МИНИСТЕРСТВО НАУКИ ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«**МИРЭА - Российский технологический университет**»

РТУ МИРЭА

Отчет по выполнению практического задания №6
Тема:
«Двунаправленные динамические списки»
Дисциплина: «Структуры и алгоритмы обработки данных»

Выполнил студент: Подоплелов А.С.

Группа: ИНБО-21-23

Москва - 2024

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ.....3
1.1 УСЛОВИЕ ЗАДАНИЯ.....3
2 РЕШЕНИЕ ЗАДАНИЯ.....3
2.1 РАЗРАБОТКА СТРУКТУРЫ УЗЛА СПИСКА И РЕАЛИЗАЦИЯ ФУНКЦИЙ ДЛЯ ВЫПОЛНЕНИЯ ОПЕРАЦИЙ НАД ДВУНАПРАВЛЕННЫМ ДИНАМИЧЕСКИМ СПИСКОМ.....3
2.2 ДОПОЛНИТЕЛЬНЫЕ ОПЕРАЦИИ НАД СПИСКОМ, УКАЗАННЫЕ ВАРИАНТОМ.7
2.3 РАЗРАБОТКА ПРОГРАММЫ, УПРАВЛЯЕМОЙ ТЕКСТОВЫМ МЕНЮ И ТЕСТИРОВАНИЕ ПРОГРАММЫ.....9
ВЫВОД.....9
СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ.....10

1 ПОСТАНОВКА ЗАДАЧИ

Цель работы: получить знания и практические навыки управления двунаправленным списком в программах на языке C++.

1.1 УСЛОВИЕ ЗАДАНИЯ

Разработать модульную программу, которая демонстрирует выполнение всех операций, определенных вариантом, над линейным двунаправленным списком. Для определения структуры узла списка, используйте тип «struct» или «class». Мой индивидуальный вариант под номером 6.

Тип информационное части узла списка:

1. Номер автобусного маршрута;

2. Время отправления (целое число);

3. Номер автобуса (формат госномера автотранспорта в России);

4. Стоимость одной поездки;

5. Дата отправления.

Дополнительные Операции:

- Вставить новый узел после последнего узла с заданным номером автобуса;

- Удалить все узлы заданного автобуса;

- Подсчитать, сколько раз автобус выходил на маршрут в течении заданного дня.

2 РЕШЕНИЕ ЗАДАНИЯ

2.1 РАЗРАБОТКА СТРУКТУРЫ УЗЛА СПИСКА И РЕАЛИЗАЦИЯ ФУНКЦИЙ ДЛЯ ВЫПОЛНЕНИЯ ОПЕРАЦИЙ НАД ДВУНАПРАВЛЕННЫМ ДИНАМИЧЕСКИМ СПИСКОМ

Инициализация списка предназначена для создания корневого узла списка, у которого поля указателей на следующий и предыдущий узлы содержат нулевое значение.

Реализуем структуру узла списка и функцию инициализации двунаправленного динамического списка на языке C++ (См. Рисунок 1).

```
struct list {
    string numb; // Формат знаков – 3 буквы, 3 цифры -> Автотранспорт в России
    struct list* next; // Указатель на следующий элемент
    struct list* prev; // Указатель на предыдущий элемент
};
//Инициализация ДЛС
struct list* init(string first) {
    struct list* lst; // выделение памяти под корень списка
    lst = (struct list*)malloc(sizeof(struct list));
    lst->numb = first;
    lst->next = NULL; // Указатель на следующий узел
    lst->prev = NULL; // Указатель на предыдущий узел
    return(lst);
}
```

Рисунок 1 - Реализация структуры узла списка и функции инициализации двунаправленного динамического списка на языке C++

Помимо этого, представим схему двусвязного динамического линейного списка в виде таблицы 1.

Таблица 1 — Схема узла двусвязного линейного списка

list	
Тип	**Имя свойства**
string	numb
Указатель на следующий элемент	next
Указатель на предыдущий элемент	prev

Реализуем функцию вставки узла в двусвязный динамический список на языке C++, а также представим схему данного алгоритма (См. Рисунок 2-3).

