Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

про виконання

**Лабораторних та практичних робіт № 10**

***з дисципліни:*** «Мови та парадигми програмування»

***з розділу***: «Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур»

***Виконала:***

студентка групи ШІ-11

Педченко Юлія Сергіївна

# **Тема роботи:**

Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур.

# **Мета роботи:**

Навчитися працювати з динамічними структурами. Засвоїти весь теоретичний матеріал по цим темам і вміти використовувати його в коді.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Динамічні структури даних (чергa).
* Тема №2: Динамічні структури даних (списки).
* Тема №3: Динамічні структури даних (стек).
* Тема №4: Динамічні структури даних (дерево).

1. Індивідуальний план опрацювання теорії:

* Тема №1: Динамічні структури даних (черга).
  + Джерела Інформації
    - <https://youtu.be/fmHyFTji-Lc?feature=shared>
  + Що опрацьовано:
    - Опрацьовано матеріал щодо динамічної структури – черги. Розуміння того, як ця структура працює і для чого вона потрібна.
  + Статус: Ознайомлений
  + Початок опрацювання теми: -
  + Звершення опрацювання теми: -
* Тема №2: Динамічні структури даних (списки).
  + Джерела Інформації:
    - <https://www.youtube.com/watch?v=C9FK1pHLnhI&list=PLQOaTSbfxUtAIipl4136nwb4ISyFk8oI4&ab_channel=%23SimpleCode>
    - <https://habr.com/ru/sandbox/153128/>
    - <https://codelessons.dev/ru/spisok-list-v-s-polnyj-material/>
  + Що опрацьовано:
    - Опрацьовано матеріал щодо динамічної структури – список. Розуміння того, як ця структура працює, які її типи існують і для чого вона потрібна. Уміння реалізації структури.
  + Статус: Ознайомлений.
  + Початок опрацювання теми: -
  + Звершення опрацювання теми: -
* Тема №3: Динамічні структури даних (стек).
  + Джерела Інформації:
    - <https://acode.com.ua/urok-111-stek-i-kupa/>
    - <https://habr.com/ru/articles/341586/>
  + Що опрацьовано:
    - Опрацьовано матеріал щодо динамічної структури – список. Розуміння того, як ця структура працює і для чого вона потрібна.
  + Статус: Ознайомлений.
  + Початок опрацювання теми: -
  + Звершення опрацювання теми: -
* Тема №4 Динамічні структури даних (дерево).
  + Джерела Інформації:
    - <https://purecodecpp.com/uk/archives/2483>
    - <https://youtu.be/qBFzNW0ALxQ?feature=shared>
    - <https://youtu.be/VRX99cNoczY?feature=shared>
    - <https://uk.myservername.com/binary-search-tree-c>
  + Що опрацьовано:
    - Опрацьовано матеріал щодо динамічної структури – список. Розуміння того, як ця структура працює і для чого вона потрібна.
  + Статус: Ознайомлений
  + Початок опрацювання теми: -
  + Звершення опрацювання теми: -

# **Виконання роботи:**

## **1. Опрацювання завдання та вимог до програм та середовища:**

Завдання №1  Lab# programming: VNS Lab 10

* Варіант завдання - 1
* Створити однонаправлений список та виконати певні дії з ним.

Завдання №2 Lab# programming: Algotester Lab 5

* Варіант завдання - 2
* В пустелі існує незвичайна печера, що є двохвимірною. Дано висоту і ширину. Всередині печери є пустота, пісок та каміння. Одного дня стався землетрус і весь пісок посипався вниз. Він падає на найнижчу клітинку з пустотою, але він не може пролетіти через каміння. Ваше завдання сказати як буде виглядати печера після землетрусу.

Завдання №3 Practice# programming: Class Practice Task

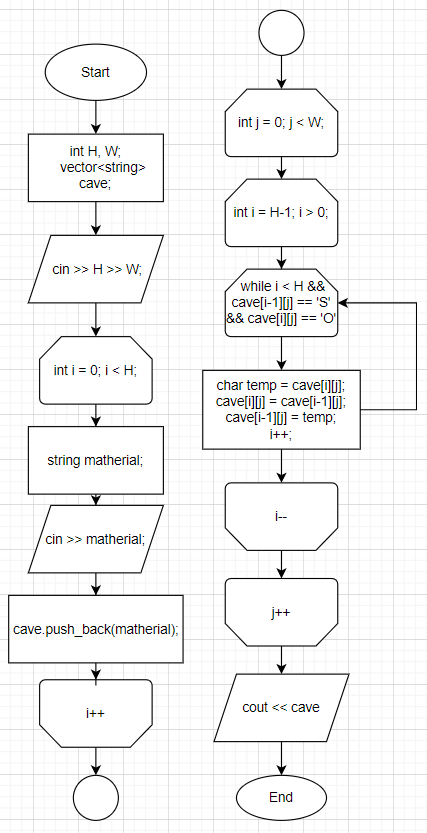
* Реалізація двохзв’язного списку. Функція виведення списку, його обертання, порівняння двох списків та додавання великих чисел, записаних за допомогою списку.

