МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра систем автоматизированного проектирования

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Программирование»

Тема: представление однонаправленного линейного списка – в элементе строка символов, его обработка

Студентка гр. 3353 _____ Карпенко А.Ю.

Преподаватель _____ Калмычков В.А.

Санкт-Петербург

2024

Оглавление

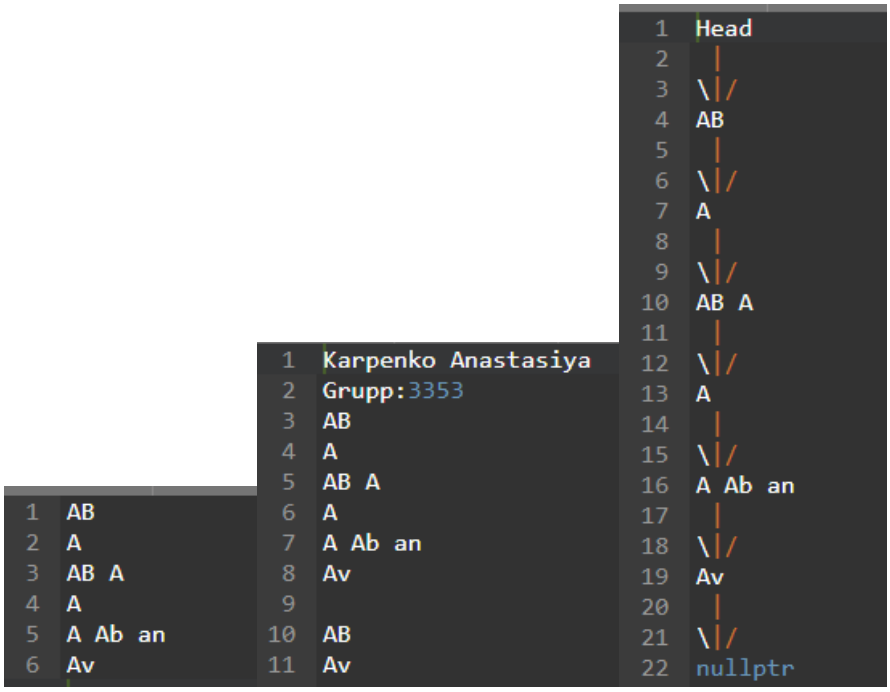
Исходная формулировка задания

Дан файл с символами. Каждая строка состоит из слов, разделенными пробелами, или является пустой. Необходимо удалить элементы, содержащие самые минимальные подэлементы (по длине слов);

Математическая постановка задания

Дано: файл с набором строк.
Необходимо: занести символы строки в массив, затем занести эти массивы в список.
Посчитать длину минимального слова во всех строках. Удалить элементы списка – строки, в которых есть слова с такой минимальной длиной.

Контрольный пример



Разработка интерфейса пользователя

Во входном файле пользователь заполняет строки словами.

Sssssss…ssss*ssss
Sssssss…ssss*ssss
…
Sssssss…ssss*ssss

Первая и вторая строки в выходном файле содержит данные об авторе:

Karpenko Anastasiya
Grupp:3353

С третьей строки выводятся строки, которые были считаны:

Sssssss…ssss*ssss
Sssssss…ssss*ssss
…
Sssssss…ssss*ssss

После этого выводятся строки после удаления строк со словами минимальной длины
Также присутствует файл с наглядным представлением списка

Head
|
\|/
Sssssss…ssss
|
\|/
…
|
\|/
Sssssss…ssss
|
\|/
nullptr

Организация ввода-вывода

Для ввода из файла используем библиотеку fstream

(название потока).get(имя переменной);

Для вывода в файл используем библиотеку fstream

(название потока)<<«Текст»<<(имя переменной);

Так же стоит упомянуть, что для работы с файлами потребуется о открывать потоки вывода и открыть файлы, а так же:

fstream (название потока) ;

(название потока).open(«(Имя файла)», ios::(in для ввода, out для ввывода и app для дополнения));

(название потока)..close();

Программа дробится на файлы посредством вынесения функций и классов в отдельные файлы. Таким образом появляются два файла. Файл расширения .h, содержащий только сами методы и их аргументы. Файл расширения .cpp с реализацией методов. В фале расширения обязательно выполнить проверку на повторное включение:

#ifndef *название*
#define *название*
основной код
#endif

Подключение происходит как подключение библиотек:

#include *название*

Организация хранения данных

Внутренние переменные методов и переменные main

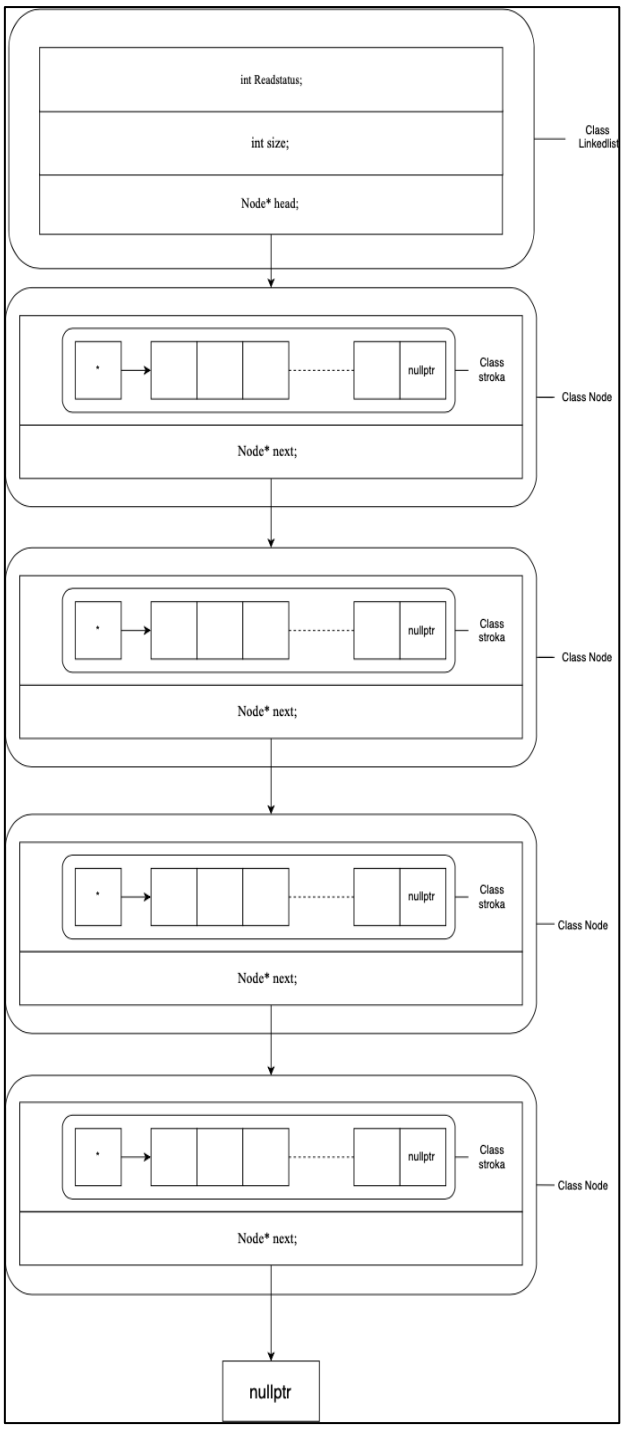
Название переменной	Тип переменной	Пояснение
s	char	хранит элемент типа char
i,j,size, cicles,curren,t_le ngth, max_length	int	Счетчики
in_word	bool	Флаг нахождения в слове
text1	LinkedList	Односвязный список-текст
Current,current1	Node*	Указатели на узел
head	Node*	указатель на первый узел списка

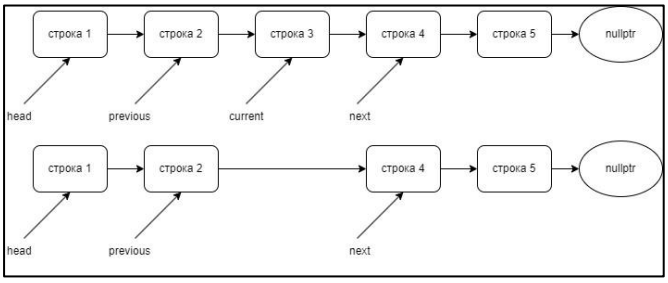
Классы:

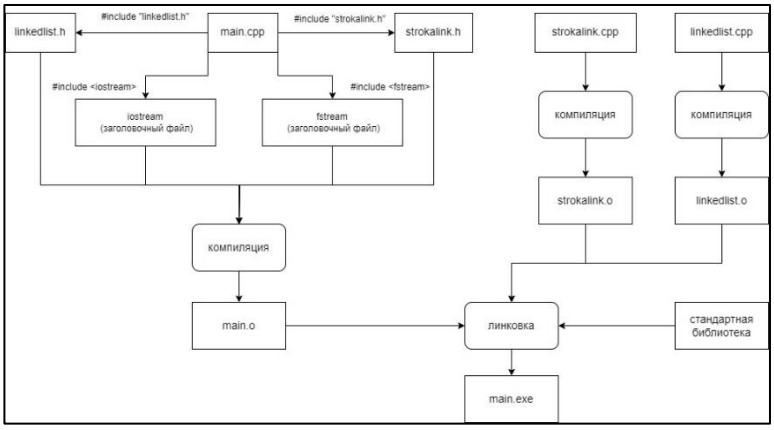
Class stroka			
Переменные	Назначение	Методы	Назначения
char* data	Указатель на массив символов	unsigned int size() const;	Получение длины строки
		int getline(istream& in);	Взятие строки
		int count_words();	Подсчет количества слов
		bool isIn(const stroka& other) const;	Метод проверки вхождения строки в строку
		int MinWord();	Метод для нахождения длины самого короткого слова в строке
		ostream& operator<<(ostream& out, stroka& str);	Перегрузка оператора вывода
		stroka& operator=(const stroka& other);	Перегрузка оператора присваивания
		char& operator[](unsigned int index);	Перегрузка метода взятие элемента(символа)
Class LinkedList			
Node* head;	Указатель на начало списка	void addStroka(stroka& str);	Вставка доп сроки и создание узла
int size;	Размер массива	int getSize() const;	Взятие размера
int Readstatus;	Статус чтения	void clearList();	Явная очистка списка
		int GetReadstatus();	Взятие статуса чтения
		void SetReadstatus(int set);	Установка статуса чтения
		void remove(int index)	Метод для удаления элемента по индексу
		istream& operator>>(istream& is, LinkedList& text);	Перегрузка оператора погружения текста
		ostream& operator<<(ostream& os, LinkedList& text);	Перегрузка оператора извлечения текста
		void process(fstream& Out,LinkedList& text);	Функция процесса
		void massage(fstream& Out,int mode,LinkedList& text);	Функция сообщений
Class Node			
stroka data;	Объект класса строка		
Node* next;	указатель на следующий объект		

