Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления
Кафедра — интеллектуальных информационных технологий

К защите допустить:

Заведующий кафедрой ИИТ
Д.В. Шункевич

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к лабораторной работе
на тему:

Динамическая структура ОЧЕРЕДЬ

Студент гр. 321701 А. С. Астахов
Руководитель С.И. Матюшкин

Минск 2024

Цель: изучить алгоритмы работы с динамическими структурами в виде очереди.
Задача: составить и отладить программу.

Индивидуальное задание

Написать программу по созданию, добавлению (в начало, в конец), просмотру (с начала, с конца) и решению приведенной в подразделе 3.3 задачи для двунаправленных линейных списков.

```
#include <iostream>
using namespace std;

struct Quene {
    int info;

    Quene *next, *before;
} *start, *theEnd, *positiveStart, *positiveEnd, *negativeStart,
*negativeEnd;

void createQuene(Quene **first, Quene **last, int in) {
    Quene *tStart = new Quene;
    Quene *tEnd = new Quene;

    tStart -> info = in;
    tStart -> before = NULL;
    tStart -> next = tEnd;

    tEnd -> before = tStart;
    tEnd ->info = in;
    tEnd -> next = NULL;

    (*first) = tStart;
    (*last) = tEnd;

}

void deleteAll(Quene* p) {
    Quene *t;
    while (p != nullptr)
    {
        t = p;
        p = p -> next;
        delete t;
    }
}

void viewWholeQueneFromFirstToLast(Quene* p) {
```

```
    Quene *t = p;

    while (t != NULL) {
        cout << t -> info << endl;
        t = t -> next;
    }
}

void viewWholeQueneFromLastToFirst(Quene *End) {
    Quene *t = End;

    while (t -> before != NULL) {
        cout << t -> info << endl;
        t = t -> before;
    }
}

void deleteValue(Quene **start) {
    Quene *t;

    t = *start;
    *start = (*start) -> next;
    delete t;

}

void addInBegin(Quene **start, int in) {
    Quene *t = new Quene;

    t -> info = in;
    t -> next = (*start);
    t -> before = NULL;

    (*start) -> before = t;

    *start = (*start) -> before;
}

void addInFinish(Quene **theEnd, int in) {
    Quene *t = new Quene;

    t -> info = in;
    t -> next = NULL;

    (*theEnd) -> next = t;
    t -> before = (*theEnd);

    *theEnd = (*theEnd) -> next;
}

Quene *negativeValues(Quene *start, Quene *negativeStart, Quene
**negativeEnd) {
    Quene *t = start;


    while (t != NULL) {
```

```
        if (t -> info < 0) {
            addInFinish(negativeEnd, t -> info);
        }
            t = t -> next;

    }

    deleteValue(&negativeStart);
    deleteValue(&negativeStart);

    return negativeStart;
}

Quene *positiveValues(Quene *start, Quene *positiveStart, Quene
**positiveEnd) {

    Quene *t = start;


    while (t != NULL) {
        if (t -> info > 0) {
            addInFinish(positiveEnd, t -> info);
        }
        t = t -> next;

    }

    deleteValue(&positiveStart);
    deleteValue(&positiveStart);

    return positiveStart;
}

int main(int argc, const char * argv[]) {
    int value, choise, localChoise;
    createQuene(&start, &theEnd, 5);
    createQuene(&positiveStart, &positiveEnd, 5);
    createQuene(&negativeStart, &negativeEnd, 5);

    for (int i = 0; i < 5; i++) {
        addInBegin(&start, random() % 10 - 6);
    }

    while (true) {
        cout << "Выберете операцию над очередью:" << endl;
        cout << "1 - Удалить всю очередь" << endl;
        cout << "2 - Добавить элемент в начало" << endl;
        cout << "3 - Добавить элемент в конец" << endl;
        cout << "4 - Добавить все отрицательные элементы в очередь" <<
endl;
        cout << "5 - Добавление всех положительных элементов в очередь"
<< endl;
        cout << "6 - Вывести очередь с начала" << endl;
        cout << "7 - Вывести очередь с конца" << endl;
        cout << "8 - Удалить следующий в очереди элемент" << endl;
        cout << "9 - Выход" << endl;
```

```
        cin >> choise;
        switch (choise) {
            case 1:
                cout << "Удаление..." << endl;
                deleteAll(start);
                break;

            case 2:
                cout << "Введите элемент, который хотите добавить" <<
endl;

                cin >> value;

                addInBegin(&start, value);
                break;

            case 3:
                cout << "Введите элемент, который хотите добавить" <<
endl;

                cin >> value;

                addInFinish(&start, value);
                break;

            case 4:
                cout << "Добавление всех отрицательных элементов в
отдельную очередь" << endl;
                negativeStart = negativeValues(start, negativeStart,
&negativeEnd);
                break;

            case 5:
                cout << "Добавление всех положительных значений в
отдельную очередь" << endl;
                positiveStart = positiveValues(start, positiveStart,
&positiveEnd);
                break;

            case 6:
                cout << "Вывести первоначальную(1), положительную(2) или
отрицательную(3) очередь?" << endl;
                cin >> localChoise;
                switch (localChoise) {
                    case 1:
                        viewWholeQueneFromFirstToLast(start);
                        break;

                    case 2:
                        viewWholeQueneFromFirstToLast(positiveStart);
                        break;

                    case 3:
                        viewWholeQueneFromFirstToLast(negativeStart);
                        break;
```

```
                    default:
                        break;
                }

                break;

            case 7:
                cout << "Вывести первоначальную(1), положительную(2) или
отрицательную(3) очередь?" << endl;
                cin >> localChoise;
                switch (localChoise) {
                    case 1:
                        viewWholeQueneFromLastToFirst(theEnd);
                        break;

                    case 2:
                        viewWholeQueneFromLastToFirst(positiveEnd);
                        break;

                    case 3:
                        viewWholeQueneFromLastToFirst(negativeEnd);
                        break;

                    default:
                        break;
                }
                break;

            case 8:
                cout << "Удаление следующего элемента очереди..." <<
endl;
                deleteValue(&start);
                break;

            case 9:
                return 0;

            default:
                cout << "Введён неправильный номер функции" << endl;
                break;
        }
    }
}
```

