Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра інформаційних систем та мереж

ЗВІТ

про виконання лабораторної роботи № 5

з дисципліни Алгоритми та структури даних

Варіант 18

Виконав студент групи СА-32

Маленчак Валентин Богданович

Прийняв доцент Щербак С.С.

*Львів 2017*

**Тема:** робота з динамічними структурами.

**Мета роботи:** набуття практичних навичок опрацювання таких динамічних структур як зв'язні списки і дерева.

**Завдання на роботу:** Розробити програми, які виконують операції вказані в індивідуальному завданні.

* Програму для роботи з двонапрааленими звязними списками. Кожен елемент списку містить зсилки на наступний і попередній елемент в списку. Програма повинна забезпечувати ввід і побудову списку.
* Програму для роботи для роботи з деревами. Кожен елемент дерева містить зсилку на батьківський елемент і зсилки на елементи-нащадки (необмежена кількість). Програма повинна забезпечувати ввід і побудову дерева.
* Кожен елемент списку містить інформаційне поле(атрибут) деякого простого типу: символ, стрічка, число.
* Всі операції над динамічними структурами повинні супроводжуватись відповідним виводом на екран.
* В контрольних прикладах забезпечити опрацювання структур з 1000 елементами. Вихідні дані завантажуйте через XML-файл.

**Індивідуальне завдання:**

1) Розбиття списку на два списки за вказаним значенням інформаційного атрибуту елементу для розбиття;

2) Розбиття дерева на два дерева за вказаним значенням інформаційного атрибуту елементу для розбиття.

# Хід роботи

Текст програми на мові С++:

#pragma once

#include <iostream>

#include <string>

#include "UInterface.h"

using namespace std;

template <class T>

class myList

{

private:

struct node

{

T data;

node\* next;

node\* prev;

};

node\* head;

void destroy()

{

node\* p, \*p1;

p = head;

p1 = p->next;

while (p1 != NULL)

{

p = p1;

p1 = p1->next;

delete p;

}

}

public:

myList()

{

head = new node;

head->next = NULL;

}

~myList()

{

destroy();

delete head;

}

void push\_back(T element)

{

node\* p, \*p1;

p = head;

while (p->next != NULL)

{

p = p->next;

}

p1 = new node;

p1->data = element;

p1->next = NULL;

p1->prev = p;

p->next = p1;

}

void print(UInterface myInterface)

{

node\* p;

p = head->next;

while (p != NULL)

{

myInterface.outputData(p->data);

p = p->next;

}

}

void push\_front(T element)

{

node\* p;

p = new node;

p->data = element;

if (head->next == NULL)

{

p->next = head->next;

head->next = p;

p->prev = head;

}

else

{

p->next = head->next;

head->next->prev = p;

head->next = p;

p->prev = head;

}

}

void pop\_back()

{

node\* p, \*p1;

p = head;

p1 = p->next;

while (p1->next != NULL)

{

p = p1;

p1 = p1->next;

}

p->next = NULL;

delete p1;

}

void pop\_front()

{

node\* p;

p = head->next;

head->next = p->next;

p->next->prev = head;

delete p;

}

void divideByTwo(T data, myList& a, myList& b)

{

node\* p;

p = head;

while (p->next != NULL)

{

p = p->next;

if (p->data == data)

break;

a.push\_back(p->data);

}

p = p->prev;

while (p->next != NULL)

{

p = p->next;

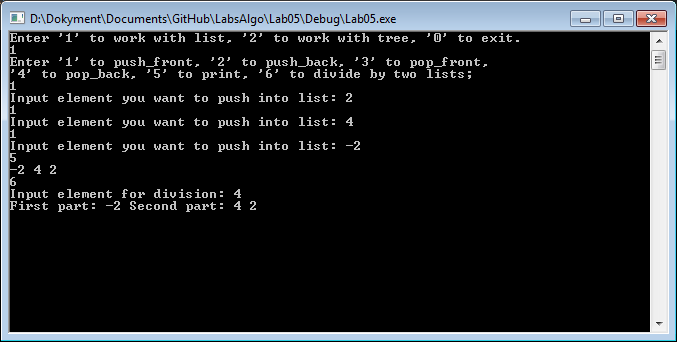
b.push\_back(p->data);

}

}

};

Результати комп'ютерної реалізації:



**Висновок:** я набув практичних навичок роботи з такими динамічними структурами даних як список та дерево.