Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп’ютерно-інформаційних систем та програмної інженерії

Кафедра програмної інженерії

**З В І Т**

до виконання лабораторної роботи №3

з дисципліни «Алгоритми та структури даних»

на тему «Абстрактні типи даних “стек“ та “черга”»

Виконав:

студент групи СП-11

Бондар Віталій

Тернопіль 2022

**Мета роботи:** набути практичних навичок по реалізації та роботі зі АТД “стек” та “черга”

**Теоретичні відомості**

**Поняття “Стеку”**

**Стек** – це спеціальний тип списку, в якому всі вставки і видалення елементів виконуються тільки на одному з його кінців, який називається вершиною стеку (top).

Стеки реалізують принцип “останнім прийшов - перший пішов” LIFO (last-in-first-out).

Приклади стеків – колода карт, книжки складені одна на одну і т.д..

Абстрактний тип даних STACK як правило використовує наступні п’ять операторів:

MAKENULL(S) – робить стек порожнім.

TOP (S) – повертає елемент, що знаходиться на вершині стеку S.

Зазвичай вершина стеку ідентифікується з позицією 1, тоді TOP(S) можна записати в термінах загальних операторів списку як RETRIEVE (FIRST(S),S)

POP(S) – видаляє елемент з вершини стеку (виштовхує із стеку).

Використовуючи оператори списку, це можна представити як DELETE(FIRST(S),S)

PUSH(x, S) – встановлює елемент *x* на вершину стеку S (заштовхує елемент в стек). Елемент, що до цього знаходився на вершині, стає наступним елементом після вершини.

Для списку це має вигляд INSERT(x, FIRST(S),S)

EMPTY(S) – повертає значення true, якщо стек порожній, і значення false – в противному випадку.

**Поняття “Черги”**

Черга (queue) – спеціальний тип списку, де елементи додаються з одного кінця, який називається хвостом (tail або rear), а вилучаються з іншого, який називається головою (head або front).

Черги реалізують принцип “перший прийшов – перший пішов” FIFO (first-in-first-out). Оператори, що виконуються над чергами подібні до операторів стеку за винятком того, що вставка нових елементів робиться в кінець списку, а не в його початок.

Основні оператори роботи із чергою:

MAKENULL(Q) – очищає чергу Q, роблячи її порожньою.

FRONT(Q) – функція повертає перший елемент черги Q.

Цю функцію можна реалізувати при допомозі операторів списку як  
RETRIEVE (FIRST(Q),Q)

ENQUEUE(x,Q) – ставить елемент *x* в кінець черги Q.

З допомогою операторів списку – INSERT (x, END(Q),Q)

DEQUEUE (Q) – видаляє перший елемент із черги Q.

Також реалізується як DELETE(FIRST(Q),Q)

EMPTY(Q) – повертає значення true тоді і лише тоді коли чергою Q є порожньою.

**Завдання 1**

Реалізувати на мові C/С++ представлення АТД “стек” при допомозі масиву (операції MAKENULL, PUSH, POP, TOP, EMPTY, PRINT) та виконати наступний код:

. . .

            STACK S;

            char c;

            MAKENULL(S);

            c = getch();

            while (c != ‘=’) {

                  switch (c) {

                                   // витерти останній символ

                        case ‘#’: POP(S); break;

                                   // витерти усю стрічку

                        case ‘@’: MAKENULL(S); break;

                                   // продублювати останній символ

                        case ‘^’: PUSH(TOP(S)); break;

                                   // додати символ до стрічки

                        default :

                                  PUSH(c, S);

                  c = getch();

}

// вивести введену стрічку на екран

PRINT(S);

. . .

IDE: Microsoft Visual Studio 2019

Лістинг програми

//main.cpp

#include <iostream>

#include "Stack.h"

#include <conio.h>

int main()

{

STACK S;

char c;

MAKENULL(S);

c = \_getch();

while (c != '=') {

switch (c) {

case '#': POP(S); break;

case '@': MAKENULL(S); break;

case '^': PUSH(TOP(S), S); break;

default:

PUSH(c, S);

c = \_getch();

}

PRINT(S);

cout << "\n";

}

return 0;

}

//stackRealisation.h

#pragma once

#include <stdio.h>

#define maxlength 1000

typedef char elementtype;

using namespace std;

struct STACK

{

int top;

elementtype elements[maxlength];

};

void MAKENULL(STACK& S) {

S.top = maxlength;

}

bool EMPTY(STACK& S) {

if (S.top > maxlength)

{

return true;

}

else {

return false;

}

}

elementtype TOP(STACK& S) {

if (EMPTY(S)) {

throw "Stack is empty";

}

else{

return S.elements[S.top];

}

}

void POP(STACK& S) {

if (EMPTY(S)) {

throw "Stack is empty";

}

else

{

S.top = S.top + 1;

}

}

void PUSH(elementtype x, STACK& S) {

if (S.top == 0)

{

throw "Stack is full";

}

else {

S.top = S.top - 1;

S.elements[S.top] = x;

}

}

void PRINT(STACK S) {

for (int i = maxlength - 1; i >= S.top; i--)

{

cout << S.elements[i];

