

ЖИТОМИРСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ
Факультет інформаційно-комп'ютерних технологій
Кафедра інженерії програмного забезпечення

ЗВІТ
З ЛАБОРАТОРНИХ РОБІТ
з «Алгоритми та структури даних»
(назва дисципліни)

Студента 1 курсу ВТ-21-1 групи
напряму підготовки Веб-технології
спеціальності 121 «Інженерія
програмного забезпечення»
Маньківського Владислава Вячеславовича
(прізвище та ініціали)
Керівник Старший викладач кафедри
ІПЗ Локтікова Т. М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала _____
Кількість балів: _____ Оцінка: ECTS _____

Зміст

ЗВІТ	1
Лабораторна робота № 1	1
Лабораторна робота № 2	1
Лабораторна робота № 3	1
Лабораторна робота № 4	1
Лабораторна робота № 5	1
Лабораторна робота № 6	1
Лабораторна робота № 7-8.....	1

Лабораторна робота № 1

РОБОТА З БАЗОВИМИ ТИПАМИ ДАНИХ

Мета : отримати практичні навички по роботі з базовими типами даних (простими і складними типами даних).

1.1 Хід роботи

Порядок виконання роботи

1. Записати і заповнити структуру даних зберігання поточного часу (включаючи секунди) і дату в найбільш компактному вигляді. Визначити обсяг пам'яті, яку займає змінна даного типу. Порівняти зі стандартною структурою *tm (time.h)*. Вивести вміст структури в зручному вигляді для користувача на дисплей..
2. Реалізувати введення цілочисельного значення типу *signed short*. Визначити знак і значення, використовуючи: 1) структури даних та об'єднання; 2) побітові логічні операції.
3. Виконати операції:
а) $5 + 127$; б) $2-3$; в) $-120-34$; г) (unsigned char) (-5) ; д) $56 \& 38$; е) $56 | 38$.

Всі значення (константи) повинні зберігатися в змінних типу *signed char*. Виконати перевірку результату в ручну. Пояснити результат, використовуючи двійкову систему числення.

4. Записати і заповнити структуру даних (об'єднання) для зберігання дійсного числа типу *float* в найбільш компактному вигляді. Реалізувати відображення на дисплей: 1) значення побитово; 2) значення побайтово; 3) знака, мантиси і ступінь значення. Виконати перевірку результату в ручну. Визначити обсяг пам'яті, яку займає змінна користувацького типу.

1.1.1

Завдання 1:

1. Записати і заповнити структуру даних зберігання поточного часу (включаючи секунди) і дату в найбільш компактному вигляді. Визначити обсяг пам'яті, яку займає змінна даного типу. Порівняти зі стандартною структурою *tm (time.h)*. Вивести вміст структури в зручному вигляді для користувача на дисплей..

Лістинг:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <windows.h>
#include <stdlib.h>
#include <time.h>
```

					ДУ«Житомирська політехніка».21.121.02.000–Лр 1			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Маньківський В.В			Звіт з лабораторної роботи		Літ.	Арк.
Перевір.		Локтікова Т.М.						1
Керівник								8
Н. контр.							ФІКТ Гр. ВТ-21-1[2]	
Зав. каф.								

```

struct date {
    unsigned short Year : 12;
    unsigned short Month : 4;
    unsigned short Day : 5;
    unsigned short Hour : 5;
    unsigned short Min : 6;
    unsigned short Sec : 6;
};
enum dow {
    Mon = 1,
    Tue,
    Wed,
    Thu,
    Fri,
    Sat,
    Sun
};
int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    time_t now = time(0);
    struct date data;
    int y, m, d, h, mm, s;
    char k[20];
    printf("Введіть номер дня тижня ");

    dow doo;
    scanf_s("%d", &doo);
    switch (doo) {
        case Mon:
            strcpy_s(k, "Понеділок");
            break;
        case Tue:
            strcpy_s(k, "Вівторок");
            break;
        case Wed:
            strcpy_s(k, "Середа");
            break;
        case Thu:
            strcpy_s(k, "Четвер");
            break;
        case Fri:
            strcpy_s(k, "П'ятниця");
            break;
        case Sat:
            strcpy_s(k, "Субота");
            break;
        case Sun:
            strcpy_s(k, "Неділя");
            break;
        default: break;
    }
    printf("Введіть рік:");
    scanf_s("%d", &y);
    printf("Введіть місяць:");
    scanf_s("%d", &m);
    printf("Введіть день:");
    scanf_s("%d", &d);
    printf("Введіть годину:");
    scanf_s("%d", &h);
    printf("Введіть хвилини:");
    scanf_s("%d", &mm);
    printf("Введіть секунди:");
    scanf_s("%d", &s);
    data.Year = y;

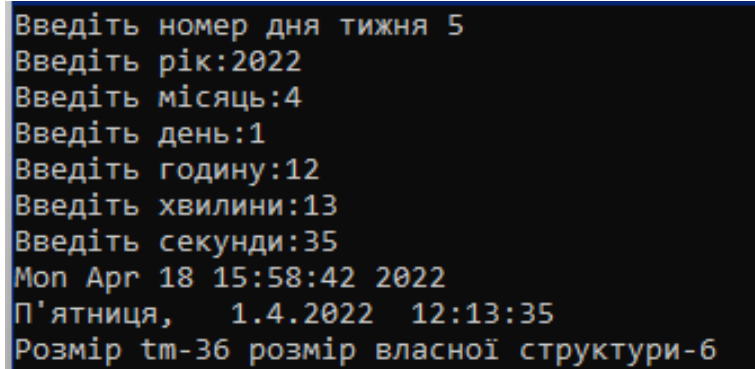
```

		Маньківський В.М			ДУ«Житомирська політехніка».21.121.02.000–Лр 1	Арк.
		Локтікова Т.М.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

```

data.Month = m;
data.Day = d;
data.Hour = h;
data.Min = mm;
data.Sec = s;
time_t rawtime;
struct tm* timeinfo;
time(&rawtime);
timeinfo = localtime(&rawtime);
printf("%s", asctime(timeinfo));
printf("%s, %d.%d.%d %d:%d:%d", k, data.Day, data.Month, data.Year, data.Hour,
data.Min, data.Sec);
int n1, n2;
n1 = sizeof(tm);
n2 = sizeof(date);
printf("\nРозмір tm-%d розмір власної структури-%d", n1, n2);
return 0;
}

```



```

Введіть номер дня тижня 5
Введіть рік:2022
Введіть місяць:4
Введіть день:1
Введіть годину:12
Введіть хвилини:13
Введіть секунди:35
Mon Apr 18 15:58:42 2022
П'ятниця, 1.4.2022 12:13:35
Розмір tm-36 розмір власної структури-6

```

Рисунок 1.1 – Результат виконання завдання 1

1.1.2

Завдання 2:

2. Реалізувати введення цілочисельного значення типу *signed short*. Визначити знак і значення, використовуючи: 1) структури даних та об'єднання; 2) побітові логічні операції.

Лістинг:

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <math.h>
#include <conio.h>

union binary {
    signed short number_binary;
    struct {
        unsigned char zero : 1;
        unsigned char one : 1;
        unsigned char two : 1;
        unsigned char three : 1;
        unsigned char four : 1;
        unsigned char five : 1;
        unsigned char six : 1;
        unsigned char seven : 1;
    };
};

```

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Лр 1	Арк.
		Локтікова Т.М.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

```

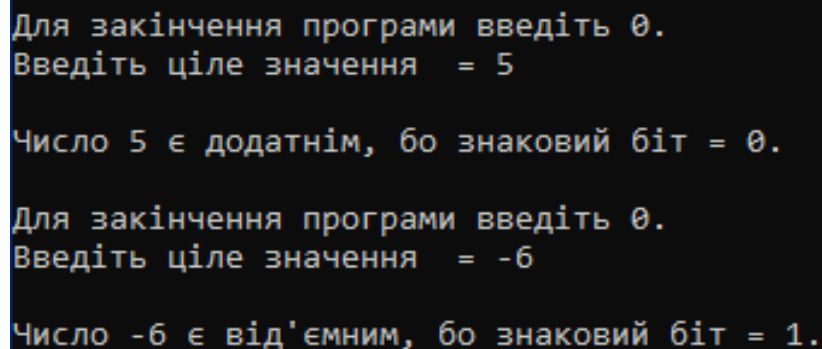
        unsigned char eight : 1;
        unsigned char nine : 1;
        unsigned char ten : 1;
        unsigned char eleven : 1;
        unsigned char twelve : 1;
        unsigned char thirteen : 1;
        unsigned char fourteen : 1;
        unsigned char fiveteen : 1;
    }bytes;
};

int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    signed short num;
    binary binary;
    do {
        printf("\n\nДля закінчення програми введіть 0.\nВведіть ціле значення = ");
        scanf("%d", &num);

        binary.number_binary = num;

        if (binary.bytes.fiveteen == 1) {
            printf("\nЧисло %d є від'ємним, бо знаковий біт = %d.",
binary.number_binary, binary.bytes.fiveteen);
        }
        else {
            printf("\nЧисло %d є додатнім, бо знаковий біт = %d.",
binary.number_binary, binary.bytes.fiveteen);
        }
    } while (num != 0);
    return 0;
}

```



```

Для закінчення програми введіть 0.
Введіть ціле значення = 5

Число 5 є додатнім, бо знаковий біт = 0.

Для закінчення програми введіть 0.
Введіть ціле значення = -6

Число -6 є від'ємним, бо знаковий біт = 1.

```

Рисунок 1.2 – Результат виконання завдання 2

1.1.2

Завдання 3:

Лістинг:

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Лр 1	Арк.
		Локтікова Т.М.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

3. Виконати операції:

а) $5 + 127$; б) $2 - 3$; в) $-120 - 34$; г) (unsigned char) (-5) ; д) $56 \& 38$; е) $56 | 38$.

Всі значення (константи) повинні зберігатися в змінних типу *signed char*. Виконати перевірку результату в ручну. Пояснити результат, використовуючи двійкову систему числення.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <math.h>
#include <Windows.h>
#include <string.h>
#include <conio.h>
#include <time.h>

struct ch {
    signed char a = 5;
    signed char b = 127;
    signed char c = 2;
    signed char d = -3;
    signed char f = -120;
    signed char e = -34;
    signed char g = -5;
    signed char l = 56;
    signed char k = 38;
    signed char rez;
}x;
int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    x.rez = x.a + x.b;
    printf("5+127=%d", x.rez);
    printf("\nВиходить за межі значень типу signed char");
    x.rez = 0;
    x.rez = x.c + x.d;
    printf("\n\n2-3=%d", x.rez);
    printf("\nНе виходить за межі значень типу signed char");
    x.rez = 0;
    x.rez = x.f + x.e;
    printf("\n\n-120-34=%d", x.rez);
    printf("\nВиходить за межі значень типу signed char");
    x.rez = 0;
    x.rez = (unsigned char)(x.g);
    printf("\n\nunsigned char -5=%d", x.rez);
    printf("\nНічого не відбувається");
    x.rez = 0;
    x.rez = x.l & x.k;
    printf("\n\n56 & 38=%d", x.rez);
    printf("\nЛогічна дія AND");
    x.rez = 0;
    x.rez = x.l | x.k;
    printf("\n\n56 | 38=%d", x.rez);
    printf("\nЛогічна дія OR\n");

    return 0;
}
```

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Пр 1	Арк.
		Локтікова Т.М.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

```

5+127=-124
Виходить за межі значень типу signed char

2-3=-1
Не виходить за межі значень типу signed char

-120-34=102
Виходить за межі значень типу signed char

unsigned char -5=-5
Нічого не відбувається

56 & 38=32
Логічна дія AND

56 | 38=62
Логічна дія OR

```

Рисунок 1.3 – Результат виконання завдання 3

1.1.4

Завдання 4:

4. Записати і заповнити структуру даних (об'єднання) для зберігання дійсного числа типу *float* в найбільш компактному вигляді. Реалізувати відображення на дисплей: 1) значення побитово; 2) значення побайтово; 3) знака, мантиї і ступінь значення. Виконати перевірку результату в ручну. Визначити обсяг пам'яті, яку займає змінна користувацького типу.

Лістинг:

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <math.h>
#include <Windows.h>
#include <string.h>
#include <conio.h>
#include <time.h>

union radiance {
    float number;
    struct {
        unsigned char zero : 1;
        unsigned char one : 1;
        unsigned char two : 1;
        unsigned char three : 1;
        unsigned char four : 1;
        unsigned char five : 1;
        unsigned char six : 1;
        unsigned char seven : 1;
        unsigned char eight : 1;
        unsigned char nine : 1;
        unsigned char ten : 1;
        unsigned char eleven : 1;
        unsigned char twelve : 1;
        unsigned char thirteen : 1;
    };
};

