**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ІМЕНІ ТАРАСА ШЕВЧЕНКА**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра інтелектуальних технологій**

Лабораторна робота №11  
*(вид роботи: лабораторна робота, індивідуальне завдання, курсова робота тощо)*

з дисципліни «Алгоритмізація та програмування»Тема роботи: «Динамічні структури даних:  
однонаправлені списки. Стек та черга.»  
**Варіант № 2**

Виконав(-ла) студент(-ка)  
групи АнД - 11  
Яковкін Микола Андрійович

Перевірив(-ла):  
ПІП викладача

Київ – 2022

**Завдання 1**

**З використанням структури Node реалізувати однонаправлений динамічний список-стек.**

**1.1 Тестування програми**

1. Для перевірки задані дані:

[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

Маємо отримати:

Стек парних чисел: [2, 4, 6, 8, 10]

Стек непарних чисел [1, 3, 5, 7, 9]

Стек парних чисел після видалення одного елемента та додавання іншого: [0, 4, 6, 8, 10]

Стек парних чисел після видалення одного елемента та додавання іншого: [-3, 3, 5, 7, 9]

Об’єднання двох стеків: [10, 9, 8, 7, 6, 5, 4, 3, 0, -3]

Переміщення з одного стека в інший: [-1 -2 -3 -4 -5 -6 -7 -8 -9 -10 -11]

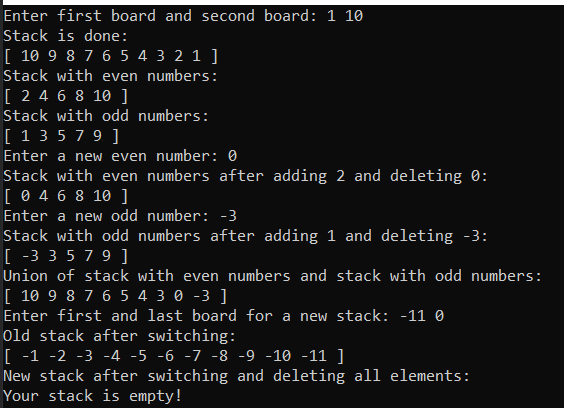


Рисунок 1.1 – Тестування програми.

2. Для перевірки задані дані:

[ 0 -1 -2 -3 -4 -5 -6 -7 -8 -9 ]

Маємо отримати:

Стек парних чисел: [ -8 -6 -4 -2 0 ]

Стек непарних чисел [ -9 -7 -5 -3 -1 ]

Стек парних чисел після видалення одного елемента та додавання іншого: [ -12 -6 -4 -2 0 ]

Стек парних чисел після видалення одного елемента та додавання іншого: [ -15 -7 -5 -3 -1 ]

Об’єднання двох стеків: [ 0 -1 -2 -3 -4 -5 -6 -7 -12 -15 ]

Переміщення з одного стека в інший: [ 10 9 8 7 6 5 4 3 2 1 ]

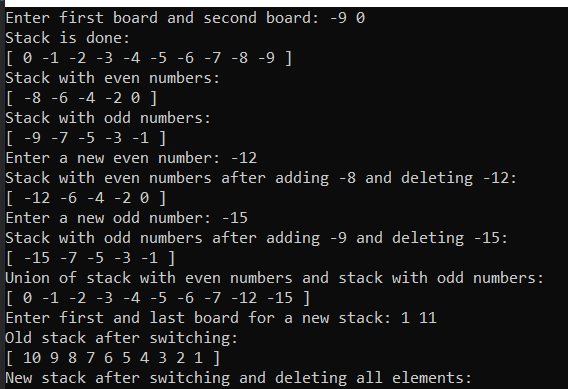


Рисунок 1.2 – Тестування програми.

