

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут ІКНІ

Кафедра ПЗ

ЗВІТ

До лабораторної роботи №8

На тему: «Лінійні структури даних»

З дисципліни: «Алгоритми та структури даних»

Лектор : доцент каф.ПЗ

Коротєєва Т.О.

Виконала: ст.гр.ПЗ-23

Кохман О.В.

Прийняв: асистент каф.ПЗ

Франко А.В.

« ____ » _____ 2022 р.

Σ _____ .

Львів – 2022

Тема: Лінійні структури даних.

Мета: познайомитися з лінійними структурами даних (стек, черга, дек, список) та отримати навички програмування алгоритмів, що їх обробляють.

Теоретичні відомості

Стек, черга, дек, список відносяться до класу лінійних динамічних структур.

Зі стеку (*stack*) можна видалити тільки той елемент, який був у нього доданий останнім: стек працює за принципом «останнім прийшов – першим пішов» (*last-in, first-out – LIFO*).

З черги (*queue*), навпаки, можна видалити тільки той елемент, який знаходився в черзі довше за всіх: працює принцип «першим прийшов – першим пішов» (*first-in, first-out – FIFO*).

Дек - це впорядкована лінійна динамічно змінювана послідовність елементів, у якій виконуються такі умови: 1) новий елемент може приєднуватися з обох боків послідовності; 2) вибірка елементів можлива також з обох боків послідовності. Дек називають **реверсивною** чергою або чергою з двома боками.

У зв'язаному списку (або просто списку; *linked list*) елементи лінійно впорядковані, але порядок визначається не номерами, як у масиві, а вказівниками, що входять до складу елементів списку. Списки є зручним способом реалізації динамічних множин.

Елемент двобічно зв'язаного списку (*doubly linked list*) – це запис, що містить три поля: *key* (ключ) і два вказівники *next* (наступний) і *prev* (попередній). Крім цього, елементи списку можуть містити додаткові дані.

У кільцевому списку (*circular list*) поле *prev* голови списку вказує на хвіст списку, а поле *next* хвоста списку вказує на голову списку.

Індивідуальне завдання

Розробити програму, яка читає з клавіатури послідовність даних, жодне з яких не повторюється, зберігає їх до структури даних (згідно з варіантом) та видає на екран такі характеристики:

- кількість елементів;

- мінімальний та максимальний елемент (для символів за кодом);
- третій елемент з початку послідовності та другий з кінця послідовності;
- елемент, що стоїть перед мінімальним елементом та елемент, що стоїть після максимального;
- знайти позицію елемента, значення якого задається з клавіатури;
- об'єднати дві структури в одну.

Всі характеристики потрібно визначити із заповненої структури даних.

Використовувати готові реалізації структур даних (наприклад, STL) **заборонено**.

Варіант 9: односторонній зв'язаний список цілих.

Код програми

Назва файлу: MyForm.h

```
#pragma once
#include <stdio.h>
#include <string>
#include <sstream>
#include "LinkedList.h"
namespace lab08 {
    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
    public ref class MyForm : public System::Windows::Forms::Form
    {
    public:
        MyForm(void)
        {
            InitializeComponent();
        }
    protected:
        ~MyForm()
        {
            if (components)
            {
                delete components;
            }
        }
    private: System::Windows::Forms::RichTextBox^ elementsRichTextBox;
    protected:
    private: System::Windows::Forms::Button^ computeStuffButton;
    private: System::Windows::Forms::Label^ countLabel;
    private: System::Windows::Forms::Label^ outputSizeLabel;
```

```

private: System::Windows::Forms::Label^ minElementLabel;
private: System::Windows::Forms::Label^ maxElementLabel;
private: System::Windows::Forms::Label^ elementBeforeMinLabel;
private: System::Windows::Forms::Label^ elementAfterMaxLabel;
private: System::Windows::Forms::Label^ outputMinElementLabel;
private: System::Windows::Forms::Label^ outputMaxElementLabel;
private: System::Windows::Forms::Label^ outputElementBeforeMinLabel;
private: System::Windows::Forms::Label^ outputElementAfterMaxLabel;
private: System::Windows::Forms::Label^ thirdElementFromStartLabel;
private: System::Windows::Forms::Label^ label2;
private: System::Windows::Forms::Label^ outputThirdElementFromStartLabel;
private: System::Windows::Forms::Label^ outputSecondElementFromEndLabel;
private: System::Windows::Forms::Label^ label3;
private: System::Windows::Forms::TextBox^ elementForPositionTextBox;
private: System::Windows::Forms::Label^ positionLabel;
private: System::Windows::Forms::Label^ outputPositionLabel;
private: System::Windows::Forms::Label^ label4;
private: System::Windows::Forms::RichTextBox^ newElementsrichTextBox;
private:
    System::ComponentModel::Container ^components;
#pragma region Windows Form Designer generated code

#pragma endregion

private: System::Void MyForm_Load(System::Object^ sender,
System::EventArgs^ e) {
}

private: System::Void computeStuffButton_Click(System::Object^ sender,
System::EventArgs^ e) {
    outputPositionLabel->Text = "";
    std::string inputElements = toStandardString(elementsRichTextBox-
>Text);

    std::stringstream stream(inputElements);
    LinkedList *linkedList = new LinkedList();
    int n;
    while (stream >> n) {
        linkedList->insert(n);
    }
    std::string newElements = toStandardString(newElementsrichTextBox-
>Text);

    std::stringstream newStream(newElements);
    while (newStream >> n) {
        linkedList->insert(n);
        elementsRichTextBox->Text += " " + n.ToString();
    }
    int size = linkedList->getCount();
    outputSizeLabel->Text = size.ToString();
    if (size > 0) {
        outputMinElementLabel->Text = linkedList-
>minElement().ToString();
        outputMaxElementLabel->Text = linkedList-
>maxElement().ToString();
    }
    if (size >= 3) {
        outputThirdElementFromStartLabel->Text = linkedList-
>thirdElementFromStart().ToString();
    }
    if (size >= 2) {
        outputSecondElementFromEndLabel->Text = linkedList-
>secondElementFromEnd().ToString();
        int elementBeforeMin = linkedList->elementBeforeMin();
        if (elementBeforeMin != INT_MAX) {
            outputElementBeforeMinLabel->Text =
elementBeforeMin.ToString();