```

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Пр 1	Арк.
		Локтікова Т.М.				6
Змн.	Арк.	№ докум.	Підпис	Дата		


```

    unsigned char fourteen : 1;
    unsigned char fifteen : 1;
    unsigned char sixteen : 1;
    unsigned char seventeen : 1;
    unsigned char eighteen : 1;
    unsigned char nineteen : 1;
    unsigned char twenty : 1;
    unsigned char twenty_one : 1;
    unsigned char twenty_two : 1;
    unsigned char twenty_three : 1;
    unsigned char twenty_four : 1;
    unsigned char twenty_five : 1;
    unsigned char twenty_six : 1;
    unsigned char twenty_seven : 1;
    unsigned char twenty_eight : 1;
    unsigned char twenty_nine : 1;
    unsigned char thirty : 1;
    unsigned char thirty_one : 1;
}bytes;
}radiancе;
int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    printf("Введіть змінну = "); scanf("%f", &radiancе.number);
    printf("\nЗначення побитово:\n");
    printf("|%d|%d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d | \n", radiancе.bytes.thirty_one,
radiancе.bytes.thirty,radiancе.bytes.twenty_nine,
radiancе.bytes.twenty_eight,radiancе.bytes.twenty_seven,
radiancе.bytes.twenty_six,radiancе.bytes.twenty_five,
radiancе.bytes.twenty_four,radiancе.bytes.twenty_three,
radiancе.bytes.twenty_two,radiancе.bytes.twenty_one, radiancе.bytes.twenty,
radiancе.bytes.nineteen,radiancе.bytes.eighteen, radiancе.bytes.seventeen,
radiancе.bytes.sixteen,radiancе.bytes.fiveteen, radiancе.bytes.fourteen,
radiancе.bytes.thirteen,radiancе.bytes.twelve, radiancе.bytes.eleven,
radiancе.bytes.ten,radiancе.bytes.nine, radiancе.bytes.eight,
radiancе.bytes.seven,radiancе.bytes.six, radiancе.bytes.five,
radiancе.bytes.four,radiancе.bytes.three, radiancе.bytes.two,
radiancе.bytes.one,radiancе.bytes.zero);
printf(" |                                     |\n");
    printf("знак          ступінь                                мантиса\n");
    printf("\nЗначення побайтово:\n");
    printf("|%d %d %d %d %d %d %d %d|%d %d %d %d %d %d %d %d|%d %d %d %d %d %d %d %d | %
d %d %d %d %d %d %d %d | \n", radiancе.bytes.thirty_one,
radiancе.bytes.thirty,radiancе.bytes.twenty_nine,
radiancе.bytes.twenty_eight,radiancе.bytes.twenty_seven,
radiancе.bytes.twenty_six,radiancе.bytes.twenty_five,
radiancе.bytes.twenty_four,radiancе.bytes.twenty_three,
radiancе.bytes.twenty_two,radiancе.bytes.twenty_one, radiancе.bytes.twenty,
radiancе.bytes.nineteen,radiancе.bytes.eighteen, radiancе.bytes.seventeen,
radiancе.bytes.sixteen,radiancе.bytes.fiveteen, radiancе.bytes.fourteen,
radiancе.bytes.thirteen,radiancе.bytes.twelve, radiancе.bytes.eleven,
radiancе.bytes.ten,radiancе.bytes.nine, radiancе.bytes.eight,
radiancе.bytes.seven,radiancе.bytes.six, radiancе.bytes.five,
radiancе.bytes.four,radiancе.bytes.three, radiancе.bytes.two,
radiancе.bytes.one,radiancе.bytes.zero);
float s;
    printf("\nОб'єм пам'яті змінної = %d;\nОб'єм пам'яті union = % d.", sizeof(s),
sizeof(radiancе));

return 0;
}
```

		Маньківський В.М			ДУ«Житомирська політехніка».21.121.02.000–Пр 1	Арк.
		Локтікова Т.М.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

Лабораторна робота № 2

ГЕНЕРУВАННЯ ПОСЛІДОВНОСТІ ПСЕВДОВИПАДКОВИХ ЗНАЧЕНЬ

Мета : ознайомитись з методами генерування випадкових чисел, а також формуванням та обробкою масивів даних.

2.1 Хід роботи

Порядок виконання роботи

Розробити програму * генерування цілочислової послідовності псевдовипадкових значень (за допомогою конгруентного методу*) та виконати обробку отриманого масиву даних наступним чином:

- розрахувати частоту інтервалів появи випадкових величин (інтервал дорівнює 1);
- розрахувати статистичну імовірність появи випадкових величин;
- розрахувати математичне сподівання випадкових величин;
- розрахувати дисперсію випадкових величин;
- розрахувати середньоквадратичне відхилення випадкових величин.

Варіант	Коефіцієнти			Діапазон випадкових величин, n	Довжина послідовності чисел, K
	a	c	m		
2	1664525	1013904223	2^{32}	10. 150	20000

2.1.1

Завдання:

Лістинг:

```

int a = 1664525, c = 1013904223, n = 150, x = 0;
double k = 20000, m = 0, d = 0, q = 0;
int mod = (int)Math.Pow(2, 32);
int[] l = new int[n];
double[] p = new double[n];
int[] mass = new int[20000];

for (int i = 0; i < k ;i)
{
    x = (a * x + c) % mod;
    mass[i] = x % n;
    if (x >= 0)
    {
        Console.WriteLine($"{i+1} - {x % n}");
        i++;
    }
}
Console.WriteLine();

for (int i = 0; i < k; i++)
{
    l[mass[i]] = l[mass[i]] + 1;
}
    
```

					ДУ«Житомирська політехніка».21.121.02.000–Лр 2			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Маньківський В.В			Звіт з лабораторної роботи	Літ.	Арк.	Аркушів
Перевір.		Локтікова Т.М.					1	2
Керівник						ФІКТ Гр. ВТ-21-1[2]		
Н. контр.								
Зав. каф.								

```

for (int i = 0; i < k; i++)
{
    p[mass[i]] = l[mass[i]] / k;
}
for (int i = 0; i < n; i++)
    Console.WriteLine($"P({i}) = {p[i]} ({l[i]})");

for (int i = 0; i < n; i++)
{
    m = m + i * p[i];
}
Console.WriteLine($"M(X)={m:f2}");

for (int i = 0; i < n; i++)
{
    d = d + Math.Pow((i - m), 2) * p[i];
}
Console.WriteLine($"D(X)={d:f2}");

q = Math.Sqrt(d);
Console.WriteLine($"q(X)={q:f2}");

```

```

19974 - 32 P(0) = 0,00675 (135)
19975 - 12 P(1) = 0,0064 (128)
19976 - 49 P(2) = 0,007 (140)
19977 - 102 P(3) = 0,0065 (130)
19978 - 3 P(4) = 0,007 (140)
19979 - 110 P(5) = 0,0061 (122)
19980 - 105 P(6) = 0,00665 (133)
19981 - 52 P(7) = 0,0067 (134)
19982 - 73 P(8) = 0,0059 (118)
19983 - 40 P(9) = 0,00695 (139)
19984 - 77 P(10) = 0,0059 (118)
19985 - 13 P(11) = 0,00675 (135)
19986 - 12 P(12) = 0,0066 (132)
19987 - 136 P(13) = 0,0077 (154)
19988 - 103 P(14) = 0,00735 (147)
19989 - 108 P(15) = 0,0074 (148)
19990 - 119 P(16) = 0,0063 (126)
19991 - 113 P(17) = 0,00665 (133)
19992 - 99 P(18) = 0,0072 (144)
19993 - 18 P(19) = 0,00595 (119)
19994 - 81 P(20) = 0,00675 (135)
19995 - 78 P(21) = 0,0064 (128)
19996 - 85 P(22) = 0,00665 (133)
19997 - 70 P(23) = 0,0074 (148)
19998 - 116 P(24) = 0,00555 (111)
19999 - 139 P(25) = 0,0064 (128)
20000 - 85 P(26) = 0,00565 (113)
P(27) = 0,00615 (123)
P(28) = 0,0057 (114)

```

```

M(X)=74,86
D(X)=1875,60
q(X)=43,31

```

Рисунок 2.1 – Результат виконання завдання

Висновки: я ознайомився з методами генерування випадкових чисел, а також формуванням та обробкою масивів даних.

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Пр 2	Арк. 2
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Лабораторна робота № 3

ОЦІНКА ЧАСОВОЇ СКЛАДНОСТІ АЛГОРИТМІВ

Мета : набуття навичок дослідження часової складності алгоритмів і визначення її асимптотичних оцінок.

3.1 Хід роботи

Порядок виконання роботи

1. Написати програму для табулювання наступних функцій: $f(n)=n$; $f(n)=\log(n)$; $f(n)=n \cdot \log(n)$; $f(n)=n^2$; $f(n)=2^n$; $f(n)=n!$. Табулювання виконати на відрізку $[0, 50]$ з кроком 1. Побудувати графіки функцій (за допомогою Excel) в одній декартовій системі координат. Значення осі ординат обмежити величиною 500.
2. Напишіть програму згідно індивідуального завдання (таблиця 3.1 та таблиця 3.2). Виміряти час виконання функцій та побудувати графіки за допомогою Excel. Провести аналіз і оцінку часової складності алгоритмів. Порівняти практично отримані результати з оцінкою часової складності алгоритмів.

2	Дано вхідне ціле число a , де $0 \leq a \leq 20$. Реалізувати функцію за допомогою рекурсії знаходження факторіалу числа a .
5	Дан масив цифр вісімкової системи числення. Обсяг масиву $m \leq 20$. Реалізувати функцію, яка повертає найбільше можливе число з даних цифр. Цифри згенерувати генератором випадкових чисел.

3.1.1

Завдання 1:

Лістинг:

```
Console.WriteLine("f(n) = n");
Console.WriteLine("x\ty");
double f;
for(int n = 0; n <= 50; n++)
{
    f = n;
    Console.WriteLine($"{n}\t{f}");
}

Console.WriteLine("f(n) = lg(n)");
Console.WriteLine("x\ty");
for (int n = 0; n <= 50; n++)
{
    f = Math.Log10(n);
    Console.WriteLine($"{n}\t{Math.Round(f, 4)}");
}
```

					ДУ«Житомирська політехніка».21.121.02.000–Лр 3			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Маньківський В.В			Звіт з лабораторної роботи		Літ.	Арк.
Перевір.		Локтікова Т.М.						1
Керівник							ФІКТ Гр. ВТ-21-1[2]	
Н. контр.								
Зав. каф.								

```

Console.WriteLine("f(n) = n * lg(n)");
Console.WriteLine("x\ty");
for (int n = 0; n <= 50; n++)
{
    f = n * Math.Log10(n);
    Console.WriteLine($"{n}\t{Math.Round(f, 4)}");
}

```

```

Console.WriteLine("f(n) = n^2");
Console.WriteLine("x\ty");
for (int n = 0; n <= 50; n++)
{
    f = Math.Pow(n, 2);
    Console.WriteLine($"{n}\t{Math.Round(f, 4)}");
}

```

```

Console.WriteLine("f(n) = 2^n");
Console.WriteLine("x\ty");
for (int n = 0; n <= 50; n++)
{
    f = Math.Pow(2, n);
    Console.WriteLine($"{n}\t{Math.Round(f, 4)}");
}

```

```

Console.WriteLine("f(n) = n!");
Console.WriteLine("x\ty");
Console.WriteLine($"{0}\t{1}");
for (int n = 1; n <= 50; n++)
{
    f = 1;
    for (int k = 1; k <= n; k++)
    {
        f = f * k;
    }
    Console.WriteLine($"{n}\t{Math.Round(f, 4)}");
}

