Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра інформаційних систем та мереж

ЗВІТ

про виконання лабораторної роботи № 4

з дисципліни Алгоритми та структури даних

Варіант 18

Виконав студент групи СА-32

Маленчак Валентин Богданович

Прийняв викладач Щербак О.Д.

*Львів 2017*

**Тема:** стек і черга. Хеш таблиця.

**Мета роботи:** набуття навичок моделювання зв’язаних динамічних структур даних та роботи з ними.

**Завдання на роботу:** розробити додатки, які забезпечують запити на запис або читання даних з черги, стека або дека. Для організації вказаних структур використовувати масиви або списки. Перевірити працездатність розроблених додатків. Послідовність виконання операцій запису або читання вибираються випадково. Порівняти результати роботи, зробити висновки.

**Індивідуальне завдання:** розробити програму для роботи з Деком. Дек організований на масиві з циклічним заповненням і з використанням двонаправленого списку. Операції виконуються з обох кінців Дека.

# Хід роботи

Текст програми на мові С++:

// Файл Lab04.cpp

#include <iostream>

#include "stdafx.h"

#include "MyDeque.h"

#include "UInterface.h"

using namespace std;

int sizeDeque(UInterface myInterface)

{

int size;

myInterface.getID(11);

myInterface.setSize(size);

return size;

}

int main()

{

UInterface myInterface;

MyDeque <int> myDeque(sizeDeque(myInterface));

char ch;

myInterface.getID(1);

myInterface.inputCharacter(ch);

do

{

switch (ch)

{

case '1':

myDeque.inputElements();

break;

case '2':

myDeque.outputElements();

break;

case '3':

myDeque.sizeD();

break;

case '4':

myDeque.clear();

break;

case '5':

myDeque.popBackElement();

break;

case '6':

myDeque.popFrontElement();

break;

case '7':

myDeque.pushBackElement();

break;

case '8':

myDeque.pushFrontElement();

break;

case '9':

myDeque.empty();

break;

default:

break;

}

myInterface.getID(1);

myInterface.inputCharacter(ch);

} while (ch > 48 && ch < 58);

system("pause");

}

// Файл MyDeque.h

#pragma once

#include <iostream>

#include <list>

#include <string>

#include "UInterface.h"

using namespace std;

template <class type> class MyDeque

{

private:

list <type> myList;

int size;

UInterface myInterface;

public:

MyDeque(int n1)

{

size = n1;

myList.resize(size);

}

void inputElements()

{

myInterface.getID(2);

for (list <type> ::iterator myIterator = myList.begin(); myIterator != myList.end(); myIterator++)

myInterface.inputDeque(myIterator);

}

void outputElements()

{

myInterface.getID(3);

for (list <type> ::iterator myIterator = myList.begin(); myIterator != myList.end(); myIterator++)

myInterface.outputDeque(myIterator);

}

type frontElement()

{

myInterface.getID(4);

return myList.front();

}

type backElement()

{

myInterface.getID(5);

return myList.back();

}

void popBackElement()

{

myList.pop\_back();

size--;

}

void popFrontElement()

{

myList.pop\_front();

size--;

}

void pushBackElement()

{

myInterface.getID(10);

myList.push\_back(myInterface.inputNewDequeElement(myList));

size++;

}

void pushFrontElement()

{

myInterface.getID(10);

myList.push\_front(myInterface.inputNewDequeElement(myList));

size++;

}

bool empty()

{

if (myList.empty())

{

myInterface.getID(6);

return true;

}

myInterface.getID(7);

return false;

}

void clear()

{

myInterface.getID(8);

myList.clear();

size = 0;

}

int sizeD()

{

myInterface.getID(9);

myInterface.getSize(size);

return size;

}

};

// Файл UInterface.h

#pragma once

#include <iostream>

using namespace std;

class UInterface

{

public:

void getID(int id)

{

switch (id)

{

case 1:

cout << "\nEnter '1' to input deque,\n'2' to output deque,";

cout << "\n'3' to get size of deque,\n'4' to clear deque,";

cout << "\n'5' to pop back,\n'6' to pop front,";

cout << "\n'7' to push back,\n'8' to push front,";

cout << "\n'9' to test whether deque is empty\n\n";

break;

case 2:

cout << "\nInput deque elements:\n";

break;

case 3:

cout << "\nDeque elements:\n";

break;

case 4:

cout << "\nFront element:\n";

break;

case 5:

cout << "\nBack element:\n";

break;

case 6:

cout << "\nDeque is empty!\n";

break;

case 7:

cout << "\nDeque isn't empty!\n";

break;

case 8:

cout << "\nDeque deleted!\n";

break;

case 9:

cout << "\nDeque's size = ";

break;

case 10:

cout << "\nEnter new deque element:\n";

break;

case 11:

cout << "\nInput the count of elements:\n";

break;

default:

break;

}

}

static void inputCharacter(char& ch)

{

cin >> ch;

cin.ignore();

}

template <class type>

void inputDeque(const type& myIterator)

{

cin >> \*myIterator;

}

template <class type>

void outputDeque(const type myIterator)

{

cout << \*myIterator << " ";

}

void setSize(int &size)

{

cin >> size;

}

void getSize(int size)

{

cout << size << endl;

}

template <class type>

type inputNewDequeElement(list <type> myList)

{

type newElement;

cin >> newElement;

return newElement;

}

};

Результати комп'ютерної реалізації:

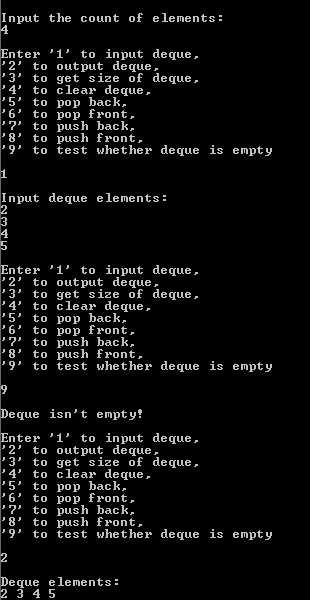


Рис.1. Приклад роботи програми.

**Висновок:** під час виконання лабораторної роботи я вдосконалив навички роботи з динамічними структурами даних. Програма створена для роботи з деком, який організований на двонапрямленому списку. Можливим варіантом вдосконалення програми є створення додаткових функцій для зручнішої роботи.