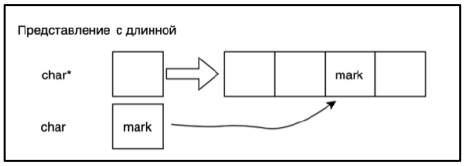
Представление алгоритма решения задачи

Имя ф-ции	Принадлежность	Назначение	Параметры			Возвращаемое значение	Внешние эффекты
			Входные	Выходные	Модифицируемые		
size	class stroka	Получение размера	.data	-	-	unsigned int Возращает длину массива	-
getline	class stroka	Взятие строки	-	.data	-	-	Выделение памяти под массив и его заполнение
bool isIn	class stroka	Проверка вхождения	const stroka& other	-	-	Bool 1-если подстрока входит 0-если не входит	-
count_words	class stroka	Подсчет количество слов	.data	-	struct StrMark& str	Int возвращает количество слов	-
operator<<	ostream	Перегрузка оператора <<	stroka& str	ostream& out	-	ostream	-
operator+	class stroka	Перегрузка оператора +	-	const stroka& other	-	stroka	-
clearlist() ;	class LinkedList	Очистка массива	-	-	.data	-	Перевыделение памяти
addStroka	LinkedList	Добавление узла	const stroka& s	-	.data	-	Перевыделение памяти
getSize	LinkedList	Получение размера	-	-	-	int Размер массива	-
GetReadstatus	LinkedList	Получение статуса чтения	-	-	-	int Статус чтения	-
SetReadstatus	LinkedList	Установка статуса чтения	int set	-	-	-	-
operator<<	ostream	Перегрузка оператора <<	ostream& os, Text& text)	-	-	ostream&	-
operator>>	istream	Перегрузка оператора >>	istream& is, Text& text	-	-	istream&	-
process	-	Основной процесс	-	-	fstream& Out,LinkedList & text	-	-
void massage	-	Метод сообщений	LinkedList & text	-	fstream& Out,int mode,	-	-

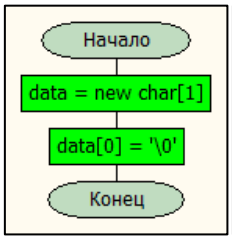

int Readstatus;
int size;
Node* head;
Class Linkedlist
*
nullptr
Class stroka
Class Node
Node* next;
nullptr

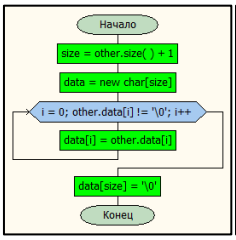

строка 1
строка 2
строка 3
строка 4
строка 5
nullptr
head
previous
current
next

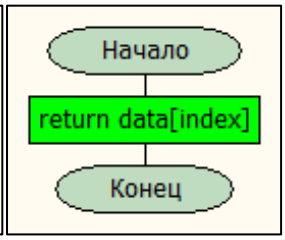

linkedlist.h
#include "linkedlist.h"
main.cpp
#include "strokalink.h"
strokalink.h
strokalink.cpp
linkedlist.cpp
#include <iostream>
#include <fstream>
iostream (заголовочный файл)
fstream (заголовочный файл)
компиляция
strokalink.o
linkedlist.o
main.o
линковка
стандартная библиотека
main.exe

