МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра АПУ

ОТЧЕТ

по лабораторной работе 2

по дисциплине «Программирование»

Тема: Поиск образца в тексте: очередь.

Студент гр. 2392 _____ Малимон Г.Д

Преподаватель _____ Власенко С.В

Санкт-Петербург

2023

Задание.

Требуется разработать программу обеспечивающую:

построение линейного односвязного линейного списка типа «очередь». (информационные части узлов должны иметь целочисленный тип и заполняться путем последовательного ввода значений с клавиатуры; признак окончания ввода – введенный 0);

обработку созданного списка с выполнением следующих функций:

а) вставку узлов с информационной частью = 1 во все позиции списка, предшествующие узлам с отрицательной информационной частью

б) удаление всех узлов списка, содержащих в информационной части отрицательные числа;

в) расчет количества вхождений заданного с клавиатуры числа в информационные поля узлов;

г) рекурсивное удаление всех узлов списка.

Выполнение

1. Структура Node для хранения чисел

```
typedef struct node {
    int num = -1;
    node* prev = NULL;
}node;
```

рис.1 - структура node

2. Функция заполнения очереди числами вводом с клавиатуры, работает через 2 указателя: в первом текущий элемент списка, во втором следующий элемент, это сделано, чтобы при добавлении нового числа в лист его можно было связать с предыдущим.

```
void lb_2_queue::set() {
    int count;
    int num = -1;
    cout << "write volume of queue: ";
    cin >> count;
    cout << "write queue: ";
    cin >> num;
    tile->set_num(num);
    count -= 1;
    while (count > 0) {
        cin >> num;
        add(num);
        count -= 1;
    }
```

Рис.2 - ввод данных

3. Функция вставки единиц после отрицательных, просто создаёт элемент связанный с отрицательным

```
void lb_2_queue::one_behind_minus() {
    line_node* cur = head;
    line_node* nw;
    while (cur->nxt()) {
        if (cur->nxt()->nm() < 0) {
            nw = new line_node(1, cur->nxt());
            cur->set_next(nw);
            cur = nw->nxt();
        }
        else cur = cur->nxt();
    }
    if (head->nm() < 0) {
        nw = new line_node(1, head);
        head = nw;
    }
```

Рис.3 - функция вставки единиц

4. Удаление отрицательных, идёт по списку и если встречает отрицательное связывает предыдущее со следующим и удаляет текущий

```
void lb_2_queue::delete_minus() {
    line_node* cur = head;
    line_node* next = cur->nxt();
    while (next) {
        if (next->nm() < 0) {
            cur->set_next(next->nxt());
            delete next;
            next = cur->nxt();
        }
        cur = next;
        next = cur->nxt();
    }
}
```

Рис.4 - удаление

5. Поиск с подсчётом просто идёт по списку и считает

```
int lb_2_queue::find(int num)
{
    line_node* cur = head;
    int n = 0;
    while (cur) {
        if (cur->nm() == num) n += 1;
        cur = cur->nxt();
    }
    return n;
}
```

Рис.5 - подсчёт

6. Вывод

```
void lb_2_queue::print() {
    line_node* cur = head;
    while (cur) {
        cout << cur->nm() << " ";
        cur = cur->nxt();
    }
    cout << endl;
}
```

Рис.6 - вывод.

7. Рекурсивная очистка каждый раз удаляет head

```
void lb_2_queue::rec_del() {
    if (head->nxt()) {
        line_node* temp = head;
        head = head->nxt();
        delete temp;
        rec_del();
    }
    else {
        delete head;
        cout << endl << "queue cleared";
    }
}
```

Рис.7 - рекурсивная очистка

Выводы.

Изучены основы работы со структурами, указателями, односвязными линейными списками и структурой данных под названием очередь.