

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ  
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра програмної інженерії та інформаційних технологій управління

Звіт з лабораторної роботи № 1  
з дисципліни «Основи теорії алгоритмів»

Виконав:

Ст. гр. КН-221в

Шулюпов Є.Р.

Перевірила:

доцент каф. ПІТУ

Солонська С.В.

Харків

2022

## ТЕМА: БАЗОВІ СТРУКТУРИ ДАНИХ

### ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ

Розробити програму, яка читає з клавіатури послідовність  $N$  цілих чисел ( $1 < N < 256$ ), жодне з яких не повторюється, зберігає їх до структури даних (згідно з завданням) та видає на екран такі характеристики:

- кількість елементів;
- середнє арифметичне збережених елементів;
- мінімальний та максимальний елемент;
- четвертий елемент послідовності;
- елемент, що йде перед мінімальним елементом.

Наголосимо, що всі характеристики потрібно визначити із заповненої структури даних. Дозволено використовувати лише ті операції, що притаманні заданій структурі, наприклад, заборонено отримувати доступ до елемента із довільною позицією у черзі, яку реалізовано на базі масиву.

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

Варіант 3: однобічно зв'язаний список.

### МЕТА РОБОТИ

Метою виконання лабораторної роботи є ознайомлення з базовими структурами даних (список, черга, стек) та отримання навичок програмування алгоритмів, що їх обробляють.

## 1 ОСНОВНІ ТЕОРЕТИЧНІ ПОЛОЖЕННЯ

Структура даних (з англ. «data structure») – програмна одиниця, яка дозволяє зберігати й обробляти множину однотипних і/чи логічно зв'язаних даних у обчислювальній техніці [1].

У програмуванні часто потрібно мати справу з множинами, які змінюються у процесі виконання алгоритмів. Роздивимось структури, необхідні для динамічних множин. До елементарних структур даних належать стеки, черги та зв'язані списки.

Стеки й черги – це динамічні множини, у яких елемент, що видаляється з множини, не задається довільно, а визначається самою структурою множини [2].

Зі стека (stack) можна видалити лише той елемент, котрий був доданий останнім. Стек працює за принципом LIFO (a last-in, first-out) – останній зайшов, перший вийшов.

З черги (queue), навпаки, можна видалити тільки той елемент, який був доданий першим, тобто який там довше всього знаходиться. Стек працює за принципом FIFO (a first-in, first-out) – перший зайшов, перший вийшов [3].

Зв'язані списки (linked list) – структури даних, у яких елементи лінійно впорядковані, але порядок визначається не номерами, а вказівниками, що входять до складу елементів списку. Зв'язані списки бувають односторонні й двосторонні.

В односторонньому списку (singly linked list) елементи мають два компонента: ключ (key), де зберігаються дані, та вказівник на наступний елемент списку. У останньому елементі списку вказівник має значення nullptr, тобто не вказує ні на який елемент. Перший елемент називається головою (head). З нього починається рух у списку через вказівники.

В двосторонньому списку (doubly linked list) елементи мають три компоненти: ключ, де зберігаються дані, та два вказівника. Один з них (prev) вказує на попередній елемент списку, а інший (next) – на наступний. Перший елемент списку називається головою, а останній – хвіст (tail). Вказівники prev голови списку та next хвоста не вказують ні на який елемент, тобто мають значення nullptr.

Також існують впорядкований список, у якому елементи на відміну від неупорядкованого списку знаходяться в порядку збільшення їх ключів, та кільцевий

список, у якому вказівник prev голови списку вказує на його хвіст, а вказівник next хвоста – на його голову [4].

## 2 ОПИСАННЯ РОЗРОБЛЕНОГО ЗАСТОСУНКУ

Весь код програми реалізований у одному файлі реалізації ОТА\_Lab1. У цьому файлі розв’язуються всі необхідні задачі відповідно до завдання.

