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 Петраков С.А.

Группа РК6-26Б

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

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2020 г.

**Вариант ООП L16**

**Задание:**

Разработать программу для разделения любой заданной символьной строки на 2 группы символов, бинарные коды которых заканчиваются нулевым и единичным разрядом. Входная строка символов должны передаваться программе через поток стандартного ввода. Результат группировки символов должен отображаться строкой потока стандартного вывода, где все символы каждой группы расположены рядом, а под начальным символом каждой группы указано значение младшего разряда его бинарного кода. Разработка программы должна быть основана на использовании информационной структуры связанного списка с наследованием операций просмотра, удаления и вставки его элементов.

**Алгоритм:**

Для реализации я использую класс в котором наследуется двусвязный список и хранится символ.

1. Вводим строку и запоминаем ее в двусвязный список.
2. Проходим по всем элементам списка и если последний бит равен единице, то переставляем его в конец.
3. Выводим полученную строку и на след строке выводим 0 и 1 под началом каждой группы символов.

**Входные данные:**

Строка символов

**Выходные данные:**

Строка символов отсортированная, на след строке 0 и 1 где символ подписан по началом группы

**Текст программы:**

**main.cpp**

#include "SymLink.h"

#include "DualLink.h"

#include <iostream>

#include <cstdio>

int main()

{

//Init module

char ch;

int count = 0;

int countOnes = 0;

SymLink\* head;

SymLink\* tail;

SymLink\* watch[2];

SymLink\* temp;

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

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

head->insertAfter(tail);

//Input module

while ((ch = std::cin.get()) != '\n')

{

temp = new SymLink(ch);

tail->insertBefore(temp);

count++;

}

//Processing module

temp = head->getNext();

for (int i = 0; i < count; i++)

{

if ((temp->getSymbol()) % 2 == 1)

{

SymLink\* t = temp->getPrev();

temp->exception();

tail->insertBefore(temp);

temp = t->getNext();

countOnes++;

}

else

temp = temp->getNext();

}

//Output module

head->printList();

if (count != 0)

{

if ((count - countOnes) == 0)

printf("1");

else if (countOnes == 0)

printf("0");

else

printf("0%\*c1", (count - countOnes - 1), ' ');

}

temp = watch[0];

//Clear memory

while (temp != NULL)

{

head = temp->getNext();

temp->exception();

delete temp;

temp = head;

}

return 0;

}

**DualLink.h**

#pragma once

#ifndef DUALLINK

#define DUALLINK

#include <cstdio>

// Dlink class

class DualLink {

protected:

DualLink\* \_next;

DualLink\* \_prev;

public:

DualLink(); //Init list

void exception(); //Except element

void insertAfter(DualLink\*); //insert after current

void insertBefore(DualLink\*); //insert before current

DualLink\* getNext(); //Get next element

DualLink\* getPrev(); //Get prev

DualLink\* getHead(); // Get head of list

DualLink\* getTail(); // Get taif of list

int count();

};

#endif

**DualLink.cpp**

#include "DualLink.h"

DualLink::DualLink()

{

\_next = NULL;

\_prev = NULL;

}

void DualLink::exception()

{

if (\_next != NULL)

\_next->\_prev = \_prev;

if (\_prev != NULL)

\_prev->\_next = \_next;

return;

}

void DualLink::insertAfter(DualLink\* q)

{

q->\_next = \_next;

q->\_prev = this;

if (\_next != NULL)

\_next->\_prev = q;

\_next = q;

}

void DualLink::insertBefore(DualLink\* q)

{

q->\_next = this;

q->\_prev = \_prev;

if (\_prev != NULL) {

\_prev->\_next = q;

}

\_prev = q;

}

DualLink\* DualLink::getNext()

{

return \_next;

}

DualLink\* DualLink::getPrev()

{

return \_prev;

}

DualLink\* DualLink::getHead()

{

DualLink\* p = this;

DualLink\* q = this;

while (p != NULL)

{

q = p;

p = p->\_prev;

}

return q;

}

DualLink\* DualLink::getTail()

{

DualLink\* p = this;

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

p = p->\_next;

return p;

}

int DualLink::count()

{

DualLink\* p = getHead();

int n = 0;

while (p->\_next != NULL)

{

n++;

p = p->\_next;

}

return n;

}

**SymLink.h**

#pragma once

#ifndef SYMLINK

#define SYMLINK

#include "DualLink.h"

class SymLink : public DualLink

{

private:

unsigned char \_sym;

public:

SymLink(unsigned char);

unsigned char getSymbol();

SymLink\* getNext();

SymLink\* getPrev();

SymLink\* getHead();

SymLink\* getTail();

void printList();

};

#endif

**SymLink.cpp**

#include "SymLink.h"

#include <iostream>

SymLink::SymLink(unsigned char c): DualLink(), \_sym(c) {}

unsigned char SymLink::getSymbol()

{

return \_sym;

}

SymLink\* SymLink::getNext()

{

return (SymLink\*)DualLink::getNext();

}

SymLink\* SymLink::getPrev()

{

return (SymLink\*)DualLink::getPrev();

}

SymLink\* SymLink::getHead()

{

return (SymLink\*)DualLink::getHead();

}

SymLink\* SymLink::getTail()

{

return (SymLink\*)DualLink::getTail();

}

void SymLink::printList()

{

SymLink\* p = getHead();

while (p->\_next != NULL)

{

std::cout << p->\_sym;

p = p->getNext();

}

std::cout << std::endl;

return;

}

**Тесты:**

|  |  |  |
| --- | --- | --- |
| **Номер** | **Вход** | **Выход** |
| 1 | ccacaaccee | ccacaaccee  1 |
| 2 | bdbfbdbf | bdbfbdbf  0 |
| 3 | abcdbca |  |
| 4 | Пустой ввод | Пустой вывод |

**Список использованной литературы:**

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