**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**КАФЕДРА САПР**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

Тема: **«Представление строки символов с маркером, ее обработка»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0336 |  | Полещук А.В. |
| Преподаватель |  | Калмычков В.А. |

Санкт-Петербург

2021

Исходная формулировка задания

Найти номер последнего элемента с заданным значением.

1. Постановка задачи:
2. Организовать представление однонаправленного линейного списка – в элементе строка символов с маркером.
3. Обеспечить эффективное выполнение операция по преобразованию списка.
4. Реализовать чтение списка из файла любого размера.

Этапы решения задачи:

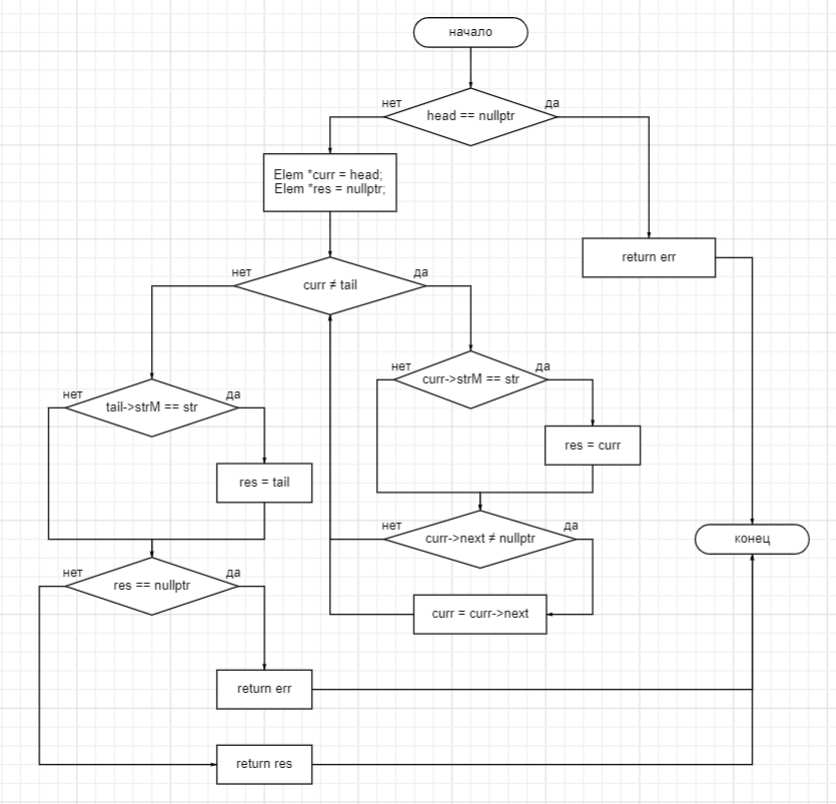
* Постановка задачи и определение особенностей реализации;
* Разработка класса для ввода и вывода строки;
* Проверка работы;
* Разработка основного алгоритма программы;
* Тестирование.

1. Анализ вариантов хранения и обработки информации.

