

Міністерство освіти і науки України  
Державний університет «Житомирська політехніка»  
Факультет інформаційно-комп'ютерних технологій  
Кафедра комп'ютерних наук

## **Звіт**

з лабораторних робіт

з дисципліни «Алгоритми та структури даних»

Виконав студент 1-го курсу, групи ІПЗ-20-3  
спеціальності 121 «Інженерії програмного забез-  
печення»

\_\_\_\_\_ М. О. Башманівський

Керівник \_\_\_\_\_ Р. В. Петросян

Житомир – 2021

## ЗМІСТ

Лабораторна робота №1 .....	2
Лабораторна робота №2 .....	10
Лабораторна робота №3 .....	15
Лабораторна робота №4 .....	30
Лабораторна робота №5 .....	40
Лабораторна робота №6 .....	49
Лабораторна робота №7 .....	59

					ДУ «Житомирська політехніка».20.121.3.000–Пр								
Змн.	Арк.	№ докум.	Підпис	Дата									
Розроб.		Башманівський М.			Звіт з лабораторної роботи				Літ.	Арк.	Аркушів		
Перевір.		Петросян Р.В.									1	63	
Керівник									ФІКТ Гр. ІПЗ-20-3[1]				
Н. контр.													
Зав. каф.													

## Лабораторна робота № 1

### Робота з базовими типами даних

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

### Хід 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.

Створюю структуру даних з полями та задаю об'єм пам'яті. Створюю змінну створеного типу(структури) та ініціалізую поля. Порівнюю розмір створеної структури та та існуючої структури tm.

Лістинг програми:

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <math.h>
#include <stdio.h>
#include <Windows.h>
#include <stdlib.h>
#include <time.h>

struct date
{
    unsigned short Rik : 5;
    unsigned short Mis : 4;
    unsigned short Tyzhd : 3;
    unsigned short Den : 5;
    unsigned short Godyn : 5;
    unsigned short Hvy1 : 6;
    unsigned short Sec : 6;
};

int main()
{
    date a;
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    a.Rik = 21;
    a.Mis = 6;
    a.Tyzhd = 4;
    a.Den = 31;
    a.Godyn = 21;
    a.Hvy1 = 45;
    a.Sec = 56;
    printf("%d.%d.%d %d:%d:%d |%d|", a.Den, a.Mis, a.Rik, a.Godyn, a.Hvy1, a.Sec, a.Tyzhd);
    int k, g;
    k = sizeof(date);
    g = sizeof(tm);
    printf("\nsizeof(date) = %d", k);
    printf("\nsizeof(tm) = %d", g);
}
```

Результат виконання програми:

		Башиманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

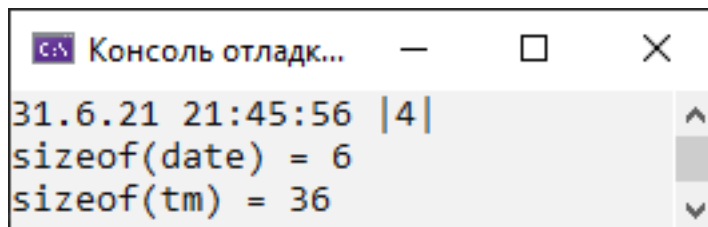


Рисунок №1 – Результат виконання програми

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

Створюю об'єднання для збереження числа типу unsigned short. При записі числа в структуру, воно розбивається на 16 бітів, останній із яких вказує на знак.

Лістинг програми:

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <math.h>
#include <stdio.h>
#include <Windows.h>
#include <stdlib.h>
#include <time.h>

union num {
    struct n {
        unsigned short a0 : 1;
        unsigned short a1 : 1;
        unsigned short a2 : 1;
        unsigned short a3 : 1;
        unsigned short a4 : 1;
        unsigned short a5 : 1;
        unsigned short a6 : 1;
        unsigned short a7 : 1;
        unsigned short a8 : 1;
        unsigned short a9 : 1;
        unsigned short a10 : 1;
        unsigned short a11 : 1;
        unsigned short a12 : 1;
        unsigned short a13 : 1;
        unsigned short a14 : 1;
        unsigned short a15 : 1;
    }number;
    signed short count;
}num1;

int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    puts("Введіть число: ");
    scanf_s("%hd", &num1);
    if (num1.count == 0)
    {
        printf("Значення заданого числа: ");
        printf("%d\n", num1.count);
        exit(0);
    }
    if (num1.number.a15 == 0) puts("Знак числа +");
    else if (num1.number.a15 == 1) puts("Знак числа -");
    printf("Значення заданого числа: ");
    printf("%d\n", num1.count);
}
```

		Башиманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

}

Результат виконання програми:

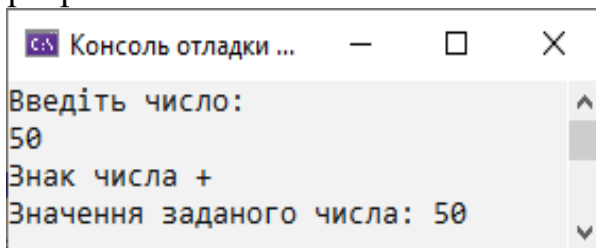


Рисунок №1.1 – Результат виконання програми

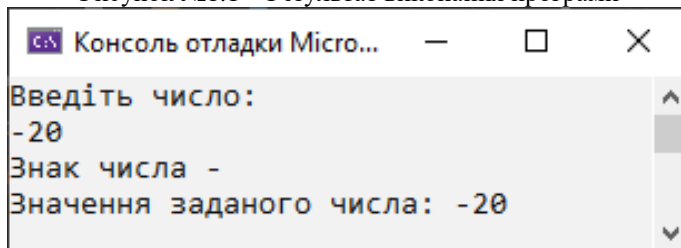


Рисунок №1.2 – Результат виконання програми

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.

Лістинінг програми:

```
#include <time.h>
#include <stdio.h>
#include <Windows.h>
#include <cstdlib>
#include <math.h>

int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    signed char res_asm;
    int Nom;
    printf("Оберіть номер завдання від а(1) до е(6): "); scanf_s("%d", &Nom);
    printf("\n");

    switch (Nom)
    {
    case 1:
    {
        printf("%d) 5 + 127\n", Nom);
        __asm
        {
            mov al, 5;
            mov bl, 127;
            add al, bl;
            mov cl, al;
            mov res_asm, cl;
        }
        printf("%d", res_asm);
    }
    }
```

		Бащманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    printf("\nПереповнення біту!");
    break;
}
case 2:
{
    printf("%d) 2-3\n", Nom);
    __asm
    {
        mov al, 2;
        mov bl, 3;
        sub al, bl;
        mov cl, al;
        mov res_asm, cl;
    }
    printf("%d", res_asm);
    break;
}
case 3:
{
    printf("%d) -120-34\n", Nom);
    __asm
    {
        mov al, -120;
        mov bl, 34;
        sub al, bl;
        mov cl, al;
        mov res_asm, cl;
    }
    printf("%d", res_asm);
    printf("\nПереповнення біту!");
    break;
}
case 4:
{
    printf("%d) (unsigned char(0-255)) (- 5)\n", Nom);
    int x;
    int y = (-5);
    printf("Введіть x:");
    scanf_s("%d", &x);
    if (x > y)
    {
        printf("%d < %d", y, x);
    }
    else
        printf("%d > %d", y, x);
    break;
}
case 5:
{
    printf("%d) 56 & 38\n", Nom);
    int x = 56, y = 38;
    if (x > y)
    {
        printf("%d > %d", x, y);
    }
    else
        printf("%d < %d", x, y);
    break;
}
case 6:
{
    printf("%d) 56 | 38\n", Nom);
    int x = 56, y = 38;
    if (x > y)

```

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