Програмний код:

```
#include <iostream>

using namespace std;

struct ListItem
{
    int data;
    ListItem* next;
};

ListItem* ReadList(int);
void arithm(ListItem*);
int max(ListItem*);
int min(ListItem*);
void printElem(ListItem*, int);
void printElemBeforeMin(ListItem*);
void DeleteList(ListItem*);
void WriteList(ListItem*);
ListItem* addElem(ListItem*&);
void DeleteElem(ListItem*&);

ListItem* ReadList(int N)
{
    while (N < 1 || N>256)
    {
        cout << "Error; Уведіть коректну кількість елементів: ";
        cin >> N;
        cout << endl;
    }
    ListItem* check = nullptr;
    ListItem* last = nullptr;
    ListItem* first = nullptr;
    for (int i = 0; i < N; i++)
    {
        cout << "Уведіть " << i+1 << "-й елемент списку: ";
        int k;
        cin >> k;
        check = first;
        while (check != nullptr)
        {
            while (k == check->data)
            {
                cout << "Error; Елементи списку повторюються" << endl <<
"Перепишіть останній елемент: ";
                cin >> k;
            }
            check = check->next;
        }
        ListItem* link = new ListItem;
```

```

        link->data = k;
        link->next = 0;
        if (last == 0)
        {
            first = link;
        }
        else
        {
            last->next = link;
        }
        last = link;
    }
    return first;
}

void arithm(ListItem* a)
{
    int sa = 0, count=0;
    while (a != nullptr)
    {
        sa += a->data;
        count++;
        a = a->next;
    }
    cout <<"Середнє арифметичне елементів списку: " << double(sa) / double(count) <<
endl;
}

int max(ListItem* a)
{
    int max=a->data;
    while (a != nullptr)
    {
        max =max > a->data ? max : a->data;
        a = a->next;
    }
    return max;
}

int min(ListItem* a)
{
    int min = a->data;
    while (a != nullptr)
    {
        min = min < a->data ? min : a->data;
        a = a->next;
    }
    return min;
}

void printElem(ListItem* a, int number)
{
    int count=0;
    for (ListItem* i = a; i != nullptr; i = i->next)
        count++;
    if (count >= number)
    {
        for (int i = 1; i < number; i++)
        {
            a = a->next;
        }
        cout << number << "-й елемент списку: " << a->data << endl;
    }
}

```

```

    }
    else
    {
        cout << number << "-го елемента немає" << endl;
    }
}

void printElemBeforeMin(ListItem* a)
{
    int count = 0;
    ListItem* first = a;
    while (first->data != min(a))
    {
        count++;
        first = first->next;
    }
    if (count == 0)
    {
        cout << "Перший елемент мінімальний: " << a->data << endl;
    }
    else {
        for (int i = 0; i < count-1; i++)
        {
            a = a->next;
        }
        cout << "Елемент, що йде перед мінімальним елементом: " << a->data << endl;
    }
}

ListItem* addElem(ListItem*& list)
{
    ListItem* check = list;
    cout << "Уведіть новий елемент списку: ";
    ListItem* item = new ListItem;
    int k;
    cin >> k;
    while (check != nullptr)
    {
        if (k == check->data)
            throw "Error; Елементи списку повторюються";
        check = check->next;
    }
    item->next = list;
    item->data = k;
    list = item;
    return list;
}

void DeleteElem(ListItem*& list)
{
    ListItem* next = list->next;
    delete list;
    list = next;
}

void DeleteList(ListItem* list)
{
    while (list != nullptr)
    {
        ListItem* next = list->next;
        delete list;
        list = next;
    }
}