## **2. Дизайн та планована оцінка часу виконання завдань:**

Програма №1 Lab# programming: VNS Lab 10

* Планований час на реалізацію: 3 години

Програма №2 Lab# programming: Algotester Lab 5



* Планований час на реалізацію: 50 хвилин

Програма №3 Practice# programming: Class Practice Task

* Планований час на реалізацію: 3 години

## **3. Конфігурація середовища до виконання завдань:**

-

## **4. Код програм з посиланням на зовнішні ресурси:**

Завдання №1 Lab# programming: VNS Lab 10

#include <iostream>

#include <fstream>

using *namespace* std;

*class* Node {

*public:*

*int* data;

    Node\* next;

};

*class* List {

*public:*

    Node\* head = nullptr;

*int* size = 0;

*bool* is\_empty() {

    if (head == nullptr) return true;

    return false;

}

*void* create\_node\_by\_number(Node\* *node*, *int* *number*) {

    if (*number* < 1) return;

    if (is\_empty()) {

*node*->next = nullptr;

        head = *node*;

        size++;

    }

    else if (*number* == 1) {

*node*->next = head;

        head = *node*;

        size++;

    }

    else if (*number* > size) {

        Node\* current = head;

        while (current->next != nullptr) {

            current = current->next;

        }

        current->next = *node*;

        size++;

    }

    else {

        Node\* current = head;

        for (*int* i = 1; i < *number*-1; i++) {

            current = current->next;

            }

*node*->next = current->next;

        current->next = *node*;

        size++;

    }

}

*void* create\_node\_by\_number(*int* *data*, *int* *number*) {

    Node\* a = new Node();

    a->data = *data*;

    a->next = nullptr;

    create\_node\_by\_number(a, *number*);

}

*void* delete\_node\_by\_number(*int* *number*) {

    if(is\_empty() || *number* < 1 || *number* > size) return;

    else if(*number* == 1 && size == 1) {

        delete head;

        head = nullptr;

        size--;

    }

    else if (*number* == 1 && size > 1) {

        Node\* temp;

        temp = head;

        head = head->next;

        delete temp;

        size--;

    }

    else if (*number* == size) {

        Node\* current = head;

        while (current->next->next != nullptr) {

            current = current->next;

        }

        delete current->next->next;

        current->next = nullptr;

        size--;

    }

    else {

        Node\* temp;

        Node\* current = head;

        for (*int* i = 1; i < *number*-1; i++) {

            current = current->next;

        }

        temp = current->next;

        current->next = current->next->next;

        delete temp;

        size--;

    }

}

*void* print() {

    if (size == 0) {

        cout << "Список порожній" << endl;

        return;

    }

    Node\* current = head;

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

        cout << current->data << " ";

        current = current->next;

    }

    cout << endl;

}

*void* tofile(string *path*) {

    ofstream tofile;

    tofile.open(*path*, ios::app);

    Node\* current = head;

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

        tofile << current->data << "\n";

        current = current->next;

    }

    tofile.close();

}

*void* destroy() {

    Node\* del;

    Node\* current = head;

    while (current->next != nullptr) {

        del = current;

        current = current->next;

        delete del;

        size--;

    }

    delete current->next;

    size--;

    head = nullptr;

}

*void* fromfile(string *path*) {

*int* number = 1;

    string line;

    ifstream fromfile;

    fromfile.open(*path*, ios::app);

    while (getline(fromfile, line)) {

*int* data = stoi(line);

        create\_node\_by\_number(data, number);

        number++;

    }

    number--;

    fromfile.close();

}

};

///////////////////////////////////////////

*int* main() {

    List list;

    list.create\_node\_by\_number(3, 1);

    list.create\_node\_by\_number(5, 2);

    list.create\_node\_by\_number(8, 3);

    list.create\_node\_by\_number(1, 4);

    list.create\_node\_by\_number(6, 5);

    list.create\_node\_by\_number(0, 6);

    list.create\_node\_by\_number(11, 7);

    list.print();

    cout << "Елемент під яким номером ви хочете знищити?(від 1 до 7)" << endl;

*int* a;

    cin >> a;

    list.delete\_node\_by\_number(a);

    list.print();

    cout << "Який елемент і яким номером ви хочете додати?(номер від 1 до 7)" << endl;

