**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ІМЕНІ ТАРАСА ШЕВЧЕНКА**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра інтелектуальних технологій**

Лабораторна робота №12  
*(вид роботи: лабораторна робота, індивідуальне завдання, курсова робота тощо)*

з дисципліни «Алгоритмізація та програмування»Тема роботи: «Динамічні структури даних:  
двонаправлені списки.»  
**Варіант № 0**

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

Перевірив(-ла):  
ПІП викладача

Київ – 2022

**Завдання 1**

* 1. **Тестування програми**

1. Для тестування заданий масив:

Діапазон (1, 10), к-сть елементів 6

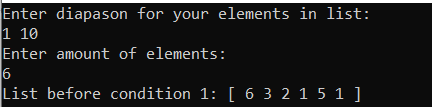


Рисунок 1.1 – Вхідні дані.

Маємо отримати:

[ 6 5 4 3 2 1 2 1 6 5 2 1 ] – після першої умови.

[ 4 2 1 2 1 2 1 ] – після другої умови.

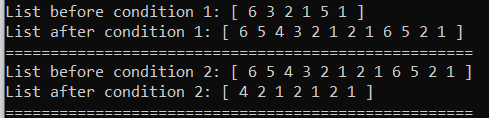


Рисунок 1.2 – Тестування програми.

1. Для тестування заданий масив:

Діапазон (6, 15), к-сть елементів 7

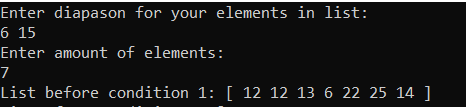


Рисунок 1.3 – Вхідні дані.

Маємо отримати:

[ 12 11 12 11 14 13 6 5 22 21 26 25 14 13 ]– після першої умови.

[ 11 11 14 13 22 26 14 13 ]– після другої умови.

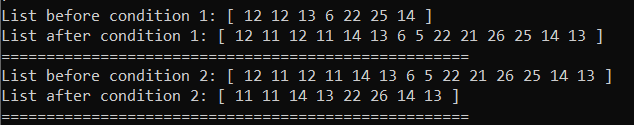


Рисунок 1.4 – Тестування програми.

1. Для тестування заданий масив:

Діапазон (-12, -6), к-сть елементів 8

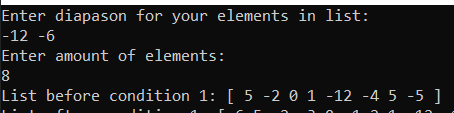


Рисунок 1.5 – Вхідні дані.

Маємо отримати:

[ 6 5 -2 -3 0 -1 2 1 -12 -13 -4 -5 6 5 -4 -5 ] – після першої умови.

[ -2 -1 2 1 -13 -4 -4 ] – після другої умови.

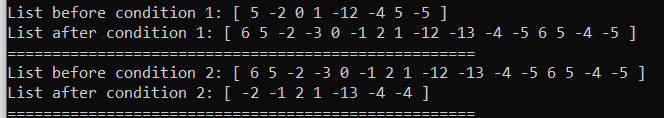


Рисунок 1.6 – Тестування програми.

* 1. **Текст програмною мовою С++.**

#**include** <iostream>

#**include** "biList.h"

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

**void** randomData(**biList**&, **int**, **int**, **int**);

**int** main()

{

**biList** list;

**int** start, stop, count;

cout << "Enter diapason for your elements in list: " << endl;

cin >> start >> stop;

cout << "Enter amount of elements: " << endl;

cin >> count;

**randomData**(list, start, stop, count);

cout << "List before condition 1: ";

list.**forward\_view()**;

**for** (**int** i = 0; i < list.**getSize**(); i++)

{

//cout << !(list.find(i)->data % 2) << " " << i << endl;

**if** (!(list.**find**(i)->data % 2))

{

list.**insert**\_**after**(i, list.**find**(i)->data - 1);

i++;

}

**else**

{

list.**insert\_before**(i, list.**find**(i)->data + 1);

i++;

}

}

cout << "List after condition 1: ";

list.forward\_view();

cout << "====================================================" << endl;

cout << "List before condition 2: ";

list.forward\_view();

list.push\_front(12);

**for** (**int** i = 0; i < list.**getSize**(); i++)

{

**if** (!(list.**find**(i)->data % 3) || !(list.**find**(i)->data % 5))

{

list.**remove**(i);

i--;

}

}

cout << "List after condition 2: ";

list.**forward\_view**();

**return** 0;

}

**void** randomData(**biList&** list, **int** start, **int** stop, **int** count)

{

**srand**(**time**(0));

**for** (; count; count--)

{

**int** element = start + **rand**() % (stop + start);

list.**push\_back**(element);

}

}

**Завдання 2**

* 1. **Тестування програми**

1. Для тестування заданий масив:

Діапазон (-6, 12), к-сть елементів 6

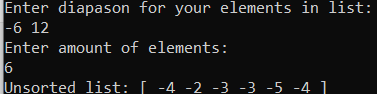


Рисунок 2.1 – Вхідні дані.