```

{
    printf("%d", x);
}
else
    printf("%d", y);
break;
}
}
}

```

1)  $5 + 127 = 132$  (Максимальні значення для signed char -127 – 127. Тому відбувається переповнення і сума досягнувши максимуму в 127 починає заповнюватись з початку із -127.)

2)  $2 - 3 = -1$

3)  $-120 - 34 = -154$  Відбувається переповнення і різниця досягнувши максимуму в -127 починає іти з максимуму із 127.

4) (unsigned char)(-5) (Число -5 перетворюється в незначковий тип, проте змінна в яку записується результат знакова, тому результат знову перетворюється в значковий)

5)  $56 \& 38 = 32$  ( $56 = 111000_2$ ,) ( $38 = 100110_2$ .)

6)  $56 | 38 = 62$  (Побітове або.  $56 = 111000_2$ ,  $38 = 100110_2$ . Якщо хоча б один біт 1, в результаті 1, Результат  $111110_2 = 62$ )

Результат виконання програми:

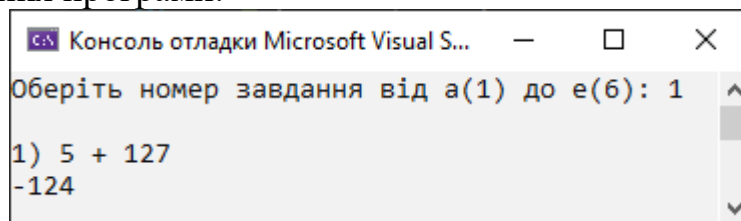


Рисунок №1.3 – Результат виконання програми

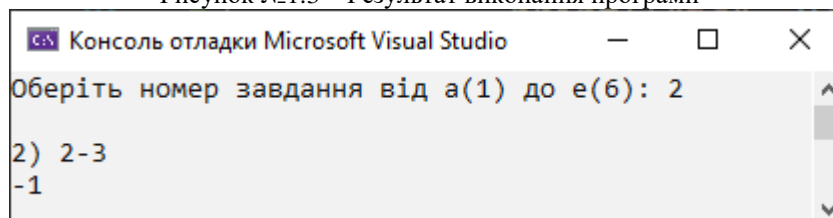


Рисунок №1.4 – Результат виконання програми

```

Консоль отладки Microsoft Visual S...
Оберіть номер завдання від а(1) до е(6): 3
3) -120-34
102
Переповнення біту!

```

Рисунок №1.5 – Результат виконання програми

```

Консоль отладки Microsoft Visual Studio
Оберіть номер завдання від а(1) до е(6): 4
4) (unsigned char(0-255)) (- 5)
Введіть х:10
-5 < 10

```

Рисунок №1.6 – Результат виконання програми

```

Консоль отладки Microsoft Visu...
Оберіть номер завдання від а(1) до е(6): 5
5) 56 & 38
56 > 38

```

Рисунок №1.7 – Результат виконання програми

```

Консоль отладки Microsoft Visual Studio
Оберіть номер завдання від а(1) до е(6): 6
6) 56 | 38
56

```

Рисунок №1.8 – Результат виконання програми

- 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.

Лістинінг програми:

```

#include<stdio.h>
#include<windows.h>

union data {
    float num;
    struct {
        unsigned char b0 : 1;
        unsigned char b1 : 1;
        unsigned char b2 : 1;
        unsigned char b3 : 1;
        unsigned char b4 : 1;
        unsigned char b5 : 1;
        unsigned char b6 : 1;
        unsigned char b7 : 1;
        unsigned char b8 : 1;
        unsigned char b9 : 1;
        unsigned char b10 : 1;
        unsigned char b11 : 1;
        unsigned char b12 : 1;
    };
};

```



```

    unsigned char b13 : 1;
    unsigned char b14 : 1;
    unsigned char b15 : 1;
    unsigned char b16 : 1;
    unsigned char b17 : 1;
    unsigned char b18 : 1;
    unsigned char b19 : 1;
    unsigned char b20 : 1;
    unsigned char b21 : 1;
    unsigned char b22 : 1;
    unsigned char b23 : 1;
    unsigned char b24 : 1;
    unsigned char b25 : 1;
    unsigned char b26 : 1;
    unsigned char b27 : 1;
    unsigned char b28 : 1;
    unsigned char b29 : 1;
    unsigned char b30 : 1;
    unsigned char b31 : 1;
} bits;

struct {
    unsigned char byte0;
    unsigned char byte1;
    unsigned char byte2;
    unsigned char byte3;
} bytes;
};

int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    data number;
    printf("Введіть число: "); scanf_s("%f", &number.num);
    printf("\nВведене число типу float в десятковій системі: number = %f\n", number.num);
    printf("\nВведене число в двійковій системі: number = %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", number.bits.b31, number.bits.b30, number.bits.b29,
number.bits.b28, number.bits.b27, number.bits.b26, number.bits.b25, number.bits.b24, num-
ber.bits.b23, number.bits.b22, number.bits.b21, number.bits.b20, number.bits.b19, num-
ber.bits.b18, number.bits.b17, number.bits.b16, number.bits.b15, number.bits.b14, num-
ber.bits.b13, number.bits.b12, number.bits.b11, number.bits.b10, number.bits.b9, num-
ber.bits.b8, number.bits.b7, number.bits.b6, number.bits.b5, number.bits.b4, number.bits.b3,
number.bits.b2, number.bits.b1, number.bits.b0);
    printf("\nВведене число в шістнадцятковій системі: number = 0x%x%x%x%x\n", num-
ber.bytes.byte3, number.bytes.byte2, number.bytes.byte1, number.bytes.byte0);
    if (number.bits.b31 == 0)
        printf("\nЗнак числа: +.\n");
    else
        printf("\nЗнак числа: -.\n");
    printf("\nМантіса: %d%d%d%d %d%d%d%d %d%d%d%d %d%d%d%d %d%d%d%d %d%d%d\n", num-
ber.bits.b22, number.bits.b21, number.bits.b20, number.bits.b19, number.bits.b18, num-
ber.bits.b17, number.bits.b16, number.bits.b15, number.bits.b14, number.bits.b13, num-
ber.bits.b12, number.bits.b11, number.bits.b10, number.bits.b9, number.bits.b8, num-
ber.bits.b7, number.bits.b6, number.bits.b5, number.bits.b4, number.bits.b3, number.bits.b2,
number.bits.b1, number.bits.b0);
    printf("\nСтупінь числа: %d%d%d%d %d%d%d%d\n", number.bits.b30, number.bits.b29, num-
ber.bits.b28, number.bits.b27, number.bits.b26, number.bits.b25, number.bits.b24, num-
ber.bits.b23);
    printf("\nОбсяг пам'яті, яку займає змінна %d біта\n", sizeof(number));

    return 0;
}
```

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

## Результат виконання програми:

```

Консоль отладки Microsoft Visual Studio
Введіть число: 234
Введене число типу float в десятковій системі: number = 234.000000
Введене число в двійковій системі: number = 0100 0011 0110 1010 0000 0000 0000 0000
Введене число в шістнадцятковій системі: number = 0x436a00
Знак числа: +.
Мантиса: 1101 0100 0000 0000 0000 000
Ступінь числа: 1000 0110
Обсяг пам'яті, яку займає змінна 4 біта
  
```

Рисунок №1.9 – Результат виконання програми

```

Консоль отладки Microsoft Visual Studio
Введіть число: -120.1
Введене число типу float в десятковій системі: number = -120.099998
Введене число в двійковій системі: number = 1100 0010 1111 0000 0011 0011 0011 0011
Введене число в шістнадцятковій системі: number = 0xc2f03333
Знак числа: -.
Мантиса: 1110 0000 0110 0110 0110 011
Ступінь числа: 1000 0101
Обсяг пам'яті, яку займає змінна 4 біта
  
```

Рисунок №1.10 – Результат виконання програми

## Лабораторна робота № 2

### «Генерування послідовності псевдовипадкових значень»

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

## 2.1 Хід роботи

### Завдання:

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.

3	16807	0	$2^{31}-1$	[0, 200)	30000
---	-------	---	------------	----------	-------

Рисунок 2 - Варіант завдання(3)

### Лістинг програми:

```
#include <iostream>
#include "Windows.h"
#include "stdio.h"
#include "math.h"
#define N 30000
int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    unsigned int x0, a = 16807, c = 0, k = 30000;
    int arr[N], arr1[250];
    float x = 1;
    int m = (pow(2, 31) - 1);
    printf("Введіть початкове значення x:");
    scanf_s("%d", &x0);
    for (int i = 0; i < k; i++)
    {
        x0 = (x0 * a + c) % m;
        x = ((float)x0 / m * 200);
        arr[i] = x;
        printf(" %d", arr[i]);
    }
    printf("\n");
    printf("\nЧастота інтервалів появи випадкових величин:\n");
    int f = 0, g = 0;
    float arr2[200];
    for (int h = 0; h < 200; h++)
    {
        for (int i = 0; i < k; i++)
        {
            if (arr[i] == h)
                f++;
        }
        printf(" (%d) | %d\n", h, f);
        arr1[h] = f;
        f = 0;
    }
}
```

		Башиманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Консоль отладки Microsoft Visual Studio

Введи́те по́чатко́е значе́ния x:5

0 131 155 41 161 25 174 80 129 139 182 46 149 39 116 29 0 81 16 175 129 143 39 139 21 88 175 2 190 69 34 36 91 133 151 60 21 80 173 59 96 126 20 103 135 135 167 162 158 62 11 25 34 106 105 128 47 161 182 114 1 31 19 199 13 97 68 67 35 92 43 52 68 76 156 26 42 82 31 89 185 97 24 69 14 166 30 61 172 122 54 112 40 117 174 152 144 30 199 141 11 91 55 98 188 35 41 167 144 180 19 113 3 73 128 196 46 141 192 154 85 129 116 53 78 118 20 191 115 76 99 1 49 58 12 101 167 52 30 7 167 76 159 153 149 145 32 190 52 123 192 118 180 34 188 107 88 177 59 61 155 44 0 16 107 19 13 156 14 142 92 40 170 56 113 12 2 159 130 50 158 85 133 88 94 103 112 71 44 41 195 151 38 130 135 27 78 193 31 50 157 107 183 116 85 165 133 65 192 141 166 57 30 175 182 87 57 94 165 74 0 83 121 38 70 25 14 178 193 182 117 164 28 165 134 88 47 116 182 24 130 46 93 147 7 7 99 49 62 143 118 155 48 126 59 90 59 122 94 154 172 49 114 181 72 30 66 49 90 12 143 174 130 62 184 180 199 44 112 62 106 25 13 6 197 75 126 2 158 139 71 193 65 113 163 114 11 76 155 93 3 81 21 129 5 46 78 173 168 88 43 184 16 178 67 70 179 17 16 94 127 132 25 54 175 101 23 123 123 81 179 79 88 90 7 126 33 60 92 158 193 40 37 45 150 78 154 145 56 83 184 80 102 36 154 91 39 106 166 87 32 18 103 117 79 177 31 121 176 67 34 196 9 78 177 98 135 161 54 53 12 39 146 23 72 0 199 68 157 178 103 65 7 23 198 172 7 0 8 71 55 75 39 4 104 183 119 134 98 39 63 79 27 27 79 25 28 106 195 160 28 134 3 10 188 67 173 91 76 31 75 122 169 20 191 109 38 52 12 107 121 152 19 6 25 122 154 57 186 3 32 83 61 137 74 69 135 5 74 147 128 41 169 193 129 132 15 72 160 101 51 112 127 140 75 52 62 94 64 156 185 189 59 153 182 125 19 95 118 126 98 45 124 34 154 181 63 160 59 28 91 122 173 15 39 31 183 40 10 70 152 150 7 81 63 3 149 40 51 87 80 98 62 50 60 119 30 172 197 27 2 102 27 133 19 79 198 58 175 152 101 126 46 103 64 187 135 111 13 142 101 32 77 47 72 58 180 139 79 127 87 124 85 158 86 67 94 71 23 143 90 21 18 71 179 16 9 95 16 168 77 128 186 178 142 133 20 108 15 181 73 32 96 142 191 20 25 101 20 63 58 189 36 166 47 137 128 166 39 163 197 16 101 38 22 94 99 83 94 36 26 95 143 163 95 161 26 104 33 88 174 99 84 126 28 171 11 178 192 33 134 174 61 116 163 54 144 119 79 8 45 145 129 139 125 43 189 57 152 193 140 61 154 5 135 195 146 91 61 19 55 143 179 51 180 128 84 67 54 175 81 82 125 180 33 173 118 9 60 23 100 44 26 20 120 186 118 154 151 65 165 55 14 130 59 1 99 41 275 29 186 187 148 43 104 136 94 150 7 155 188 161 110 196 110 93 162 68 163 157 43 8 113 112 196 26 1 115 93 170 128 81 191 45 56 153 121 18 142 75 111 71 144 160 15 136 142 25 102 153 57 28 150 113 74 89 88 31 77 168 70 183 100 189 129 129 57 29 128 68 178 24 83 166 92 48 60 171 196 11 158 47 172 108 64 81 140 79 162 3 2 148 156 199 74 158 121 124 13 142 53 90 102 183 28 51 167 110 27 124 51 88 126 75 8 96 148 191 102 139 147 177 1 3 134 139 128 168 168 131 165 21 127 96 125 189 37 11 164 155 75 149 43 154 190 124 35 155 196 170 186 136 152 84 16 121 106 117 22 107 171 83 184 14 167 125 42 102 8 84 75 92 163 52 137 105 29 175 144 82 193 198 55 9 111 38 58 60 12 60 7 133 35 59 195 56 83 13 190 14 144 154 82 178 38 161 162 164 102 167 36 99 179 184 49 14 159 76 32 83 115 80 57 25 189 9 57 79 159 11 42 33 37 136 168 95 134 77 8 1 159 171 75 117 118 167 85 52 162 159 85 85 98 120 154 61 99 181 48 183 26 12 91 63 85 135 143 149 82 168 3 91 1 179 88 97 194 36 100 8 39 33 4 162 23 174 58 37 14 176 151 102 84 51 148 76 1 57 48 41 198 79 119 76 37 9 77 50 151 137 15 126 44 44 101 144 105 54 191 165 62 52 158 50 190 8 34 99 138 115 110 6 79 113 7 177 191 176 129 186 8 16 1 10 110 126 112 94 123 163 9 77 335 347 144 126 14 135 10 16 56 60 127 139 151 156 132 111 14 173 19 120 96 174 115 72 112 142 177 67 111 125 112 177 105 155 142 29 179 3 3 101 104 119 141 42 139 8 121 114 137 99 31 78 59 181 104 198 88 135 118 30 48 25 142 80 86 163 30 107 126 123 116 142 180 67 126 170 199 5 99 111 16 120 61 121 119 91 35 154 157 63 77 185 138 195 89 92 199 35 18 69 176 83 115 78 191 129 175 62 93 15 143 121 147 59 47 1

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

Частота інтервалів появи випадкових величин (інтервал дорівнює 1):

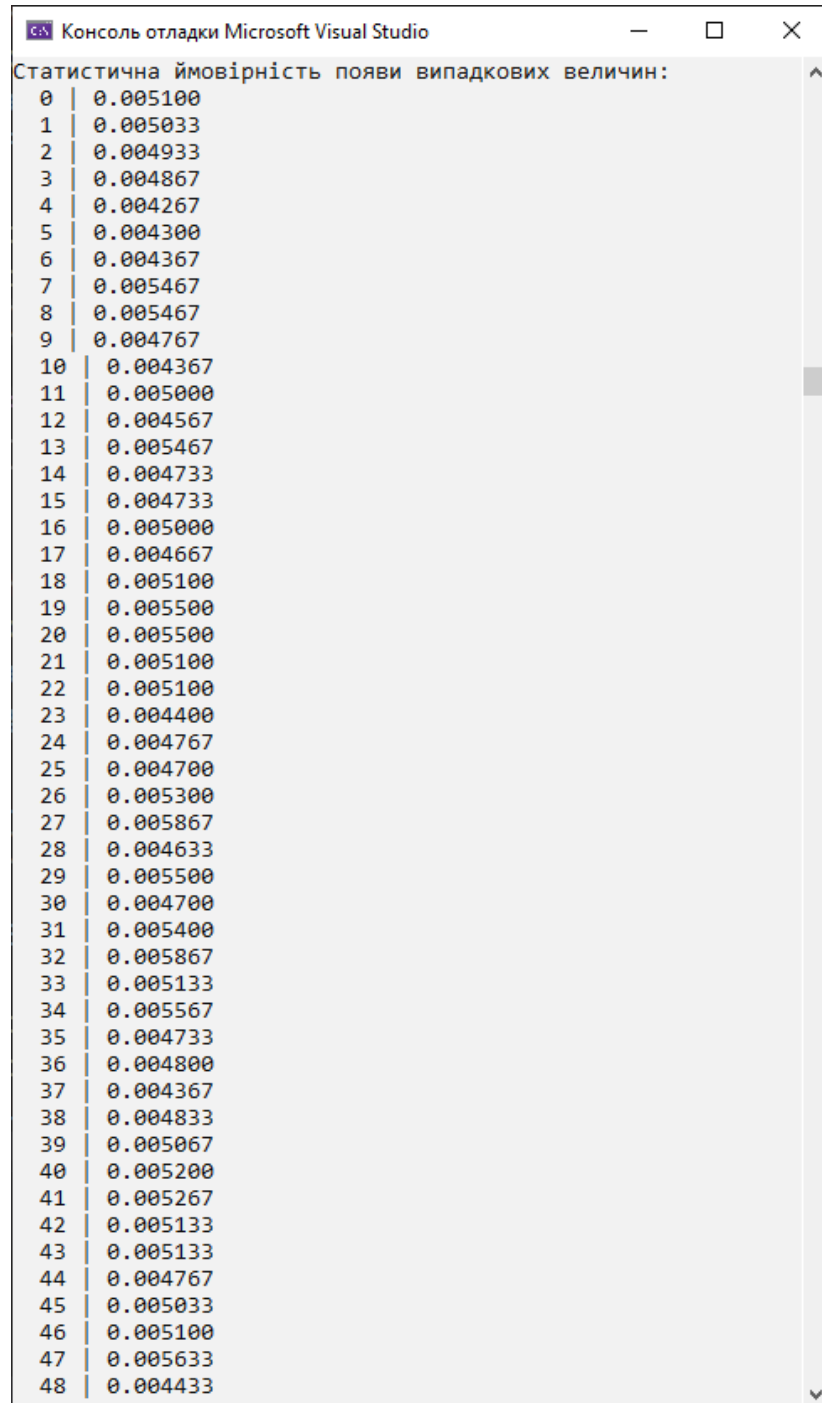
```
int f = 0, g = 0;
float arr2[200];
for (int h = 0; h < 200; h++)
{
    for (int i = 0; i < k; i++)
    {
        if (arr[i] == h)
            f++;
    }
    printf("  (%d) | %d\n", h, f);
    arr1[h] = f;
    f = 0;
}
```