*int* b, c;

    cin >> b >> c;

    list.create\_node\_by\_number(b, c);

    list.print();

    ofstream destroydata;

    destroydata.open("Dota2.txt", ios::trunc);

    destroydata.close();

    list.tofile("Dota2.txt");

    list.destroy();

    list.print();

    list.fromfile("Dota2.txt");

    list.print();

    list.destroy();

    list.print();

    return 0;

}

Завдання №2 Lab# programming: Algotester Lab 5

#include <iostream>

#include <vector>

using *namespace*  std;

*int* main() {

*int* H, W;

    vector<string> cave;

    cin >> H >> W;

    if ( H<1 || W<1 || H>1000 || W>1000) return 0;

    for(*int* i = 0; i < H; i++) {

        string matherial;

        cin >> matherial;

        cave.push\_back(matherial);

    }

    for(*int* j = 0; j < W; j++) {

        for(*int* i = H-1; i > 0; i--) {

            while(i < H && cave[i-1][j] == 'S' && cave[i][j] == 'O') {

*char* temp = cave[i][j];

                cave[i][j] = cave[i-1][j];

                cave[i-1][j] = temp;

                i++;

            }

        }

    }

    for (*int* i = 0; i < H; i++) {

        for (*int* j = 0; j < W; j++) {

            cout << cave[i][j];

        }

        cout << endl;

    }

return 0;

}

Завдання №3 Practice# programming: Class Practice Task

#include <iostream>

using *namespace* std;

*class* Node {

*public:*

*int* data;

    Node\* next;

    Node\* previous;

};

*class* List {

*public:*

    Node\* head = nullptr;

    Node\* tail = nullptr;

*int* size = 0;

*bool* is\_empty() {

    if (head == nullptr) return true;

    return false;

}

*void* reverse() {

*int* i = 1;

    Node\* start = head;

    Node\* end = tail;

*int* temp;

    while (i <= (size/2)) {

        temp = start->data;

        start->data = end->data;

        end->data = temp;

        start = start->next;

        end = end->previous;

        i++;

    }

}

*void* create\_node\_by\_number(Node\* *node*, *int* *number*) {

    if (*number* < 1) return;

    if (is\_empty()) {

*node*->next = nullptr;

*node*->previous = nullptr;

        head = *node*;

        tail = *node*;

        size++;

    }

    else if (*number* == 1) {

*node*->next = head;

*node*->previous = nullptr;

        head = *node*;

        size++;

    }

    else if (*number* > size) {

*node*->previous = tail;

        tail->next = *node*;

        tail = *node*;

*node*->next = nullptr;

        size++;

    }

    else {

        Node\* current = head;

        for (*int* i = 1; i < *number*-1; i++) {

            current = current->next;

            }

*node*->next = current->next;

*node*->previous = current;

        current->next = *node*;

        size++;

    }

}

*void* create\_node\_by\_number(*int* *data*, *int* *number*) {

    Node\* a = new Node();

    a->data = *data*;

    a->next = nullptr;

    create\_node\_by\_number(a, *number*);

}

*void* print() {

    if (size == 0) {

        cout << "Список порожній" << endl;

        return;

    }

    Node\* current = head;

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

        cout << current->data << " ";

        current = current->next;

    }

    cout << endl;

}

*void* print\_without\_space() {

    if (size == 0) {

        cout << "Список порожній" << endl;

        return;

    }

    Node\* current = head;

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

        cout << current->data;

        current = current->next;

    }

    cout << endl;

}

*bool* compare(List *list1*, List *list2*) {

    Node\* start1 = *list1*.head;

    Node\* start2 = *list2*.head;

    if (*list1*.size != *list2*.size) return false;

    for (*int* i = 0; i < *list1*.size; i++) {

        if (*list1*.head->data != *list2*.head->data) return false;

    }

    return true;

}

List addition(List *list3*, List *list4*, List *list5*) {

    if (*list3*.size < *list4*.size) {

        for (*int* i = *list3*.size; i < *list4*.size; i++) {

*list3*.create\_node\_by\_number(0, i+1);

        }

    }

    else if (*list4*.size < *list3*.size) {

        for (*int* i = *list4*.size; i < *list3*.size; i++) {

*list4*.create\_node\_by\_number(0, i+1);

        }

    }

    Node\* start1 = *list3*.head;

    Node\* start2 = *list4*.head;

*int* plus = 0;

*int* a;

    for (*int* i = 1; i <= *list3*.size; i++) {

        if ((start1->data + start2->data + plus)<10) {

*list5*.create\_node\_by\_number(start1->data + start2->data + plus, i);

            plus = 0;