```

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Пр 3	Арк.
		Локтікова Т.М.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

f(n) = n		f(n) = lg(n)		f(n) = n * lg(n)		f(n) = n^2	
x	y	x	y	x	y	x	y
0	0	0	-?	0	не число	0	0
1	1	1	0	1	0	1	1
2	2	2	0,301	2	0,6021	2	4
3	3	3	0,4771	3	1,4314	3	9
4	4	4	0,6021	4	2,4082	4	16
5	5	5	0,699	5	3,4949	5	25
6	6	6	0,7782	6	4,6689	6	36
7	7	7	0,8451	7	5,9157	7	49
8	8	8	0,9031	8	7,2247	8	64
9	9	9	0,9542	9	8,5882	9	81
10	10	10	1	10	10	10	100
11	11	11	1,0414	11	11,4553	11	121
12	12	12	1,0792	12	12,9502	12	144
13	13	13	1,1139	13	14,4813	13	169
14	14	14	1,1461	14	16,0458	14	196
15	15	15	1,1761	15	17,6414	15	225
16	16	16	1,2041	16	19,2659	16	256
17	17	17	1,2304	17	20,9176	17	289
18	18	18	1,2553	18	22,5949	18	324
19	19	19	1,2788	19	24,2963	19	361
20	20	20	1,301	20	26,0206	20	400
21	21	21	1,3222	21	27,7666	21	441
22	22	22	1,3424	22	29,5333	22	484
23	23	23	1,3617	23	31,3197	23	529
24	24	24	1,3802	24	33,1251	24	576
25	25	25	1,3979	25	34,9485	25	625
26	26	26	1,415			26	676
27	27	27	1,4314			27	729

Рисунок 3.1 – Результат виконання завдання 1

$f(n) = 2^n$		$f(n) = n!$	
x	y	x	y
0	1	0	1
1	2	1	1
2	4	2	2
3	8	3	6
4	16	4	24
5	32	5	120
6	64	6	720
7	128	7	5040
8	256	8	40320
9	512	9	362880
10	1024	10	3628800
11	2048	11	39916800
12	4096	12	479001600
13	8192	13	6227020800
14	16384	14	87178291200
15	32768	15	1307674368000
16	65536	16	20922789888000
17	131072	17	355687428096000
18	262144	18	6402373705728000
19	524288	19	1,21645100408832E+17
20	1048576	20	2,43290200817664E+18
21	2097152	21	5,109094217170944E+19
22	4194304	22	1,1240007277776077E+21
23	8388608	23	2,585201673888498E+22
24	16777216	24	6,204484017332394E+23
25	33554432	25	1,5511210043330986E+25
26	67108864		
27	134217728		

Рисунок 3.2 – Результат виконання завдання 1

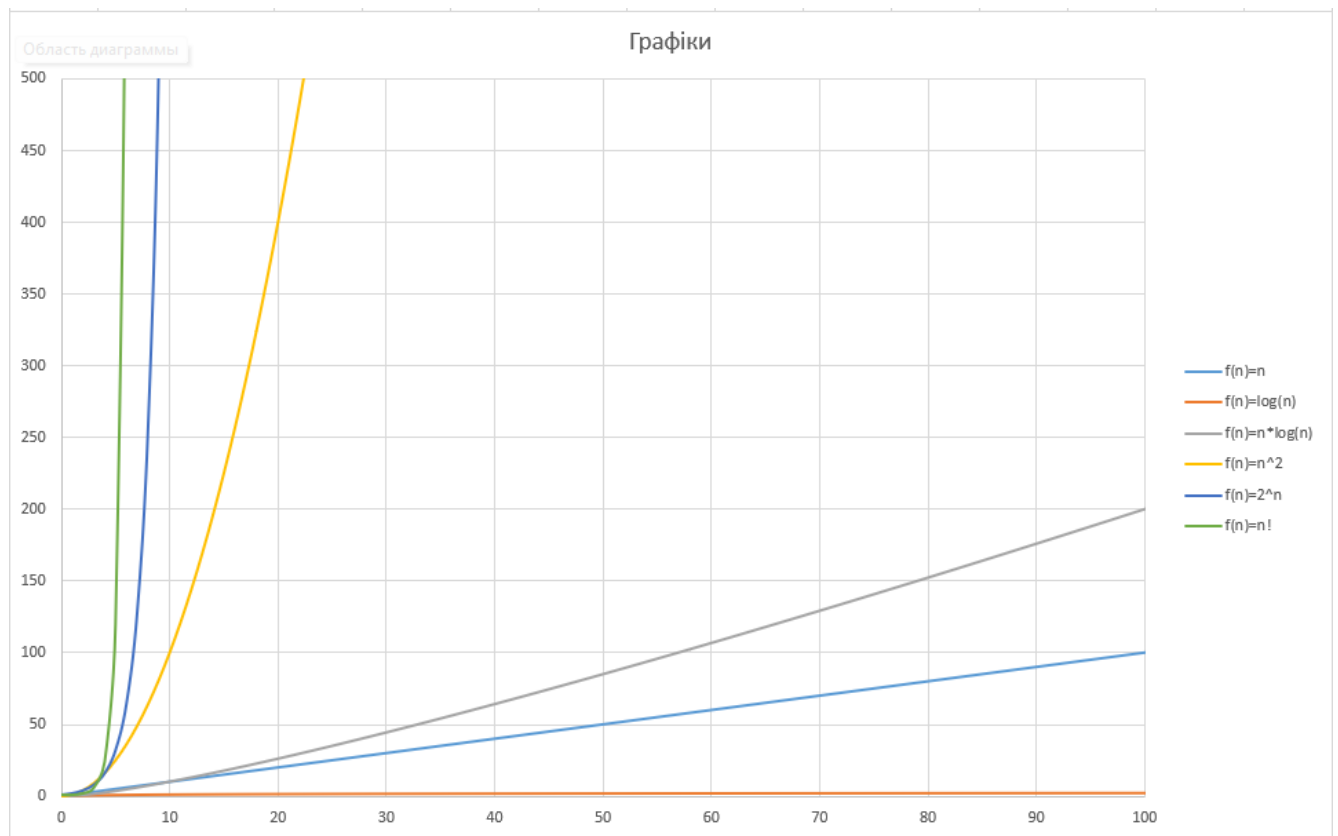


Рисунок 3.3 – Результат виконання завдання 1

Завдання 2.1:

2	Дано вхідне ціле число a , де $0 \leq a \leq 20$. Реалізувати функцію за допомогою рекурсії знаходження факторіалу числа a .
---	---

Лістинг:

```
using System;
using System.Text;
using System.Diagnostics;

class Program
{
    static void Main()
    {
        Console.OutputEncoding = System.Text.Encoding.Default;
        Stopwatch stopwatch = new Stopwatch();
        Console.WriteLine("Число a, де  $0 \leq a \leq 20$ ");
        bool n0;
        int a;
        do
        {
            n0 = true;
            if (int.TryParse(Console.ReadLine(), out a) && a >= 0 && a <= 20)
            {
                n0 = false;
            }
        }
        else
        {
            // Loop continues until valid input is received
        }
    }
}
```

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Пр 3	Арк.
		Локтікова Т.М.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

```

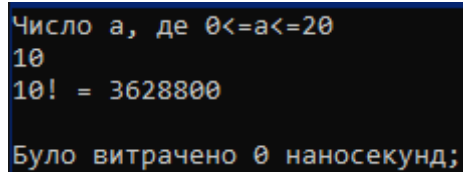
        Console.WriteLine("Введіть ще раз");
    }
} while (n0);
stopwatch.Start();
Console.WriteLine($"{a}! = {fact(a)}");

stopwatch.Stop();
Console.WriteLine($"{n}Було витрачено {stopwatch.ElapsedMilliseconds * 1000000}
наносекунд;");
}

static double fact(int a)
{
    if (a <= 1) return 1;

    return a * fact(a - 1);
}

```



```

Число a, де 0<=a<=20
10
10! = 3628800

Було витрачено 0 наносекунд;

```

Рисунок 3.4 – Результат виконання завдання 2.1

Завдання 2.2:

5	Дан масив цифр вісімкової системи числення. Обсяг масиву $m \leq 20$. Реалізувати функцію, яка повертає найбільше можливе число з даних цифр. Цифри згенерувати генератором випадкових чисел.
---	--

Лістинг:

```

using System;
using System.Text;
using System.Diagnostics;

class Program
{
    static void Main()
    {
        Console.OutputEncoding = System.Text.Encoding.Default;
        Stopwatch stopwatch = new Stopwatch();
        Console.WriteLine("Число m, де m<=20");
        bool n0;
        int m;
        double max = 0, zero = 1, maxn = 0;
        do
        {
            n0 = true;
            if (int.TryParse(Console.ReadLine(), out m) && m >= 0 && m <= 20)
            {
                n0 = false;
            }
            else
            {
                Console.WriteLine("Введіть ще раз");
            }
        } while (n0);
    }
}

```

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Пр 3	Арк.
		Локтікова Т.М.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

```

stopwatch.Start();
int []mass = new int[m];
Random rnd = new Random();
for (int i = 0; i < m; i++)
{
    mass[i] = rnd.Next(0, 8);
    Console.Write($"{mass[i]}\t");
    zero *= 10;
}
int s = 0;
for (int i = 0; i < m; i++)
{
    zero /= 10;
    maxn = 0;
    for (int j = 0; j < m; j++)
    {
        if (mass[j] > maxn && mass[j] != -1)
        {
            maxn = mass[j];
            s = j;
        }
    }
    max += maxn * zero;
    mass[s] = -1;
}
Console.WriteLine($"{n}\nНайбільш можливе -> {max}");
stopwatch.Stop();
Console.WriteLine($"{n}\nБуло витрачено {stopwatch.ElapsedMilliseconds * 1000000}
наносекунд;");
}
}

```

```

Число m, де m<=20
10
1      0      6      4      5      0      3      4      0      6
Найбільш можливе -> 6654431000
Було витрачено 0 наносекунд;

```

Рисунок 3.3 – Результат виконання завдання 2.1

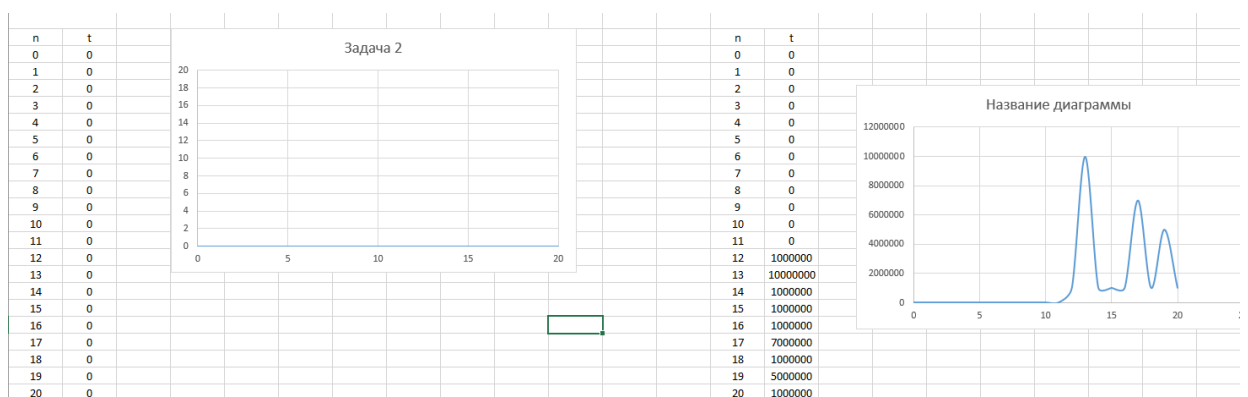


Рисунок 3.4 – Результат виконання завдання 2

Висновки: я набув навичок дослідження часової складності алгоритмів і визначення її асимптотичних оцінок.

