Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра інформаційних систем та мереж

ЗВІТ

про виконання лабораторних робіт №4

**“ Стек і черга. Хеш таблиця.”**

з дисципліни "Алгоритми та структури даних ”

В-15

Виконала студентка групи СА-32

Катеринчук Тетяна Андріївна

Прийняв доцент

Щербак С.С.

*2017*

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

**Завдання:**

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 пріоритетною чергою. Встановлення запитів в чергу виконується по пріоритету, зняття - зі старших адрес (кінець черги). Черга організована на масиві або на списку. Пріоритет: мах значення числового параметра, при збігу параметрів - FIFO

**Текст програми**

#include <iostream>

#include <stdexcept>

#include <cstdlib>

#include <stdio.h>

#include <string>

using namespace std;

template <typename TypeQueue>

class PriorityQueue

{

protected:

struct Node

{

TypeQueue data;

int priority;

Node\* next;

}\*list;

public:

PriorityQueue() : list(NULL) { ; }

PriorityQueue\* Push(TypeQueue data, int priority = 0)

{

Node\* node = new Node;

node->data = data;

node->priority = priority;

if ((list == NULL) || (list->priority < node->priority))

{

node->next = list;

list = node;

}

else

{

Node\* cur;

for (cur = list; cur->next && (cur->next->priority >= node->priority); cur = cur->next) { ; }

node->next = cur->next;

cur->next = node;

}

return this;

}

TypeQueue Pop()

{

if (IsEmpty())

{

cout << "Черга пуста" << endl;

return 0;

}

else {

TypeQueue data = list->data;

Node\* node = list;

list = list->next;

delete node;

return data;

}

}

TypeQueue Front()

{

if (IsEmpty())

{

cout << "Черга пуста" << endl;

return 0;

}

else {

TypeQueue data = list->data;

return data;

}

}

bool IsEmpty() const

{

return (list == NULL);

}

friend ostream& operator << (ostream& os, const PriorityQueue<TypeQueue>& queue)

{

typename PriorityQueue<TypeQueue>::Node\* node;

for (node = queue.list; node; node = node->next)

{

if (node != queue.list)

{

os << ",\t";

}

os << node->data << '[' << node->priority << ']';

}

return os;

}

};

void main()

{

setlocale(LC\_CTYPE, "ukr");

PriorityQueue<int> list;

int menu = -1;

while (menu != 0) {

cout << "1 - додати елемент\n2 - видалити елемент\n3 - переглянути всi елементи черги\n4 - вивести перший елемент черги\n0 - вихiд\n";

cin >> menu;

switch (menu) {

case 1:

int priority;

int elementQueue;

cout << "Введiть елемент черги: ";

cin >> elementQueue;

cout << "Введiть прiорiтет черги: ";

cin >> priority;

list.Push(elementQueue, priority);

break;

case 2:

list.Pop();

break;

case 3:

cout << "Елементи черги:\n" << list << endl;

break;

case 4:

cout << "Перший елемент черги:\n" << list.Front() << endl;

break;

case 0:

exit(0);

default:

cout << "Помилка" << endl;

break;

}

}

}

**Результат програми**

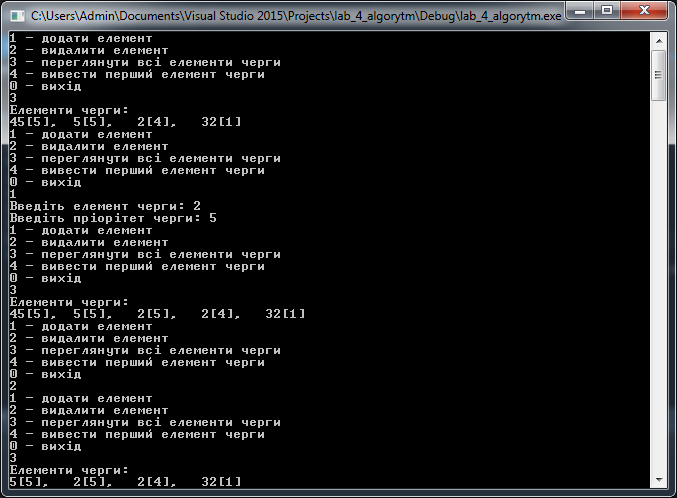


Рис.1 Реалізація додавання, видалення елементів з черги

**Висновок:** При виконання даної лабараторної роботи я набула навичок моделювання зв’язаних динамічних структур даних та роботи з ними. Розробила програму, яка реалізовує пріоритет чергу, у якій встановлення запитів в чергу виконується по пріоритету, зняття - зі старших адрес.