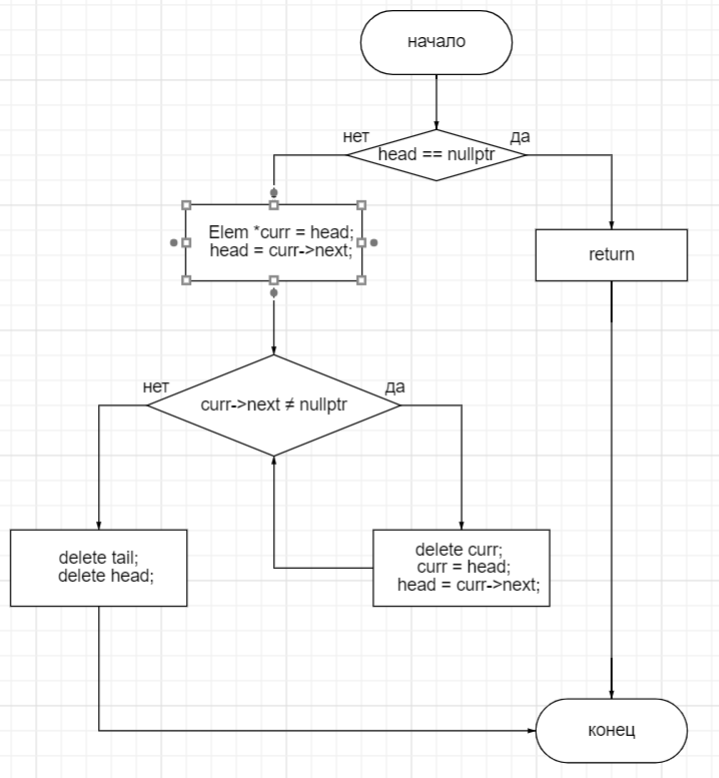
Целью является реализация задачи представления однонаправленного линейного списка – в элементе строка символов с маркером, для этого реализуем класс StrMarkList и структуру Elem. Первый будет хранить в себе указатели на первый и последний элементы списка, а также реализацию методов для взаимодействия со списком. Вторая же будет содержать поля с номером элемента, строкой, маркером и указателем на следующий элемент списка.

В классе списка необходимо создать указатель на строку, являющуюся началом списка. В структуре создать указатель на следующий элемент.

1. Описание методов.
2. Конструктор класса StrMarkList: корень и конец списка приравниваем к NULL.
3. Метод void StrMarkList::addNew(int id, char \*str, int size) добавляет новый элемент в конец списка. Если добавляемый элемент является первым (корень), то указатель на этот объект записывается в поля head и tail класса StrMarkList.
4. Метод Elem\* StrMarkList::search(char \*str) находит переданную строку среди элементов списка, после чего возвращает указатель на этот элемент. В случае отсутствия такой строки(или отсутствия элементов) возвращает элемент «ошибка».



1. Метод void StrMarkList::delList() удаляет все элементы списка, если список не пуст.



1. Метод Elem\* StrMarkList::getElem(int num) возвращает указатель на элемент с номером num.
2. Метод Elem\* StrMarkList::getHead() возвращает указатель на корневой элемент.

Класс LabIO реализует ввод/вывод из файла.

1. Метод int LabIO::loadFile() так же считывает строку из файла, однако из-за изменившейся задачи теперь при нахождении в строке маркера (в данном случае символ @, однако структура программы позволяет заменить его на любой другой без особых усилий), метод не заканчивает свою работу. При первом проходе по строке заполняется массив длин int LabIO::strLens[], далее, проходя по строке еще раз получаем каждую строку(элемент) и добавляем ее в список при помощи метода void StrMarkList::addNew(int id, char \*str, int size). В случае отсутствия строки(пустая строка, либо строка содержит только один символ - маркер) возвращается -1;
2. Метод int LabIO::saveFile(StrMarkList \*strM) сохраняет все элементы в файл, сохраняя исходную структуру файла.
3. Метод StrMarkList\* LabIO::getStr() возвращает указатель на объект списка
4. Метод int LabIO::getErrCode() возвращает err, который задается методом int LabIO::loadFile().
5. Программа на C++:

Файл main.cpp:

#include "labio.h"  
#include "strmarklist.h"  
  
using namespace std;  
  
int main(){  
 StrMarkList \*strL;  
 LabIO \*io = new LabIO();  
 if(io->getErrCode() != 0) return -1;  
 strL = io->getStr();  
 char findstr[100];  
 cout << "Input the word you need to find: " << endl;  
 cin >> findstr;  
 Elem \*str = strL->search(findstr);  
 cout << "string and element number " << str->strM << " " << str->num << endl;  
 strL->delList();  
 io->saveFile(strL);  
 return 0;  
}

Файл labio.cpp:

#include "labio.h"  
  
using namespace std;  
  