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

Порядок виконання роботи

1. Розробити всі основні функції роботи з двозв'язним списком (доповнити функції, які відсутні у прикладі, що розглядався на лекції для тих, хто претендує на оцінку "відмінно").
2. Розробити програму роботи з двозв'язним списком. Створення та заповнення динамічних структур даних повинно виконуватися в діалоговому режимі. Програма повинна виконувати наступні операції: створення списку, додавання елементів, видалення елементів, виведення списку на дисплей, знищення списку. Протестуйте програму для 7 – 10 елементів.
3. Розробити програму обчислення арифметичного виразу (використати зворотну польську запис). Операнди у виразі розділяти пробілами. Операції: додавання (+), віднімання (-), множення (*), ділення (/), зведення в ступінь (^), корінь квадратний (sqrt). Допускається використати готові класи роботи з динамічними структурами даних.

4.1.1

Завдання 1:

1. Розробити всі основні функції роботи з двозв'язним списком (доповнити функції, які відсутні у прикладі, що розглядався на лекції для тих, хто претендує на оцінку "відмінно").

Лістинг:

```
#include <stdio.h>
#include <stdlib.h>
```

```
typedef int elemtype;
struct elem {
    elemtype value;
    struct elem* next;
    struct elem* prev;
};
```

					ДУ«Житомирська політехніка».21.121.02.000–Лр 4			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Маньківський В.В			Звіт з лабораторної роботи		Літ.	Арк.
Перевір.		Локтікова Т.М.						1
Керівник							Аркушів	
Н. контр.							14	
Зав. каф.							ФІКТ Гр. ВТ-21-1[2]	

```

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) {

```

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Пр 4	Арк.
		Локтікова Т.М.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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)) {

```

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Пр 4	Арк.
		Локтікова Т.М.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        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 початок 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;
}

```

		Маньківський В.М			ДУ«Житомирська політехніка».21.121.02.000–Лр 4	Арк.
		Локтікова Т.М.				5
Змн.	Арк.	№ докум.	Підпис	Дата		


```

Список пустий

1)додавання елемента на початої списку (1)
2)додавання елемента в середину списку (2)
3)додавання елемента в кінець списку (3)
4)видалення елемента з початку списку
5)видалення елемента з середини списку
6)видалення елемента з кінця списку
Введіть дію: 1

1. 1

1)додавання елемента на початої списку (1)
2)додавання елемента в середину списку (2)
3)додавання елемента в кінець списку (3)
4)видалення елемента з початку списку
5)видалення елемента з середини списку
6)видалення елемента з кінця списку
Введіть дію: 3

1. 1
2. 3

```

Рисунок 4.1 – Результат виконання завдання 1

```

1)додавання елемента на початої списку (1)
2)додавання елемента в середину списку (2)
3)додавання елемента в кінець списку (3)
4)видалення елемента з початку списку
5)видалення елемента з середини списку
6)видалення елемента з кінця списку
Введіть дію: 2

1. 1
2. 2
3. 3

1)додавання елемента на початої списку (1)
2)додавання елемента в середину списку (2)
3)додавання елемента в кінець списку (3)
4)видалення елемента з початку списку
5)видалення елемента з середини списку
6)видалення елемента з кінця списку
Введіть дію: 4

1. 2
2. 3

```

Рисунок 4.2 – Результат виконання завдання 1

		Маньківський В.М			ДУ«Житомирська політехніка».21.121.02.000–Лр 4	Арк.
		Локтікова Т.М.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

```

1. 1
2. 2
3. 3

1)додавання елемента на початої списку (1)
2)додавання елемента в середину списку (2)
3)додавання елемента в кінець списку (3)
4)видалення елемента з початку списку
5)видалення елемента з середини списку
6)видалення елемента з кінця списку
Введіть дію: 5

1. 1
2. 3

1)додавання елемента на початої списку (1)
2)додавання елемента в середину списку (2)
3)додавання елемента в кінець списку (3)
4)видалення елемента з початку списку
5)видалення елемента з середини списку
6)видалення елемента з кінця списку
Введіть дію: 6

1. 1

```

Рисунок 4.3 – Результат виконання завдання 1

Завдання 2:

2. Розробити програму роботи з двозв'язним списком. Створення та заповнення динамічних структур даних повинно виконуватися в діалоговому режимі. Програма повинна виконувати наступні операції: створення списку, додавання елементів, видалення елементів, виведення списку на дисплей, знищення списку. Протестуйте програму для 7 – 10 елементів.

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
    {

```

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Пр 4	Арк.
		Локтікова Т.М.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        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
            }

```

		Маньківський В.М			ДУ«Житомирська політехніка».21.121.02.000–Пр 4	Арк.
		Локтікова Т.М.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        {
            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);

```

		Маньківський В.М			ДУ«Житомирська політехніка».21.121.02.000–Пр 4	Арк.
		Локтікова Т.М.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Введіть кількість елементів списку:
8

1
2
3
4
5
6
7
8

1)додати елемент
2)видалити елемент списку
3)видалити список
1
Введіть елемент після якого додати новий
3
Введіть новий елемент
100

1
2
3
100
4
5
6
7
8

```

Рисунок 4.4 – Результат виконання завдання 2

```

1)додати елемент
2)видалити елемент списку
3)видалити список
2
Введіть елемент який хочете видалити:
4

1
2
3
100
5
6
7
8

```

Рисунок 4.5 – Результат виконання завдання 2

		Маньківський В.М			ДУ«Житомирська політехніка».21.121.02.000–Лр 4	Арк.
		Локтікова Т.М.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

```

1)додати елемент
2)видалити елемент списку
3)видалити список
3
0

```

Рисунок 4.6 – Результат виконання завдання 2

Завдання 3:

3. Розробити програму обчислення арифметичного виразу (використати зворотну польську запис). Операнди у виразі розділяти пробілами. Операції: додавання (+), віднімання (-), множення (*), ділення (/), зведення в ступінь (^), корінь квадратний (sqrt). Допускається використати готові класи роботи з динамічними структурами даних.

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($"{result[j]} ");
        }
        Console.Write("\nCтек: ");
        foreach (Object obj in stack)
        {
            Console.Write($"{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;
                }
            }
        }
    }
}

```

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Пр 4	Арк.
		Локтікова Т.М.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        }
        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($"Обчислення {str} = {stack.Peek()}\n");
}
}

```

		Маньківський В.М			ДУ«Житомирська політехніка».21.121.02.000–Лр 4	Арк.
		Локтікова Т.М.				13
Змн.	Арк.	№ докум.	Підпис	Дата		


```

Введіть вираз: ( 36 / 4 + 7 * 3 ) / 2
Рядок:
Стек: (

Рядок: 36
Стек: (

Рядок: 36
Стек: / (

Рядок: 36 4
Стек: / (

Рядок: 36 4 /
Стек: + (

Рядок: 36 4 / 7
Стек: + (

Рядок: 36 4 / 7
Стек: * + (

Рядок: 36 4 / 7 3
Стек: * + (

Рядок: 36 4 / 7 3 * +
Стек:

Рядок: 36 4 / 7 3 * +
Стек: /

Рядок: 36 4 / 7 3 * + 2
Стек: /

Зворотній польський запис: 36 4 / 7 3 * + 2 /
Обчислення ( 36 / 4 + 7 * 3 ) / 2 = 15

```

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.

		Маньківський В.М			ДУ«Житомирська політехніка».21.121.02.000–Пр 4	Арк.
		Локтікова Т.М.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

Лабораторна робота № 5

ПРОСТІ МЕТОДИ СОРТУВАННЯ

Мета : реалізація простих алгоритмів сортування та дослідження їх характеристик (швидкодія, необхідний обсяг пам'яті, застосування тощо).

5.1 Хід роботи

Зміст звіту

1. Опис алгоритму (словесна форма або блок-схема алгоритму).
2. Текст функцій сортування з коментарями.
3. Таблиця результатів вимірів часу.
4. Графіки результатів вимірів часу.
5. Висновки по роботі (опис досліджених характеристик кожного алгоритму, порівняння алгоритмів, відзначити достоїнства та недоліки).

Порядок виконання роботи

1. Реалізувати алгоритми сортування:
 - а) сортування вибором (структура даних – двусвязний список);
 - б) сортування вставками (структура даних – масив);
 - в) сортування вставками (структура даних – двусвязний список);
2. Вивчити засоби вимірювання інтервалів часу (можна використовувати клас Stopwatch з простору імен System.Diagnostics для C # або використати бібліотеку C++ <chrono>).
3. Виміряти час сортування даних різної розмірності: 10, 100, 500, 1000, 2000, 5000, 10000. Дані сформувати з використанням генератора випадкових чисел.
4. За отриманими даними побудувати графіки залежностей часу сортування від кількості вхідних даних (з використанням Excel).

5.1.1

Завдання:

Лістинг:

```
using System.Diagnostics;
namespace lab_5
{
    class Program
    {
```

					ДУ«Житомирська політехніка».21.121.02.000–Лр 5			
Змн.	Арк.	№ докум.	Підпис	Дата	Звіт з лабораторної роботи			
Розроб.		Маньківський В.В						
Перевір.		Локтікова Т.М.						
Керівник								
Н. контр.								
Зав. каф.					ФІКТ Гр. ВТ-21-1[2]			
					Літ.	Арк.	Аркушів	
						1	13	

```

static void Main()
{
    Console.OutputEncoding = System.Text.Encoding.Unicode;
    Console.InputEncoding = System.Text.Encoding.Unicode;
    Stopwatch stopwatch = new Stopwatch();
    Random random = new Random();
    LinkedList<int> list = new LinkedList<int>();
    int[] mass_1 = new int[10];
    int[] mass_2 = new int[100];
    int[] mass_3 = new int[500];
    int[] mass_4 = new int[1000];
    int[] mass_5 = new int[2000];
    int[] mass_6 = new int[5000];
    int[] mass_7 = new int[10000];
    Console.WriteLine("a) сортування вибором (структура даних – двусвязний
список);");

    Console.WriteLine("б) сортування вставками (структура даних –
масив);");

    Console.WriteLine("в) сортування вставками (структура даних –
двусвязний список); ");
    int x = 10, num;
    Console.WriteLine($"Позмірність {x}:");
    for (int i = 0; i < x; i++)
    {
        num = random.Next(-x, x);
        list.AddLast(num);
        mass_1[i] = num;
    }
    stopwatch.Reset();
    stopwatch.Start();
    listSelectionSort(list, x);
    stopwatch.Stop();
    list.Clear();
    for (int i = 0; i < x; i++)
    {
        list.AddLast(mass_1[i]);
    }
    Console.WriteLine($"a) {stopwatch.Elapsed.TotalMilliseconds} мс");

    stopwatch.Reset();
    stopwatch.Start();
    massInsertionSort(mass_1);
    stopwatch.Stop();

```

		Маньківський В.М			ДУ«Житомирська політехніка».21.121.02.000–Лр 5	Арк.
		Локтікова Т.М.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Console.WriteLine($"6) {stopwatch.Elapsed.TotalMilliseconds} мс");

for (int i = 0; i < x; i++)
    list.AddLast(random.Next(-x, x));
stopwatch.Reset();
stopwatch.Start();
listInsertionSort(list, x);
stopwatch.Stop();
list.Clear();
Console.WriteLine($"8) {stopwatch.Elapsed.TotalMilliseconds} мс");

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

x = 100;
Console.WriteLine($"Розмірність {x}:");
for (int i = 0; i < x; i++)
{
    num = random.Next(-x, x);
    list.AddLast(num);
    mass_2[i] = num;
}
stopwatch.Reset();
stopwatch.Start();
listSelectionSort(list, x);
stopwatch.Stop();
list.Clear();
for (int i = 0; i < x; i++)
{
    list.AddLast(mass_2[i]);
}
Console.WriteLine($"a) {stopwatch.Elapsed.TotalMilliseconds} мс");

stopwatch.Reset();
stopwatch.Start();
massInsertionSort(mass_2);
stopwatch.Stop();
Console.WriteLine($"6) {stopwatch.Elapsed.TotalMilliseconds} мс");

for (int i = 0; i < x; i++)
    list.AddLast(random.Next(-x, x));

```

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Лр 5	Арк.
		Локтікова Т.М.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

```

stopwatch.Reset();
stopwatch.Start();
listInsertionSort(list, x);
stopwatch.Stop();
list.Clear();
Console.WriteLine($"Б) {stopwatch.Elapsed.TotalMilliseconds} мс");