```
Консоль отладки Microsoft Visual Studio
Частота інтервалів появи випадкових величин:
(0) | 153
(1) | 151
(2) | 148
(3) | 146
(4) | 128
(5) | 129
(6) | 131
(7) | 164
(8) | 164
(9) | 143
(10) | 131
(11) | 150
(12) | 137
(13) | 164
(14) | 142
(15) | 142
(16) | 150
(17) | 140
(18) | 153
(19) | 165
(20) | 165
(21) | 153
(22) | 153
(23) | 132
(24) | 143
(25) | 141
(26) | 159
(27) | 176
(28) | 139
(29) | 165
(30) | 141
(31) | 162
(32) | 176
(33) | 154
(34) | 167
(35) | 142
(36) | 144
(37) | 131
(38) | 145
(39) | 152
(40) | 156
(41) | 158
(42) | 154
(43) | 154
(44) | 143
(45) | 151
(46) | 153
(47) | 169
(48) | 133
```

Рисунок 2.2 - Частота інтервалів появи випадкових величин

Статистична імовірність появи випадкових величин:

```
for (int h = 0; h < 200; h++)
{
    for (int i = 0; i < k; i++)
    {
        if (arr[i] == h)
            g++;
    }
    printf(" %d | %lf\n", h, (float)g / 30000);
    arr2[h] = (float)g / 30000;
    g = 0;
}
```



```
Статистична ймовірність появи випадкових величин:
0 | 0.005100
1 | 0.005033
2 | 0.004933
3 | 0.004867
4 | 0.004267
5 | 0.004300
6 | 0.004367
7 | 0.005467
8 | 0.005467
9 | 0.004767
10 | 0.004367
11 | 0.005000
12 | 0.004567
13 | 0.005467
14 | 0.004733
15 | 0.004733
16 | 0.005000
17 | 0.004667
18 | 0.005100
19 | 0.005500
20 | 0.005500
21 | 0.005100
22 | 0.005100
23 | 0.004400
24 | 0.004767
25 | 0.004700
26 | 0.005300
27 | 0.005867
28 | 0.004633
29 | 0.005500
30 | 0.004700
31 | 0.005400
32 | 0.005867
33 | 0.005133
34 | 0.005567
35 | 0.004733
36 | 0.004800
37 | 0.004367
38 | 0.004833
39 | 0.005067
40 | 0.005200
41 | 0.005267
42 | 0.005133
43 | 0.005133
44 | 0.004767
45 | 0.005033
46 | 0.005100
47 | 0.005633
48 | 0.004433
```

Рисунок 2.3 - Статистична імовірність появи випадкових величин

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

Математичне сподівання випадкових величин:

```
double Expvalue = 0;
for (int i = 0; i < 200; i++)
{
    Expvalue += arr[i] * arr2[i];
}
```

Дисперсія випадкових величин:

```
double variance = 0;
for (int i = 0; i < k; i++)
{
    variance += pow(arr[i] - Expvalue, 2) * arr[i];
}
```

Середньоквадратичне відхилення випадкових величин:

```
double deviation = 0;
deviation = sqrt(variance);
```

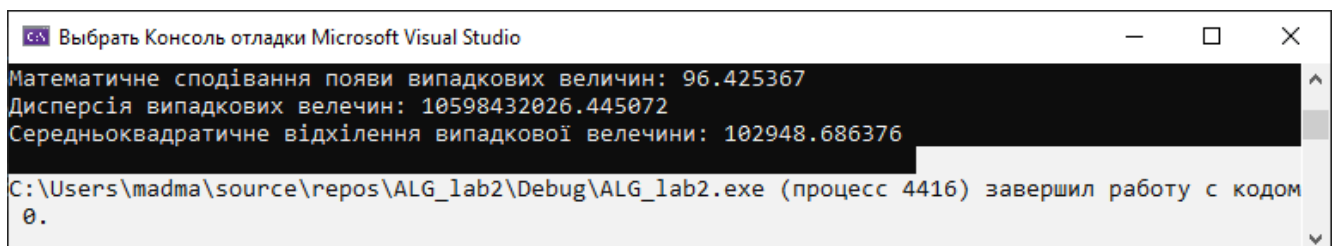


Рисунок 2.4 - Математичне сподівання випадкових величин , дисперсія випадкових величин , середньоквадратичне відхилення випадкових величин

		Башиманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

## Лабораторна робота № 3

### Оцінка часової складності алгоритмів

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

### 3.1 Хід роботи

Завдання 1: Написати програму для табулювання наступних функцій:  $f(n)=n$ ;  $f(n)=\log(n)$ ;  $f(n)=n \cdot \log(n)$ ;  $f(n)=n^2$ ;  $f(n)=2n$ ;  $f(n)=n!$ . 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.

Лістинг програми:

```
#include<stdio.h>
#include<windows.h>
#include <iostream>
#include <chrono>
#include <math.h>
#define GETTIME std::chrono::steady_clock:: now
#define CALCTIME std::chrono::duration_cast<std::chrono::nanoseconds>

unsigned __int64 factorial(int count)
{
    unsigned __int64 vlfct = 1;
    for (int i = 1; i <= count; i++)
        vlfct = vlfct * i;
    return vlfct;
}
void func_1()
{
    int y = 0;
    for (int x = 0; x <= 50; x++)
    {
        y = x;
        printf("\ny = %d , x = %d", y, x);
    }
}
void func_2()
{
    float y = 0;
    for (int x = 0; x <= 50; x++)
    {
        y = log(x);
        printf("\ny = %f , x = %d", y, x);
    }
}
void func_3()
{
    float y = 0;
    for (int x = 0; x <= 50; x++)
    {
        y = x * log(x);
        printf("\ny = %f , x = %d", y, x);
    }
}
void func_4()
{
    int y = 0;
    for (int x = 0; x <= 50; x++)
    {
```

		Башиманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				15
Змн.	Арк.	№ докум.	Підпис	Дата		



```

        y = pow(x,2);
        printf("\ny = %d , x = %d", y, x);
    }
}
void func_5()
{
    long long int y = 0;
    for (int x = 0; x <= 50; x++)
    {
        y = pow(2, x);
        printf("\ny = %lld , x = %d", y, x);
    }
}
void func_6()
{
    unsigned __int64 y = 0;
    for (int x = 0; x <= 50; x++)
    {
        y = factorial(x);
        printf("\ny = %llu , x = %d", y, x);
    }
}
int main()
{
    auto begin = GETTIME();
    func_6();
    auto end = GETTIME();
    auto elapsed_ns = CALCTIME(end - begin);
    printf("\nThe time: %lld ns\n", elapsed_ns.count());
}

```

		Башиманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

Результат виконання програми:

```

Консоль отладки Microso...
y = 0 , x = 0
y = 1 , x = 1
y = 2 , x = 2
y = 3 , x = 3
y = 4 , x = 4
y = 5 , x = 5
y = 6 , x = 6
y = 7 , x = 7
y = 8 , x = 8
y = 9 , x = 9
y = 10 , x = 10
y = 11 , x = 11
y = 12 , x = 12
y = 13 , x = 13
y = 14 , x = 14
y = 15 , x = 15
y = 16 , x = 16
y = 17 , x = 17
y = 18 , x = 18
y = 19 , x = 19
y = 20 , x = 20
y = 21 , x = 21
y = 22 , x = 22
y = 23 , x = 23
y = 24 , x = 24
y = 25 , x = 25
y = 26 , x = 26
y = 27 , x = 27
y = 28 , x = 28
y = 29 , x = 29
y = 30 , x = 30
y = 31 , x = 31
y = 32 , x = 32
y = 33 , x = 33
y = 34 , x = 34
y = 35 , x = 35
y = 36 , x = 36
y = 37 , x = 37
y = 38 , x = 38
y = 39 , x = 39
y = 40 , x = 40
y = 41 , x = 41
y = 42 , x = 42
y = 43 , x = 43
y = 44 , x = 44
y = 45 , x = 45
y = 46 , x = 46
y = 47 , x = 47
y = 48 , x = 48
y = 49 , x = 49
y = 50 , x = 50
The time: 4250900 ns
  
```

Рисунок 4.1 - Табулювання функції  $f(n)=n$

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

```

y = 0.000000 , x = 1
y = 1.386294 , x = 2
y = 3.295837 , x = 3
y = 5.545177 , x = 4
y = 8.047190 , x = 5
y = 10.750557 , x = 6
y = 13.621371 , x = 7
y = 16.635532 , x = 8
y = 19.775021 , x = 9
y = 23.025850 , x = 10
y = 26.376848 , x = 11
y = 29.818880 , x = 12
y = 33.344341 , x = 13
y = 36.946804 , x = 14
y = 40.620754 , x = 15
y = 44.361420 , x = 16
y = 48.164627 , x = 17
y = 52.026691 , x = 18
y = 55.944340 , x = 19
y = 59.914646 , x = 20
y = 63.934971 , x = 21
y = 68.002937 , x = 22
y = 72.116364 , x = 23
y = 76.273293 , x = 24
y = 80.471893 , x = 25
y = 84.710510 , x = 26
y = 88.987595 , x = 27
y = 93.301727 , x = 28
y = 97.651581 , x = 29
y = 102.035919 , x = 30
y = 106.453606 , x = 31
y = 110.903549 , x = 32
y = 115.384750 , x = 33
y = 119.896255 , x = 34
y = 124.437180 , x = 35
y = 129.006683 , x = 36
y = 133.603958 , x = 37
y = 138.228271 , x = 38
y = 142.878906 , x = 39
y = 147.555176 , x = 40
y = 152.256454 , x = 41
y = 156.982117 , x = 42
y = 161.731598 , x = 43
y = 166.504349 , x = 44
y = 171.299805 , x = 45
y = 176.117508 , x = 46
y = 180.956940 , x = 47
y = 185.817642 , x = 48
y = 190.699188 , x = 49
y = 195.601151 , x = 50
The time: 6067700 ns

```

Рисунок 4.2 – Табулювання функції  $f(n)=\log(n)$

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

```

y = 0.000000 , x = 1
y = 1.386294 , x = 2
y = 3.295837 , x = 3
y = 5.545177 , x = 4
y = 8.047190 , x = 5
y = 10.750557 , x = 6
y = 13.621371 , x = 7
y = 16.635532 , x = 8
y = 19.775021 , x = 9
y = 23.025850 , x = 10
y = 26.376848 , x = 11
y = 29.818880 , x = 12
y = 33.344341 , x = 13
y = 36.946804 , x = 14
y = 40.620754 , x = 15
y = 44.361420 , x = 16
y = 48.164627 , x = 17
y = 52.026691 , x = 18
y = 55.944340 , x = 19
y = 59.914646 , x = 20
y = 63.934971 , x = 21
y = 68.002937 , x = 22
y = 72.116364 , x = 23
y = 76.273293 , x = 24
y = 80.471893 , x = 25
y = 84.710510 , x = 26
y = 88.987595 , x = 27
y = 93.301727 , x = 28
y = 97.651581 , x = 29
y = 102.035919 , x = 30
y = 106.453606 , x = 31
y = 110.903549 , x = 32
y = 115.384750 , x = 33
y = 119.896255 , x = 34
y = 124.437180 , x = 35
y = 129.006683 , x = 36
y = 133.603958 , x = 37
y = 138.228271 , x = 38
y = 142.878906 , x = 39
y = 147.555176 , x = 40
y = 152.256454 , x = 41
y = 156.982117 , x = 42
y = 161.731598 , x = 43
y = 166.504349 , x = 44
y = 171.299805 , x = 45
y = 176.117508 , x = 46
y = 180.956940 , x = 47
y = 185.817642 , x = 48
y = 190.699188 , x = 49
y = 195.601151 , x = 50
The time: 5549000 ns

```

Рисунок 4.3 – Табулювання функції  $f(n)=n \cdot \log(n)$

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

```
Консоль отла...
y = 0 , x = 0
y = 1 , x = 1
y = 4 , x = 2
y = 9 , x = 3
y = 16 , x = 4
y = 25 , x = 5
y = 36 , x = 6
y = 49 , x = 7
y = 64 , x = 8
y = 81 , x = 9
y = 100 , x = 10
y = 121 , x = 11
y = 144 , x = 12
y = 169 , x = 13
y = 196 , x = 14
y = 225 , x = 15
y = 256 , x = 16
y = 289 , x = 17
y = 324 , x = 18
y = 361 , x = 19
y = 400 , x = 20
y = 441 , x = 21
y = 484 , x = 22
y = 529 , x = 23
y = 576 , x = 24
y = 625 , x = 25
y = 676 , x = 26
y = 729 , x = 27
y = 784 , x = 28
y = 841 , x = 29
y = 900 , x = 30
y = 961 , x = 31
y = 1024 , x = 32
y = 1089 , x = 33
y = 1156 , x = 34
y = 1225 , x = 35
y = 1296 , x = 36
y = 1369 , x = 37
y = 1444 , x = 38
y = 1521 , x = 39
y = 1600 , x = 40
y = 1681 , x = 41
y = 1764 , x = 42
y = 1849 , x = 43
y = 1936 , x = 44
y = 2025 , x = 45
y = 2116 , x = 46
y = 2209 , x = 47
y = 2304 , x = 48
y = 2401 , x = 49
y = 2500 , x = 50
The time: 5938800 ns
```

Рисунок 4.4 – Табулювання функції  $f(n)=n^2$

```

Консоль отладки Microsoft Visual Studio
y = 1 , x = 0
y = 2 , x = 1
y = 4 , x = 2
y = 8 , x = 3
y = 16 , x = 4
y = 32 , x = 5
y = 64 , x = 6
y = 128 , x = 7
y = 256 , x = 8
y = 512 , x = 9
y = 1024 , x = 10
y = 2048 , x = 11
y = 4096 , x = 12
y = 8192 , x = 13
y = 16384 , x = 14
y = 32768 , x = 15
y = 65536 , x = 16
y = 131072 , x = 17
y = 262144 , x = 18
y = 524288 , x = 19
y = 1048576 , x = 20
y = 2097152 , x = 21
y = 4194304 , x = 22
y = 8388608 , x = 23
y = 16777216 , x = 24
y = 33554432 , x = 25
y = 67108864 , x = 26
y = 134217728 , x = 27
y = 268435456 , x = 28
y = 536870912 , x = 29
y = 1073741824 , x = 30
y = 2147483648 , x = 31
y = 4294967296 , x = 32
y = 8589934592 , x = 33
y = 17179869184 , x = 34
y = 34359738368 , x = 35
y = 68719476736 , x = 36
y = 137438953472 , x = 37
y = 274877906944 , x = 38
y = 549755813888 , x = 39
y = 1099511627776 , x = 40
y = 2199023255552 , x = 41
y = 4398046511104 , x = 42
y = 8796093022208 , x = 43
y = 17592186044416 , x = 44
y = 35184372088832 , x = 45
y = 70368744177664 , x = 46
y = 140737488355328 , x = 47
y = 281474976710656 , x = 48
y = 562949953421312 , x = 49
y = 1125899906842624 , x = 50
The time: 5543900 ns

```

Рисунок 4.5 – Табулювання функції  $f(n)=2^n$

