Министерство науки и высшего образования РФ

Федеральное государственное автономное образовательное   
учреждение высшего профессионального образования

«Нижегородский государственный университет им. Н. И. Лобачевского»

Институт информационных технологий математики и механики

**Отчет по учебной практике**

**Лабораторная работа № 6**

**Редактирование текстов (иерархический связный список)**

**Выполнили**:

студенты группы 381806-4:

Журавлёв Роман

Икромов Ином

Тареев Даниил

**Преподаватель**:

Шестакова Наталья Валерьевна

**Содержание**

[Введение 3](#__RefHeading__3039_1807997502)

[Постановка учебной задачи 4](#__RefHeading__3041_1807997502)

[Руководство пользователя](#__RefHeading__3043_1807997502) 5

[Руководство программиста](#__RefHeading__3045_1807997502) 6

[Описание структуры программы](#__RefHeading__3047_1807997502) 7

[Описание алгоритмов](#__RefHeading__3049_1807997502) 8

[Заключение 9](#__RefHeading__3051_1807997502)

[Литература](#__RefHeading__3053_1807997502) 10

# **Введение**

Обработка текстовой информации на компьютере широко применяется в различных областях человеческой деятельности: образование, наука, документооборот, кадровый и бухгалтерский учет и др. Вне зависимости от назначения текста типовыми операциями обработки являются создание, просмотр, редактирование и сохранение информации. В связи с тем, что объем текстовой информации может являться очень большим, для эффективного выполнения операций с ней необходимо выбрать представление текста, обеспечивающее структурирование и быстрый доступ к различным элементам текста. Так, текст можно представить в виде линейной последовательности страниц, каждая из которых есть линейная последовательность строк, которые в свою очередь являются линейными последовательностями слов. Такое представление можно осуществлять с любой степенью детализации в зависимости от особенностей прикладной задачи. В рамках лабораторной работы рассматривается задача разработки учебного редактора текстов, в котором для представления данных используется иерархический связный список. Подобная иерархическая структура представления может применяться при компьютерной реализации математических моделей в виде деревьев и, тем самым, может иметь самое широкое применение в самых различных областях приложений.

# **Постановка учебной задачи**

В рамках лабораторной работы ставится задача разработки учебного редактора текстов с поддержкой следующих операций:

• выбор текста для редактирования (или создание нового текста);

• демонстрация текста на экране дисплея;

• поддержка средств указания элементов (уровней) текста;

• вставка, удаление и замена строк текста;

• запись подготовленного текста в файл.

При выполнении операций чтения (при выборе для редактирования существующего текста) и записи редактор должен использовать стандартный формат, принятый в файловой системе для представления текстовых файлов, обеспечивая тем самым совместимость учебного редактора текстов и существующего программного обеспечения.

# **Руководство пользователя**

Данная программа написана в среде разработки Microsoft Visual Studio 2017 на языке С++ и является консольным приложением. При запуске программы будет предложено ввести имя документа, см. рис. 1.

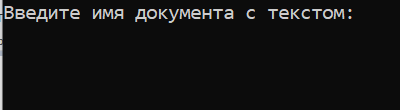


Рис. 1. Запрос на введение имени документа.

После инициализации имени документа произойдет вывод меню, позволяющего совершать операции над текстом, см. рис. 2.

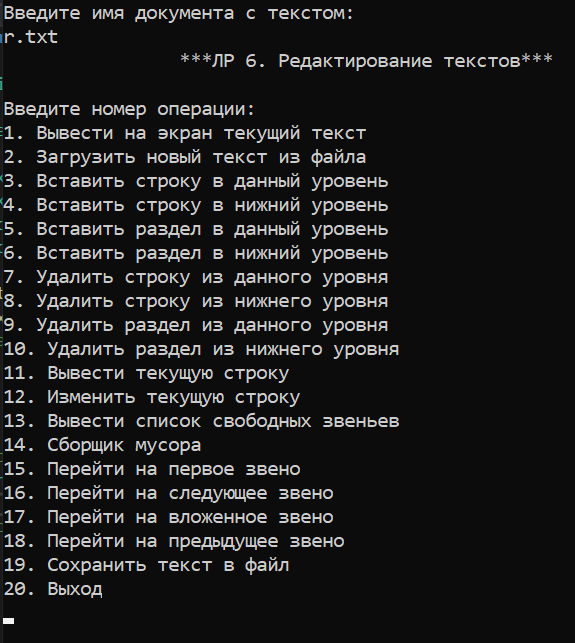


Рис. 2. Вывод меню. При выборе первого пункта «вывести на экран текущий текст, на консоль выведется содержимое выбранного документа, см. рис. 3.

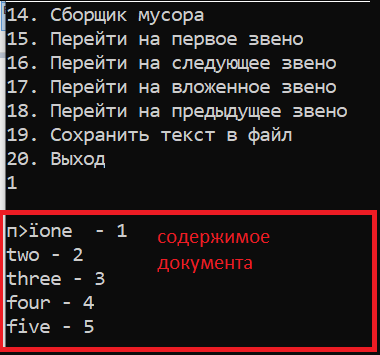
****

Рис. 3. Содержимое выбранного документа.

# **Руководство программиста**

Программа написана при помощи двух классов TLink, TText одного шаблонного TStack.

