Министерство образование Республики Беларусь

Учреждение образования «Полоцкий государственный университет»

Информационно-технологический факультет

Кафедра технологий программирования

Лабораторная работа №8

По дисциплина «Основы алгоритмизации и программирования»

Выполнил: Крутилин С.Д.

Группа 19-ИТ-1.

Проверил: Адамовский Е.

Новополоцк

2019 г.

**Динамические структуры данных**

**Вариант 2**

**Цель работы**: изучить методы организации списочных структур в динамической памяти. Реализовать алгоритмы помещения и изъятия элементов из стека, дека или очереди.

**Задание 1.** Функции для создания новой ДСД и ее удаления.

**Описание:** выделение динамической памяти и присваивание указателям на NULL.

Queue\* Creation()

{

Queue\* q = (Queue\*)malloc(sizeof(Queue));

q->head = NULL;

q->tail = NULL;

return q;

}

**Описание:** Вытаскиваем из очереди элементы до тех пор, пока head не будет указывать на NULL(параллельно выводим вытаскиваемые элементы для удобства); освобождаем память.

void DeleteQ(Queue\* q) {

int i;

while (popHead(q , &i) == 0) {

cout << i << " ";

}

cout << endl;

free(q);

}

**Задание 2.** Функции добавления элементов.

**Описание:** Создаем временную переменную; проверяем на выделение памяти; присваиваем данные; затем вставляем в очередь.

int PushTail(Queue\* q, int v) {

Node\* n;

if ((n = (Node\*)malloc(sizeof(Node))) == NULL)

return -1;

n->value = v;

n->next = NULL;

if (q->head == NULL) {

q->head = n;

}

else {

q->tail->next = n;

}

q->tail = n;

**Задание 3.** Функции удаления элементов.

**Описание:** если наш указатель на “голову” очереди указывает на NULL, то вытаскивать уже нечего и мы возвращаем -1, иначе возвращаем 1 (возвращаемое значение храниться в vptr.

int popHead(Queue\* q, int\* vptr) {

if (q->head == NULL) {

return -1;

}

else {

Node\* n = q->head->next;

\*vptr = q->head->value;

free(q->head);

q->head = n;

}

return 0;

}

**Задание 4.** Функции печати всех элементов ДСД.

**Описание:** сначала вытаскиваем элементы и выводим их, складывая во временную очередь; затем возвращаем обратно в нашу искомую очередь.

void printAll(Queue\* q) {

Queue\* temp = Creation();

int num;

while (popHead(q, &num) == 0) {

cout << num << " ";

PushTail(temp, num);

}

cout << endl;

while (popHead(temp, &num) == 0) {

PushTail(q, num);

}

free(temp);

}

**Задание 5.** Функция очистки ДСД

**Описание:** Вытаскивает все элементы и освобождает память.

void clear(Queue\* q) {

int num;

while (popHead(q, &num) == 0) {

}

}

#include <iostream>

#include <cstdlib>

#include <clocale>

using namespace std;

struct Node {

int value;

Node\* next;

};

struct Queue {

Node\* head;

Node\* tail;

};

int PushTail(Queue\* q, int v) {

Node\* n;

if ((n = (Node\*)malloc(sizeof(Node))) == NULL)

return -1;

n->value = v;

n->next = NULL;

if (q->head == NULL) {

q->head = n;

}

else {

q->tail->next = n;

}

q->tail = n;

return 0;

}

int popHead(Queue\* q, int\* vptr) {

if (q->head == NULL) {

return -1;

}

else {

Node\* n = q->head->next;

\*vptr = q->head->value;

free(q->head);

q->head = n;

}

return 0;

}

Queue\* Creation()

{

Queue\* q = (Queue\*)malloc(sizeof(Queue));

q->head = NULL;

q->tail = NULL;

return q;

}

void DeleteQ(Queue\* q) {

int num;

while (popHead(q , &num) == 0) {

cout << num << " ";

}

cout << endl;

free(q);

}

void printAll(Queue\* q) {

Queue\* temp = Creation();

int num;

while (popHead(q, &num) == 0) {

cout << num << " ";

PushTail(temp, num);

}

cout << endl;

while (popHead(temp, &num) == 0) {

PushTail(q, num);

}

free(temp);

}

void clear(Queue\* q) {

int num;

while (popHead(q, &num) == 0) {

}

}

int menu(void) {

setlocale(LC\_ALL, "Russian");

cout << "1 - добавить в конец" << endl;

cout << "2 - удалить из головы" << endl;

cout << "3 - печать всех элементов" << endl;

cout << "4 - очистка ДСД" << endl;

cout << "0 - выход" << endl << "> ";

int ret;

cin >> ret;

if (ret < 0 || ret > 4) ret = -1;

return ret;

}

int main() {

Queue\* q = Creation();

int ret;

while ((ret = menu()) > 0) {

switch (ret) {

case 1: {

int val;

cout << endl << "Value: ";

cin >> val;

if (PushTail(q, val))

cout << "Memory error!" << endl;

break;

}

case 2: {

int val;

if (popHead(q, &val))

cout << endl << "Empty queue." << endl;

else

cout << endl << "Returned: " << val << endl;

break;

}

case 3: {

printAll(q);

break;

}

case 4: {

clear(q);

break;

}

case 0: {

exit(1);

}

default: {

cout << "Wrong choice!" << endl;

}

}

}

system("pause");

return 0;

}