```

Консоль отладки Microsoft Visual S...
y = 1 , x = 0
y = 1 , x = 1
y = 2 , x = 2
y = 6 , x = 3
y = 24 , x = 4
y = 120 , x = 5
y = 720 , x = 6
y = 5040 , x = 7
y = 40320 , x = 8
y = 362880 , x = 9
y = 3628800 , x = 10
y = 39916800 , x = 11
y = 479001600 , x = 12
y = 6227020800 , x = 13
y = 87178291200 , x = 14
y = 1307674368000 , x = 15
y = 20922789888000 , x = 16
y = 355687428096000 , x = 17
y = 6402373705728000 , x = 18
y = 121645100408832000 , x = 19
y = 2432902008176640000 , x = 20
y = 14197454024290336768 , x = 21
y = 17196083355034583040 , x = 22
y = 8128291617894825984 , x = 23
y = 10611558092380307456 , x = 24
y = 7034535277573963776 , x = 25
y = 16877220553537093632 , x = 26
y = 12963097176472289280 , x = 27
y = 12478583540742619136 , x = 28
y = 11390785281054474240 , x = 29
y = 9682165104862298112 , x = 30
y = 4999213071378415616 , x = 31
y = 12400865694432886784 , x = 32
y = 3400198294675128320 , x = 33
y = 4926277576697053184 , x = 34
y = 6399018521010896896 , x = 35
y = 9003737871877668864 , x = 36
y = 1096907932701818880 , x = 37
y = 4789013295250014208 , x = 38
y = 2304077777655037952 , x = 39
y = 18376134811363311616 , x = 40
y = 15551764317513711616 , x = 41
y = 7538058755741581312 , x = 42
y = 10541877243825618944 , x = 43
y = 2673996885588443136 , x = 44
y = 9649395409222631424 , x = 45
y = 1150331055211806720 , x = 46
y = 17172071447535812608 , x = 47
y = 12602690238498734080 , x = 48
y = 8789267254022766592 , x = 49
y = 15188249005818642432 , x = 50
The time: 5881900 ns

```

