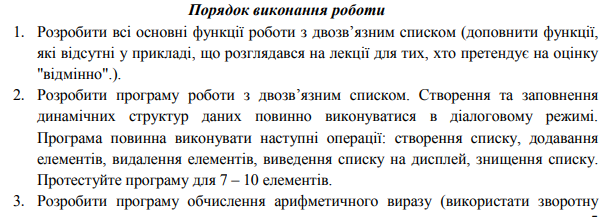
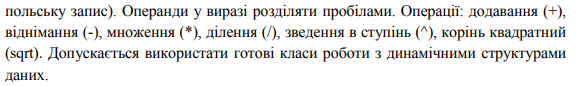
**Лабораторна робота № 4**

ЗВ’ЯЗНИЙ СПИСОК, СТЕК, ЧЕРГА. ЗВОРОТНІЙ ПОЛЬСЬКИЙ ЗАПИС

***Мета*** : ознайомитися з основами роботи з двозв’язним списком, однозв’язним списком, стеком та чергою. Розробити основні функції для обчислення арифметичного виразу, записаного з використанням зворотного польського запису.

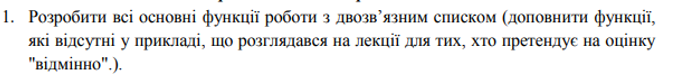
**4.1 Хід роботи**

****

****

4.1.1

**Завдання 1**:



Лістинг:

#include <stdio.h>

#include <stdlib.h>

typedef int elemtype;

struct elem {

elemtype value;

struct elem\* next;

struct elem\* prev;

};

struct myList {

struct elem\* head;

struct elem\* tail;

int size;

};

typedef struct elem cNode;

typedef struct myList cList;

cList\* createList(void) {

cList\* list = (cList\*)malloc(sizeof(cList));

if (list) {

list->size = 0;

list->head = list->tail = NULL;

}

return list;

}

bool isEmptyList(cList\* list) {

return ((list->head == NULL) || (list->tail == NULL));

}

cNode\* getNode(cList\* list, int index) {

cNode\* node = NULL;

int i;

if (index >= list->size) {

return (NULL);

}

if (index < list->size / 2) {

i = 0;

node = list->head;

while (node && i < index) {

node = node->next;

i++;

}

}

else {

i = list->size - 1;

node = list->tail;

while (node && i > index) {

node = node->prev;

i--;

}

}

return node;

}

int pushFront(cList\* list, elemtype\* data) {

cNode\* node = (cNode\*)malloc(sizeof(cNode));

if (!node) {

return(-1);

}

node->value = \*data;

node->next = list->head;

node->prev = NULL;

if (!isEmptyList(list)) {

list->head->prev = node;

}

else {

list->tail = node;

}

list->head = node;

list->size++;

return(0);

}

int pushBack(cList\* list, elemtype\* data) {

cNode\* node = (cNode\*)malloc(sizeof(cNode));

if (!node) {

return(-3);

}

node->value = \*data;

node->next = NULL;

node->prev = list->tail;

if (!isEmptyList(list)) {

list->tail->next = node;

}

else {

list->head = node;

}

list->tail = node;

list->size++;

return(0);

}

int pushPosition(cList\* list, elemtype\* data)

{

int index = (int)(list->size / 2);

if (index == 0)

{

return pushFront(list, data);

}

if (index<0 || index>list->size - 1)

{

return -1;

}

cNode\* next = getNode(list, index - 1)->next;

cNode\* prev = getNode(list, index)->prev;

cNode\* node = (cNode\*)malloc(sizeof(cNode));

node->value = \*data;

getNode(list, index - 1)->next = node;

getNode(list, index)->prev = node;

node->next = next;

node->prev = prev;

list->size++;

}

int popFront(cList\* list) {

cNode\* node;

if (isEmptyList(list)) {

return(-2);

}

node = list->head;

list->head = list->head->next;

if (!isEmptyList(list)) {

list->head->prev = NULL;

}

else {

list->tail = NULL;

}

list->size--;

free(node);

return(0);