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

x = 500;
Console.WriteLine($"Позмірність {x}:");
for (int i = 0; i < x; i++)
{
    num = random.Next(-x, x);
    list.AddLast(num);
    mass_3[i] = num;
}
stopwatch.Reset();
stopwatch.Start();
listSelectionSort(list, x);
stopwatch.Stop();
list.Clear();
for (int i = 0; i < x; i++)
{
    list.AddLast(mass_3[i]);
}
Console.WriteLine($"А) {stopwatch.Elapsed.TotalMilliseconds} мс");

stopwatch.Reset();
stopwatch.Start();
massInsertionSort(mass_3);
stopwatch.Stop();
Console.WriteLine($"Б) {stopwatch.Elapsed.TotalMilliseconds} мс");

for (int i = 0; i < x; i++)
    list.AddLast(random.Next(-x, x));
stopwatch.Reset();
stopwatch.Start();
listInsertionSort(list, x);
stopwatch.Stop();

```

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Лр 5	Арк.
		Локтікова Т.М.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

```

list.Clear();
Console.WriteLine($"В) {stopwatch.Elapsed.TotalMilliseconds} мс");

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

x = 1000;
Console.WriteLine($"nРозмірність {x}:");
for (int i = 0; i < x; i++)
{
    num = random.Next(-x, x);
    list.AddLast(num);
    mass_4[i] = num;
}
stopwatch.Reset();
stopwatch.Start();
listSelectionSort(list, x);
stopwatch.Stop();
list.Clear();
for (int i = 0; i < x; i++)
{
    list.AddLast(mass_4[i]);
}
Console.WriteLine($"a) {stopwatch.Elapsed.TotalMilliseconds} мс");

stopwatch.Reset();
stopwatch.Start();
massInsertionSort(mass_4);
stopwatch.Stop();
Console.WriteLine($"6) {stopwatch.Elapsed.TotalMilliseconds} мс");

for (int i = 0; i < x; i++)
    list.AddLast(random.Next(-x, x));
stopwatch.Reset();
stopwatch.Start();
listInsertionSort(list, x);
stopwatch.Stop();
list.Clear();
Console.WriteLine($"В) {stopwatch.Elapsed.TotalMilliseconds} мс");

```

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Пр 5	Арк.
		Локтікова Т.М.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

```
x = 2000;
Console.WriteLine($"\\nРозмірність {x}:");
for (int i = 0; i < x; i++)
{
    num = random.Next(-x, x);
    list.AddLast(num);
    mass_5[i] = num;
}
stopwatch.Reset();
stopwatch.Start();
listSelectionSort(list, x);
stopwatch.Stop();
list.Clear();
for (int i = 0; i < x; i++)
{
    list.AddLast(mass_5[i]);
}
Console.WriteLine($"a {stopwatch.Elapsed.TotalMilliseconds} mc");
```

```
stopwatch.Reset();
stopwatch.Start();
massInsertionSort(mass_5);
stopwatch.Stop();
Console.WriteLine($"6) {stopwatch.Elapsed.TotalMilliseconds} ms");
```

```
for (int i = 0; i < x; i++)
    list.AddLast(random.Next(-x, x));
stopwatch.Reset();
stopwatch.Start();
listInsertionSort(list, x);
stopwatch.Stop();
list.Clear();
Console.WriteLine($"B {stopwatch.Elapsed.TotalMilliseconds} ms");
```

```
x = 5000;
```

		Маньківський В.М			ДУ«Житомирська політехніка».21.121.02.000–Пр 5	Арк.
		Локтікова Т.М.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Console.WriteLine($"\\nРозмірність {x}:");
for (int i = 0; i < x; i++)
{
    num = random.Next(-x, x);
    list.AddLast(num);
    mass_6[i] = num;
}
stopwatch.Reset();
stopwatch.Start();
listSelectionSort(list, x);
stopwatch.Stop();
list.Clear();
for (int i = 0; i < x; i++)
{
    list.AddLast(mass_6[i]);
}
Console.WriteLine($"a) {stopwatch.Elapsed.TotalMilliseconds} mc");

```

```

stopwatch.Reset();
stopwatch.Start();
massInsertionSort(mass_6);
stopwatch.Stop();
Console.WriteLine($"6) {stopwatch.Elapsed.TotalMilliseconds} mc");

```

```

for (int i = 0; i < x; i++)
    list.AddLast(random.Next(-x, x));
stopwatch.Reset();
stopwatch.Start();
listInsertionSort(list, x);
stopwatch.Stop();
list.Clear();
Console.WriteLine($"b) {stopwatch.Elapsed.TotalMilliseconds} mc");

```

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

```

x = 10000;
Console.WriteLine($"\\nРозмірність {x}:");
for (int i = 0; i < x; i++)
{
    num = random.Next(-x, x);

```

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Лр 5	Арк.
		Локтікова Т.М.				7
Змн.	Арк.	№ докум.	Підпис	Дата		


```

        list.AddLast(num);
        mass_7[i] = num;
    }
    stopwatch.Reset();
    stopwatch.Start();
    listSelectionSort(list, x);
    stopwatch.Stop();
    list.Clear();
    for (int i = 0; i < x; i++)
    {
        list.AddLast(mass_7[i]);
    }
    Console.WriteLine($"a) {stopwatch.Elapsed.TotalMilliseconds} mc");

    stopwatch.Reset();
    stopwatch.Start();
    massInsertionSort(mass_7);
    stopwatch.Stop();
    Console.WriteLine($"6) {stopwatch.Elapsed.TotalMilliseconds} mc");

    for (int i = 0; i < x; i++)
        list.AddLast(random.Next(-x, x));
    stopwatch.Reset();
    stopwatch.Start();
    listInsertionSort(list, x);
    stopwatch.Stop();
    list.Clear();
    Console.WriteLine($"в) {stopwatch.Elapsed.TotalMilliseconds} mc");

    //foreach (var person in list) Console.WriteLine(person);

}

static void listInsertionSort(LinkedList<int> list, int x)
{
    int count = 1;
    int k;
    int j;

```

елемента

```
int temp;
var currentNode = list.First;
for (int i = 1; i < x; i++)//початок сортування вставок для масиву
{
    currentNode = list.First;
    for(count = 0; count < i; count++)//повернення до теперішнього
        currentNode = currentNode.Next;

    k = currentNode.Value;//теперішній елемент
    j = i;//індекс теперішнього елемента
    while (j > 0 && currentNode.Previous.Value > k)//заміна
    {
        temp = currentNode.Value;
        currentNode.Value = currentNode.Previous.Value;
        currentNode.Previous.Value = temp;
        j--;
        currentNode = currentNode.Previous;
    }
    currentNode.Value = k;
}
```

масиву

```
static void massInsertionSort(int[] mass)
{
    int k;
    int j;
    int temp;
    for (int i = 1; i < mass.Length; i++)//початок сортування вставок для
        масиву
        {
            k = mass[i];//теперішній елемент
            j = i;//індекс теперішнього елемента
            while (j > 0 && mass[j - 1] > k)//заміна
            {
                temp = mass[j];
                mass[j] = mass[j-1];
                mass[j-1] = temp;
                j--;
            }
            mass[j] = k;
        }
}
```

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Пр 5	Арк.
		Локтікова Т.М.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

```

static void listSelectionSort(LinkedList<int> list, int x, int
currentIndex = 0, int count_1 = 50000, int index = 0, int i = -1, int count_2 = 50000)
{
    for (currentIndex = 0; currentIndex < x; currentIndex++)//початок
    сортування вибором для списків
    {
        count_2 = 50000;
        i = -1;
        index = 0;
        foreach (var person in list)//пошук теперішнього та найменшого
        елемента
        {
            i++;//індекс теперішнього елемента
            if (person < count_2 && i > currentIndex)//найменший елемент
            {
                count_2 = person;//найменший знайдений елемент
                index = i;//індекс найменшого елемента
            }
            if (i == currentIndex)//теперішній елемент
                count_1 = person;
        }
        if (index != currentIndex && count_2 <= count_1)//заміна
        {
            i = -1;
            var currentNode = list.First;
            while (currentNode != null)//пошук цих елементів
            {
                i++;
                if (i == currentIndex)//заміна
                    currentNode.Value = count_2;
                if (i == index)//заміна
                    currentNode.Value = count_1;
                currentNode = currentNode.Next;
            }
        }
    }
}

```

		Маньківський В.М			ДУ«Житомирська політехніка».21.121.02.000–Лр 5	Арк.
		Локтікова Т.М.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

```

а) сортування вибором (структура даних – двусвязний список);
б) сортування вставками (структура даних – масив);
в) сортування вставками (структура даних – двусвязний список);

Розмірність 10:
а) 1,7511 мс
б) 0,1768 мс
в) 0,3299 мс

Розмірність 100:
а) 0,2416 мс
б) 0,0199 мс
в) 0,1275 мс

Розмірність 500:
а) 5,9849 мс
б) 0,4368 мс
в) 3,0519 мс

Розмірність 1000:
а) 25,3935 мс
б) 1,7003 мс
в) 14,6873 мс

Розмірність 2000:
а) 100,2654 мс
б) 7,4698 мс
в) 36,6659 мс

Розмірність 5000:
а) 513,2568 мс
б) 52,6816 мс
в) 341,3473 мс

Розмірність 10000:
а) 2038,0403 мс
б) 171,0827 мс
в) 884,282 мс

```

Рисунок 5.1 – Результат виконання завдання

Словесний опис алгоритмів:

Сортування вибором – алгоритм сортування масиву, який по швидкості виконання можна зрівняти з сортуванням бульбашкою.

Алгоритм сортування вибором складається з наступних кроків:

- Для початку визначаємо позицію мінімального елементу масиву;
- Здійснюємо обмін мінімального елементу з елементом на початку масиву. Виходить, що перший елемент масиву вже відсортовано;

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Лр 5	Арк.
		Локтікова Т.М.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

- Зменшуємо робочу область масиву, відкидаючи перший елемент, а для підмасиву, що залишився, повторюємо сортування.

Алгоритмічна складність: $O(n) = n^2$.

Сортування вставками - це алгоритм сортування, в якому всі елементи масиву по чергові переглядаються, при цьому кожен елемент переміщається у відповідне місце серед раніше впорядкованих значень.

Алгоритм роботи сортування включенням наступний:

- На початку роботи впорядкована частина порожня;
- Додаємо в неї перший елемент масиву з не впорядкованих даних;
- Переходимо до наступного елементу в невідсортованих даних, і знаходимо йому правильну позицію у відсортованій частині масиву, цим ми розширюємо область впорядкованих даних;
- Повторюємо попередній крок для всіх елементів, що залишилися.

Алгоритмічна складність: $O(n) = n^2$.

