedStr = new char[ln + 1];

read(readedStr, ln + 1);

cout << "Finded string: ";

cout << readedStr << endl;

close();

delete[]readedStr;

return;

}

seekg(next);

}

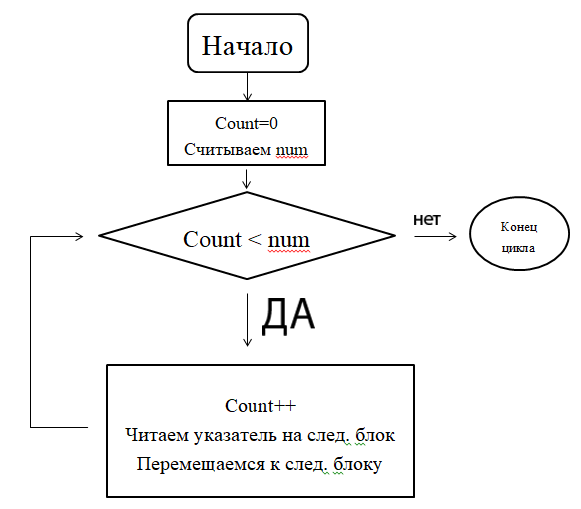
close();

cout << "String this ur number wasn't founded" << endl;

return;

}

## Блок–схема для перемещения по блокам заданное количество раз

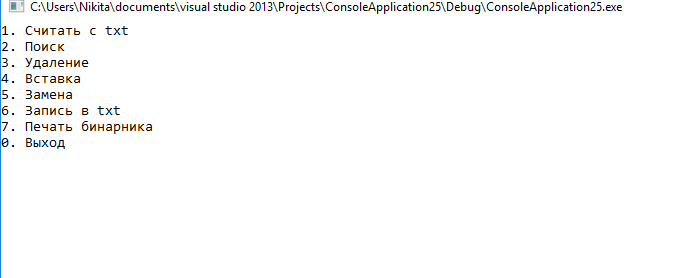
****

# Описание пользовательского интерфейса

Для демонстрации работы класса был создан user-интерфейс, позволяющий

показать работу всех методов класса.

## Меню программы



Для работы с программой требуется ввести нужные команды. Полный перечень

команд:

1. Считать с txt

2. Поиск

3. Удаление

4. Вставка

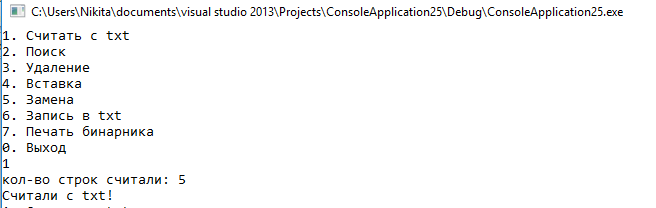
5. Замена

6. Запись в txt

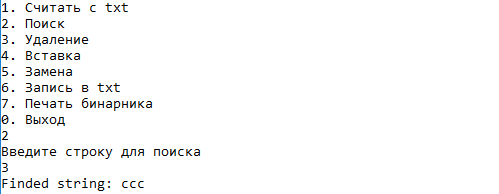
7. Печать бинарника

0. Выход

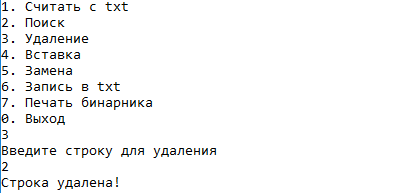
## Считывание с txt



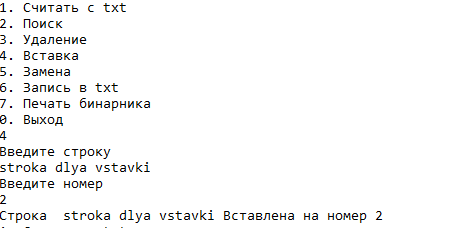
## Поиск



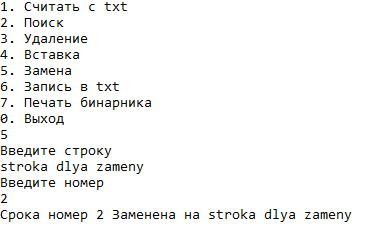
## Удаление



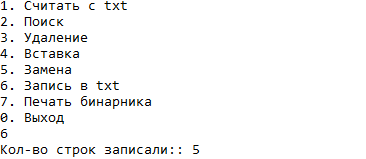
## Вставка



## Замена



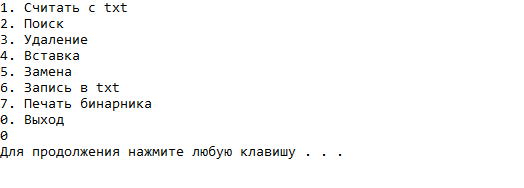
## Запись в txt



## Печать бинарника



## Выход



# Описание работы программы на контрольных примерах

Для тестирования программы был создан txt файл, хранящий 1000000 строк . Производились поиск строки по номеру и измерение времени выполнения поиска.