}

int popBack(cList\* list) {

cNode\* node = NULL;

if (isEmptyList(list)) {

return(-4);

}

node = list->tail;

list->tail = list->tail->prev;

if (!isEmptyList(list)) {

list->tail->next = NULL;

}

else {

list->head = NULL;

}

list->size--;

free(node);

return(0);

}

int popPosition(cList\* list)

{

int index = (int)(list->size / 2);

if (index == 0)

{

elemtype tmp;

return popFront(list);

}

if (index == list->size - 1)

{

elemtype tmp;

return popBack(list);

}

if (index<0 || index>list->size - 1)

{

return -1;

}

cNode\* next = getNode(list, index)->next;

cNode\* prev = getNode(list, index)->prev;

free(getNode(list, index));

getNode(list, index - 1)->next = next;

getNode(list, index + 1)->prev = prev;

list->size--;

return 0;

}

void printList(myList\* numbers) {

if(!isEmptyList(numbers)){

cNode\* node = numbers->head;

for(int i = 1; i <= numbers->size; i++){

printf("\n%i. %i", i, node->value);

node = node->next;

}

} else {

printf("Список пустий");

}

}

int main()

{

system("chcp 1251");

system("cls");

createList();

int x = 0;

elemtype first = 1;

elemtype second = 2;

elemtype third = 3;

elemtype tmp;

cList\* list = createList();

do {

printList(list);

printf("\n\n1)додавання елемента на початої списку (1)\n");

printf("2)додавання елемента в середину списку (2)\n");

printf("3)додавання елемента в кінець списку (3)\n");

printf("4)видалення елемента з початку списку\n");

printf("5)видалення елемента з середини списку\n");

printf("6)видалення елемента з кінця списку\n");

printf("Введіть дію: ");

scanf\_s("%d", &x);

switch (x)

{

case 1:

pushFront(list, &first);

break;

case 2:

pushPosition(list, &second);

break;

case 3:

pushBack(list, &third);

break;

case 4:

popFront(list);

break;

case 5:

popPosition(list);

break;

case 6:

popBack(list);

break;

default:

x = -1;

break;

}

} while (x != -1);

return 0;

}

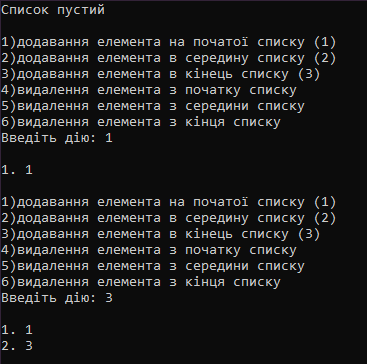


Рисунок 4.1 – Результат виконання завдання 1

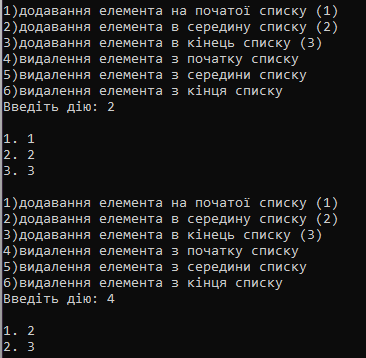


Рисунок 4.2 – Результат виконання завдання 1

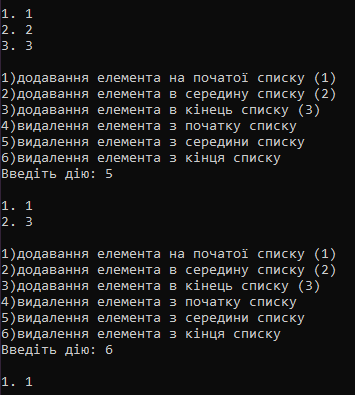
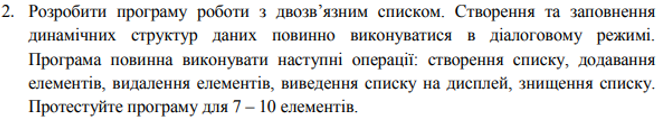