```

```

    }
    int elementAfterMax = linkedList->elementAfterMax();
    if (elementAfterMax != INT_MIN) {
        outputElementAfterMaxLabel->Text =
elementAfterMax.ToString();
    }
}
if (elementForPositionTextBox->Text != "") {
    int elementForPosition =
System::Convert::ToInt64(elementForPositionTextBox->Text);
    int position = linkedList-
>searchPosition(elementForPosition);
    if (position != INT_MIN) {
        outputPositionLabel->Text = position.ToString();
    }
}
}
static std::string toStandardString(System::String^ string) {
    using System::Runtime::InteropServices::Marshal;
    System::IntPtr pointer = Marshal::StringToHGlobalAnsi(string);
    char* charPointer = reinterpret_cast<char*>(pointer.ToPointer());
    std::string returnString(charPointer, string->Length);
    Marshal::FreeHGlobal(pointer);
    return returnString;
}
};
}

```

Назва файлу: MyForm.cpp

```

#include "MyForm.h"
using namespace lab08;
int main() {
    Application::EnableVisualStyles();
    Application::SetCompatibleTextRenderingDefault(false);
    Application::Run(gcnew MyForm());
    return 0;
}

```

Назва файлу: LinkedList.h

```

#ifndef LINKEDLIST_H
#define LINKEDLIST_H
#include <iostream>
using namespace std;
struct Node {
    int data;
    Node* next;
    Node(int val = 0) :data(val), next(nullptr) {}
    Node(int val, Node* tempNext) :data(val), next(tempNext) {}
};
class LinkedList{
    Node* head;
public:
    LinkedList();
    void insert(int val);
    int searchPosition(int val);
    int getCount();
    int maxElement();
    int minElement();
    int thirdElementFromStart();
    int secondElementFromEnd();
}

```

```

        int elementBeforeMin();
        int elementAfterMax();
        Node* getHead();
};
#endif

```

Назва файлу: LinkedList.cpp

```

#include "LinkedList.h"
LinkedList::LinkedList() :head(nullptr){}
void LinkedList::insert(int val) {
    Node* new_node = new Node(val);
    if (head == nullptr) {
        head = new_node;
    }
    else{
        new_node->next = head;
        head = new_node;
    }
}
int LinkedList::getCount() {
    Node* temp = head;
    int count = 0;
    while (temp != nullptr) {
        count++;
        temp = temp->next;
    }
    return count;
}
int LinkedList::searchPosition(int val) {
    Node* temp = head;
    int length = 0;
    while (temp != nullptr) {
        temp = temp->next;
        length++;
    }
    temp = head;
    while (temp != nullptr) {
        if (temp->data == val)
            return length;
        length--;
        temp = temp->next;
    }
    return INT_MIN;
}
int LinkedList::maxElement() {
    Node* temp = head;
    int max = INT_MIN;
    while (temp != nullptr) {
        if (max < temp->data)
            max = temp->data;
        temp = temp->next;
    }
    return max;
}
int LinkedList::elementAfterMax() {
    Node* temp = head;
    Node* maxNode = head;
    Node* afterMaxNode = head;
    int max = INT_MIN;
    while (temp != nullptr) {
        if (max < temp->data) {
            afterMaxNode = maxNode;