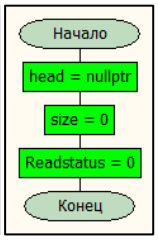

Представление с длинной
char*
mark
char
mark

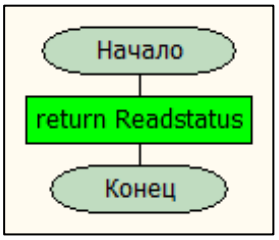
Описание алгоритма

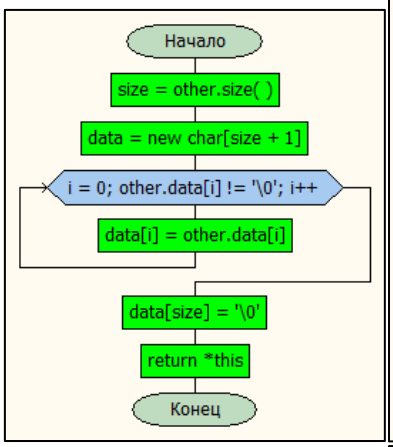


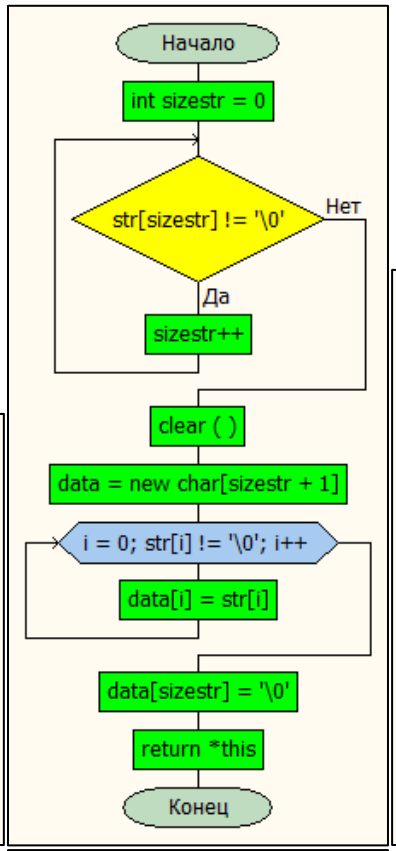


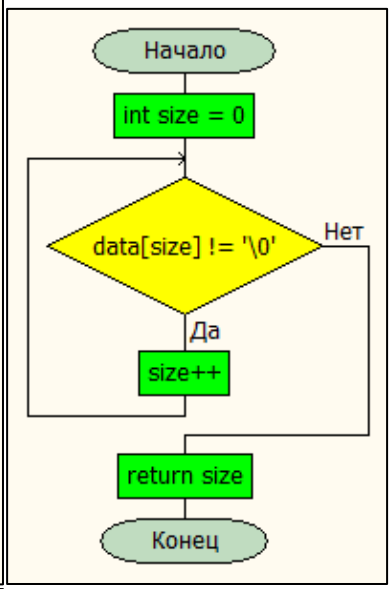


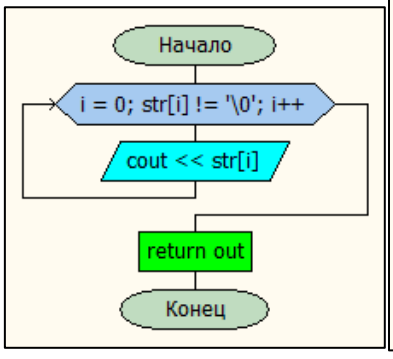


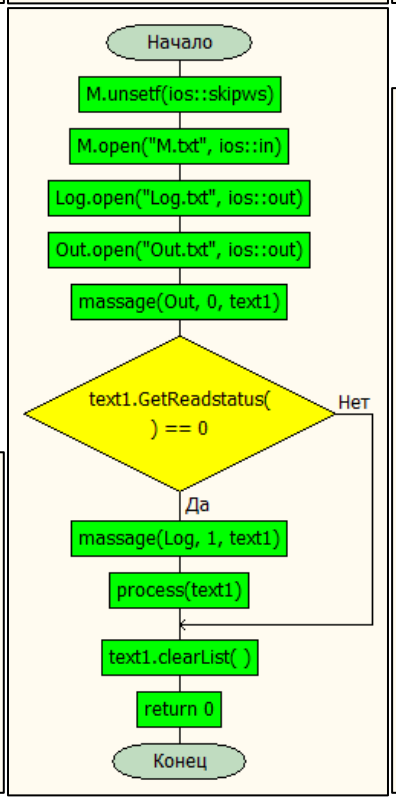


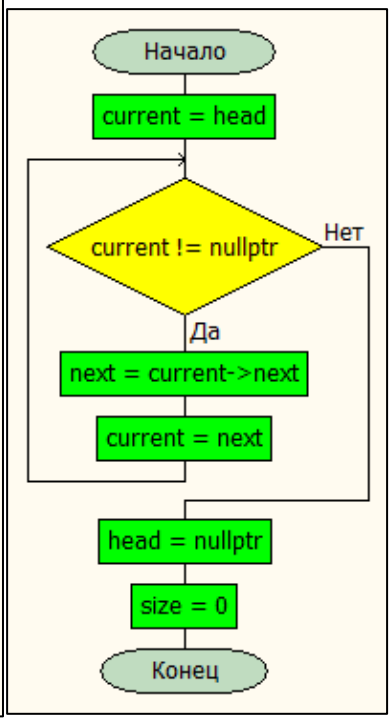


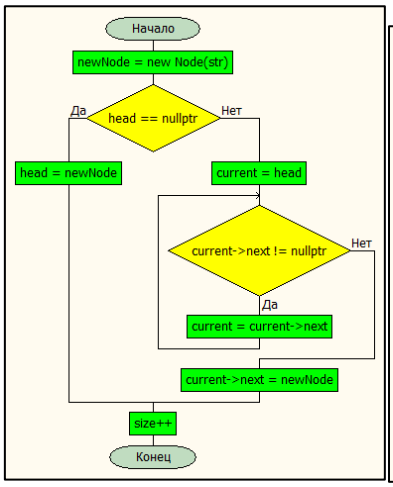


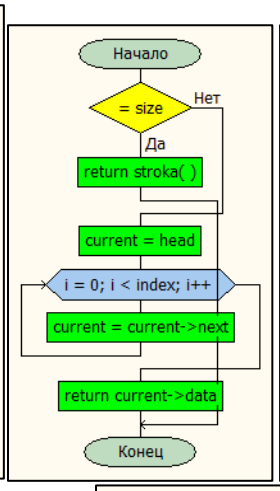


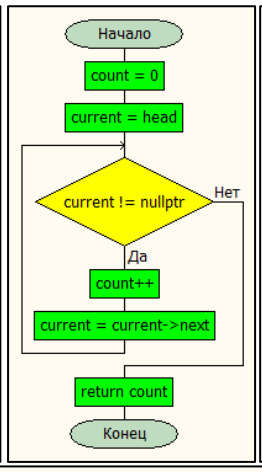


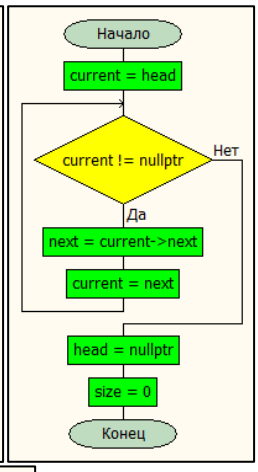


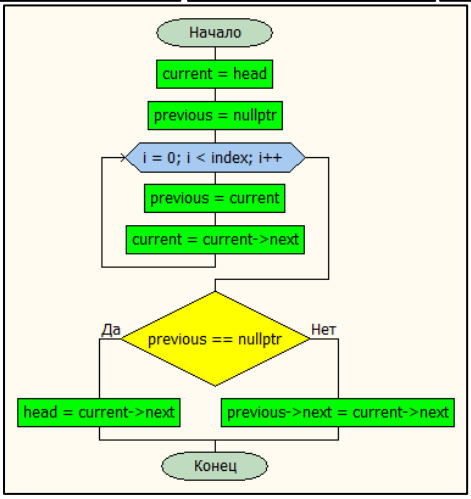


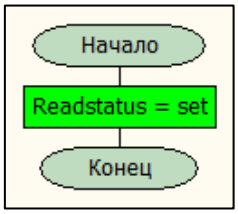

Начало
newNode = new Node(str)
head == nullptr
Да
Нет
head = newNode
current = head
current->next != nullptr
Нет