3. Для перевірки задані дані:

[ 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 ]

Маємо отримати:

Стек парних чисел: [ 20 22 24 26 28 30 32 34 36 38 40 ]

Стек непарних чисел [ 21 23 25 27 29 31 33 35 37 39 ]

Стек парних чисел після видалення одного елемента та додавання іншого: [ 4 22 24 26 28 30 32 34 36 38 40 ]

Стек парних чисел після видалення одного елемента та додавання іншого: [ 5 23 25 27 29 31 33 35 37 39 ]

Об’єднання двох стеків:

[ 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 5 4 ]

Переміщення з одного стека в інший:

[ -21 -22 -23 -24 -25 -26 -27 -28 -29 -30 -31 -32 -33 -34 -35 -36 -37 -38 -39 -40 ]

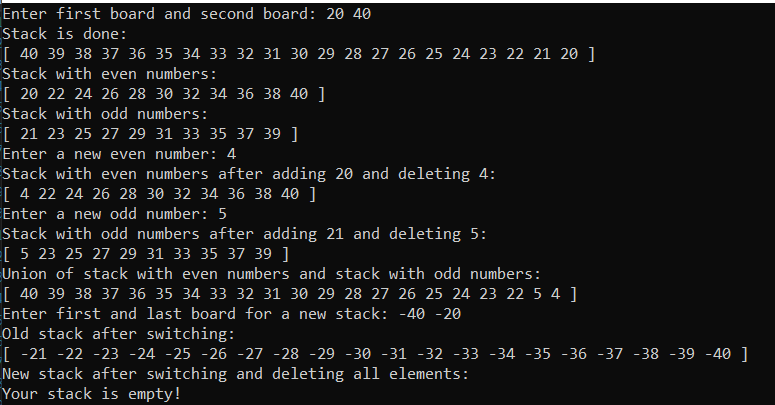


Рисунок 1.3 – Тестування програми.

**1.2 Текст програмною мовою С++.**

stack.h

#**pragma** **once**

#**include** <iostream>

**class** Stack

{

**public**:

Stack();

~Stack();

void push(const int& x);

void push(Stack& T1, Stack& T2);

bool empty() const;

void pop();

int top();

void view();

void clear();

**private**:

**struct** Node

{

**int** data;

**Node**\* pNext;

};

**Node**\* root;

};

stack.cpp

#**include** "stack.h"

**Stack**::Stack() : root(**nullptr**) {}

**Stack**::~Stack() { **clear**(); };

**void** **Stack**::push(**const** **int**& x)

{

**Node**\* newnode = **new** Node;

newnode->data = x;

newnode->pNext = root;

root = newnode;

}

**void** **Stack**::push(**Stack**& T1, **Stack**& T2)

{

**Stack** temp;

**while** (!T1.**empty**())

{

temp.**push**(T1.**top**());

T1.**pop**();

}

**while** (!T2.**empty**())

{

T1.**push**(T2.**top**());

T2.**pop**();

}

**while** (!temp.**empty**())

{

T2.**push**(temp.**top**());

temp.**pop**();

}

}

**bool** **Stack**::empty() **const**

{

**return** root == **nullptr**;

}

**void** Stack::pop()

{

**if** (**empty**())

std::cerr << "Error! Your stack is empty!";

**else**

{

**Node**\* delnode = root;

root = delnode->pNext;

**delete** delnode;

}

}

**int** **Stack**::top()

{

**if** (**empty**())

std::cerr << "Error! Your stack is empty." << std::endl;

**return** root->data;

}

**void** **Stack**::view()

{

**if** (**empty**())

std::cerr << "Your stack is empty!";

**else**

{

std::cout << "[ ";

**Node**\* view = root;

**while** (view->pNext != **nullptr**)

{

std::cout << view->data << " ";

view = view->pNext;

}

std::cout << view->data << " ";

std::cout << "]" << std::endl;

}

}

**void** Stack::clear()

{

**while** (!**empty**()) { **pop**(); }

}

main.cpp

#**include** <iostream>

#**include** "stack.h"

using namespace std;