Рисунок 4.3 – Результат виконання завдання 1

**Завдання 2**:



Лістинг:

System.Globalization.CultureInfo customCulture = (System.Globalization.CultureInfo)

System.Threading.Thread.CurrentThread.CurrentCulture.Clone();

customCulture.NumberFormat.NumberDecimalSeparator = ".";

System.Threading.Thread.CurrentThread.CurrentCulture = customCulture;

Console.OutputEncoding = System.Text.Encoding.Default;

bool n0;

int x;

int n;

LinkedList<int> list = new LinkedList<int>();

Console.WriteLine("Введіть кількість елементів списку:");

do

{

n0 = true;

if (int.TryParse(Console.ReadLine(), out n) && n > 0)

{

n0 = false;

}

else

{

Console.WriteLine("Введіть ще раз");

}

} while (n0);

for (int i = 1; i <= n; i++)

{

list.AddLast(i);

}

do

{

Console.WriteLine();

foreach (int i in list)

{

Console.WriteLine(i);

}

Console.WriteLine("\n\n1)додати елемент\n2)видалити елемент списку\n3)видалити список");

do

{

n0 = true;

if (int.TryParse(Console.ReadLine(), out x))

{

n0 = false;

}

else

{

Console.WriteLine("Введіть ще раз");

}

} while (n0);

switch (x)

{

case 1:

Console.WriteLine("Введіть елемент після якого додати новий");

int k;

do

{

n0 = true;

if (int.TryParse(Console.ReadLine(), out k))

{

foreach (int i in list)

{

if (k == i)

{

n0 = false;

break;

}

}

}

else

{

Console.WriteLine("Введіть ще раз");

}

} while (n0);

Console.WriteLine("Введіть новий елемент");

int g;

do

{

n0 = true;

if (int.TryParse(Console.ReadLine(), out g))

{

n0 = false;

}

else

{

Console.WriteLine("Введіть ще раз");

}

} while (n0);

LinkedListNode<int> lln = list.Find(k);

list.AddAfter(lln, g);

break;

case 2:

Console.WriteLine("Введіть елемент який хочете видалити:");

int l;

do

{

n0 = true;

if (int.TryParse(Console.ReadLine(), out l))

{

n0 = false;

}

else

{

Console.WriteLine("Введіть ще раз");

}

} while (n0);

LinkedListNode<int> lll = list.Find(l);

list.Remove(lll);

break;

case 3:

list.Clear();

list.AddLast(0);

break;

default:

x = 0;

break;

}

} while (x != 0);

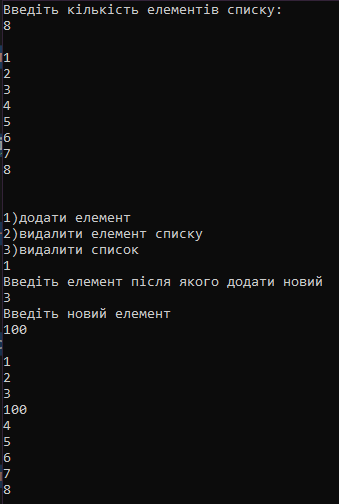


Рисунок 4.4 – Результат виконання завдання 2

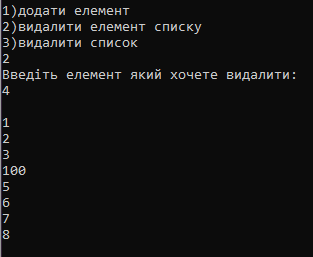


Рисунок 4.5 – Результат виконання завдання 2

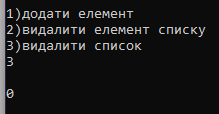
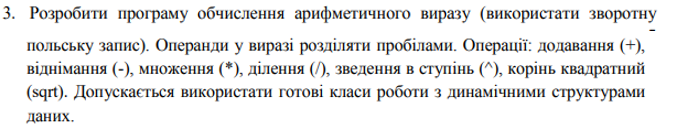


Рисунок 4.6 – Результат виконання завдання 2

**Завдання 3**:



Лістинг:

class Program

{

static int Priority(string symbol)