```

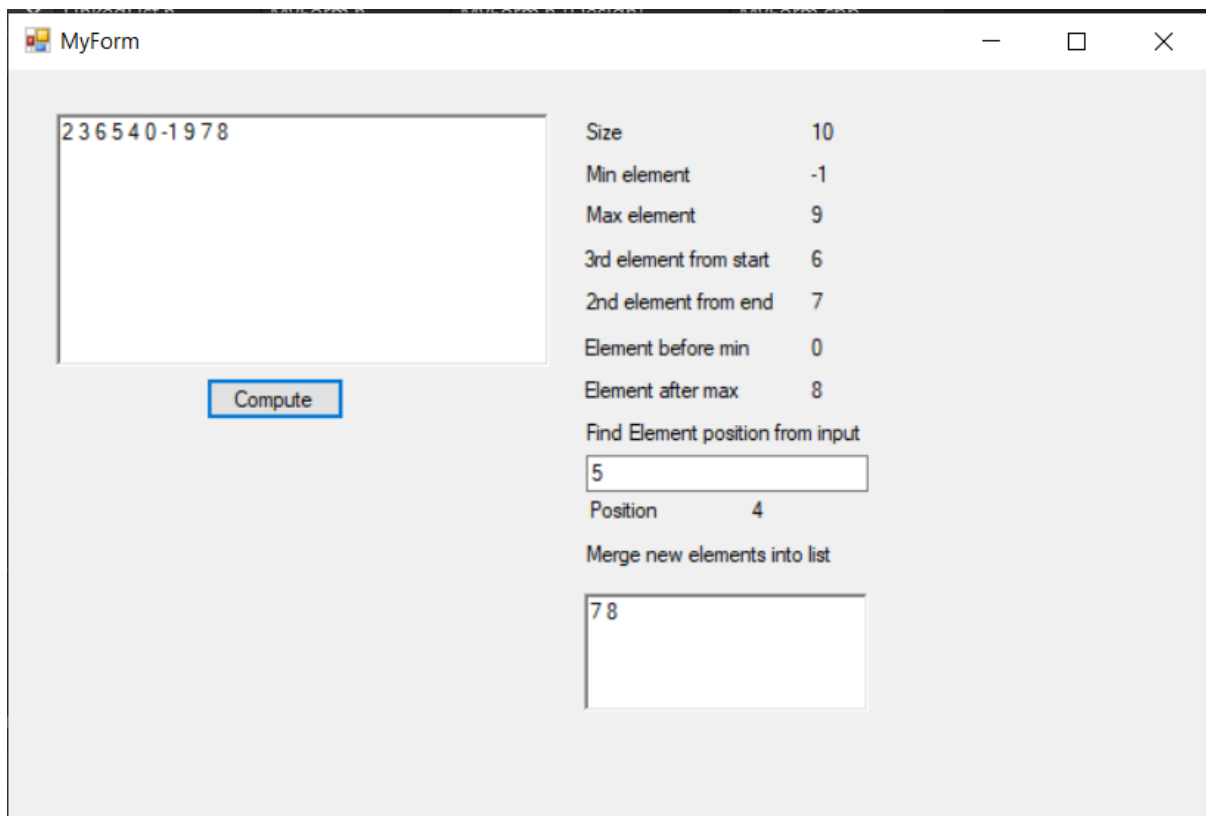
```

        maxNode = temp;
        max = temp->data;
    }
    temp = temp->next;
}
if (afterMaxNode->data == maxNode->data) {
    return INT_MIN;
}
return afterMaxNode != nullptr ? afterMaxNode->data : INT_MIN;
}
int LinkedList::elementBeforeMin() {
    Node* temp = head;
    Node* minNode = head;
    int min = INT_MAX;
    while (temp != nullptr) {
        if (min > temp->data) {
            minNode = temp;
            min = temp->data;
        }
        temp = temp->next;
    }
    return minNode->next != nullptr ? minNode->next->data : INT_MAX;
}
int LinkedList::minElement() {
    Node* temp = head;
    int min = INT_MAX;
    while (temp != nullptr) {
        if (min > temp->data)
            min = temp->data;

        temp = temp->next;
    }
    return min;
}
int LinkedList::thirdElementFromStart() {
    Node* temp = head;
    int length = 0;
    while (temp != nullptr) {
        temp = temp->next;
        length++;
    }
    if (length < 3)
        return INT_MIN;
    temp = head;
    for (int i = 1; i < length - 3 + 1; i++)
        temp = temp->next;
    return temp->data;
}
int LinkedList::secondElementFromEnd() {
    Node* temp = head;
    return temp->next == nullptr ? INT_MIN : temp->next->data;
}
Node* LinkedList::getHead() {
    return head;
}

```

Протокол роботи



Size	10
Min element	-1
Max element	9
3rd element from start	6
2nd element from end	7
Element before min	0
Element after max	8
Find Element position from input	5
Position	4
Merge new elements into list	78

Рис. 1 Результат виконання програми.

Висновок

На цій лабораторній роботі я дізналась про лінійні структури даних та навчилась використовувати на практиці одну з них, а саме – лінійний однозв'язний список. Продемонструвала результати роботи у віконному додатку у Visual Studio 2022.