**int** main()

{

**Stack** st;

**int** firstBoard, secondBoard;

**int** newNumber;

cout << "Enter first board and second board: ";

cin >> firstBoard >> secondBoard;

**for** (**int** i = firstBoard; i <= secondBoard; i++)

{

st.**push**(i);

}

cout << "Stack is done: " << endl;

st.**view**();

**Stack** evenSt;

**Stack** oddSt;

**while** (!st.**empty**())

{

**if** (!(st.**top**() % 2))

{

evenSt.**push**(st.**top**());

st.**pop**();

}

**else**

{

oddSt.**push**(st.top());

st.**pop**();

}

}

cout << "Stack with even numbers: " << endl;

evenSt.**view**();

cout << "Stack with odd numbers: " << endl;

oddSt.**view**();

cout << "Enter a new even number: ";

cin >> newNumber;

cout << "Stack with even numbers after adding " << evenSt.top() << " and deleting " <<

newNumber << ": " << endl;

evenSt.**pop**();

evenSt.**push**(newNumber);

evenSt.**view**();

cout << "Enter a new odd number: ";

cin >> newNumber;

cout << "Stack with odd numbers after adding " << oddSt.top() << " and deleting " <<

newNumber << ": " << endl;

oddSt.**pop**();

oddSt.**push**(newNumber);

oddSt.**view**();

while (!oddSt.**empty**() && !evenSt.**empty**())

{

if (oddSt.**top**() > evenSt.**top**())

{

st.**push**(evenSt.**top**());

evenSt.**pop**();

}

else

{

st.**push**(oddSt.**top**());

oddSt.**pop**();

}

}

**while** (!oddSt.**empty**())

{

st.**push**(oddSt.**top**());

oddSt.**pop**();

}

while (!evenSt.**empty**())

{

st.push(evenSt.**top**());

evenSt.**pop**();

}

cout << "Union of stack with even numbers and stack with odd numbers:" << endl;

st.**view**();

cout << "Enter first and last board for a new stack: ";

cin >> firstBoard >> secondBoard;

**Stack** test;

**for** (**int** i = firstBoard; i < secondBoard; i++) { test.**push**(i); }

cout << "Old stack after switching: " << endl;

st.**push**(test, st);

st.**view**();

cout << "New stack after switching and deleting all elements: " << endl;

test.**clear**();

test.**view**();

**return** 0;

}

**Завдання 2**

**З використанням структури Node реалізувати однонаправлений динамічний список-черга.**

**2.1 Тестування програми**

1. Для перевірки задані дані:

[ 1 2 3 4 5 6 7 8 9 10 ]

Маємо отримати:

Черга парних чисел: [2, 4, 6, 8, 10]

Черга непарних чисел [1, 3, 5, 7, 9]

Черга парних чисел після видалення одного елемента та додавання іншого: [ 4 6 8 10 12 ]

Черга парних чисел після видалення одного елемента та додавання іншого: [ 3 5 7 9 11 ]

Об’єднання двох черг: [ 3 4 5 6 7 8 9 10 11 12 ]

Переміщення з одної черги в іншу: [ -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 ]

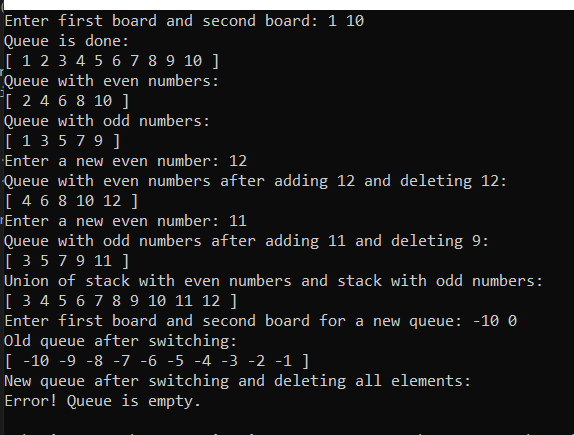


Рисунок 2.1 – Тестування програми.