Графіки та таблиця:

Кількість елементів	Метод вибору	Метод вставками	
	список (мс)	масив (мс)	список (мс)
10	1,7511	0,1768	0,3299
100	0,2416	0,0199	0,1275
500	5,9849	0,4368	3,0519
1000	25,3935	1,7003	14,6873
2000	100,2654	7,4698	36,6659
5000	513,2568	52,6816	341,3473
10000	2038,0403	171,0827	884,282

Рисунок 5.2 – Таблиця

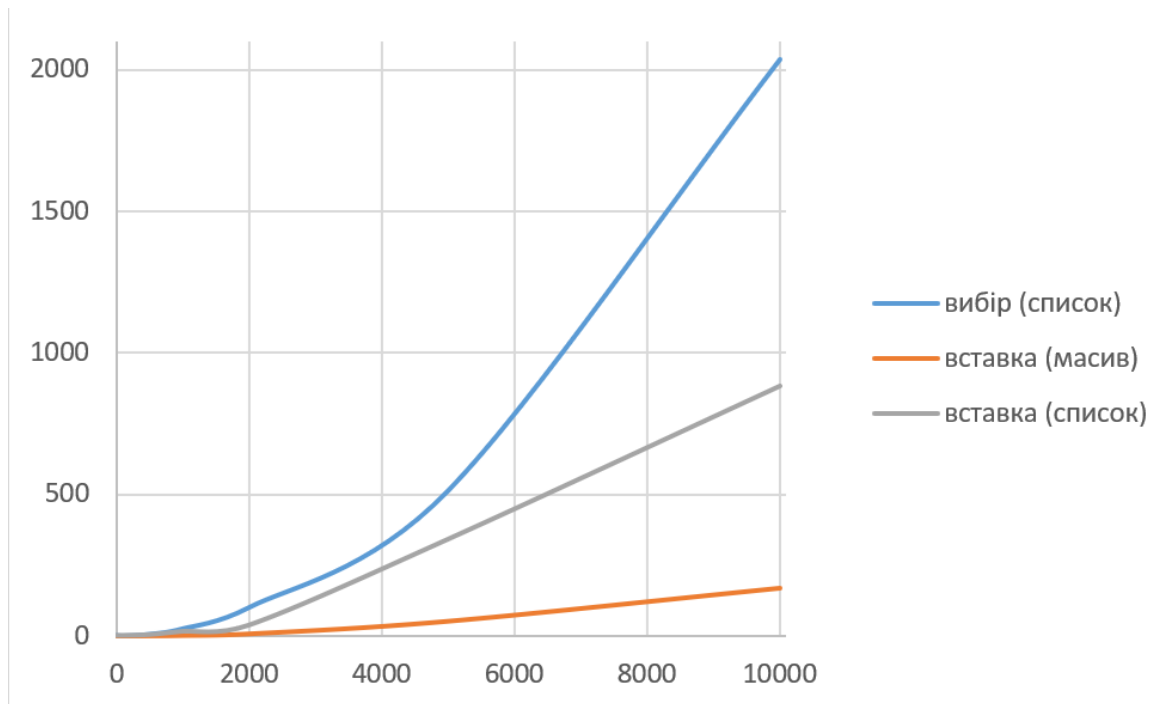


Рисунок 5.3 – Графіки

Висновки: я дослідив та порівняв характеристики кожного з алгоритмів, та дійшов висновку, що сортування вибором найповільніше сортування, а вставка, навпаки виграє як для масиву, так і для списку, на мою думку, це відбувається так, тому що вставка, може бути частково відсортована, це значно допомагає в сортуванні, а під час вибірки, сортування не залежить від розміщення елемента, на малій кількості елементів, різниця невелика, але зі збільшенням їх кількості, збільшується і різниця, яка продемонстрована на графіку.

Лабораторна робота № 6

ШВИДКІ МЕТОДИ СОРТУВАННЯ

Мета : реалізація швидких алгоритмів сортування та дослідження їх характеристик (швидкодія, необхідний обсяг пам'яті, застосування тощо).

5.1 Хід роботи

Зміст звіту

1. Опис алгоритму (словесна форма або блок-схема алгоритму).
2. Текст функцій сортування з коментарями.
3. Таблиця результатів вимірів часу.
4. Графіки результатів вимірів часу.
5. Висновки по роботі (опис досліджених характеристик кожного алгоритму, порівняння алгоритмів, відзначити переваги та недоліки).

Порядок виконання роботи

1. Реалізувати алгоритми сортування у відповідності за таблицею 6.1:
 - а) пірамідальне сортування (структура даних – масив);
 - б) сортування Шелла (структура даних – масив);
 - в) сортування підрахунком (структура даних – масив).
2. Виміряти час сортування даних різної розмірності: 10, 100, 500, 1000, 2000, 5000, 10000. Дані сформувати з використанням генератора випадкових чисел.
3. За отриманими даними побудувати графіки залежностей часу сортування від кількості вхідних даних (з використанням Excel).

Варіант	Пірамідальне сортування		Сортування Шелла			Сортування підрахунком	
	Тип даних	Діапазон	Тип даних	Діапазон	Формула приросту	Тип даних	Діапазон
2, 17	double	[-10, 100]	float	[0, 200]	$\text{inc}(s)=(3^s-1)/2$	char	[-100, -10]

					ДУ«Житомирська політехніка».21.121.02.000–Лр 6							
Змн.	Арк.	№ докум.	Підпис	Дата								
Розроб.		Маньківський В.В			Звіт з лабораторної роботи				Літ.	Арк.	Аркушів	
Перевір.		Локтікова Т.М.									1	14
Керівник									ФІКТ Гр. ВТ-21-1[2]			
Н. контр.												
Зав. каф.												

6.1.1

Завдання:

Лістинг:

```
using System.Diagnostics;
namespace lab_5
{
    class Program
    {
        static void Main()
        {
            Console.OutputEncoding = System.Text.Encoding.Unicode;
            Console.InputEncoding = System.Text.Encoding.Unicode;
            Stopwatch stopwatch = new Stopwatch();
            Random random = new Random();
            Console.WriteLine("а) пірамідальне сортування (структура даних – масив);");
            Console.WriteLine("б) сортування Шелла(структура даних – масив);");
            Console.WriteLine("в) сортування підрахунком(структура даних – масив);\n");

            double[] piramida_10 = new double[10];
            double[] piramida_100 = new double[100];
            double[] piramida_500 = new double[500];
            double[] piramida_1000 = new double[1000];
            double[] piramida_2000 = new double[2000];
            double[] piramida_5000 = new double[5000];
            double[] piramida_10000 = new double[10000];

            float[] szela_10 = new float[10];
            float[] szela_100 = new float[100];
            float[] szela_500 = new float[500];
            float[] szela_1000 = new float[1000];
            float[] szela_2000 = new float[2000];
            float[] szela_5000 = new float[5000];
            float[] szela_10000 = new float[10000];

            char[] liczba_10 = new char[10];
            char[] liczba_100 = new char[100];
            char[] liczba_500 = new char[500];
            char[] liczba_1000 = new char[1000];
            char[] liczba_2000 = new char[2000];
            char[] liczba_5000 = new char[5000];
```

		Маньківський В.М			ДУ«Житомирська політехніка».21.121.02.000–Лр 6	Арк.
		Локтікова Т.М.				2
Змн.	Арк.	№ докум.	Підпис	Дата		


```
char[] liczba_10000 = new char[10000];
```

```
HeapSort heapSort = new HeapSort();
```

```
ShellSort shellSort = new ShellSort();
```

```
CountSort countSort = new CountSort();
```

```
////////////////////////////////////  
////////////////////////////////////
```

```
int count = 10;
```

```
Console.WriteLine($"Розмірність: {count}");
```

```
for (int i = 0; i < count; i++)
```

```
{
```

```
    piramida_10[i] = random.NextDouble() * (100 - (-10)) - 10;
```

```
}
```

```
stopwatch.Start();
```

```
heapSort.sort(piramida_10);
```

```
stopwatch.Stop();
```

```
Console.WriteLine($"a) {stopwatch.Elapsed.TotalMilliseconds} мс");
```

```
stopwatch.Reset();
```

```
for (int i = 0; i < count; i++)
```

```
{
```

```
    szela_10[i] = (float)random.NextDouble() * (200 - 0) - 0;
```

```
}
```

```
stopwatch.Start();
```

```
shellSort.Sort(szela_10);
```

```
stopwatch.Stop();
```

```
Console.WriteLine($"6) {stopwatch.Elapsed.TotalMilliseconds} мс");
```

```
stopwatch.Reset();
```

```
for (int i = 0; i < count; i++)
```

```
{
```

```
    liczba_10[i] = (char)random.Next(-100, -10);
```

```
}
```

```
stopwatch.Start();
```

```
countSort.Sort(liczba_10);
```

```
stopwatch.Stop();
```

```
Console.WriteLine($"b) {stopwatch.Elapsed.TotalMilliseconds} мс");
```

```
stopwatch.Reset();
```

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Пр 6	Арк.
		Локтікова Т.М.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

//
//

```
count = 100;
Console.WriteLine($"\\nРозмірність: {count}");
for (int i = 0; i < count; i++)
{
    piramida_100[i] = random.NextDouble() * (100 - (-10)) - 10;
}
stopwatch.Start();
heapSort.sort(piramida_100);
stopwatch.Stop();
Console.WriteLine($"a) {stopwatch.Elapsed.TotalMilliseconds} мс");
stopwatch.Reset();
for (int i = 0; i < count; i++)
{
    szela_100[i] = (float)random.NextDouble() * (200 - 0) - 0;

}
stopwatch.Start();
shellSort.Sort(szela_100);
stopwatch.Stop();
Console.WriteLine($"6) {stopwatch.Elapsed.TotalMilliseconds} мс");
stopwatch.Reset();
for (int i = 0; i < count; i++)
{
    liczba_100[i] = (char)random.Next(-100, -10);
}
stopwatch.Start();
countSort.Sort(liczba_100);
stopwatch.Stop();
Console.WriteLine($"b) {stopwatch.Elapsed.TotalMilliseconds} мс");
stopwatch.Reset();
```

//
//

count = 500;

		Маньківський В.М			ДУ«Житомирська політехніка».21.121.02.000–Лр 6	Арк.
		Локтікова Т.М.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

[illegible]

		Маньківський В.М			ДУ«Житомирська політехніка».21.121.02.000–Пр 6	Арк.
		Локтікова Т.М.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

```

heapSort.sort(piramida_1000);
stopwatch.Stop();
Console.WriteLine($"a) {stopwatch.Elapsed.TotalMilliseconds} мс");
stopwatch.Reset();
for (int i = 0; i < count; i++)
{
    szela_1000[i] = (float)random.NextDouble() * (200 - 0) - 0;

}
stopwatch.Start();
shellSort.Sort(szela_1000);
stopwatch.Stop();
Console.WriteLine($"б) {stopwatch.Elapsed.TotalMilliseconds} мс");
stopwatch.Reset();
for (int i = 0; i < count; i++)
{
    liczba_1000[i] = (char)random.Next(-100, -10);
}
stopwatch.Start();
countSort.Sort(liczba_1000);
stopwatch.Stop();
Console.WriteLine($"в) {stopwatch.Elapsed.TotalMilliseconds} мс");
stopwatch.Reset();

```

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

```

count = 2000;
Console.WriteLine($"nПозмірність: {count}");
for (int i = 0; i < count; i++)
{
    piramida_2000[i] = random.NextDouble() * (100 - (-10)) - 10;
}
stopwatch.Start();
heapSort.sort(piramida_2000);
stopwatch.Stop();
Console.WriteLine($"a) {stopwatch.Elapsed.TotalMilliseconds} мс");
stopwatch.Reset();
for (int i = 0; i < count; i++)
{

```

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Лр 6	Арк.
		Локтікова Т.М.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        szela_2000[i] = (float)random.NextDouble() * (200 - 0) - 0;

    }
    stopwatch.Start();
    shellSort.Sort(szela_2000);
    stopwatch.Stop();
    Console.WriteLine($"6) {stopwatch.Elapsed.TotalMilliseconds} мс");
    stopwatch.Reset();
    for (int i = 0; i < count; i++)
    {
        liczba_2000[i] = (char)random.Next(-100, -10);
    }
    stopwatch.Start();
    countSort.Sort(liczba_2000);
    stopwatch.Stop();
    Console.WriteLine($"8) {stopwatch.Elapsed.TotalMilliseconds} мс");
    stopwatch.Reset();

```

```

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

```

```

count = 5000;
Console.WriteLine($"Розмірність: {count}");
for (int i = 0; i < count; i++)
{
    piramida_5000[i] = random.NextDouble() * (100 - (-10)) - 10;
}
stopwatch.Start();
heapSort.sort(piramida_5000);
stopwatch.Stop();
Console.WriteLine($"a) {stopwatch.Elapsed.TotalMilliseconds} мс");
stopwatch.Reset();
for (int i = 0; i < count; i++)
{
    szela_5000[i] = (float)random.NextDouble() * (200 - 0) - 0;

}
stopwatch.Start();
shellSort.Sort(szela_5000);
stopwatch.Stop();

```

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Лр 6	Арк.
		Локтікова Т.М.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

[illegible]