```

```

}

void WriteList(ListItem* list)
{
    cout << "Структура: ";
    while (list != nullptr) {
        cout.width(6);
        cout << list->data;
        list = list->next;
    }
    cout << endl;
}

int main()
{
    setlocale(LC_ALL, "UKRAINIAN");

    int N;
    cout << "Уведіть кількість елементів однозв'язного списку: ";
    cin >> N;
    ListItem* list = ReadList(N);
    WriteList(list);
    arithm(list);
    cout << "Мінімальний елемент списку: " << min(list) << endl;
    cout << "Максимальний елемент списку: " << max(list) << endl;
    printElem(list, 4);
    printElemBeforeMin(list);
    cout << endl;

    cout << "Видалили перший елемент структури" << endl;
    DeleteElem(list);
    WriteList(list);
    arithm(list);
    cout << "Мінімальний елемент списку: " << min(list) << endl;
    cout << "Максимальний елемент списку: " << max(list) << endl;
    printElem(list, 4);
    printElemBeforeMin(list);
    cout << endl;

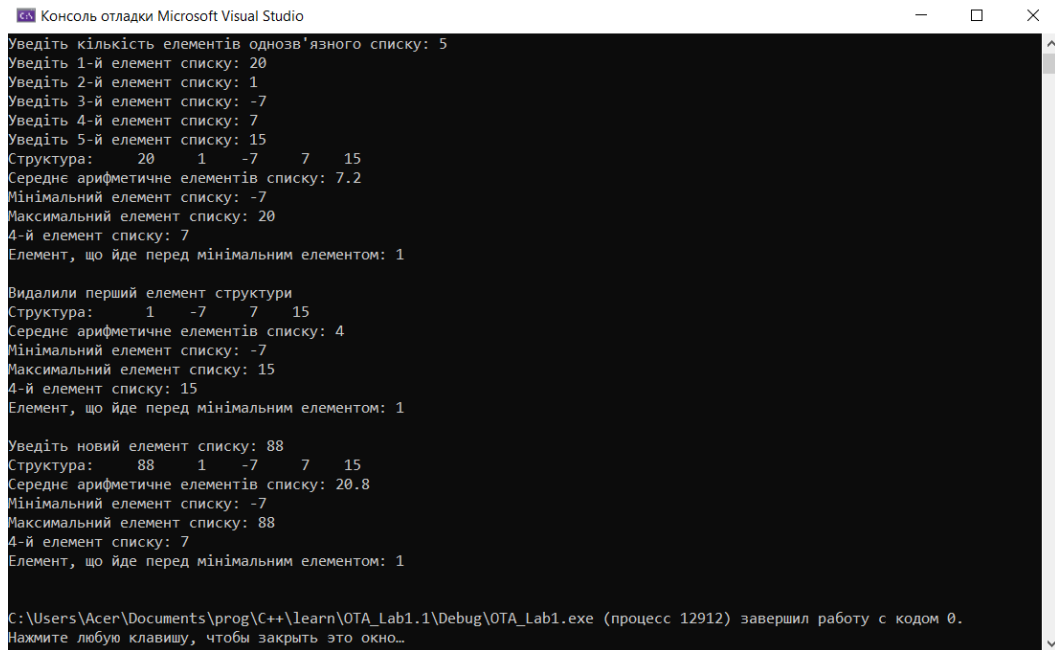
    addElem(list);
    WriteList(list);
    arithm(list);
    cout << "Мінімальний елемент списку: " << min(list) << endl;
    cout << "Максимальний елемент списку: " << max(list) << endl;
    printElem(list, 4);
    printElemBeforeMin(list);
    cout << endl;

    DeleteList(list);
    return 0;
}