2. Для перевірки задані дані:

[ -9 -8 -7 -6 -5 -4 -3 -2 -1 ]

Маємо отримати:

Черга парних чисел: [ -8 -6 -4 -2 ]

Черга непарних чисел [ -9 -7 -5 -3 -1 ]

Черга парних чисел після видалення одного елемента та додавання іншого: [ -6 -4 -2 14 ]

Черга парних чисел після видалення одного елемента та додавання іншого: [ -7 -5 -3 -1 15 ]

Об’єднання двох черг: [ -7 -6 -5 -4 -3 -2 -1 14 15 ]

Переміщення з одної черги в іншу:

[ 22 23 24 25 26 27 28 29 30 31 32 33 34 35 ]

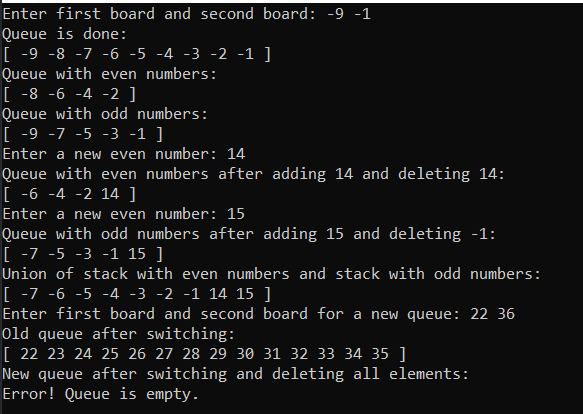


Рисунок 2.2 – Тестування програми.

3. Для перевірки задані дані:

[ 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 ]

Маємо отримати:

Черга парних чисел: [ 100 102 104 106 108 110 112 114 116 118 120 ]

Черга непарних чисел [ 101 103 105 107 109 111 113 115 117 119 ]

Черга парних чисел після видалення одного елемента та додавання іншого: [ 102 104 106 108 110 112 114 116 118 120 600 ]

Стек парних чисел після видалення одного елемента та додавання іншого: [ 103 105 107 109 111 113 115 117 119 599 ]

Об’єднання двох черг:

[ 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 599 600 ]

Переміщення з одної черги в іншу:

[ -120 -119 -118 -117 -116 -115 -114 -113 -112 -111 -110 -109 -108 -107 -106 -105 -104 -103 -102 -101 -100 ]

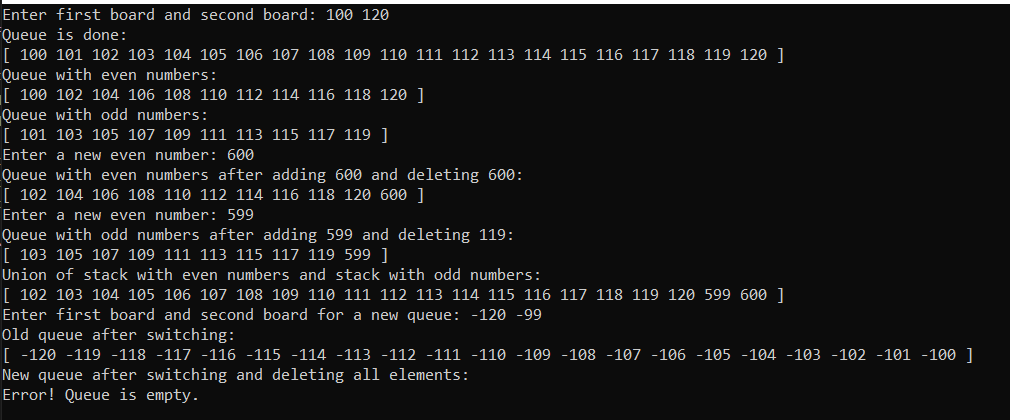


Рисунок 2.3 – Тестування програми.

