Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н. Э. Баумана (национальный исследовательский университет)

Факультет «Робототехника и комплексная автоматизация»

Кафедра «Системы автоматизированного проектирования»

**Отчет по лабораторной работе**

По курсу «Объектно-ориентированное программирование»

Выполнил: Студент Сергеева Д.К.

Группа РК6-26Б

Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2020 г.

**ООП Вариант L20**

**Задание:**

Разработать программу для разделения любой заданной символьной строки на латинские буквы, десятичные цифры и знаки пунктуации. Заданная входная символьная строка должна передаваться программе через поток стандартного ввода. Результаты классифицирования ее символов должна отображать строка потока стандартного вывода, где все символы одного класса расположены рядом друг с другом. При этом первый символ каждого класса должен быть указан знаком ^. При разработке программы следует использовать информационную структуру связанного списка с наследованием операций просмотра, удаления и вставки его элементов.

**Алгоритм:**

Для разделения заданной символьной строки на латинские буквы, цифры и знаки пунктуации сначала запомним их в двусвязный список, реализованный с помощью класса. Введём указатели на начало и конец списка, а также дополнительный указатель для того, чтобы запоминать 3 группу необходимых символов. Идя по всем значениям списка, будем сортировать каждый символ в определенную группу: если встретим букву, то её не будем трогать, если встретили знак пунктуации, то запишем его перед хвостом, если встретилась буква, записываем её перед дополнительным указателем. Таким образом в отсортированном списке сначала у нас будут все буквы, потом цифры, потом знаки пунктуации.

**Входные данные:**

Символьная строка

**Выходные данные:**

Отсортированный двусвязный список, где начало каждой группы символов помечено символом ‘^’.

**Текст программы:**

**Dlink.h**

#ifndef DLINK

#define DLINK

#include <cstdio>

class Dlink

{

protected:

Dlink\* \_next; //адрес следующего элемента

Dlink\* \_prev; //адрес предыдущего элемента

public:

Dlink(); //инициализация ссылок начала и конца

void exclusion(); //ф-ия исключения текущего элемента

Dlink\* after(Dlink\*); //ф-ия вставки элемента после текущего

Dlink\* before(Dlink\*); //ф-ия вставки элемента до текущего

Dlink\* get\_next(); //ф-ия получения адреса следующего элемента

Dlink\* get\_prev(); // ф-ия получения адреса предыдущего элемента

};

#endif

**Dlink.cpp**

#include "Dlink.h"

Dlink::Dlink()

{

\_next = NULL;

\_prev = NULL;

}

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::get\_next()

{

return \_next;

}

Dlink \* Dlink::get\_prev()

{

return \_prev;

}

void Dlink::exclusion()

{

if (\_next != NULL)

\_next->\_prev = \_prev;

if (\_prev != NULL)

\_prev->\_next = \_next;

}

**Symlink.cpp**

#include <cstdio>

#include <cstdlib>

#include <cctype>

#include "Dlink.h"

using namespace std;

class SymLink : public Dlink

{

private:

unsigned char \_sym;

public:

SymLink(unsigned char ch) : Dlink(), \_sym(ch) {};

void print();

unsigned char getsym() { return \_sym; };

SymLink\* get\_next() { return (SymLink\*)Dlink::get\_next(); };

SymLink\* get\_prev() { return (SymLink\*)Dlink::get\_prev(); };

};

void SymLink::print()

{

SymLink\* p = this;

p = (SymLink\*)p->\_next;

printf("Link:\n");

while ((p->\_next) != NULL)

{

printf("%c", p->\_sym);

p = (SymLink\*)p->\_next;

}

printf("\n");

}

int main()

{

SymLink\* head; //указатель на начало списка

SymLink\* tail; //указатель на конец списка

SymLink\* sep; //дополнительный указатель

head = new SymLink('\n');

tail = new SymLink('\n');

sep = tail;

tail->before(head);

SymLink\* p;

unsigned char ch;

int len = 0;

printf("Input string and click enter:\n");

while ((ch = getchar()) != '\n') //ввод символьной строки

{

if (isalnum(ch) || ispunct(ch))

{

p = new SymLink(ch);

sep->before(p);

++len;

}

else

printf("\"%c\" is not correct symbol.\n", ch);

}

p = head->get\_next();

int countNum = 0, countPunct = 0;

for (int i = 0; i < len; ++i) //разделение символов на 3 группы

{

ch = p->getsym();

if (isdigit(ch))

{

SymLink\* q;

q = p;

p = p->get\_next();

q->exclusion();

sep->before(q);

++countNum;

}

else if (ispunct(ch))

{

SymLink\* q;

q = p;

p = p->get\_next();

q->exclusion();

tail->before(q);

if (!countPunct)

sep = q;

++countPunct;

}

else

p = p->get\_next();

}

head->print();

if ((len - countNum - countPunct) > 0)

printf("^");

if (countNum != 0)

printf("%\*c", (len - countNum - countPunct), '^');

if (countPunct != 0)

printf("%\*c", (countNum > 0) ? (countNum) : (len - countPunct), '^');

printf("\n");

SymLink\* q;

p = tail;

while (p != NULL)

{

q = p->get\_prev();

p->exclusion();

delete p;

p = q;

}

return 0;

}

**Тесты:**

|  |  |  |
| --- | --- | --- |
| **Номер** | **Вход** | **Выход** |
| 1 | !g4 | g4!  ^^^ |
| 2 | d45!;rty | drty45!:  ^ ^ ^ |

**Список использованной литературы:**

* Волосатова Т.М., Родионов С.В. Лекции по курсу «Объектно-ориентированное программирование»
* bigor.bmstu.ru