```

### 3 РЕЗУЛЬТАТИ

3.1 Розглянемо ситуацію, коли всі елементи вводяться коректно згідно з умовою завдання (рис. 3.1).



```

Консоль отладки Microsoft Visual Studio
Уведіть кількість елементів однозв'язного списку: 5
Уведіть 1-й елемент списку: 20
Уведіть 2-й елемент списку: 1
Уведіть 3-й елемент списку: -7
Уведіть 4-й елемент списку: 7
Уведіть 5-й елемент списку: 15
Структура: 20 1 -7 7 15
Середнє арифметичне елементів списку: 7.2
Мінімальний елемент списку: -7
Максимальний елемент списку: 20
4-й елемент списку: 7
Елемент, що йде перед мінімальним елементом: 1

Видалили перший елемент структури
Структура: 1 -7 7 15
Середнє арифметичне елементів списку: 4
Мінімальний елемент списку: -7
Максимальний елемент списку: 15
4-й елемент списку: 15
Елемент, що йде перед мінімальним елементом: 1

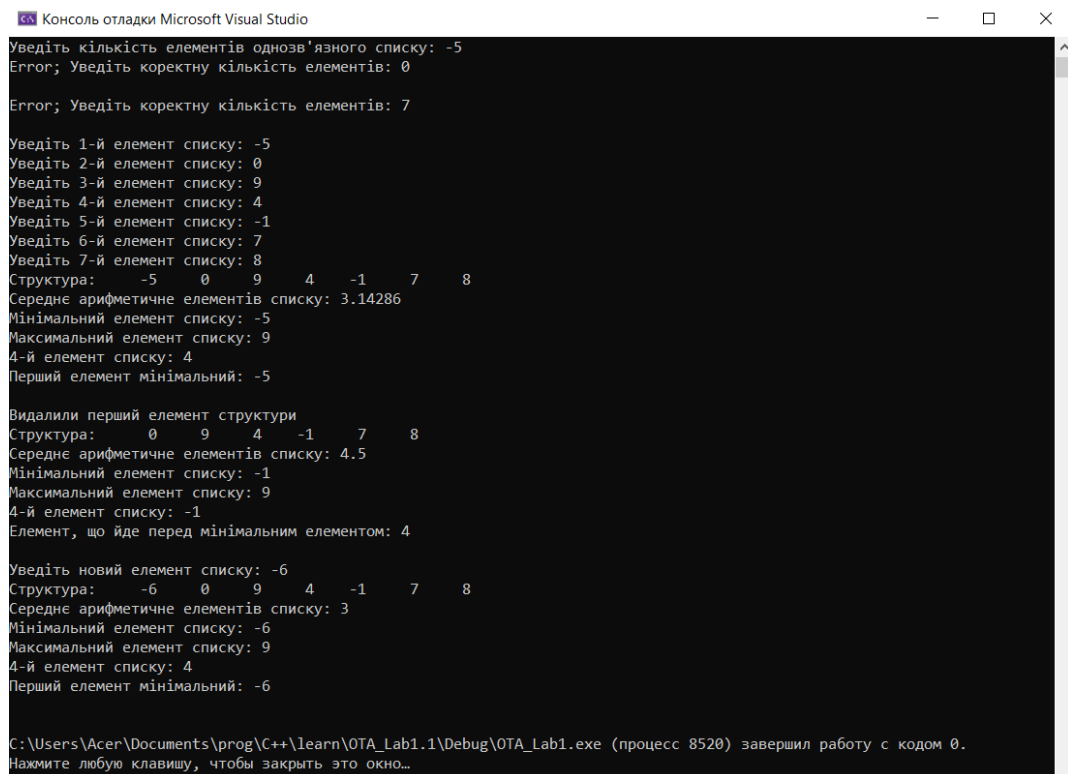
Уведіть новий елемент списку: 88
Структура: 88 1 -7 7 15
Середнє арифметичне елементів списку: 20.8
Мінімальний елемент списку: -7
Максимальний елемент списку: 88
4-й елемент списку: 7
Елемент, що йде перед мінімальним елементом: 1

C:\Users\Acer\Documents\prog\C++\learn\OTA_Lab1.1\Debug\OTA_Lab1.exe (процесс 12912) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно...

```

Рисунок 3.1 – Вводяться коректні дані

3.2 При введенні кількості елементів  $N$ , що не задовольняє інтервал, програма попереджає про це й пропонує перезаписати цю змінну. Якщо не існує елемента, який стоїть перед мінімальним, то програма виводить повідомлення, що перший елемент мінімальний (рис. 3.2).



```

Консоль отладки Microsoft Visual Studio
Уведіть кількість елементів однозв'язного списку: -5
Error; Уведіть коректну кількість елементів: 0

Error; Уведіть коректну кількість елементів: 7

Уведіть 1-й елемент списку: -5
Уведіть 2-й елемент списку: 0
Уведіть 3-й елемент списку: 9
Уведіть 4-й елемент списку: 4
Уведіть 5-й елемент списку: -1
Уведіть 6-й елемент списку: 7
Уведіть 7-й елемент списку: 8
Структура: -5 0 9 4 -1 7 8
Середнє арифметичне елементів списку: 3.14286
Мінімальний елемент списку: -5
Максимальний елемент списку: 9
4-й елемент списку: 4
Перший елемент мінімальний: -5

Видалили перший елемент структури
Структура: 0 9 4 -1 7 8
Середнє арифметичне елементів списку: 4.5
Мінімальний елемент списку: -1
Максимальний елемент списку: 9
4-й елемент списку: -1
Елемент, що йде перед мінімальним елементом: 4

Уведіть новий елемент списку: -6
Структура: -6 0 9 4 -1 7 8
Середнє арифметичне елементів списку: 3
Мінімальний елемент списку: -6
Максимальний елемент списку: 9
4-й елемент списку: 4
Перший елемент мінімальний: -6

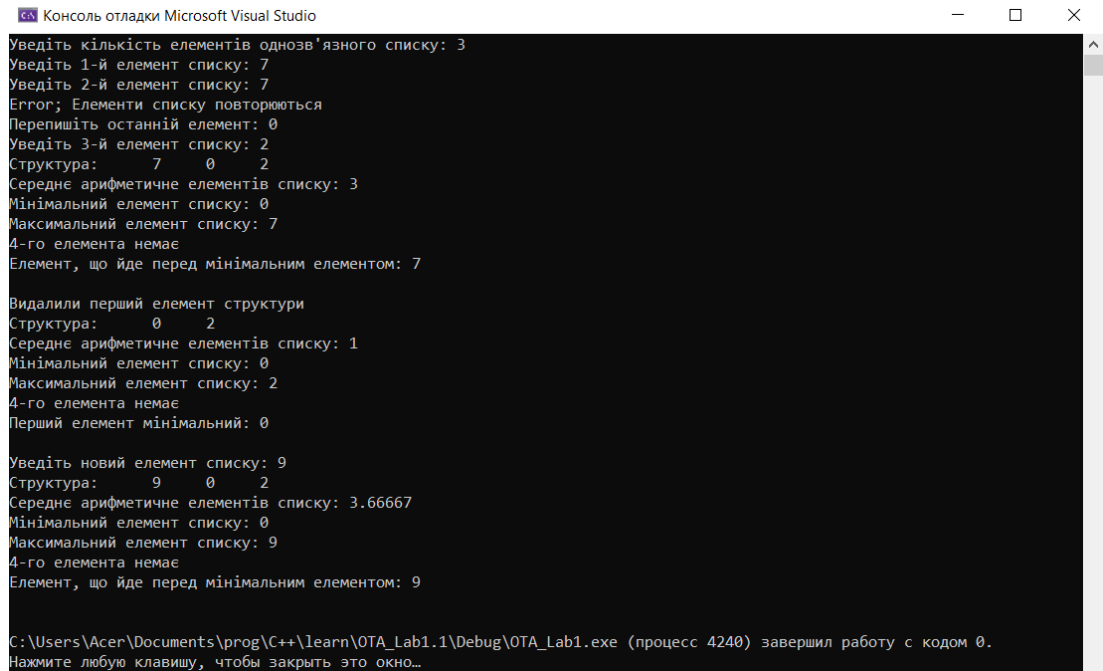
C:\Users\Acer\Documents\prog\C++\learn\OTA_Lab1.1\Debug\OTA_Lab1.exe (процесс 8520) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно...