{

switch (symbol)

{

case ")": return 0;

case "(": return 1;

case "-": return 2;

case "+": return 2;

case "/": return 3;

case "\*": return 3;

case "^": return 4;

case "sqrt": return 5;

default: return 0;

}

}

static List<string> Station(string expression)

{

Stack<string> stack = new();

List<string> result = new();

string[] symbolsArr = expression.Split(" ");

for (int i = 0; i < symbolsArr.Length; i++)

{

if (float.TryParse(symbolsArr[i], out float num))

{

result.Add(symbolsArr[i]);

}

else if (symbolsArr[i] == "(")

{

stack.Push(symbolsArr[i]);

}

else if (symbolsArr[i] == ")")

{

string symbol = stack.Pop();

while (symbol != "(")

{

result.Add(symbol);

symbol = stack.Pop();

}

}

else

{

if (stack.Count > 0)

{

if (Priority(symbolsArr[i]) <= Priority(stack.Peek()))

{

result.Add(stack.Pop());

}

}

stack.Push(symbolsArr[i]);

}

Console.Write("Рядок: ");

for (int j = 0; j < result.Count; j++)

{

Console.Write($"{result[j]} ");

}

Console.Write("\nСтек: ");

foreach (Object obj in stack)

{

Console.Write($"{obj} ");

}

Console.Write("\n\n");

}

while (stack.Count > 0)

{

result.Add(stack.Pop());

}

return result;

}

static void Main()

{

Console.OutputEncoding = System.Text.Encoding.Default;

Stack<float> stack = new();

Console.Write("Введіть вираз: ");

string str = Console.ReadLine();

List<string> expressionList = Station(str);

float num1, num2;

Console.Write($"Зворотній польський запис: ");

for (int i = 0; i < expressionList.Count; i++)

{

Console.Write($"{expressionList[i]} ");

}

Console.Write("\n");

for (int i = 0; i < expressionList.Count; i++)

{

if (float.TryParse(expressionList[i], out float num))

{

stack.Push(num);

}

else

{

switch (expressionList[i])

{

case "+":

{

num1 = stack.Pop();

num2 = stack.Pop();

stack.Push(num2 + num1);

break;

}

case "-":

{

num1 = stack.Pop();

num2 = stack.Pop();

stack.Push(num2 - num1);

break;

}

case "\*":

{

num1 = stack.Pop();

num2 = stack.Pop();

stack.Push(num2 \* num1);

break;

}

case "/":

{

num1 = stack.Pop();

num2 = stack.Pop();

stack.Push(num2 / num1);

break;

}

case "^":

{

num1 = stack.Pop();

num2 = stack.Pop();

stack.Push((float)Math.Pow(num2, num1));

break;

}

case "sqrt":

{

stack.Push((float)Math.Sqrt(stack.Pop()));

break;

}

default: break;

}

}

}

Console.WriteLine($"\nОбчислення {str} = {stack.Peek()}\n");

}

}

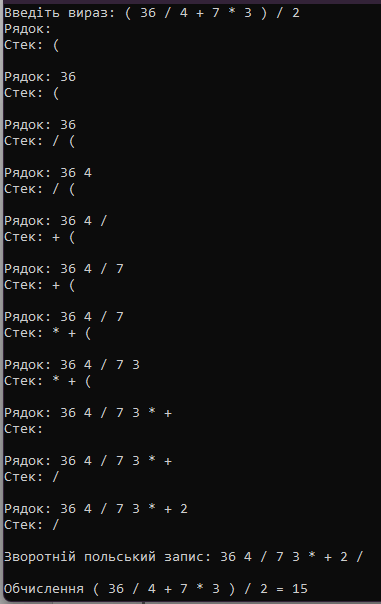


Рисунок 4.7 – Результат виконання завдання 3

***Висновки:*** я ознайомитися з основами роботи з двозв’язним списком, однозв’язним списком, стеком та чергою. Розробив основні функції для обчислення арифметичного виразу, записаного з використанням зворотного польського запису.