LabIO::LabIO(){  
 err = loadFile();  
}  
  
LabIO::~LabIO(){  
 delete fistr;  
}  
  
int LabIO::loadFile(){  
 int len;  
 fileIn.open("C:\\lab2\\lab2.txt", ios\_base::in);  
 if(fileIn.is\_open()) {  
 cout << "File is open to load." << endl;  
 }  
 else {  
 cout << "File isn't open." << endl;  
 return -1;  
 }  
 stringstream buf;  
 string str;  
 buf << fileIn.rdbuf();  
 str = buf.str();  
 len = str.length();  
 if(len <= 1) {  
 fistr = new StrMarkList();  
 cout << "Loaded string is empty!" << endl;  
 return -1;  
 }  
  
 fistr = new StrMarkList();  
  
 for(int i = 0; i < len; i++){  
 if(str[i] != mark) eLen++;  
 else {  
 eLen++;  
 strLens[eNum] = eLen;  
 eNum++;  
 eLen = 0;  
 }  
 }  
 len = 0;  
 char ch;  
 cout << "Loaded string is: " << endl;  
 for(int i = 0; i < eNum; i++) {  
 char tmpStr[strLens[i]];  
 int j;  
 for(j = 0; j < strLens[i]-1; j++){  
 if(str[len + j] != mark) ch = str[len + j];  
 tmpStr[j] = ch;  
 cout << tmpStr[j];  
 }  
 cout << " ";  
 tmpStr[j] = 0;  
 len += strLens[i];  
 fistr->addNew(i, tmpStr, strLens[i]);  
 }  
 cout << endl;  
 fileIn.close();  
 return 0;  
}  
  
int LabIO::saveFile(StrMarkList \*strM){  
 fileOut.open("C:\\lab2\\lab2.txt", ios\_base::out);  
 if(fileOut.is\_open()) {  
 cout << "File is open to save." << endl;  
 }  
 else {  
 cout << "File isn't open." << endl;  
 return -1;  
 }  
 if(strM->getHead() == nullptr){  
 cout << "List is empty";  
 fileOut << "@";  
 fileOut.close();  
 return 0;  
 }  
 Elem \*curr = strM->getHead();  
 for(int i = 0; i < eNum; i++) {  
 for(int j = 0; j < strLens[i]-1; j++){  
 fileOut << curr->strM[j];  
 }  
 fileOut << mark;  
 curr = fistr->getElem(i+1);  
 }  
 cout << "File has been saved";  
 fileOut.close();  
 return 0;  
}  
  
StrMarkList\* LabIO::getStr(){  
 return fistr;  
}  
int LabIO::getErrCode() {  
 return err;  
}

Файл labio.h:

#ifndef **LABIO\_H**#define **LABIO\_H**#include "strmarklist.h"  
#include <iostream>  
#include <fstream>  
#include <sstream>  
#include <cstring>  
#include <string>  
  
using namespace std;  
  
class LabIO{  
  
public:  
 LabIO();  
 ~LabIO();  
 int loadFile();  
 int saveFile(StrMarkList \*strM);  
 StrMarkList\* getStr();  
 int getErrCode();  
  
 private:  
 StrMarkList \*fistr;  
 ifstream fileIn;  
 ofstream fileOut;  
 char mark = '@';  
 int strLens[1000];  
 int eNum = 0;  
 int eLen = 0;  
  
 int err;  
};  
  
#endif

Файл strmarklist.cpp:

#include "strmarklist.h"  
  
using namespace std;  
  
StrMarkList::StrMarkList(){  
 head = nullptr;  
 tail = nullptr;  
}  
  
StrMarkList::~StrMarkList(){  
 delete head;  
 delete tail;  
}  
  
void StrMarkList::addNew(int id, char \*str, int size) {  
 Elem \*el = new Elem();  
 el->num = id;  
 size = strlen(str);  
 el->strM = new char[size];  
 strcpy\_s(el->strM, size+1, str);  
 el->mark = mark;  
 el->next = nullptr;  
 if(head == nullptr) {  
 head = el;  
 tail = el;  
 }  
 else {  
 tail->next = el;  
 tail = el;  
 }  
}  
  