```

Рисунок 3.2 –  $N$  вводиться некоректно



3.3 При введенні елементу, який вже був записаний, програма попереджає про це й пропонує перезаписати цей елемент. Якщо не існує четвертого елемента, то програма виводить відповідне повідомлення (рис. 3.3).



```
Консоль отладки Microsoft Visual Studio
Уведіть кількість елементів однозв'язного списку: 3
Уведіть 1-й елемент списку: 7
Уведіть 2-й елемент списку: 7
Error; Елементи списку повторюються
Перепишіть останній елемент: 0
Уведіть 3-й елемент списку: 2
Структура:      7      0      2
Середнє арифметичне елементів списку: 3
Мінімальний елемент списку: 0
Максимальний елемент списку: 7
4-го елемента немає
Елемент, що йде перед мінімальним елементом: 7

Видалили перший елемент структури
Структура:      0      2
Середнє арифметичне елементів списку: 1
Мінімальний елемент списку: 0
Максимальний елемент списку: 2
4-го елемента немає
Перший елемент мінімальний: 0

Уведіть новий елемент списку: 9
Структура:      9      0      2
Середнє арифметичне елементів списку: 3.66667
Мінімальний елемент списку: 0
Максимальний елемент списку: 9
4-го елемента немає
Елемент, що йде перед мінімальним елементом: 9

C:\Users\Acer\Documents\prog\C++\learn\OTA_Lab1.1\Debug\OTA_Lab1.exe (процесс 4240) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно...
```

Рисунок 3.3 – Вводяться однакові елементи

## ВИСНОВКИ

Виконавши лабораторну роботу, було закріплено ряд знань щодо реалізації одностороннього списку у мові програмування C++. Отримано практичні навички створення та використання базових структур даних (однобічно зв'язний список) та вдосконалено навички програмування алгоритмів, що їх обробляють. Закріплено створення та опрацювання статичних та динамічних структур даних, а також використання типових алгоритмів на цих структурах. У процесі виконання лабораторної роботи виконано ряд завдань на реалізацію функції очищення пам'яті, у якій зберігалися елементи списку.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Алгоритми і структури даних: практикум: навч. посіб./ Н.К. Стратієнко, М.Д. Годлевський, І.О. Бородіна.- Харьков: НТУ "ХПИ", 2017. - 224 с.