**2.2 Текст програмною мовою С++.**

queue.h

#**pragma** **once**

#**include** <iostream>

#**include** <cassert>

**class** Queue

{

**public**:

Queue();

void enque(int);

void enque(Queue T2);

void deque();

int first();

int last();

bool empty();

void view();

void clear();

**private**:

**struct** Node

{

**int** data;

**Node**\* pNext;

**Node**(**int** v = 0, **Node**\* n = nullptr)

{

data = v;

pNext = n;

}

};

**Node**\* head;

**Node**\* tail;

};

queue.cpp

#**include** "queue.h"

**Queue**::**Queue**() : head(NULL), tail(NULL) {}

**bool** **Queue**::empty()

{

**return** head == NULL;

}

**void** **Queue**::enque(**int** x)

{

**if** (head == NULL)

head = tail = **new** Node(x, NULL);

**else**

tail = tail->pNext = **new** Node(x, NULL);

}

**void** **Queue**::enque(Queue T2)

{

**Queue** temp;

**while** (!**empty**())

{

temp.**enque**(**first**());

**deque**();

}

**while** (!T2.**empty**())

{

**enque**(T2.**first**());

T2.**deque**();

}

**while** (!temp.**empty**())

{

T2.**enque**(temp.first());

temp.**deque**();

}

}

**void** **Queue**::deque()

{

**assert**(head != NULL);

**Node**\* tmp = head;

head = head->pNext;

**if** (head == NULL)

tail = NULL;

**delete** tmp;

}

**int** **Queue**::last()

{

**return** tail->data;

}

**int** **Queue**::first()

{

**return** head->data;

}

**void** **Queue**::view()

{

**if** (**empty**())

std::cerr << "Error! Queue is empty." << std::endl;

**else**

{

**Node**\* temp = head;

std::cout << "[ ";

**while** (temp->pNext != NULL)

{

std::cout << temp->data << " ";

temp = temp->pNext;

}

std::cout << tail->data << " ]" << std::endl;

}

}

**void** **Queue**::clear()

{

**Node**\* temp;

**while** (!**empty**())

{

**deque**();

}

}

main.cpp

#**include** <iostream>

#**include** "queue.h"

**using** **namespace** std;

**int** main()

{

**Queue** qu;

**int** firstBoard, secondBoard;

**int** newNumber;

cout << "Enter first board and second board: ";

cin >> firstBoard >> secondBoard;

**for** (**int** i = firstBoard; i <= secondBoard; i++)

{

qu.**enque**(i);

}

cout << "Queue is done: " << endl;

qu.**view**();

**Queue** evenQu;

**Queue** oddQu;

**while** (!qu.**empty**())

{

**if** (!(qu.**first**() % 2))

{

evenQu.enque(qu.**first**());

qu.**deque**();

}

**else**

{

oddQu.**enque**(qu.**first**());

qu.**deque**();

}

}

cout << "Queue with even numbers: " << endl;

evenQu.**view**();

cout << "Queue with odd numbers: " << endl;

oddQu.**view**();

cout << "Enter a new even number: ";

cin >> newNumber;

cout << "Queue with even numbers after adding " << newNumber << " and deleting " <<

newNumber << ": " << endl;

evenQu.**deque**();

evenQu.**enque**(newNumber);

evenQu.**view**();

cout << "Enter a new even number: ";

cin >> newNumber;

cout << "Queue with odd numbers after adding " << newNumber << " and deleting " <<

oddQu.last() << ": " << endl;

oddQu.**deque**();

oddQu.**enque**(newNumber);

oddQu.**view**();

**while** (!oddQu.**empty**() && !evenQu.**empty**())

{

**if** (oddQu.**first**() > evenQu.**first**())

{

qu.**enque**(evenQu.**first**());

evenQu.**deque**();

}

**else**

{

qu.**enque**(oddQu.**first**());

oddQu.**deque**();