Да
current = current->next
current->next = newNode
size++
Конец

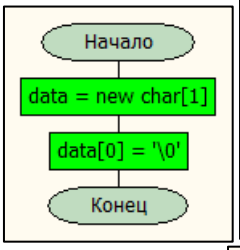

Начало
= size
Нет
Да
return stroka( )
current = head
i = 0; i < index; i++
current = current->next
return current->data
Конец

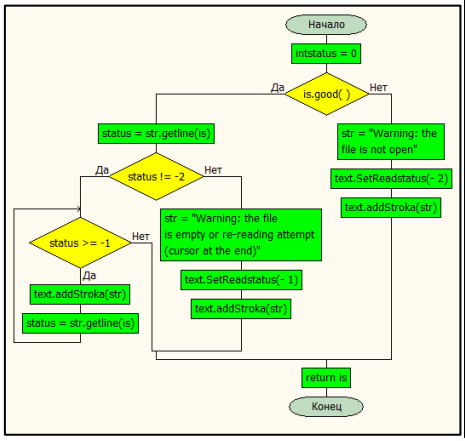

Начало
count = 0
current = head
current != nullptr
Нет
Да
count++
current = current->next
return count
Конец

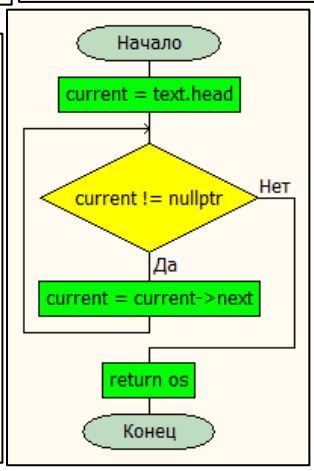

Начало
current = head
current != nullptr
Нет
Да
next = current->next
current = next
head = nullptr
size = 0
Конец

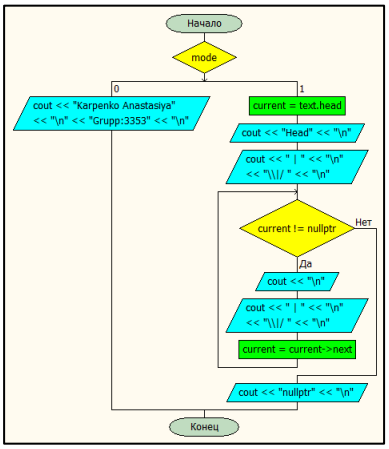

Начало
current = head
previous = nullptr
i = 0; i < index; i++
previous = current
current = current->next
Да
previous == nullptr
Нет
head = current->next
previous->next = current->next
Конец