```
count = 10000;
Console.WriteLine($" \nРозмірність: {count}");
for (int i = 0; i < count; i++)
{
    piramida_10000[i] = random.NextDouble() * (100 - (-10)) - 10;
}
stopwatch.Start();
heapSort.sort(piramida_10000);
stopwatch.Stop();
Console.WriteLine($"a) {stopwatch.Elapsed.TotalMilliseconds} mc");
stopwatch.Reset();
for (int i = 0; i < count; i++)
{
    szela_10000[i] = (float)random.NextDouble() * (200 - 0) - 0;
}
stopwatch.Start();
shellSort.Sort(szela_10000);
stopwatch.Stop();
Console.WriteLine($"6) {stopwatch.Elapsed.TotalMilliseconds} mc");
stopwatch.Reset();
for (int i = 0; i < count; i++)
{
    liczba_10000[i] = (char)random.Next(-100, -10);
}
```

		Маньківський В.М			ДУ«Житомирська політехніка».21.121.02.000–Пр 6	Арк.
		Локтікова Т.М.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        stopwatch.Start();
        countSort.Sort(liczba_10000);
        stopwatch.Stop();
        Console.WriteLine($"В) {stopwatch.Elapsed.TotalMilliseconds} мс");
        stopwatch.Reset();
    }
}

public class HeapSort
{
    public void sort(double[] arr)
    {
        int n = arr.Length;
        //перегрупування масиву
        for (int i = n / 2 - 1; i >= 0; i--)
            heapify(arr, n, i);
        for (int i = n - 1; i >= 0; i--)
        {
            var temp = arr[0];
            arr[0] = arr[i];
            arr[i] = temp;
            heapify(arr, i, 0); //найбільше в кінець
        }
    }

    void heapify(double[] arr, int n, int i)
    {
        int largest = i;
        int l = 2 * i + 1; //ліва частина
        int r = 2 * i + 2; //права частина
        if (l < n && arr[l] > arr[largest]) //якщо лівий елемент більше
            largest = l;
        if (r < n && arr[r] > arr[largest]) //якщо правий елемент більше
            largest = r;
        if (largest != i) //якщо не найбільше не корінь
        {
            var swap = arr[i];
            arr[i] = arr[largest];
            arr[largest] = swap;
            heapify(arr, n, largest);
        }
    }
}

public class ShellSort
{

```

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Пр 6	Арк.
		Локтікова Т.М.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

```

static void Swap(ref float a, ref float b)//заміна
{
    var t = a;
    a = b;
    b = t;
}
public void Sort(float[] array)
{
    var d = array.Length / 2;//відстань між елементами, які порівнюються
    while (d >= 1)
    {
        for (var i = d; i < array.Length; i++)
        {
            var j = i;
            while ((j >= d) && (array[j - d] > array[j]))//пошук більшого
            {
                Swap(ref array[j], ref array[j - d]);
                j = j - d;
            }
        }
        d = d / 2;//зменшення відстані
    }
}
}
public class CountSort
{
    public void Sort(char[] inputArray)
    {
        int[] countArray = new int[inputArray.Max() + 1];//мазив з кінцевим
індексом максимального елемента
        for (int i = 0; i < inputArray.Length; i++)//елемент стає на індекс,
відповідно свого числа
        {
            countArray[inputArray[i]]++;
        }

        int sortedArrayIndex = 0;//індекс відсортованого елемента
        for (int i = countArray.Length - 1; i >= 0; i--)//сортування
        {
            for (int j = 0; j < countArray[i]; j++)
            {
                inputArray[sortedArrayIndex++] = (char)i;
            }
        }
    }
}

```

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Лр 6	Арк.
		Локтікова Т.М.				10
Змн.	Арк.	№ докум.	Підпис	Дата		


```

    }
  }
}

```

```

а) пірамідальне сортування (структура даних – масив);
б) сортування Шелла(структура даних – масив);
в) сортування підрахунком(структура даних – масив);

Розмірність: 10
а) 0,5155 мс
б) 0,3777 мс
в) 3,295 мс

Розмірність: 100
а) 0,0272 мс
б) 0,0156 мс
в) 0,3546 мс

Розмірність: 500
а) 0,1282 мс
б) 0,12 мс
в) 0,5443 мс

Розмірність: 1000
а) 0,3839 мс
б) 0,422 мс
в) 0,4845 мс

Розмірність: 2000
а) 0,9182 мс
б) 0,6573 мс
в) 0,3426 мс

Розмірність: 5000
а) 1,8241 мс
б) 1,9742 мс
в) 0,3923 мс

Розмірність: 10000
а) 3,9212 мс
б) 4,1629 мс
в) 0,8003 мс

```

Рисунок 6.1 – Результат виконання завдання

Словесний опис алгоритмів:

Пірамідальне сортування - алгоритм сортування, який використовує бінарну кучу. Куча (англ. Heap) – структура даних типу дерево, яка задовольняє наступній властивості: для будь-якого заданого вузла В, який є нащадком вузла А виконується умова: ключ (А) \geq ключ (В). Таким чином, кореневий вузол кучи буде зберігати найбільше значення, тому іноді таку кучу називають max-кучою. Якщо змінити порівняння на протилежне, то

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Лр 6	Арк.
		Локтікова Т.М.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

кореневий вузол буде зберігати найменше значення і таку кучу називають min-кучою. Поширеною реалізацією кучи є бінарна куча (англ. Binary heap), в якій дерево є двійковим деревом (рисунок 6.1 та рисунок 6.2). Двійкова куча задовольняє трьом умовам:

- Значення в будь-якій вершині не менш, ніж значення її нащадків (max-куча);
- Глибина всього листя відрізняється не більше ніж на 1 шар;
- Останній шар заповнюється зліва направо. Висота кореневого вузла (дерева): $\text{int}(\log_2 N)$.

Сортування Шелла є модифікацією алгоритму сортування вставками та класифікується як сортування вставками з убиваючим кроком.

Ефективність алгоритму полягає в тому, що на кожному з проміжних кроків сортується або невелике число елементів, або вже досить добре впорядковані набори елементів. Впорядкованість масиву зростає після кожного проходу. В ході вивчення алгоритму досліджувалася залежність середнього числа перестановок від розміру масивів при N від 100 до 60000 для декількох типів послідовностей кроків, для яких були отримані відповідні залежності часу роботи від розміру масиву.

Сортування підрахунком - алгоритм впорядкування, що застосовується при малій кількості різних елементів (ключів) у масиві даних. Час його роботи лінійно залежить як від загальної кількості елементів у масиві так і від кількості різних елементів.

Графіки та таблиця:

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Лр 6	Арк.
		Локтікова Т.М.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

Кількість елементів	Пірамідальне	Шелла	Підрахунком
	масив (мс)	масив (мс)	масив (мс)
10	0,5155	0,3777	3,295
100	0,0272	0,0156	0,3546
500	0,1282	0,12	0,5443
1000	0,3839	0,422	0,4845
2000	0,9182	0,6573	0,3426
5000	1,8241	1,9742	0,3923
10000	3,9212	4,9742	0,8003

Рисунок 6.2 – Таблиця

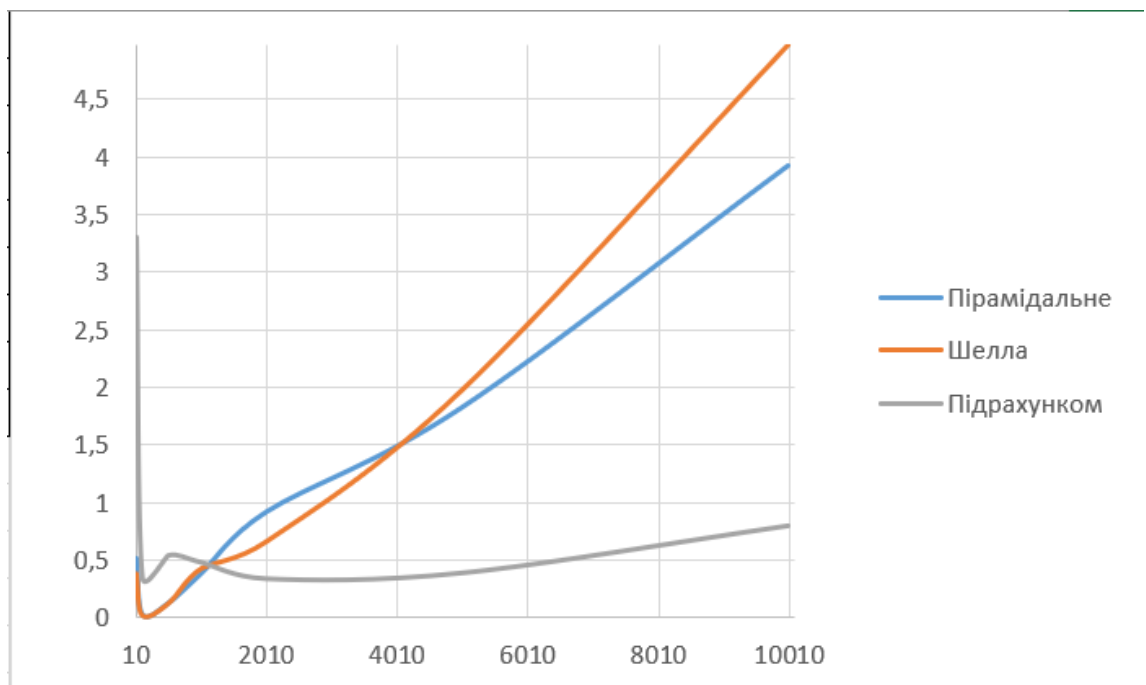


Рисунок 6.3 – Графіки

Висновки: я дослідив та порівняв характеристики кожного з алгоритмів, та дійшов висновку, що на різних проміжках, швидкість сортування різна, наприклад, Шелла найшвидше від 10 до 500, пірамідальне на 1000, підрахунком від 2000 до 10000. Перш за все, ці результати не точні, окрім підрахунку, так як в різних масивах були різні числа. Отже необхідно проаналізувати кожен алгоритм. Пірамідальне сортування залежить від кількості першин під коренем, та чи відсортованні вже елементи, якщо так, то складність алгоритму збільшується до квадрату. Сортування Шелла навпаки більш ефективне для більш впорядкованих

елементів. В свою чергу метод підрахунку не подібний на інші, так як він повністю залежить від кількості елементів, а не від їх значення.

		Маньківський В.М			ДУ«Житомирська політехніка».21.121.02.000–Пр 6	Арк.
		Локтікова Т.М.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

Лабораторна робота № 7-8

ГРАФИ. ДЕРЕВА. АЛГОРИТМИ ПОШУКУ В ГЛИБИНУ ТА В ШИРИНУ

Мета: Освоїти та закріпити прийоми роботи з даними різного типу, організованими у вигляді дерев та їх окремого випадку – бінарних дерев. Здобути практичні навички роботи з графами.

8.1 Хід роботи

Завдання

Маршрути руху автобусів зі станції Києва;

1. Київ –(135) Житомир –(80) Новоград-Волинський –(100) Рівно –(68) Луцьк
2. Київ –(135) Житомир –(38) Бердичів –(73) Вінниця –(110) Хмельницький –(104) Тернопіль
3. Київ –(135) Житомир –(115) Шепетівка
4. Київ –(78) Біла церква –(115) Умань
5. Київ –(78) Біла церква –(146) Черкаси –(105) Кременчук
6. Київ –(78) Біла церква –(181) Полтава – (130) Харків
7. Київ –(128) Прилуки –(175) Суми
8. Київ –(128) Прилуки –(109) Миргород

Порядок виконання роботи

1. Записати матрицю суміжності відповідно до завдання.
2. Реалізувати алгоритми DFS та BFS (самостійно для підвищеної оцінки).
3. Записати можливі маршрути руху та відстані всього маршруту від центрального вокзалу (кореня) до всіх інших вершин.
4. Результати вивести на екран.

Зміст звіту

1. Описати алгоритм (словесна форма, блок-схема алгоритму).
2. Побудувати граф.
3. Привести текст функцій DFS та BFS із коментарями.
4. Висновки щодо роботи.

					ДУ«Житомирська політехніка».21.121.02.000–Лр 7-8			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Маньківський В.В			Звіт з лабораторної роботи		Літ.	Арк.
Перевір.		Локтікова Т.М.						1
Керівник								6
Н. контр.							ФІКТ Гр. ВТ-21-1[2]	
Зав. каф.								

