Министерство образования и науки Российской Федерации

МГТУ им .Н.Э Баумана

Факультет «Робототехника и комплексная автоматизация»

Кафедра «РК6»

**Отчет по лабораторной работе**

По курсу «Объектно-ориентированное программирование»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Выполнил: |  |  | Студент | Журавлев Н.В. |
|  |  |  | Группа | РК6-12Б |
|  |  |  |  |  |
| Проверил: |  |  |  | Кузьмина И.А. |
|  |  |  | Дата | 23.04.2020 |
|  |  |  | Подпись | \_\_\_\_\_\_\_\_\_\_\_ |

**Лабораторная работа №6**

Разработать программу, которая в каждой входной символьной строке переставляет символы с минимальным кодом в конец строки без изменения взаимного расположения остальных символов. Все входные символьные строки должны передаваться программе через поток стандартного ввода. Результирующие строки символов должны отображаться через поток стандартного вывода. Во всех входных и результирующих строках на каждый символ с минимальным кодом должен указывать знак ^. Разработка программы должна быть основана на использовании информационной структуры связанного списка с наследованием операций просмотра, удаления и вставки его элементов

**Описание алгоритма**

1)Объявление переменных

2) Проверяется первый символ

3)Создание строки

3) Заполнение строки, нахождение кода минимального символа.   
4) Перестановка каждого минимального символав конец

5) Очистка памяти

**Текст программы**

**dlink.h**

#ifndef DLINK

#define DLINK

class Dlink{

protected:

Dlink\* next; // адрес следующей записи

Dlink\* prev; // адрес предыдущей записи

public:

Dlink(); // функция инициализации ссылок начала и конца

Dlink\* append(Dlink\*); // функция добавления новой записи

Dlink\* incr(); // функция получения адреса следующей записи

Dlink\* decr(); // функция получения адреса предыдущей записи

void excluse(); // функция исключения текущей записи

Dlink\* after(Dlink\*); // функция вставки записи после текущей

Dlink\* before(Dlink\*); // функция вставки записи перед текущей

Dlink\* tohead(void); // функция получения адреса начала списка

Dlink\* tohead(int); // функция сдвига в направлении начала списка

Dlink\* totail(void); // функция получения адреса конца списка

Dlink\* totail(int); // функция сдвига в направлении конца списка

};

#endif

dlink.cc

#include <stdio.h>

#include "dlink.h"

Dlink::Dlink()

{

next=prev=NULL;

}

Dlink\* Dlink::incr()

{

return(next);

}

Dlink\* Dlink::decr() {

return(prev);

}

Dlink\* Dlink::append(Dlink\* p) {

p->next = this;

prev = p;

return(p);

}

void Dlink::excluse(){

if(next != NULL)

next->prev = prev;

if(prev != NULL)

prev->next = next;

return;

}

Dlink\* Dlink::after(Dlink\* p){

p->next = next;

p->prev = this;

if(next != NULL)

next->prev = p;

next = p;

return(p->next);

}

Dlink\* Dlink::before(Dlink\* p){

p->next = this;

p->prev = prev;

if(prev != NULL)

prev->next = p;

prev = p;

return(p->prev);

}

Dlink\* Dlink::tohead(){

Dlink\* p = this;

Dlink\* q = NULL;

while(p != NULL){

q = p;

p = p->prev;

}

return(q);

}

Dlink\* Dlink::tohead(int n) {

Dlink\* p = this;

Dlink\* q = this;

int i = 0;

while(p != NULL) {

q = p;

if(i == n)

break;

p = p->prev;

i++;

}

return(q);

}

Dlink\* Dlink::totail() {

Dlink\* p = this;

Dlink\* q = NULL;

while(p != NULL) {

q = p;

p = p->next;

}

return(q);

}

Dlink\* Dlink::totail(int n){

Dlink\* p = this;

Dlink\* q = this;

int i = 0;

while(p != NULL){

q = p;

if(i == n)

break;

p = p->next;

i++;

}

return(q);

}

symlink.cc

#include <stdio.h>

#include <stdlib.h>

#include "dlink.h"

using namespace std;

class SymLink : public Dlink {

private:

unsigned char sym;

public:

SymLink(unsigned char c) : Dlink(), sym(c) {};

unsigned char get\_sym(){return sym;}

SymLink\* incr() {return((SymLink\*) Dlink::incr());};

SymLink\* decr() {return((SymLink\*) Dlink::decr());};

SymLink\* seek(int);

int print();

};

SymLink\* SymLink::seek(int n) {

if(n > 0)

return((SymLink\* ) Dlink::totail(n));

if(n < 0)

return((SymLink\* ) Dlink::tohead(abs(n)));

return(this);

}

int SymLink::print() {

SymLink\* p = this;

SymLink\* q;

int n = 0;

while(p != NULL) {

putchar(p->sym);

q = p->incr();

p = q;

n++;

}

return(n-2);

}

int main(int argc, char\* argv[]) {

unsigned seed = 0;

int count = 0;

int length;

int ch;

int p\_ch;

unsigned pos;

int side = 1;

int s = 1;

SymLink\* watch[2];

SymLink\* head;

SymLink\* tail;

SymLink\* p, \*q, \*sym;

Dlink\* (Dlink::\* insert[])(Dlink\*) = {&Dlink::after, &Dlink::before};

while( ch != '\n' && (ch = getchar()) != '\n') {

if(ch != ' ' && ch != '\t') {

printf("Слово №%d:",s++);

watch[0] = head = new SymLink('\n');

watch[1] = tail = new SymLink('\n');

tail ->before(head);

q = new SymLink(ch);

tail->before(q);

p\_ch = ch;

while((ch = getchar()) != ' ' && ch != '\t' && ch != '\n') {

if (ch < p\_ch) {

p\_ch = ch;

}

q = new SymLink(ch);

tail->before(q);

}

if((length = head->print ( ) - 1) < 2)

count = length;

count = 1;

int end = length + 2;

int r = 1;

while(count < end) {

sym = head->seek(count);

if(p\_ch == sym->get\_sym()){

printf("Действие №%d:", r++);

head->print();

pos = count;

printf("%\*c\n", pos, '^');

q = head->seek(pos);

q->excluse();

(watch[side]->\*insert[side])(q);

head->print();

printf("%\*c\n", length + 1, '^');

count--;

end--;

}

count++;

}

p = tail;

while(p != NULL) {

q = p->decr();

p->excluse();

delete p;

p = q;

}

}

}

return(length+1);

}

**Тесты**

**Вывод Ввод**

123 312 4321 3113 123 312 4321 3113

Слово №1:

123

Действие №1:

123

^

231

^

Слово №2:

312

Действие №1:

312

^

321

^

Слово №3:

4321

Действие №1:

4321

^

4321

^

Слово №4:

3113

Действие №1:

3113

^

3131

^

Действие №2:

3131

^

3311

^

**Результат**

Строки, у которых минимальные символы в конце.

**Списки литературы**

1)Волосатова Т. М., Родионов С. В. «Объектно-ориентированное программирование на С++» 2)http://bigor.bmstu.ru/?cnt/?doc=VU/base.cou