Класс TLink содержит следующие поля и методы:

Public:

char str[MaxLen]; - строка текста

TLink\* pNext; - указатель на следующую строку текста

TLink\* pDown; - указатель на подуровень

static TTextMem MemHeader; - для управления памятью

static void InitMem(int size); - инициализация памяти

void\* operator new(size\_t size); - выделение звена

void operator delete(void\* p); - освобождение звена

static void MemCleaner(TText& txt, int& count) - "сборщик мусора"

static void PrintFree(); - печать свободных звеньев

int isAtom(); - проверка атомарности элемента

Класс TText содержит следующие поля и методы:

Private:

TLink\* pFirst; - первое звено списка

TLink\* pCurr; - текущее звено списка

TStack <TLink\*> path; - стек для обхода по списку

TStack <TLink\*> St; - стек для итератора

Public:

void GoFirstLink(); - переход к первой строке

void GoDownLink(); - переход к следующей строке по Down

void GoNextLink(); - переход к следующей строке по Next

void GoPrevLink(); - переход к предыдущей позиции в тексте

string GetLine(); - чтение текущей строки

void SetLine(string s); - замена текущей строки

void InsDownLine(string s); - вставка строки в подуровень

void InsDownSection(string s); - вставка раздела в подуровень

void InsNextLine(string s); - вставка строки в тот же уровень

void InsNextSection(string s); - вставка раздела в тот же уровень

void DelDownLine(); - удалить строку в подуровне

void DelDownSection(); - удалить раздел в подуровне

void DelNextLine(); - удалить строку в том же уровне

void DelNextSection(); - удалить раздел в том же уровне

void Reset(); - установить на первую запись

int IsTextEnded() const; - завершён текст или нет?

void GoNext(); - переход к следующей записи

void LoadText(string f\_name); - загрузить новый текст

void SaveText(string f\_name); - сохранить текст в файл

TLink\* ReadSection(ifstream& ifs); - чтение текста из файла

void SaveSection(TLink\* ptemp, ofstream& ofs); - запись текста со звена в файл

void PrintSection(TLink\* ptemp); - печать текста со звена ptemp

void PrintText(); - напечатать текст

## **Описание структуры программы**

Программа содержит один основной проект – TextLab.

• TStack.h – модуль с шаблонным классом, реализующим операции над стеком;

• TTextLink.h, TTextLink.cpp – модуль с классом для звена текста;

• TText.h, TText.cpp – модуль с классом, реализующим операции над текстом;

• TDatValue.h – абстрактный базовый класс для объектов значений.

## **Описание алгоритмов**

**Алгоритмы класса TLink**

**static void InitMem(int size)** - в этом методе выделяется память для хранения текстов. Указатель pFirst устанавливается на начало этого массива (после приведения типа к типу указателя на звено), указатель pLast на последний элемент массива. Далее этот массив размечается как список свободных звеньев, в самом начале работы список свободных звеньев может быть упорядочен по памяти. Для выделения памяти под звено перегружается оператор new, который выделяет новое звено из списка свободных звеньев. При освобождении звена в перегруженном операторе delete происходит возвращение памяти в список свободных звеньев.

**static TTextMem MemHeader -** определена статическая переменная для фиксации состояния памяти в классе

**void MemCleaner(TText& txt, int& count) -** Алгоритм «сборки мусора» состоит из трех этапов. На первом этапе происходит обход текста и маркирование звеньев текста специальными символами (например, «$$$»). На втором этапе происходит обход и маркирование списка свободных звеньев. На третьем этапе происходит проход по всему непрерывному участку памяти как по непрерывному набору звеньев. Если звено промаркировано, то маркер снимается. В противном случае, найдено звено, на которое нет ссылок («мусор»), и это звено возвращается в список свободных звеньев.

**Алгоритмы класса TText**

Для индикации позиции в тексте, относительно которой выполняются операции перемещения по тексту, вводится понятие текущей строки. Для перемещения по тексту предлагается реализовать следующие операции:

**void GoFirstLink()** - переход к первой строке текста;

**void GoDownLink()** - переход к следующему элементу в том же уровне;

**void GoNextLink()** - переход к элементу в нижерасположенном уровне;

**void GoPrevLink()** - переход к предыдущей позиции в тексте.

Разработанная структура хранения текста обеспечивает возможность передвижения по структуре только в направлении следующих или нижерасположенных элементов. Движение по элементам текста в обратном направлении возможно только при условии, если адреса звеньев, на которые необходимо переместиться, были каким-либо способом запомнены ранее. Возможный способ организации запоминания этих адресов состоит в использовании стека.

**void Reset() -** инициализация (установка на корневое звено). В этом методе происходит опустошение стека, после чего в стек помещаются указатель на корневое звено, указатели на следующее звено и следующее в подуровне.

**int IsTextEnded() const -** проверка завершения текста. Если стек пуст, то текст завершен.

**void GoNext() -** переход к следующему звену. В стек помещаются указатели на следующее звено и на звено, следующее в подуровне.

# **Заключение**

Данная лабораторная работа в совокупности с классными занятиями сформировала более глубокое понимание о способах хранения и алгоритмах создания структур данных «иерархический связный список», создав практическое и более детализированное представление о работе как текстовых редакторов, так и многих вычислительных моделей и прикладных задач, использующих иерархические структуры данных.

# **Литература**

1. Курс «Алгоритмы и Структуры Данных» - Шестакова Наталья Валерьевна
2. Материалы по лабораторным работам

<http://www.unn.ru/books/met_files/Pract_ADS.pdf>

1. Герберт Шилдт - Полный справочник по C