8.1.1

Завдання:

Лістинг:

```
namespace lab_7_8
{
    class Program
    {
        static void Main()
        {
            Console.OutputEncoding = System.Text.Encoding.Unicode;
            Console.InputEncoding = System.Text.Encoding.Unicode;
            int v = 19;
            int[] dfs = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18 };
            string[] city = { "Київ", "Житомир", "Новоград-Волинський", "Рівне", "Луцьк",
                            "Бердичів", "Вінниця", "Хмельницький", "Тернопіль",
                            "Шепетівка",
                            "Біла церква", "Умань",
                            "Черкаси", "Кременчук",
                            "Полтава", "Харків",
                            "Прилуки", "Суми",
                            "Миргород"};
            int[] km = { 135, 80, 100, 68, 38, 73, 110, 104, 115, 78, 115, 146, 105, 181,
130, 128, 175, 109 };
            int[,] edges = new int[v, v];
            bool[] visited = new bool[v];
            edges[0, 1] = 1;
            edges[1, 2] = 1;
            edges[2, 3] = 1;
            edges[3, 4] = 1;
            edges[1, 5] = 1;
            edges[5, 6] = 1;
            edges[6, 7] = 1;
            edges[7, 8] = 1;
            edges[1, 9] = 1;
            edges[0, 10] = 1;
            edges[10, 11] = 1;
            edges[10, 12] = 1;
            edges[12, 13] = 1;
            edges[10, 14] = 1;
            edges[14, 15] = 1;
            edges[0, 16] = 1;
            edges[16, 17] = 1;
            edges[16, 18] = 1;
            Console.WriteLine("DFS");
            for (int i = 0; i < v; i++)
            {
                if (!visited[i])
                    DFS(edges, v, visited, i, city);
            }
            for (int i = 0; i < v; i++)
            {
                visited[i] = false;
            }
            Console.WriteLine();
            Console.WriteLine("BFS");
            for (int i = 0; i < v; i++)
            {
                if (!visited[i])
```

		Маньківський В.М			ДУ«Житомирська політехніка».21.121.02.000–Лр 7-8	Арк.
		Локтікова Т.М.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        BFS(edges, v, visited, i, dfs[i], city);
    }
    Console.WriteLine($"\\nКиїв - Житомир:{km[0]}");
    Console.WriteLine($"Київ - Житомир - Новоград-Волинський:{km[0] + km[1]}");
    Console.WriteLine($"Київ - Житомир - Новоград-Волинський - Рівне:{km[0] +
km[1] + km[2]}");
    Console.WriteLine($"Київ - Житомир - Новоград-Волинський - Рівне -
Луцьк:{km[0] + km[1] + km[2] + km[3]}");
    Console.WriteLine($"Київ - Житомир - Бердичів:{km[0] + km[4]}");
    Console.WriteLine($"Київ - Житомир - Бердичів - Вінниця:{km[0] + km[4] +
km[5]}");
    Console.WriteLine($"Київ - Житомир - Бердичів - Вінниця - Хмельницький:{km[0]
+ km[4] + km[5] + km[6]}");
    Console.WriteLine($"Київ - Житомир - Бердичів - Вінниця - Хмельницький -
Тернопіль:{km[0] + km[4] + km[5] + km[6] + km[7]}");
    Console.WriteLine($"Київ - Житомир - Шепетівка: {km[0] + km[8]}");
    Console.WriteLine($"Київ - Біла церква - Умань: {km[9] + km[10]}");
    Console.WriteLine($"Київ - Біла церква - Черкаси - Кременчук: {km[9] + km[11]
+ km[12]}");
    Console.WriteLine($"Київ - Біла церква - Полтава - Харків: {km[9] + km[13] +
km[14]}");
    Console.WriteLine($"Київ - Прилуки - Суми: {km[15] + km[16]}");
    Console.WriteLine($"Київ - Прилуки - Миргород: {km[15] + km[17]}");
}
static void DFS(int[,] edges, int v, bool[] visited, int si, string[] city)
{
    visited[si] = true; //відвідане
    Console.Write($"{city[si]} ");
    for (int i = 0; i < v; i++)
    {
        if (i == si)
            continue;
        if (!visited[i] && edges[si, i] == 1)
        {
            DFS(edges, v, visited, i, city); //пошук зв'язаних графів
        }
    }
}
static void BFS(int[,] edges, int v, bool[] visited, int j, int si, string[]
city)
{
    Queue<int> queue = new Queue<int>(); //створення черги (перший зайшов, перший
вийшов)
    queue.Enqueue(si); //запис в кінець черги
    visited[si] = true;
    while (queue.Count != 0)
    {
        int currentVertex = queue.Dequeue();
        Console.Write($"{city[currentVertex]} ");
        for (int i = 0; i < v; i++) //пошук графів на цьому рівні
        {
            if (i == j)
                continue;
            if (!visited[i] && edges[currentVertex, i] == 1)
            {
                queue.Enqueue(i);
                visited[i] = true;
            }
        }
    }
}
}
}
}

```

```

DFS
Київ Житомир Новоград-Волинський Рівне Луцьк Бердичів Вінниця Хмельницький Тернопіль Шепетівка Біла церква Умань Черкаси
Кременчук Полтава Харків Прилуки Суми Миргород
BFS
Київ Житомир Біла церква Прилуки Новоград-Волинський Бердичів Шепетівка Умань Черкаси Полтава Суми Миргород Рівне Вінниц
я Кременчук Харків Луцьк Хмельницький Тернопіль
Київ - Житомир:135
Київ - Житомир - Новоград-Волинський:215
Київ - Житомир - Новоград-Волинський - Рівне:315
Київ - Житомир - Новоград-Волинський - Рівне - Луцьк:383
Київ - Житомир - Бердичів:173
Київ - Житомир - Бердичів - Вінниця:246
Київ - Житомир - Бердичів - Вінниця - Хмельницький:356
Київ - Житомир - Бердичів - Вінниця - Хмельницький - Тернопіль:460
Київ - Житомир - Шепетівка: 250
Київ - Біла церква - Умань: 193
Київ - Біла церква - Черкаси - Кременчук: 329
Київ - Біла церква - Полтава - Харків: 389
Київ - Прилуки - Суми: 303
Київ - Прилуки - Миргород: 237

```

Рисунок 8.1 – Результат виконання завдання

Словесний опис алгоритмів:

Алгоритм пошуку в глибину (DFS) — алгоритм для обходу дерева, структури подібної до дерева, або графу. Робота алгоритму починається з кореня дерева (або іншої обраної вершини в графі) і здійснюється обхід в максимально можливу глибину до переходу на наступну вершину.

Наведемо кроки алгоритму

1. Почати з довільної вершини v . Виконати $DFS(v):=1$. Включити цю вершину в стек.
2. Розглянути вершину у верхівці стеку: нехай це вершина x . Якщо всі ребра, інцидентні вершині x , позначено, то перейти до кроку 4, інакше — до кроку 3.
3. Нехай $\{x, y\}$ — непозначене ребро. Якщо $DFS(y)$ уже визначено, то позначити ребро $\{x, y\}$ штриховою лінією та перейти до кроку 2. Якщо $DFS(y)$ не визначено, то позначити ребро $\{x, y\}$ потовщеною суцільною лінією, визначити $DFS(y)$ як черговий DFS-номер, включити цю вершину в стек і перейти до кроку 2.
4. Виключити вершину x зі стеку. Якщо стек порожній, то зупинитись, інакше — перейти до кроку 2.

Обчислювальна складність: $O(n+m)$

Пошук у ширину — алгоритм пошуку на графі. Алгоритм має назву пошуку в ширину, оскільки «фронт» пошуку (між пройденими та непройденими вершинами) одноманітно розширюється вздовж всієї своєї ширини. Тобто, алгоритм проходить всі вершини на відстані k перед тим як пройти вершини на відстані $k+1$.

Наведемо кроки алгоритму

		Маньківський В.М.			ДУ«Житомирська політехніка».21.121.02.000–Лр 7-8	Арк.
		Локтікова Т.М.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

1. Почати з довільної вершини v . Виконати $BFS(v):=1$. Включити вершину v у чергу.
2. Розглянути вершину, яка перебуває на початку черги; нехай це буде вершина x . Якщо для всіх вершин, суміжних із вершиною x , уже визначено BFS-номери, то перейти до кроку 4, інакше - до кроку 3.
3. Нехай $\{x,y\}$ - ребро, у якому номер $BFS(y)$ не визначено. Позначити це ребро потовщеною суцільною лінією, визначити $BFS(y)$ як черговий BFS-номер, включити вершину y у чергу й перейти до кроку 2.
4. Виключити вершину x із черги. Якщо черга порожня, то зупинитись, інакше - перейти до кроку 2.

Обчислювальна складність: $O(|V|+|E|)$

Граф та матриця суміжності:

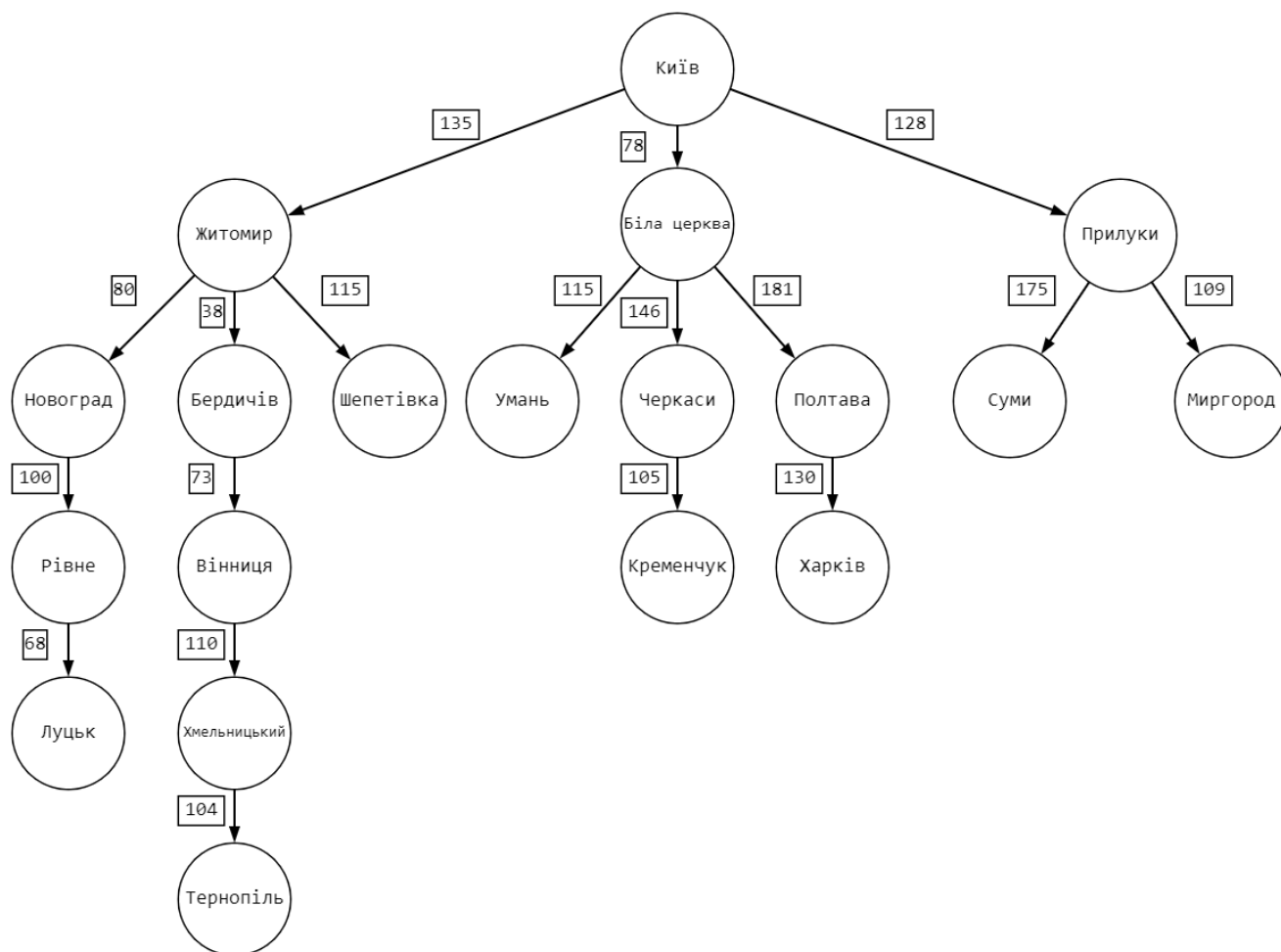


Рисунок 8.2 – Граф