Результаты исследования были внесены в таблицу.

Таблица:

|  |  |
| --- | --- |
| кол-во строк | t вып., мс |
| 100т | 594 |
| 200т | 1236 |
| 300т | 1695 |
| 400т | 2184 |
| 500т | 2689 |
| 600т | 3259 |
| 700т | 3808 |
| 800т | 4269 |
| 900т | 4817 |
| 1м | 5345 |

По данной таблице был построен график зависимости времени выполнения поиска от объема данных.

Проанализировав график нетрудно заметить, что он имеет практически линейную зависимость. Причиной тому является разработанный метод поиска: производится поиск по номеру строки, что является линейным перебором строк.

# Вывод

Была разработана программа, представляющая простой текстовый редактор, использующая бинарный файл для промежуточного хранения и редактирования данных. Были реализованы функции чтения из текстового файла в бинарный, изменения, добавления, удаления и поиска строки, а также функция печати бинарного файла.

Программа протестирована на текстовом файле объемом в 1000000 строк.

Особенность разработки заключается в том, что память, отводимая под список, ограничивается пространством диска, а не оперативной памятью, что является преимуществом разработки. С другой стороны мы имеем проигрыш по времени, так как работа с оперативной памятью производится быстрее.

# Приложение

*Текст программы*

*Header.h*

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <fstream>

#include <ctime>

using namespace std;

class binfile : public std::fstream

{

public:

binfile(char\* name)

{

ios\_base::seekdir ptr1, ptr2;

char\* binname = new char[strlen(name) + 1];

strcpy(binname, name);

open(binname, ios::out | ios::binary);

seekp(0, ios::beg);

ptr1 = tellp();

write((char\*)&ptr1, sizeof(ptr1));

seekg(0, ios::end);

ptr2 = tellg();

seekp(ptr1);

write((char\*)&ptr2, sizeof(ptr2));

}

~binfile(){ close(); }

void readtxt(char \*txt\_name);

void find(int num);

void deleteStr(int num);

void eject(char\* streject, int num);

void change(char\* change, int num);

void writeout();

void print();

};

void binfile::print()

{

ios\_base::seekdir next, end;

int len;

open("binary.bin", ios::in | ios::binary);

seekp(0, ios::end);

end = tellp();

seekp(0, ios::beg);

next = tellp();

read((char\*)&next, sizeof(next));

seekg(next);

cout << "Содержимое bin файла: " << endl;

while (next != end)

{

char\* tmp;

read((char\*)&next, sizeof(next));

read((char\*)&len, sizeof(int));

tmp = new char[len+1];

read(tmp, len + 1);

cout << tmp << endl;

delete [] tmp;

seekg(next);

}

cout << endl;

close();

}

void binfile::change(char\* str, int num)

{

deleteStr(num);

eject(str, num);

}

void binfile::deleteStr(int num)

{

if (num == 0)

{

cout << "Введите номер > 1!!!" << endl;

return;

}

open("binary.bin", ios::in | ios::out | ios::binary);

int count = 1;

ios\_base::seekdir tmp1, tmp2, next, end;

seekp(0, ios::end);

end = tellp();

seekp(0, ios::beg);

next = tellp();

while (count < num)

{

count++;

read((char\*)&next, sizeof(next));

seekp(next);

}

tmp1 = next;

read((char\*)&next, sizeof(next));

seekp(next);

read((char\*)&tmp2, sizeof(tmp2));

seekg(tmp1);

write((char\*)&tmp2, sizeof(tmp2));

close();

}

void binfile::eject(char\* str, int num)

{

open("binary.bin", ios::in | ios::out | ios::binary);

int len = strlen(str);

int count = 0;

ios\_base::seekdir tmp1, tmp2, next, end, endblock, newend, oldend;

seekg(0, ios::end);

end = tellg();

seekg(0, ios::beg);

next = tellg();

read((char\*)&next, sizeof(next));

seekg(next);

while (next != end)

{

oldend = next;

read((char\*)&next, sizeof(next));

seekg(next);

}

seekg(0, ios::beg);

next = tellg();

if (num != 1){

read((char\*)&next, sizeof(next));

seekg(next);

}

count += 2;

while (count < num)

{

count++;

read((char\*)&next, sizeof(next));

seekg(next);