Маємо отримати: [ -2 -3 -3 -4 -4 -5 ]

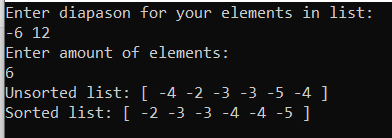


Рисунок 2.2 – Тестування програми.

1. Для тестування заданий масив:

Діапазон (1, 10), к-сть елементів 5

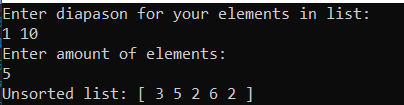


Рисунок 2.3 – Вхідні дані.

Маємо отримати: [ 6 5 3 2 2 ]

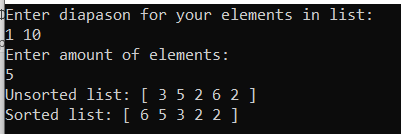


Рисунок 2.4 – Тестування програми.

1. Для тестування заданий масив:

Діапазон (2, 100), к-сть елементів 8

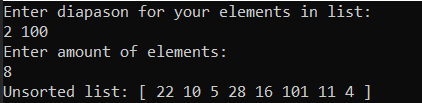


Рисунок 2.5 – Вхідні дані.

Маємо отримати: [ 101 22 16 28 11 10 5 4 ]

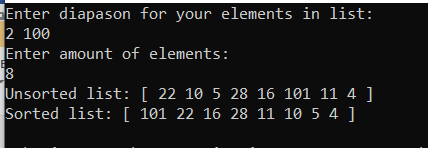


Рисунок 2.6 – Тестування програми.

* 1. **Текст програмною мовою С++.**

#**include** <iostream>

#**include** "biList.h"

**using namespace** std;

**void** randomData(**biList&,** **int**, **int**, **int**);

**int** main()

{

**biList** list;

**int** start, stop, count;

cout << "Enter diapason for your elements in list: " << endl;

cin >> start >> stop;

cout << "Enter amount of elements: " << endl;

cin >> count;

**randomData**(list, start, stop, count);

cout << "Unsorted list: ";

list**.forward\_view**();

cout << "Sorted list: ";

list.**sort**();

list.f**orward\_view**();

**return** 0;

}

**void** randomData(**biList&** list, **int** start, **int** stop, **int** count)

{

**srand**(**time**(0));

**for** (; count; count--)

{

**int** element = start + **rand**() % (stop + start);

list.**push\_back**(element);

}

}

**Додаток А**

**Текст програмною мовою С++.**

biList.h

#pragma once

#**include** <iostream>

**struct** biNode

{

**int** data;

**biNode**\* next;

**biNode**\* prev;

**biNode**(**int** value)

{

data = value;

next = prev = **nullptr**;

}

};

**class** biList

{

**public**:

**biList**();

~**biList**();

**int** getSize();

**biNode**\* get\_back();

**void** push\_back(**int**);

**void** push\_front(**int**);

**void** insert\_before(**int**, **int**);

**void** insert\_after(int, int);

**void** remove(int);

**void** forward\_view();

**void** reverse\_view();

**void** clear();

**biNode**\* find(**int**);

**void** splice\_in(biList& T2, **int** index);

**void** splice\_out(**biList**& T2, **int** start, **int** end);

**void** swap(**int**, **int**);

**void** sort();

**private**:

**biNode**\* head;

**biNode**\* tail;

**int** size;

**bool** is\_empty()

{

**return** head == **NULL**;

}

};

biList.cpp

#**include** "biList.h"

**biList**::biList() : head(**NULL**), tail(**NULL**), size(0) {}

**biList**::~biList()

{

**if** (**is\_empty**())

**return**;

**else**

**clear**();

}

**int** **biList**::getSize() { **return** size; }

**biNode**\* biList::get\_back()

{

**return** tail;

}

**void** **biList**::push\_back(**int** value)

{

**biNode**\* temp = **new** **biNode**(value);

**if** (tail == **NULL**)

head = tail = temp;

**else** **if** (size == 1)

{

tail = temp;

tail->prev = head;

head->next = tail;

}

**else**

{

tail->next = temp;

temp->prev = tail;

tail = temp;

}

**size**++;

}

**void** **biList**::push\_front(**int** value)

{

**biNode**\* temp = **new** biNode(value);

**if** (head == **NULL**)

head = tail = temp;

**else** **if** (size == 1)

{

head = temp;

head->next = tail;

tail->prev = head;

}

**else**

{

head->prev = temp;

temp->next = head;

head = temp;