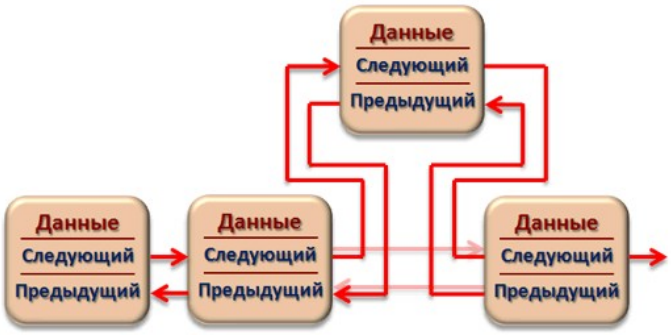

Рисунок 2 — Схема функции вставки узла в двусвязный динамический список

```
struct list* addelem(list* lst, string number) {
    struct list* temp, * p;
    temp = (struct list*)malloc(sizeof(list));
    p = lst->next; // сохранение указателя на следующий узел
    lst->next = temp; // предыдущий узел указывает на создаваемый
    temp->numb = number; // сохранение поля данных добавляемого узла
    temp->next = p; // созданный узел указывает на следующий узел
    temp->prev = lst; // созданный узел указывает на предыдущий узел
    if (p != NULL)
        p->prev = temp;
    return(temp);
}
```

Рисунок 3 - Реализация функции вставки узла в двусвязный динамический список на языке C++

Удаление узла двусвязного линейного списка включает в себя следующие этапы:

- установка указателя «следующий» предыдущего узла на узел, следующий за удаляемым;
- установка указателя «предыдущий» следующего узла на узел, предшествующий удаляемому;
- освобождение памяти удаляемого узла.

В качестве аргументов функции удаления узла двусвязного линейного списка передается указатель на удаляемый узел.

Реализуем функцию удаление узла двусвязного линейного списка, а также представим схему данного алгоритма (См. Рисунок 4-5).

Рисунок 4 — Схема функции удаления узла двусвязного линейного списка

```
struct list* deletelem(list* lst) {
    struct list* prev, * next;
    prev = lst->prev; // узел, предшествующий lst
    next = lst->next; // узел, следующий за lst
    if (prev != NULL)
        prev->next = lst->next; // переставляем указатель
    if (next != NULL)
        next->prev = lst->prev; // переставляем указатель
    free(lst); // освобождаем память удаляемого элемента
    return(prev);
}
```

Рисунок 5 — Реализация функции удаления узла двусвязного линейного списка на языке C++

Реализуем функцию вывода элементов двусвязного линейного списка на языке C++ (См. Рисунок 6).

```
void listprint(list* lst) {
    struct list* p;
    p = lst;
    do {
        printf("%d ", p->numb); // вывод значения элемента p
        p = p->next; // переход к следующему узлу
    } while (p != NULL); // условие окончания обхода
}
```

Рисунок 6 — Реализация функции вывода элементов двусвязного линейного списка на языке C++

Аналогично, реализуем функцию обратного вывода элементов двусвязного линейного списка на языке C++ (См. Рисунок 7).

```
void listprintr(list* lst)
{
    struct list* p;
    p = lst;
    while (p->next != NULL)
        p = p->next;  // переход к концу списка
    do {
        printf("%d ", p->numb); // вывод значения элемента p
        p = p->prev; // переход к предыдущему узлу
    } while (p != NULL); // условие окончания обхода
}
```

Рисунок 7 — Реализация функции обратного вывода элементов двусвязного линейного списка на языке C++

Реализуем функцию поиска узла с заданным значением, где операция должна возвращать указатель на узел с заданным значением, выполненным на языке C++ (См. Рисунок 8).

```
list* search(list l, string* n) {
    while (l.next != NULL) {
        if (!(l.next->numb, n))
            return l.next;
        l.next = l.next->next;
    }
    return l.next;
}
```

Рисунок 8 — Реализация функции поиска узла с заданным значением двусвязного линейного списка на языке C++

2.2 ДОПОЛНИТЕЛЬНЫЕ ОПЕРАЦИИ НАД СПИСКОМ, УКАЗАННЫЕ ВАРИАНТОМ

Реализация функции вставки нового узла после последнего узла с заданным номером автобуса (См. Рисунок 9). Перед тем, как проводить какие-то операции с компонентами, их необходимо добавить в список, чем мы сейчас и займемся. Прежде всего, нам необходимо создать новый компонент и заполнить его информационную часть. Так как мы будем помещать его в конец списка, то ссылочная часть узла будет иметь значение «NULL».

```
void comp_in(list& l, string* n, string* v) {
    list* c = new list();
    (c->numb, 20, n);
    (c->_numb, 10, v);
    c->next = NULL;
    if (chk_empty(l))
        l.next = c;
    else
        l.prev->next = c;
    l.prev = c;
}
```

Рисунок 9 — Реализация функции вставки нового узла после последнего узла с заданным номером автобуса двусвязного линейного списка на языке C++

Реализация функции удаления всех узлов заданного автобуса (См. Рисунок 10).

```
void comp_del(list& l, list* c) {
    if (c == l.next) {
        l.next = c->next;
        return;
    }
    list* r = new list();
    r = l.next;
    while (r->next != c)
        r = r->next;
    r->next = c->next;
    delete[] c;
};
```

Рисунок 10 — Реализация функции удаления всех узлов заданного автобуса в двусвязном линейном списке на C++