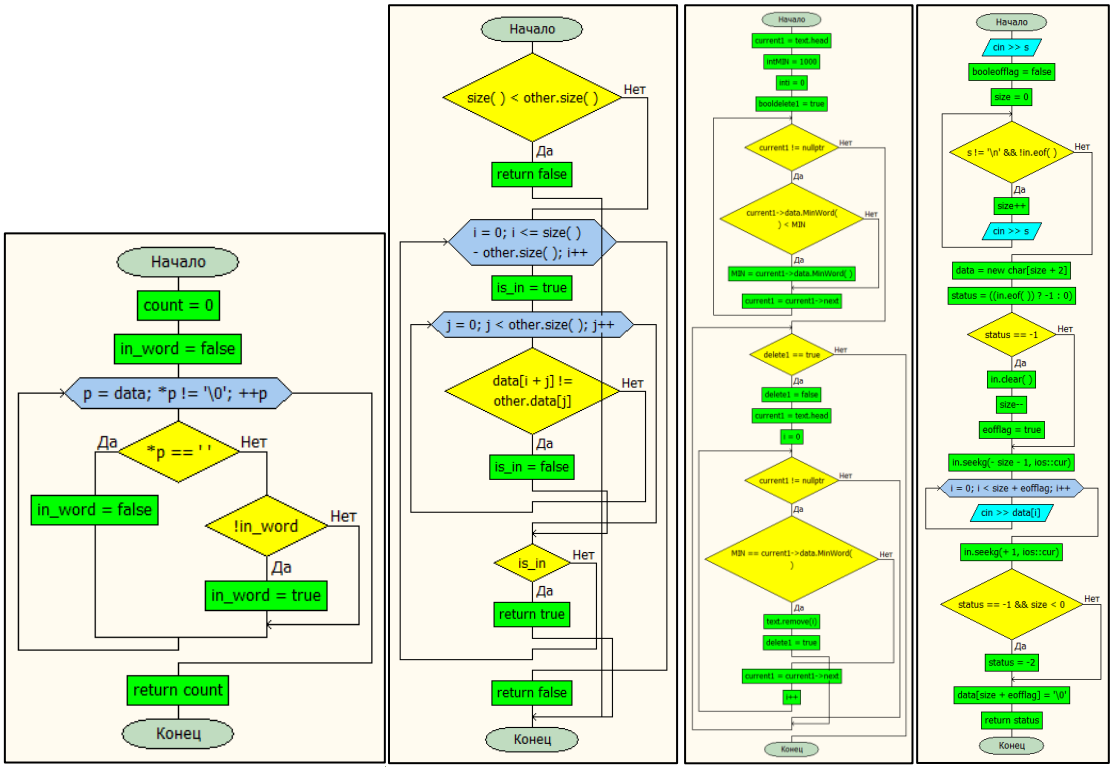

Начало
Readstatus = set
Конец


Начало
data = new char[1]
data[0] = '\0'
Конец


Начало
intstatus = 0
Да
is.good( )
Нет
status = str.getline(is)
str = "Warning: the file is not open"
Да
status != -2
Нет
text.SetReadstatus(- 2)
text.addStroka(str)
status >= -1
Нет
str = "Warning: the file is empty or re-reading attempt (cursor at the end)"
Да
text.addStroka(str)
text.SetReadstatus(- 1)
status = str.getline(is)
text.addStroka(str)
return is
Конец


Начало
current = text.head
current != nullptr
Нет
Да
current = current->next
return os
Конец


Начало
mode
0
1
cout << "Karpenko Anastasiya" << "\n" << "Grupp:3353" << "\n"
current = text.head
cout << "Head" << "\n"
cout << " | " << "\n" << "\\|/ " << "\n"
current != nullptr
Нет
Да
cout << "\n"
cout << " | " << "\n" << "\\|/ " << "\n"
current = current->next
cout << "nullptr" << "\n"
Конец



Текст программы

Main.cpp

```
#include "linkedlist.h"
#include "strokalink.h"
using namespace std;
int main(){
   fstream M;
   fstream Log;
   fstream Out;
   LinkedList text1;
   M.unsetf(ios::skipws);
   M.open("M.txt", ios::in);
   Log.open("Log.txt", ios::out);
   Out.open("Out.txt", ios::out);
   M>>text1;
   massage(Out,0,text1);
   Out<<text1<<"\n";
   if(text1.GetReadstatus()==0){
   massage(Log,1,text1);
   process(text1);
   Out<<text1<<"\n";}
   text1.clearList();
   return 0;
}
```