}

}

**while** (!oddQu.**empty**())

{

qu.**enque**(oddQu.**first**());

oddQu.**deque**();

}

**while** (!evenQu.**empty**())

{

qu.**enque**(evenQu.**first**());

evenQu.**deque**();

}

cout << "Union of stack with even numbers and stack with odd numbers:" << endl;

qu.**view**();

**Queue** test;

cout << "Enter first board and second board for a new queue: ";

cin >> firstBoard >> secondBoard;

**for** (**int** i = firstBoard; i < secondBoard; i++) { test.**enque**(i); }

cout << "Old queue after switching: " << endl;

qu.**enque**(test);

qu.**view**();

cout << "New queue after switching and deleting all elements: " << endl;

test.**clear**();

test.**view**();

**return** 0;

}

**Завдання 3**

**Постфіксний запис**

**3.1 Тестування програми**

1. Для тестування задані дані

1 7 + 8 3 + \*

Маємо отримати:

(1 + 7) \* (8 + 3) = 88

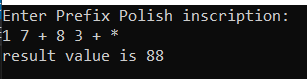


Рисунок 3.1 – Тестування програми.

2. Для тестування задані дані

1 7 8 \* + 3 +

Маємо отримати:

1 + 7 \* 8 + 3

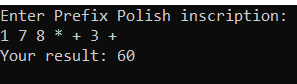


Рисунок 3.2 – Тестування програми.

3. Для тестування задані дані

1 7 + 8 \* 9 -

Маємо отримати:

((1 + 7) \* 8) - 9= 55

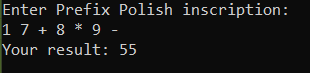


Рисунок 3.3 – Тестування програми.

**3.2 Текст програмною мовою С++.**

#**include** <iostream>

#**include** <string>

#**include** <sstream>

#**include** "stack.h"

**using namespace** std;

**int** main()

{

**Stack** numbers;

**string** example;

cout << "Enter Prefix Polish inscription: " << endl;

**getline**(**cin**, example);

**istringstream** ss(example);

**int** findInt;

**char** temp[10];

**char** sign;

**while** (!ss.**eof**())

{

**while** (ss >> findInt) { numbers.**push**(findInt); }

**if** (!ss.**eof**())

{

ss.**clear**();

ss.**unget**();

ss >> sign;

**if** (numbers.**getSize**() < 2)

{

cerr << "Error! Not enough values!" << endl;

**return** 0;

}

**int** right = numbers.**top**();

numbers.**pop**();

int **left** = numbers.**top**();

numbers.**pop**();

**switch** (sign)

{

**case** '+': numbers.push(left + right); **break**;

**case** '-': numbers.push(left - right); **break**;

**case** '\*': numbers.push(left \* right); **break**;

**case** '/': numbers.push(left / right); **break**;

**default**:

cerr << "Error! Unsupported sign." << endl;

**return** 0;

**break**;

}

}

**if** (ss.**eof**() && numbers.**getSize**() > 1)

{

**if** (numbers.**getSize**() != 2)

{

cerr << "Error! We can calculate your result!" << endl;

**return** 0;

}

ss.**clear**();

ss.**unget**();

ss >> sign;

**int** right = numbers.**top**();

numbers.**pop**();

**int** left = numbers.**top**();

numbers.**pop**();

**switch** (sign)

{

**case** '+': numbers.**push**(left + right); **break**;

**case** '-': numbers.**push**(left - right); **break**;

**case** '\*': numbers.**push**(left \* right); **break**;

**case** '/': numbers.**push**(left / right); **break**;

**default**:

cerr << "Error! Unsupported sign." << endl;

**return** 0;

**break**;

}

}

}

**if** (numbers.**getSize**() != 1)

{

cerr << "Error! You haven`t got a result" << endl;

**return** 0;

}

cout << "Your result: " << numbers.**top**() << endl;

**return** 0;

}