}

tmp1 = tellg();

read((char\*)&next, sizeof(next));

seekg(next);

tmp2 = tellg();

seekp(0, ios::end);

endblock = tellp();

write((char\*)&tmp2, sizeof(tmp2));

write((char\*)&len, sizeof(len));

write(str, len + 1);

newend = tellp();

seekp(tmp1);

write((char\*)&endblock, sizeof(endblock));

seekp(oldend);

write((char\*)&newend, sizeof(newend));

close();

}

void binfile::writeout()

{

int len;

int kol = 0;

open("binary.bin", ios::in | ios::binary);

ofstream os("newfile.txt", ios::out);

if (!os)

{

cout << "chto-to poshlo ne tak..." << endl;

return;

}

else

{

ios\_base::seekdir next, end;

seekg(0, ios::end);

end = tellg();

seekg(0, ios::beg);

next = tellg();

read((char\*)&next, sizeof(next));

seekg(next);

while (next != end)

{

char \*tmp;

read((char\*)&next, sizeof(next));

read((char\*)&len, sizeof(int));

tmp = new char[len + 1];

read(tmp, len + 1);

os << tmp;

os << "\n";

seekg(next);

delete[]tmp;

kol++;

}

}

cout << "Кол-во строк записали:: " << kol << endl;

close();

}

void binfile::readtxt(char \*name)

{

int kol = 0;

ios\_base::seekdir ptr, helptr;

ifstream is(name, ios::in);

if (!is)

{

cout << "can't open txt file!" << endl;

close();

return;

}

else

{

while (!is.eof()){

char strm[100];

is.getline(strm, 100);

seekp(0, ios::end);

ptr = tellp();

write((char\*)&ptr, sizeof(ptr));

int len = strlen(strm);

write((char\*)&len, sizeof(len));

write(strm, len + 1);

helptr = tellp();

seekp(ptr);

write((char\*)&helptr, sizeof(helptr));

kol++;

}

is.close();

close();

cout << "кол-во строк считали: " << kol << endl;

}

}

void binfile::find(int num)

{

unsigned int T0 = clock();

open("binary.bin", ios::in | ios::binary);

int ln;

int findednumber = 0;

seekg(0, ios::end);

ios\_base::seekdir end, next;

end = tellg();

seekg(0, ios::beg);

next = tellg();

read((char\*)&next, sizeof(next));

seekg(next);

while (next != end)

{

read((char\*)&next, sizeof(next));

findednumber += 1;

if (findednumber == num)

{

char \*readedStr;

read((char\*)&ln, sizeof(int));

readedStr = new char[ln + 1];

read(readedStr, ln + 1);

cout << "Finded string: ";

cout << readedStr << endl;

close();

unsigned int T1 = clock() - T0;

cout << "Время выполнения: " << T1 << endl;

delete[]readedStr;

return;

}

seekg(next);

}

close();

cout << "String this ur number wasn't founded" << endl;

return;

}

Main.cpp

#include "Header.h"

void main()

{

system("color F0");

setlocale(LC\_ALL, "rus");

binfile bin("binary.bin");

int\* a = NULL;

int menu;

point:;

cout << "1. Считать с txt" << endl;

cout << "2. Поиск" << endl;

cout << "3. Удаление" << endl;

cout << "4. Вставка" << endl;

cout << "5. Замена" << endl;

cout << "6. Запись в txt" << endl;

cout << "7. Печать бинарника" << endl;

cout << "0. Выход" << endl;

cin >> menu;

switch (menu)

{

case 1:

{

bin.readtxt("file.txt");

cout << "Считали с txt!" << endl;

goto point;

}

case 2:

{

cout << "Введите строку для поиска" << endl;

int num;

cin >> num;

bin.find(num);

goto point;

}

case 3:

{

cout << "Введите строку для удаления" << endl;

int num;

cin >> num;

bin.deleteStr(num);

cout << "Строка удалена!" << endl;

goto point;

}

case 4:

{

char str[100];

cout << "Введите строку" << endl;

cin.get();

cin.getline(str, 100);

cout << "Введите номер" << endl;

int num;

cin >> num;

bin.eject(str, num);

cout << "Строка " << str << " Вставлена на номер " << num << endl;

goto point;

}

case 5:

{

char str[100];

cout << "Введите строку" << endl;

cin.get();

cin.getline(str, 100);

cout << "Введите номер" << endl;

int num;

cin >> num;

bin.change(str, num);

cout << "Срока номер " << num << " Заменена на " << str << endl;

goto point;

}

case 6:

{

bin.writeout();

goto point;

}

case 7:

{

bin.print();

goto point;

}

case 0:

{

break;

}

default: break;

}

system("pause");

}