}

}

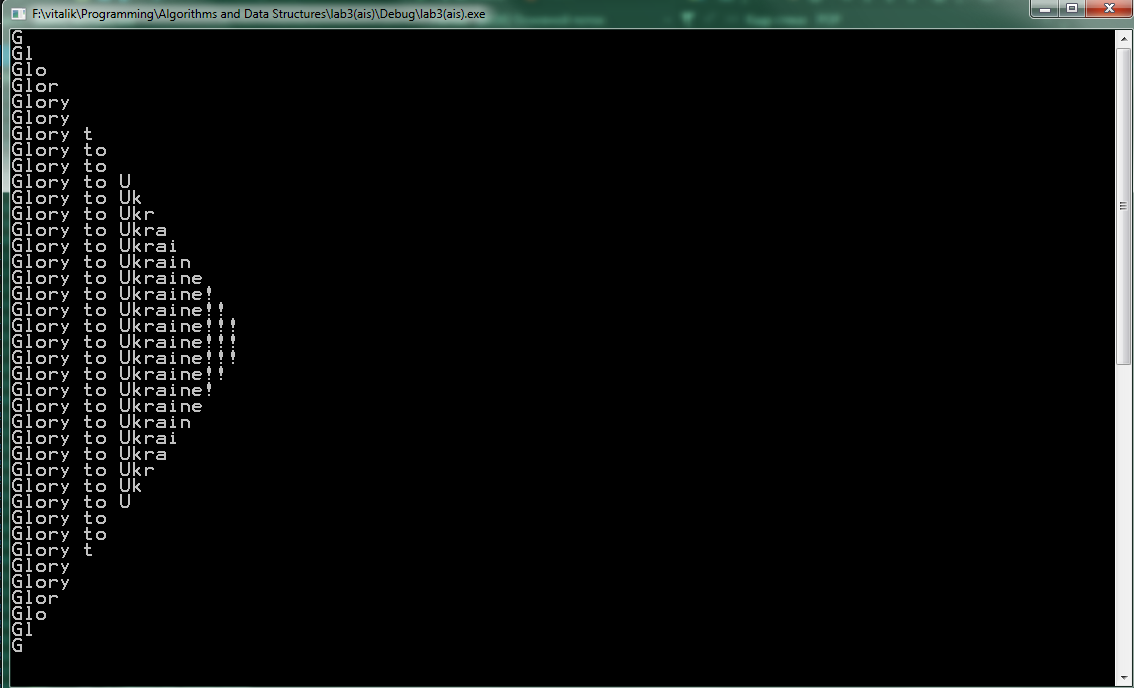


Рис.1 – результат виконання завдання 1

**Завдання 2**

Реалізувати на мові C/С++ представлення АТД “черга” при допомозі циклічного масиву (операції MAKENULL, FRONT, ENQUEUE, DEQUEUE, EMPTY) та виконати наступний код:

. . .

            QUEUE Q;

            char c;

            MAKENULL(Q);

            c = getch();

            while (c != ‘=’) {

                  ENQUEUE (c, Q);

}

DEQUEUE (Q);

DEQUEUE (Q);

ENQUEUE (‘<’, Q) ;

while (!EMPTY(Q)) {

      printf (“%c, ”, FRONT(Q));

      DEQUEUE (Q);

}

Printf (“\n”);

. . .

Лістинг програми

//main.cpp

#include <stdio.h>

#include <conio.h>

#include "queueRealisation.h"

int main()

{

QUEUE Q = QUEUE();

char c;

MAKENULL(Q);

c = \_getch();

while (c != '=') {

ENQUEUE(c, Q);

c = \_getch();

}

DEQUEUE(Q);

DEQUEUE(Q);

ENQUEUE('<', Q);

while (!EMPTY(Q)) {

printf("%c,", FRONT(Q));

DEQUEUE(Q);

}

printf("\n");

}

//queueRealisation.h

#pragma once

#define maxlength 1000

#include <iostream>

using namespace std;

typedef char elementtype;

struct QUEUE

{

int front;

int rear;

elementtype elements[maxlength];

};

int addone(int i) {

return i % maxlength + 1;

}

void MAKENULL(QUEUE& Q) {

Q.front = 1;

Q.rear = maxlength;

}

bool EMPTY(QUEUE& Q) {

if (addone(Q.rear) == Q.front) {

return true;

}

else {

return false;

}

}

elementtype FRONT(QUEUE& Q) {

if (EMPTY(Q)) {

cout<< "Queue is empty";

}

else {

return Q.elements[Q.front];

}

}

void ENQUEUE(elementtype x, QUEUE& Q) {

if (addone(addone(Q.rear)) == Q.front) {

cout << "Queue is full";

}

else {

Q.rear = addone(Q.rear);

Q.elements[Q.rear] = x;

}

}

void DEQUEUE(QUEUE& Q) {

if (EMPTY(Q)) {

cout << "Queue is empty";

}

else {

Q.front = addone(Q.front);

}

}

****

Рис.2 – результат виконання завдання 2(“Glory to Ukraine”)

**Висновок:** набув практичних навичок по реалізації та роботі із АТД “стек” та “черга”