Реализация функции подсчета, сколько раз автобус выходил на маршрут в течении заданного дня (См. Рисунок 11).

```
struct list* swap(struct list* lst1, struct list* lst2, struct list* head) {
    struct list* prev1, * prev2, * next1, * next2;
    prev1 = lst1->prev;  // узел предшествующий lst1
    prev2 = lst2->prev;  // узел предшествующий lst2
    next1 = lst1->next; // узел следующий за lst1
    next2 = lst2->next; // узел следующий за lst2
    if (lst2 == next1) { // обмениваются соседние узлы
        lst2->next = lst1;
        lst2->prev = prev1;
        lst1->next = next2;
        lst1->prev = lst2;
        if (next2 != NULL)
            next2->prev = lst1;
        if (lst1 != head)
            prev1->next = lst2;
    }
    else if (lst1 == next2)  // обмениваются соседние узлы
    {
        lst1->next = lst2;
        lst1->prev = prev2;
        lst2->next = next1;
        lst2->prev = lst1;
        if (next1 != NULL)
            next1->prev = lst2;
        if (lst2 != head)
            prev2->next = lst1;
    }
    else  { // обмениваются отстоящие узл
        if (lst1 != head)  // указатель prev можно установить только для элемента,
            prev1->next = lst2; // не являющегося корневым
        lst2->next = next1;
        if (lst2 != head)
            prev2->next = lst1;
        lst1->next = next2;
        lst2->prev = prev1;
        if (next2 != NULL) // указатель next можно установить только для элемента,
            next2->prev = lst1; // не являющегося последним
        lst1->prev = prev2;
        if (next1 != NULL)
            next1->prev = lst2;
    }
    if (lst1 == head)
        return(lst2);
    if (lst2 == head)
        return(lst1);
    return(head);
}
```

Рисунок 11 — Реализация подсчета, сколько раз автобус выходил на маршрут в течении заданного дня в двусвязном линейном списке на C++

2.3 РАЗРАБОТКА ПРОГРАММЫ, УПРАВЛЯЕМОЙ ТЕКСТОВЫМ МЕНЮ И ТЕСТИРОВАНИЕ ПРОГРАММЫ

Управление программой будет реализовано с помощью редактирования кода, путем вноса и выноса фрагментов в качестве комментариев.

Тестирование программы (См. Рисунок 12-13).

```
#1 - Создание списка. Генерация списка.
Введите количество номеров.
5
#4 - Вывод списка в двух направлениях.
4h0q7 l554ae c725xd b223cs 1q7f8
1q7f8 b223cs c725xd l554ae 4h0q7
#2 - Вставка узла.
4h0q7 l554ae c725xd b223cs 1q7f8 59f2xu #3 - Удаление узла.
4h0q7 l554ae c725xd b223cs 1q7f8
Задайте номер автобуса. Индивидуальное задание #1
342fse
4h0q7 l554ae c725xd b223cs 1q7f8 342fse
Удаление всех узлов. Индивидуальное задание #2
Done!/
Result = 0. Индивидуальное задание #3
```

Рисунок 12 — Тестирование №1

```
#1 - Создание списка. Генерация списка.
Введите количество номеров.
10
#4 - Вывод списка в двух направлениях.
4h0q7 l554ae c725xd b223cs 1q7f8 o597wx 4t3s1 g478rp y913pr 06c2mt
06c2mt y913pr g478rp 4t3s1 o597wx 1q7f8 b223cs c725xd l554ae 4h0q7
#2 - Вставка узла.
4h0q7 l554ae c725xd b223cs 1q7f8 o597wx 4t3s1 g478rp y913pr 06c2mt 05q8yy #3 - Удаление узла.
4h0q7 l554ae c725xd b223cs 1q7f8 o597wx 4t3s1 g478rp y913pr 06c2mt
Задайте номер автобуса. Индивидуальное задание #1
tw745l
4h0q7 l554ae c725xd b223cs 1q7f8 o597wx 4t3s1 g478rp y913pr 06c2mt tw745l
Удаление всех узлов. Индивидуальное задание #2
Done!/
Result = 0. Индивидуальное задание #3
```

Рисунок 13 — Тестирование №2

Тестирование показало, что все функции работают успешно.

ВЫВОД

В ходе выполнения работы были получены знания и практические навыки управления динамическим двусвязным списком. Были реализованы все

необходимые методы для работы с двунаправленным списком, а также функции согласно индивидуальному варианту.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Структуры данных и проектирование программ : Пер. с англ. / Р. Круз. — М.: БИНОМ. Лаборатория знаний, 2017. — 766 с.

2. Полный справочник по C++ : Пер. с англ. / Г. Шилдт. — М.: ООО "И.Д.Вильямс", 2016. — 796 с.: ил. — Предм. указ.: с. 787-796