}

**size**++;

}

**void** **biList**::insert\_before(**int** index, **int** value)

{

**biNode**\* insertNode = **new** biNode(value);

**biNode**\* cursor = head;

**if** (index)

{

**for** (**int** i = index; i; i--)

cursor = cursor->next;

**biNode**\* temp = cursor->prev;

temp->next = insertNode;

insertNode->prev = temp;

insertNode->next = cursor;

cursor->prev = insertNode;

size++;

}

**else**

**push**\_**front**(value);

}

**void** **biList**::insert\_after(**int** index, **int** value)

{

**biNode**\* insertAfter = new biNode(value);

**biNode**\* cursor = head;

**if** (index < size - 1)

{

**if** (index + 1 == size)

**push**\_**back**(value);

**for** (**int** i = index; i; i--)

cursor = cursor->next;

**biNode**\* temp = cursor->next;

temp->prev = insertAfter;

cursor->next = insertAfter;

insertAfter->prev = cursor;

insertAfter->next = temp;

size++;

}

**else**

**push**\_**back**(value);

}

**void** **biList**::remove(int index)

{

**if** (**is**\_**empty**())

{

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

**return**;

}

**if** (!index)

{

**biNode**\* cursor = head;

head = cursor->next;

head->prev = nullptr;

**delete** cursor;

}

**else** **if**(index + 1 == size)

{

**biNode**\* cursor = tail;

tail = cursor->prev;

tail->next = nullptr;

**delete** cursor;

}

**else**

{

**biNode**\* cursor = head;

**for** (; index; index--)

cursor = cursor->next;

**biNode**\* temp = cursor->prev;

temp->next = cursor->next;

temp = cursor->next;

temp->prev = cursor->prev;

**delete** cursor;

}

**size**--;

}

**void** **biList**::forward\_view()

{

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

**if** (**is\_empty**())

{

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

**return**;

}

**biNode**\* temp = head;

cout << "[ ";

**while** (temp->next != **nullptr**)

{

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

temp = temp->next;

}

cout << temp->data << " ]" << endl;

temp = **NULL**;

**delete** temp;

}

**void** **biList**::reverse\_view()

{

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

**if** (**is**\_**empty**())

{

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

**return**;

}

**biNode**\* temp = tail;

cout << "[ ";

**while** (temp->prev != **nullptr**)

{

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

temp = temp->prev;

}

cout << temp->data << " ]" << endl;

temp = **NULL**;

**delete** temp;

}

**void** **biList**::clear()

{

if (**is**\_**empty**())

{

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

**return**;

}

**if** (size <= 2)

{

head = tail = **NULL**;

}

**else**

{

**biNode**\* cursor = head;

**for** (**int** i = 0; i < size - 1; i++)

{

**biNode**\* temp = cursor;

cursor = cursor->next;

**delete** temp;

}

**delete** cursor;

head = tail = **NULL**;

}

size = 0;

}

**biNode**\* **biList**::find(int index)

{

**if** (index >= size)

return **nullptr**;

**else**

{

**biNode**\* cursor = head;

**for** (; index; index--)

cursor = cursor->next;

**return** cursor;

}

}

**void** **biList**::splice\_in(**biList&** T2, **int** index)

{

**biNode**\* cursor = T2.find(0);

**if** (cursor)

{

**while** (cursor->next)

{

**insert**\_**after**(index, cursor->data);

cursor = cursor->next;

index++;

}

**insert**\_**after**(index, cursor->data);

}

cursor = **NULL**;

**delete** cursor;

}

**void** **biList**::splice\_out(**biList&** T2, **int** start, **int** end)

{

**biNode**\* frontCursor = find(start);

**biNode**\* backCursor = find(end);

**if** (frontCursor && backCursor)

{

**while** (frontCursor->next != backCursor->next)

{

T2.**push**\_**back**(frontCursor->data);

frontCursor = frontCursor->next;

}

T2.**push**\_**back**(backCursor->data);

frontCursor = backCursor = **NULL**;

}

**delete** backCursor, frontCursor;

}

**void** **biList**::swap(**int** firstIndex, **int** secondIndex)

{

**if** (is\_empty())

{

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

**return**;

}

**biNode**\* firstItem = find(firstIndex);

**biNode**\* secondItem = find(secondIndex);

**if** (firstItem && secondItem)

{

**int** temp = firstItem->data;

firstItem->data = secondItem->data;

secondItem->data = temp;

}

}

**void** biList::sort()

{

**for** (**int** count = 0; count < 3; count++)

{

**for** (**int** i = 0; i < size - 1; i++)

{

**int** max\_index = i;

for (**int** j = i + 1; j < size; j++)

{

**if** (**find**(j)->data > **find**(i)->data)

{

max\_index = j;

}

}

**if** (i != max\_index)

{

**swap**(i, max\_index);

}

}

}

}