linkedlist.cpp

```
#include "linkedlist.h"
#include "strokalink.h"

using namespace std;
   LinkedList::LinkedList() {
      head = nullptr;
      size=0;
      Readstatus=0;

   }
   LinkedList::~LinkedList(){
      Node* current = head;
      while (current != nullptr) {
         Node* next = current->next;
         delete current;
         current = next;
      }
      head = nullptr;
      size = 0;
   };
void LinkedList::addStroka(stroka& str) {
      Node* newNode = new Node(str);
      if (head == nullptr) {
         head = newNode;
      } else {
         Node* current = head;
         while (current->next != nullptr) {
            current = current->next;
         }
         current->next = newNode;
      }
      size++;
   }
   stroka LinkedList::getStrokaAtIndex(int index) {
      if (index < 0 || index >= size) {
         // handle out of bounds index
         return stroka();
      }
      Node* current = head;
      for (int i = 0; i < index; i++) {
         current = current->next;
      }
      return current->data;
   }
   int LinkedList::getSize() const {
      int count = 0;
      Node *current = head;

      while (current != nullptr) {
         count++;
         current = current->next;
      }

      return count;
   }
   void LinkedList::clearList() {
      Node* current = head;
      while (current != nullptr) {
         Node* next = current->next;
         delete current;
         current = next;
      }
      head = nullptr;
      size = 0;
   }
   int LinkedList::GetReadstatus(){
     return Readstatus;
   }
   void LinkedList::SetReadstatus(int set){
    Readstatus=set;

}
   void LinkedList::remove(int index) {
      Node *current = head;
      Node *previous = nullptr;

      for (int i = 0; i < index; i++) {
         previous = current;
         current = current->next;
      }

      if (previous == nullptr) {
         head = current->next;
      } else {
         previous->next = current->next;
      }

      delete current;
   }
istream& operator>>(istream& is,  LinkedList& text){

      static stroka str;

      int status=0;

      if(is.good()){

        status=str.getline(is);

        if(status!=-2)
        {
          while(status>=-1)
          {
          text.addStroka(str);
          status=str.getline(is);
          }

        }else{
          str="Warning: the file is empty or re-reading attempt (cursor at the end)";
          text.SetReadstatus(-1);
          text.addStroka(str);
        }

      }else{
      str="Warning: the file is not open";
      text.SetReadstatus(-2);
      text.addStroka(str);
      }


      return is;
}
ostream& operator<<(ostream& os,  LinkedList& text) {
```

```
        Node *current = text.head;

        while (current != nullptr) {
            os << current->data;
            os << "\n";
            current = current->next;
        }

        return os;
    }
void process(LinkedList& text){

    Node *current1 = text.head;

    int MIN=1000;
    int i=0;
    bool delete1=true;

    while (current1 != nullptr){
        if(current1->data.MinWord()<MIN) {MIN=current1->data.MinWord();}
        current1 = current1->next;
    }


    while(delete1==true){
        delete1=false;

        current1 = text.head;
        i=0;
        while (current1 != nullptr){
        if(MIN==current1->data.MinWord()){text.remove(i);delete1=true; break;}
        current1 = current1->next;
        i++;
        }


    }
}
void massage(fstream& Out,int mode,LinkedList& text){

  switch(mode)
  {
    case 0:
    Out<<"Karpenko Anastasiya"<< "\n"
      <<"Grupp:3353"<< "\n";
      break;
    case 1:
    Node *current = text.head;
      Out<<"Head" << "\n";
      Out<<" | " << "\n"
        <<"\|/ " << "\n";
      while (current != nullptr) {
          Out << current->data;
          Out << "\n";
          Out<<" | " << "\n"
            <<"\|/ " << "\n";
          current = current->next;
      }
      Out<<"nullptr" << "\n";
      break;

  }
}
```

linkedlist.h

```
#ifndef P3_LINKEDLIST_H
#define P3_LINKEDLIST_H

#include <iostream>
#include <fstream>

#include "strokalink.h"
class Node {
public:
    stroka data;
    Node* next;
    Node(const stroka& str);
};
class LinkedList {
public:
    LinkedList();                   // Конструктор
    ~LinkedList();                  // Деструктор
    int getSize() const;            // Метод получения размера
    void addStroka(stroka& str);    // Метод добавления строки
    stroka getStrokaAtIndex(int index); // Метод получения строки по индексу
    void remove(int index);         // Метод удаления элемента по индексу
    int GetReadstatus();            // Метод получения статуса чтения
    void SetReadstatus(int set);    // Метод установки статуса чтения
    void clearList();               // Метод очистки списка
};
istream& operator>>(istream& is,  LinkedList& text);
ostream& operator<<(ostream& os,  LinkedList& text);
void process(LinkedList& text);
void massage(fstream& Out,int mode,LinkedList& text);
#endif
```

