Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет инфомационных технологий и управления

Кафедра интеллектуальных информационных технологий

Основы алгоритмизации и программирования

Отчёт по лабораторной работе №4

**ДИНАМИЧЕСКАЯ СТРУКТУРА ОЧЕРЕДЬ**

Студент гр. 321701 В. В. Перминова

Преподаватель С. И. Матюшкин

Минск 2024

Цель работы: изучить возможности работы со списками, организованными в виде очереди

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.

8. Перенести из созданного списка в новый список все элементы, находящиеся между вершиной и элементом с минимальным значением.

Код программы:

#include <iostream>

using namespace std;

struct Stack {

int info;

Stack\* next, \* prev;

} \*top, \* down, \* t;

struct N\_Stack {

int n\_info;

N\_Stack\* n\_next;

} \*n\_top, \* k;

void CreateStack(Stack\*\* begin, Stack\*\* down, int in) { //функция создания первого элемента

t = new Stack;

t->info = in;

t->next = t->prev = NULL;

\*begin = \*down = t;

}

void InStack(int kod, Stack\*\* begin, Stack\*\* down, int in) {//функция добавления элементов

t = new Stack;

t->info = in;

if (kod == 0) {

t->prev = NULL;

t->next = \*begin;

(\*begin)->prev = t;

\*begin = t;

}

else {

t->next = NULL;

t->prev = \*down;

(\*down)->next = t;

\*down = t;

}

}

void View(int kod, Stack\* t) {//функция просмотра элементов списка

while (t != NULL) {

cout << t->info << endl;

if (kod == 0) t = t->next;

else t = t->prev;

}

}

void Del\_All(Stack\*\* p) {//функция удаления стека

while (\*p != NULL) {

Stack\* t = \*p;

\*p = (\*p)->next;

delete t;

}

}

N\_Stack\* New\_Stack(Stack\* p) {// функция создания стека, необходимого по заданию

Stack\* t = p->next;

Stack\* min = p;

N\_Stack\* k = new N\_Stack;

while (t) { //пока t существует ищем минимальный элемент

if (t->info < min->info) {

min = t;

}

t = t->next;

}

if ((p == min) || (p->next == min)) { //проверяем есть ли элементы между вершиной и минимальным элементом

cout << "Элементов между вершиной и минимальным элементом - нет" << endl;

}

else { //если такие элементы есть, то выводим их на экран

Stack\* i = p->next;

while (i->info != min->info) {

k->n\_info = i->info;

cout << " " << k->n\_info;

i = i->next;

k->n\_next = new N\_Stack;

}

cout << endl;

}

return k;

}

void main()

{

setlocale(LC\_ALL, "Russian");

int in, n, kod, kod1;

while (true) {

cout << "\n\tСоздание - 1.\n\tДобавление - 2.\n\tПросмотр - 3.\n\tУдаление - 4.\n\tВыполнение задания - 5.\n\tExit - 0. : ";//выводим на экран меню выбора функции

cin >> kod;

switch (kod) {

case 1:

if (top != NULL) {

cout << "Очистите память!" << endl;

break;

}

cout << "Значение первого элемента = ";

cin >> in;

CreateStack(&top, &down, in);

cout << "Создан первый элемент очереди = " << top->info << endl;

break;

case 2:

cout << "Добавить в начало - 0, Добавить в конец - 1 : ";

cin >> kod1;

cout << "Количество добавляемых элементов: ";

cin >> n;

for (int i = 1; i <= n; i++) {

cout << "Элемент = ";

cin >> in;

InStack(kod1, &top, &down, in);

}

if (kod1 == 0)

t = top;

else

t = down;

if (kod1 == 0)

cout << "В начало добавлено " << n << " элемента";

else

cout << "В конец добавлено " << n << " элемента";

break;

case 3:

if (!top) {//проверяем, есть ли элементы в стеке

cout << "Очередь пуста!" << endl;

break;

}

cout << "Просмотр с начала - 0, Просмотр с конца - 1 : ";

cin >> kod1;

if (kod1 == 0) {

t = top;

cout << "--- Начало очереди ---" << endl;

}

else {

t = down;

cout << "--- Конец очереди ---" << endl;

}

View(kod1, t);

break;

case 4:

Del\_All(&top);

cout << "Память освобождена!" << endl;

break;

case 5:

New\_Stack(top);

cout << "Данные элементы перенесены" << endl;

break;

case 0:

if (top != NULL)

Del\_All(&top);

while (k != NULL) {

N\_Stack\* d = k;

k = k->n\_next;

delete d;

}

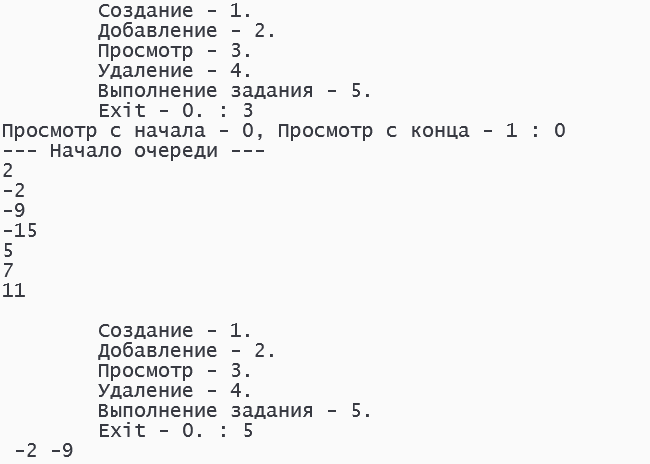
return;

}

}

}

Результат:



Вывод: в ходе данной лабораторной работы мы изучили алгоритмы работы с динамическими структурами данных в виде очереди.