Elem\* StrMarkList::search(char \*str) {  
 Elem\* err = new Elem();  
 err->strM = (char\*)"!!!error!!!";  
 err->num = -1;  
 if(head == nullptr) return err;  
 Elem \*curr = head;  
 Elem \*res = nullptr;  
 while(curr != tail){  
 if (!strcmp(curr->strM, str))res = curr;  
 if (curr->next != nullptr) curr = curr->next;  
 }  
 if(!strcmp(tail->strM, str)) res = tail;  
 if(res == nullptr) res = err;  
 return res;  
}  
  
void StrMarkList::delList() {  
 if(head == nullptr) return;  
 Elem \*curr = head;  
 head = curr->next;  
 for(;;){  
 if(curr->next != nullptr) {  
 delete curr;  
 curr = head;  
 head = curr->next;  
 }  
 else {  
 delete tail;  
 delete head;  
 break;  
 }  
 }  
}  
  
Elem\* StrMarkList::getElem(int num){  
 Elem\* curr = head;  
 for(int i = 0; i < num; i++) {  
 if(curr->next == nullptr) break;  
 curr = curr->next;  
 }  
 return curr;  
}  
  
Elem\* StrMarkList::getHead(){  
 return head;  
}

Файл strmarklist.h:

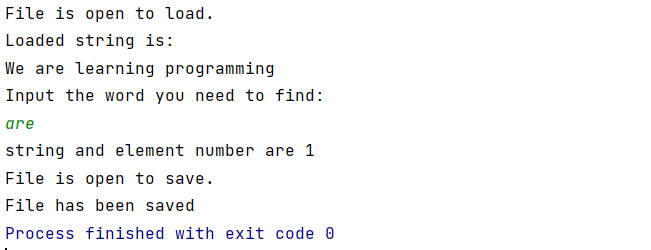
#ifndef **STRMARK\_H**#define **STRMARK\_H**#include <iostream>  
#include <cstring>  
  
  
struct Elem{  
 int num;  
 char \*strM;  
 char mark;  
 Elem \*next;  
};  
  
class StrMarkList{  
  
public:   
 StrMarkList();  
 ~StrMarkList();  
 void addNew(int id, char \*str, int size);  
 void delList();  
 Elem\* search(char \*str);  
 Elem\* getElem(int num);  
 Elem\* getHead();  
   
private:  
 Elem \*head;  
 Elem \*tail;  
 const char mark = '@';  
};  
  
#endif

1. Тестирование

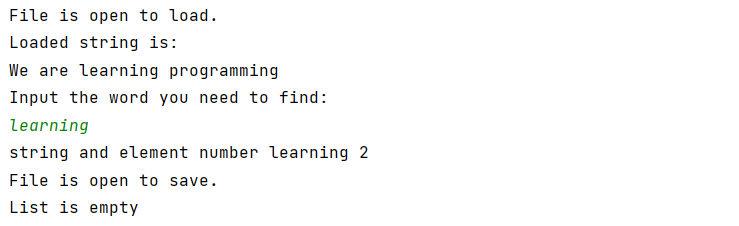
Файл *lab2.txt*

*“We@are@learning@programming@”*

1. Результат без удаления всех элементов списка:



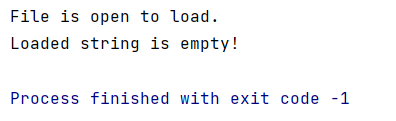
1. Результат с удалением списка:



Содержимое исходного файла после удаления списка:

«@»

1. Результат при загрузке пустого файла:



1. Вывод:
2. Организовано представление однонаправленного линейного списка – в элемент строка символов с маркером.
3. Обеспечено эффективное выполнение операций по преобразованию списка.
4. Реализовано чтение списка из файла любого размера.