strokalink.cpp

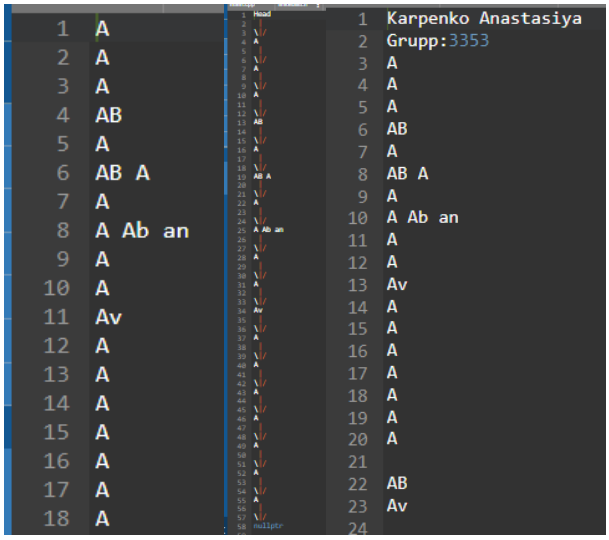
```
#include "strokalink.h"

using namespace std;

class Node {
public:
    stroka data;
    Node* next;

    Node(const stroka& str);
};
Node::Node(const stroka& str) : data(str), next(nullptr) {}
stroka::stroka() {
    data = new char[1];
    data[0] = '\0';

 //   cout<<"сработал конструктор строки"<<"\n";
}
stroka::~stroka() {
        delete[] data;
    //cout<<"сработал ДЕСТРУКТОР строки"<<"\n";
}
void stroka::clear() {

        delete[] data;
        data = new char[1];
        data[0] = '\0';
}
stroka::stroka(const stroka& other) {

int size = other.size() + 1;

data = new char[size];

for (int i = 0; other.data[i] != '\0'; i++) {
        data[i] = other.data[i];
    }
    data[size] = '\0';
}
int stroka::getline(istream& in) {
    char s;

    in >> s;

    bool eofflag=false;

    int size = 0;

    while (s != '\n' && !in.eof()) {
        size++;
        in >> s;
    }

    data = new char[size + 2];

    int status = ((in.eof()) ? -1 : 0);

    if (status == -1) {
        in.clear();
        size--;
        eofflag=true;
    }

    in.seekg(-size-1, ios::cur);



    for (int i = 0; i < size+eofflag; i++) {
        in >> data[i];
    }

    in.seekg(+1, ios::cur);

    if(status == -1 && size < 0)
    {
      status=-2;
    }

    data[size+eofflag]='\0';

    return status;
}
char& stroka::operator[](unsigned int index) {
    return data[index];
}
stroka& stroka::operator=(const stroka& other) {
    int size = other.size();

    delete[] data;

    data = new char[size + 1];

    for (int i = 0; other.data[i] != '\0'; i++) {
        data[i] = other.data[i];
    }

    data[size] = '\0';  // Добавляем нулевой символ в конец
    return *this;
}
stroka& stroka::operator=(const char* str) {
    unsigned int sizestr = 0;

    while (str[sizestr] != '\0') {
        sizestr++;
    }

    clear();

    data = new char[sizestr + 1];

    for (int i = 0; str[i] != '\0'; i++) {
        data[i] = str[i];
    }

    data[sizestr] = '\0';  // Добавляем нулевой символ в конец
    return *this;
}
unsigned int stroka::size() const {
    unsigned int size = 0;

    while (data[size] != '\0') {
        size++;
    }
    return size;
}
ostream& operator<<(ostream& out, stroka& str) {

    for (int i = 0; str[i]!='\0'; i++) {
        out<< str[i];
    }

    return out;
}
int stroka::count_words() {
    int count = 0;
    bool in_word = false;
    for (char* p = data; *p != '\0'; ++p) {
        if (*p == ' ') {
            in_word = false;
        } else if (!in_word) {
            ++count;
            in_word = true;
        }
    }
    return count;
}
int stroka::MinWord() {
        if (data == nullptr) {
            return 0;
        }

        int minWordLength = 10000;
        int currentWordLength = 0;

        for (int i = 0; data[i] != '\0'; i++) {
            if (data[i] != ' ') {
                currentWordLength++;
            } else {
                if (currentWordLength > 0 && currentWordLength < minWordLength) {
```

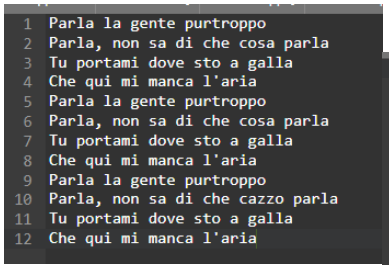
```
                minWordLength = currentWordLength;
            }
            currentWordLength = 0;
        }
    }
    if (currentWordLength > 0 && currentWordLength < minWordLength) {
        minWordLength = currentWordLength;
    }

    return minWordLength;
}
bool stroka::isIn(const stroka& other) const{
    if (size() < other.size())
        return false;

    for (int i = 0; i <= size() - other.size(); i++)
    {
        bool is_in = true;
        for (int j = 0; j < other.size(); j++)
        {
            if (data[i + j] != other.data[j])
            {
                is_in = false;
                break;
            }
        }
        if (is_in)
            return true;
    }
    return false;
}
```

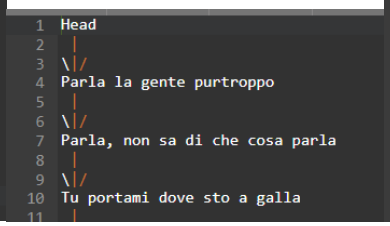
strokalink.h

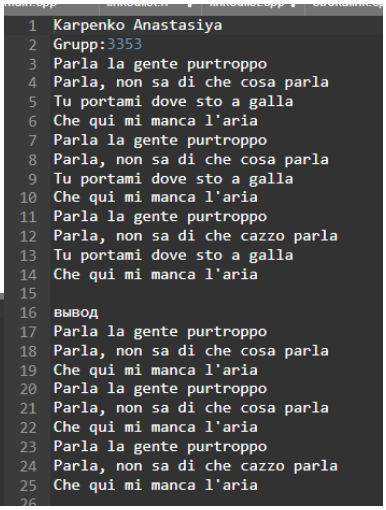
```
#ifndef P3_STROKALINK_H
#define P3_STROKALINK_H
#include <iostream>
#include <fstream>


using namespace std;

class stroka {
private:
    char* data;

public:
    //конструкторы
    stroka();
    stroka(const stroka& other);
    ~stroka();

    //перегрузка оператора
    char& operator[](unsigned int index);
    stroka& operator=(const stroka& other);
    stroka& operator=(const char* str);

    unsigned int size() const; //подсчет размера(очень бы хотелось его просто хранить:)

    int getline(istream& in); //взятие одной строки

    int count_words();//вспомогательный метод подсчета количество слов
    void clear();
    bool isIn(const stroka& other) const;
    int MinWord();
};
ostream& operator<<(ostream& out, stroka& str);
#endif
```

Результаты работы программы









Вывод

В ходе работы было освоено разбиение на несколько файлов: .cpp и .h. Была проведена работа с представлением однонаправленного линейного списка – в элементе строка символов, его обработка