        }

        else if ((start1->data + start2->data + plus)>=10) {

            a = (start1->data + start2->data + plus) % 10;

*list5*.create\_node\_by\_number(a, i);

            plus = 1;

        }

        start1 = start1->next;

        start2 = start2->next;

    }

    return *list5*;

}

};

*int* main() {

    List list1;

    list1.create\_node\_by\_number(1, 1);

    list1.create\_node\_by\_number(2, 2);

    list1.create\_node\_by\_number(3, 3);

    list1.create\_node\_by\_number(4, 4);

    list1.create\_node\_by\_number(5, 5);

    list1.create\_node\_by\_number(6, 6);

    list1.create\_node\_by\_number(7, 7);

    list1.create\_node\_by\_number(8, 8);

    cout << "Список з парною кількістю елементів: " << endl;

    list1.print();

    List list2;

    list2.create\_node\_by\_number(1, 1);

    list2.create\_node\_by\_number(2, 2);

    list2.create\_node\_by\_number(3, 3);

    list2.create\_node\_by\_number(4, 4);

    list2.create\_node\_by\_number(5, 5);

    list2.create\_node\_by\_number(6, 6);

    list2.create\_node\_by\_number(7, 7);

    // list2.create\_node\_by\_number(8, 8);

    //////////////////////  R E V E R S E  //////////////////////

    cout << "Список з непарною кількістю елементів: " << endl;

    list2.print();

    list1.reverse();

    cout << "Обернений список з парною кількістю елементів: " << endl;

    list1.print();

    list2.reverse();

    cout << "Обернений список з парною кількістю елементів: " << endl;

    list2.print();

    //////////////////////  C O M P A R E  //////////////////////

    cout << boolalpha << "Списки є однаковими - " << list1.compare(list1, list2) << endl;

    //////////////////////  B I G   N U M B E R S  //////////////////////

    List list3;

    List list4;

    List list5;

    list3.create\_node\_by\_number(6, 1);

    list3.create\_node\_by\_number(8, 2);

    list3.create\_node\_by\_number(3, 3);

    list3.create\_node\_by\_number(5, 4);

    list3.create\_node\_by\_number(1, 5);

    list3.create\_node\_by\_number(2, 6);

    list3.create\_node\_by\_number(1, 7);

    cout << "Перше велике число: " << endl;

    list3.print\_without\_space();

    list4.create\_node\_by\_number(3, 1);

    list4.create\_node\_by\_number(9, 2);

    list4.create\_node\_by\_number(3, 3);

    list4.create\_node\_by\_number(5, 4);

    list4.create\_node\_by\_number(6, 5);

    list4.create\_node\_by\_number(2, 6);

    list4.create\_node\_by\_number(3, 7);

    list4.create\_node\_by\_number(1, 7);

    cout << "Друге велике число: " << endl;

    list4.print\_without\_space();

    list5 = list5.addition(list3, list4, list5);

    cout << "Сума двох чисел: " << endl;

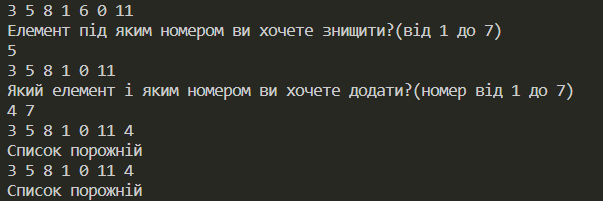
    list5.print\_without\_space();

    return 0;

}

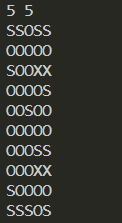
## **5. Результати виконання завдань, тестування та фактично затрачений час:**

Завдання №1 Lab# programming: VNS Lab 10



Час затрачений на виконання завдання: 5 годин

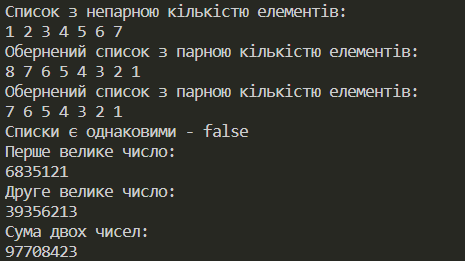
Завдання №2 Lab# programming: Algotester Lab 5



Підпис та № до блоку з виконанням та тестуванням програми

Час затрачений на виконання завдання: 50 хвилин

Завдання №1 №3 Practice# programming: Class Practice Task



Підпис та № до блоку з виконанням та тестуванням програми

Час затрачений на виконання завдання: 4 години.

# **Висновки:**

За цей епік я навчилася працювати з деякими динамічними структурами, зрозуміла для чого вони зроблені та написала код з використанням цих структур.