Результат выполнения:

```
Выберете операцию над очередью:
1 – Удалить всю очередь
2 – Добавить элемент в начало
3 – Добавить элемент в конец
4 – Добавить все отрицательные элементы в очередь
5 – Добавление всех положительных элементов в очередь
6 – Вывести очередь с начала
7 – Вывести очередь с конца
8 – Удалить следующий в очереди элемент
9 – Выход
4

Добавление всех отрицательных элементов в отдельную очередь
Выберете операцию над очередью:
1 – Удалить всю очередь
2 – Добавить элемент в начало
3 – Добавить элемент в конец
4 – Добавить все отрицательные элементы в очередь
5 – Добавление всех положительных элементов в очередь
6 – Вывести очередь с начала
7 – Вывести очередь с конца
8 – Удалить следующий в очереди элемент
9 – Выход
5

Добавление всех положительных значений в отдельную очередь
Выберете операцию над очередью:
1 – Удалить всю очередь
2 – Добавить элемент в начало
3 – Добавить элемент в конец
4 – Добавить все отрицательные элементы в очередь
5 – Добавление всех положительных элементов в очередь
6 – Вывести очередь с начала
7 – Вывести очередь с конца
8 – Удалить следующий в очереди элемент
9 – Выход
6

Вывести первоначальную(1), положительную(2) или отрицательную(3) очередь?
2

1
5
5
Выберете операцию над очередью:
1 – Удалить всю очередь
2 – Добавить элемент в начало
3 – Добавить элемент в конец
4 – Добавить все отрицательные элементы в очередь
5 – Добавление всех положительных элементов в очередь
6 – Вывести очередь с начала
7 – Вывести очередь с конца
8 – Удалить следующий в очереди элемент
9 – Выход
```

Вывод: изучил алгоритмы работы с динамическими структурами в виде очереди, составил и отладил программу.