Рисунок 4.6 – Табулювання функції  $f(n)=n!$

При табулюванні функції  $f(n)=n!$  відбувається переповнення після  $x=22$  так як unsigned long long має діапазон значень від 0 до 18 446 744 073 709 551 615.

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

## Графіки функцій

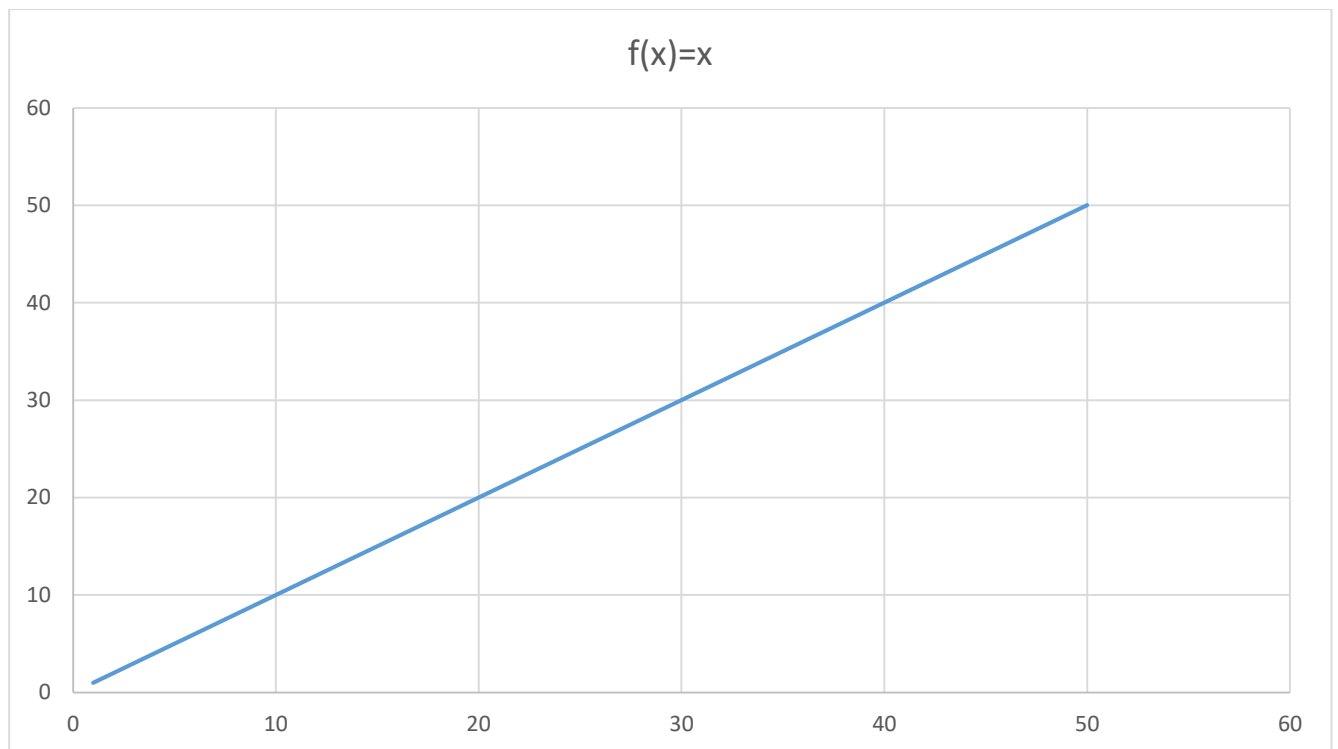


Рисунок 4.7 – Графік функції  $f(n)=n$

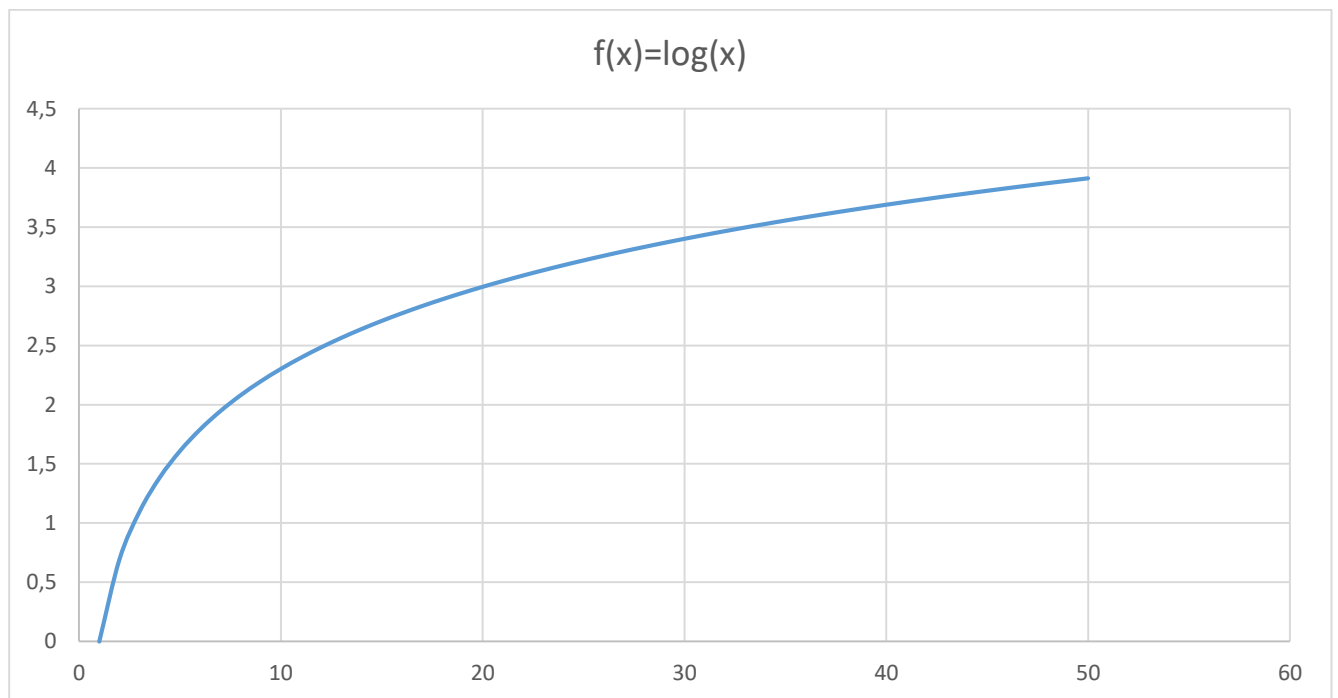


Рисунок 4.8 – Графік функції  $f(n)=\log(n)$



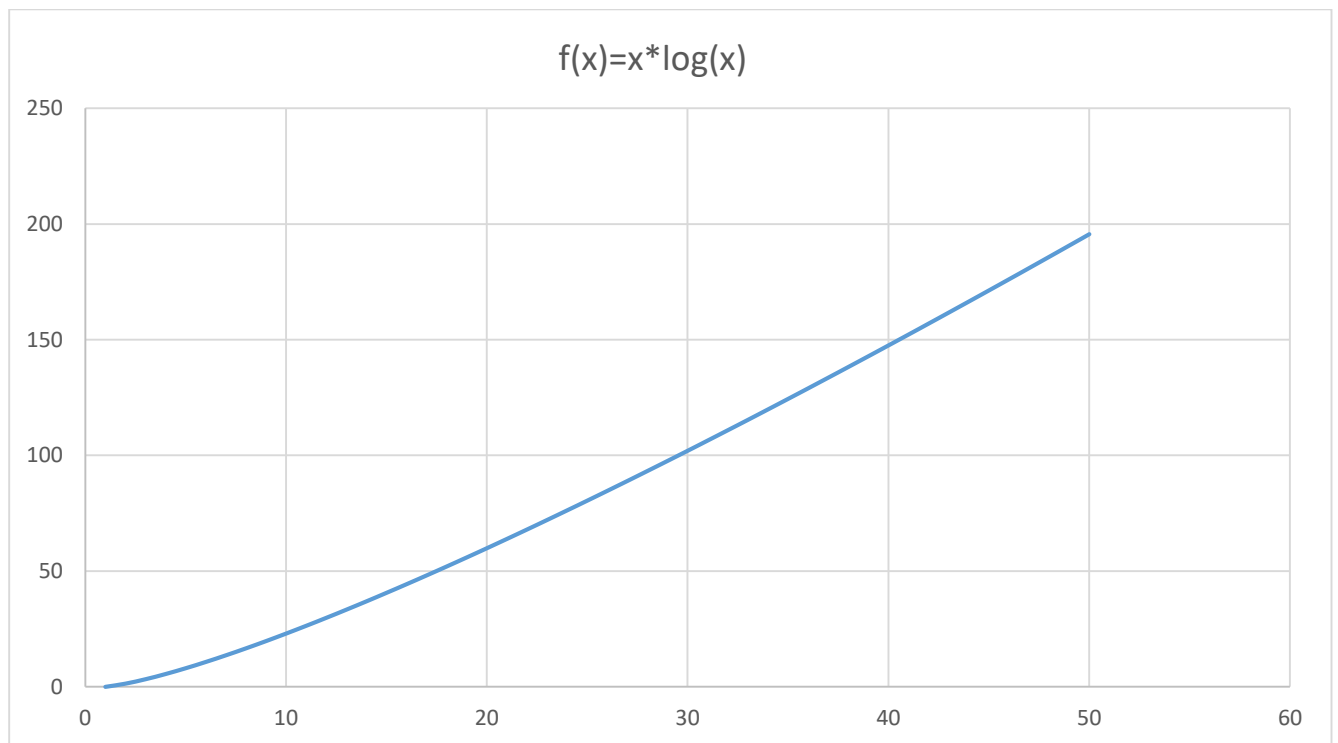


Рисунок 4.9 – Графік функції  $f(n)=n * \log(n)$

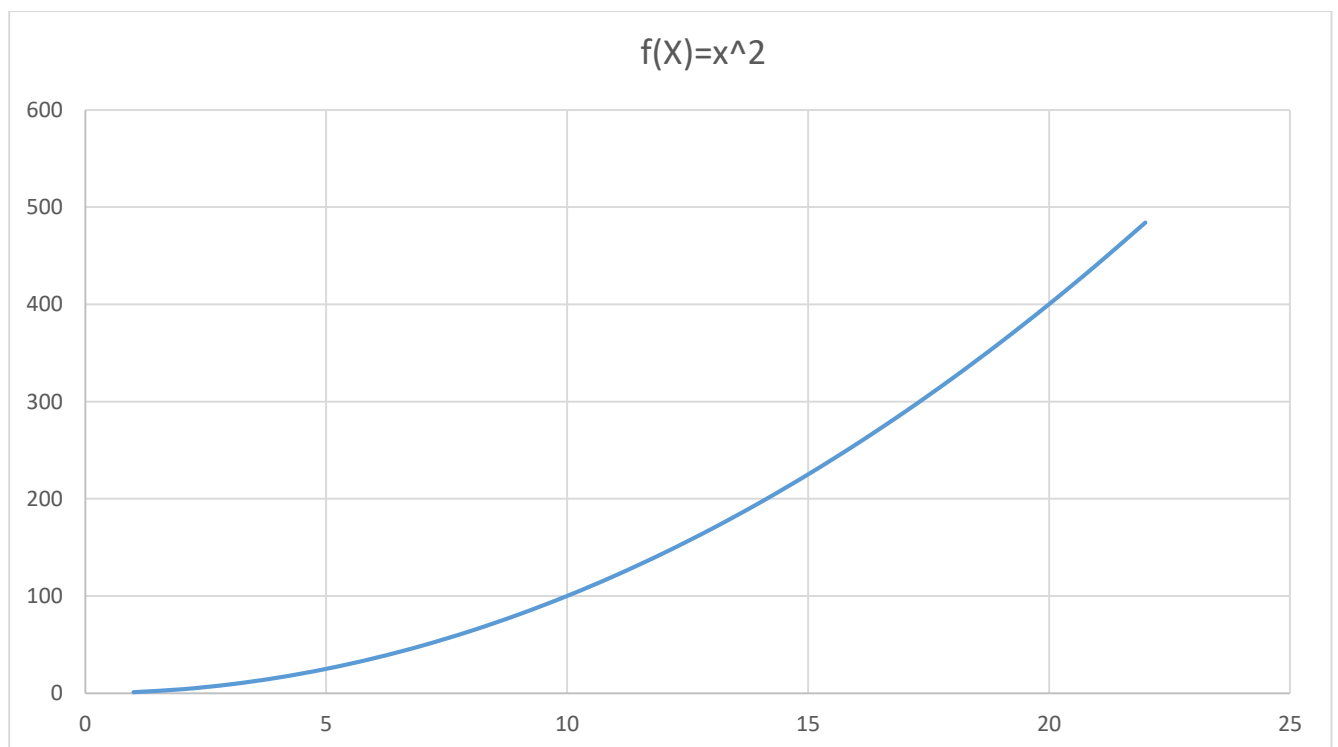


Рисунок 4.10 – Графік функції  $f(n)=n^2$

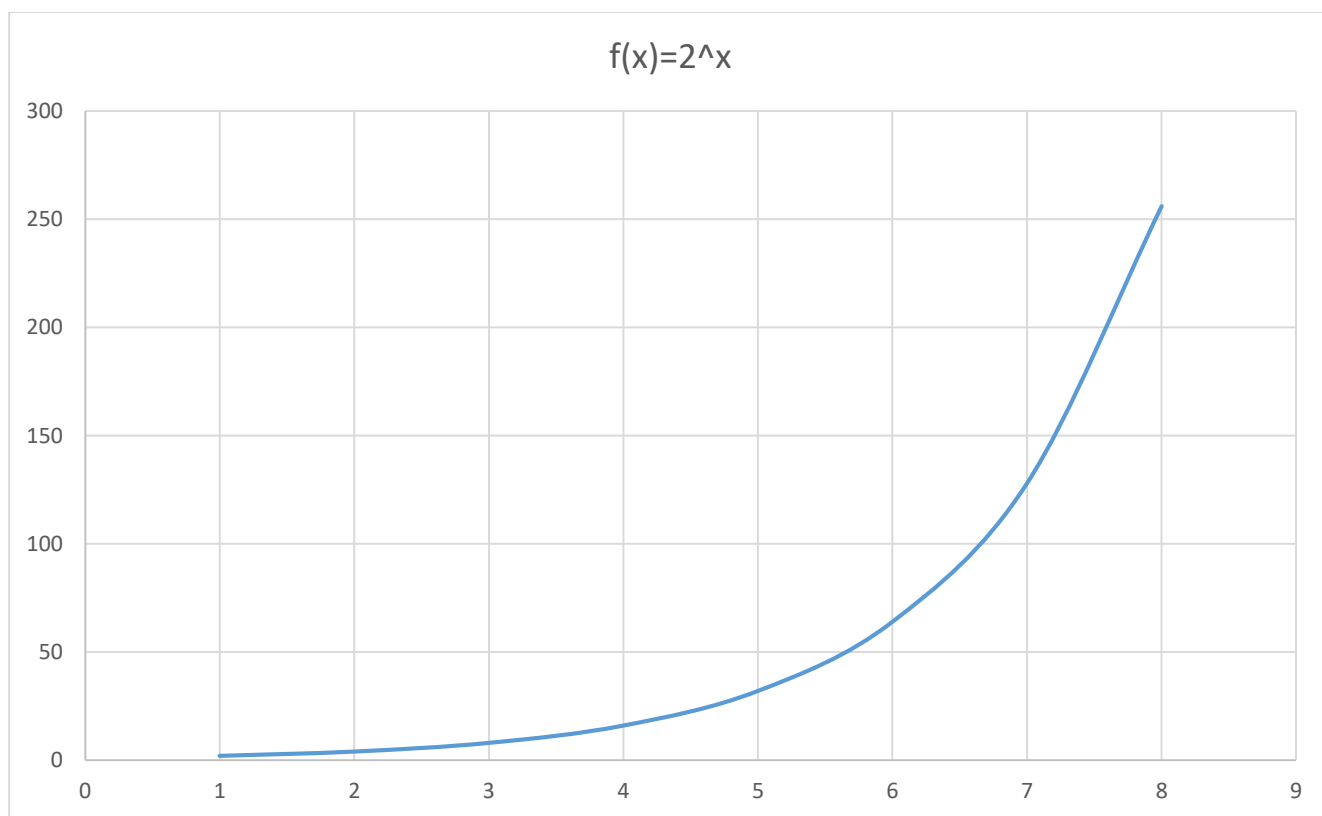


Рисунок 4.11 – Графік функції  $f(n)=2^n$

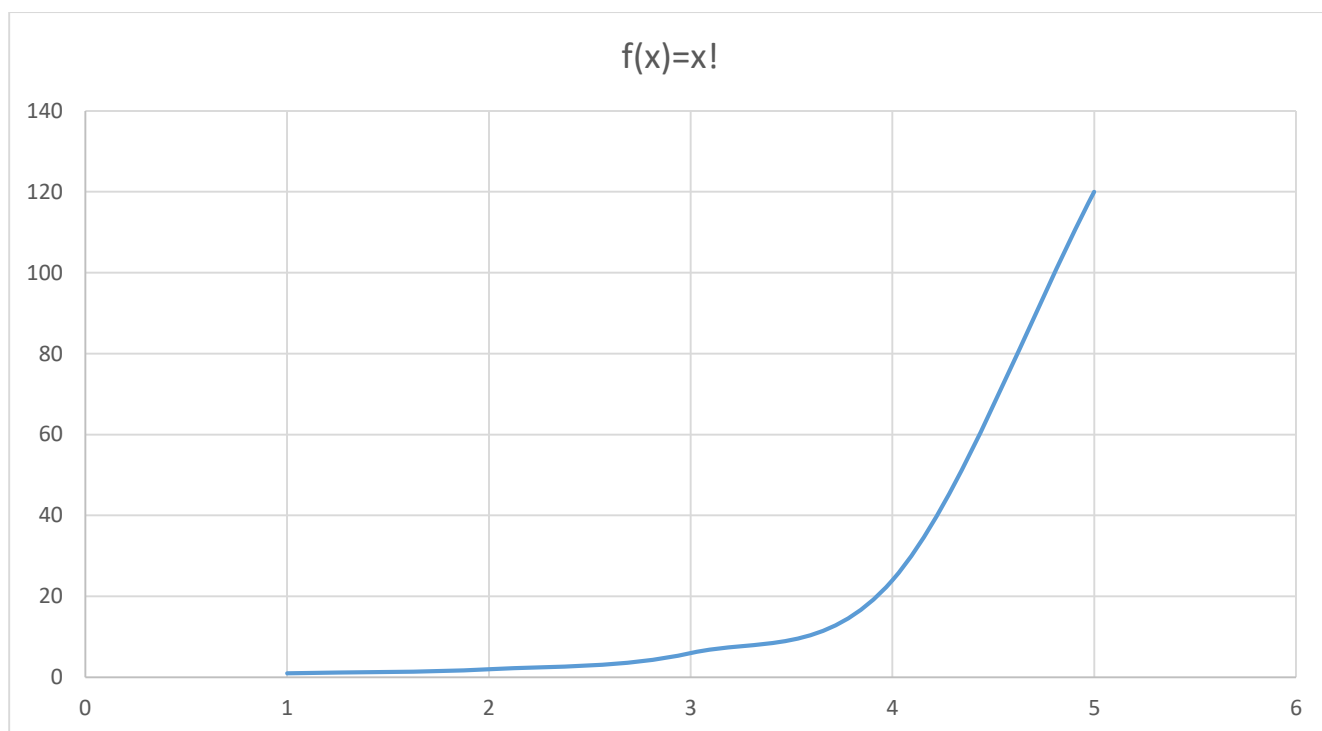


Рисунок 4.12 – Графік функції  $f(n)=n!$

Завдання 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

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

оцінку часової складності алгоритмів. Порівняти практично отримані результати з оцінкою часової складності алгоритмів.

№ варіанту	Номери задач
1	1, 3
2	2, 5
3	4, 8

Рисунок 4.13 – Номер варіанту

4	Реалізувати функцію обчислення $n$ -го числа Фібоначчі, де $n \leq 90$ , $n$ – ціле число. Обраховуються числа Фібоначчі за виразом: $F_0=0$ , $F_1=1$ , $F_n=F_{n-1}+F_{n-2}$ , де $n \geq 2$ .
---	--

Рисунок 4.14 - Варіант завдання 1

8	Дан масив чисел типу float. Обсяг масиву $m \leq 1000$ . Реалізувати функцію бульбашкового сортування масиву в порядку спадання чисел.
---	--

Рисунок 4.15 - Варіант завдання 2

Лістинг програми 1:

```
#include<stdio.h>
#include<windows.h>
#include <iostream>
#include <chrono>
#include <math.h>
#define GETTIME std::chrono::steady_clock::now
#define CALCTIME std::chrono::duration_cast<std::chrono::nanoseconds>

int main()
{
    auto begin = GETTIME();
    printf("Послідовність Фібоначчі : \n0\n1");
    unsigned long long a1 = 0, a2 = 1, temp;
    for (int n = 2; n <= 90; n++)
    {
        temp = a2;
        a2 = a1 + a2;
        a1 = temp;
        printf(" \n%llu", a2);
    }

    auto end = GETTIME();
    auto elapsed_ns = CALCTIME(end - begin);
    printf("\nThe time: %lld ns\n", elapsed_ns.count());
}
```

		Башиманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

Результат виконання програми:

```

Выбрать Консоль отладки Microsoft Visual Studio
23416728348467685
37889062373143906
61305790721611591
99194853094755497
160500643816367088
259695496911122585
420196140727489673
679891637638612258
1100087778366101931
1779979416004714189
2880067194370816120
The time: 30659200 ns
  
```

Рисунок 4.15 – Результат виконання програми 1

Аналіз і оцінка часової складності алгоритмів:

$$t(10) = 535900 \text{ ns}$$

$$T(n) = Ct \cdot t(n), \text{ де } t(n) = O(n)$$

$$T(n) \leq Ct \cdot Cxg(n) = Ct \cdot Cx \cdot n$$

$$t_2/t_1 \leq Ct \cdot Cx \cdot n_2 / Ct \cdot Cx \cdot n_1 = n_2/n_1.$$

$$t_2 \leq t_1 \cdot n_2/n_1 = 535900 \cdot 80/10 = 4287200 \text{ ns}$$

За стільки часу має виконуватись алгоритм – послідовність Фібоначчі з 80 елементів

Послідовність Фібоначчі має оцінку складності алгоритму –  $O(n)$  так як зі збільшенням вхідних даних час роботи збільшується лінійно.

Практично отримане значення(середнє арифметичне 5 дослідів) - 9887200 ns

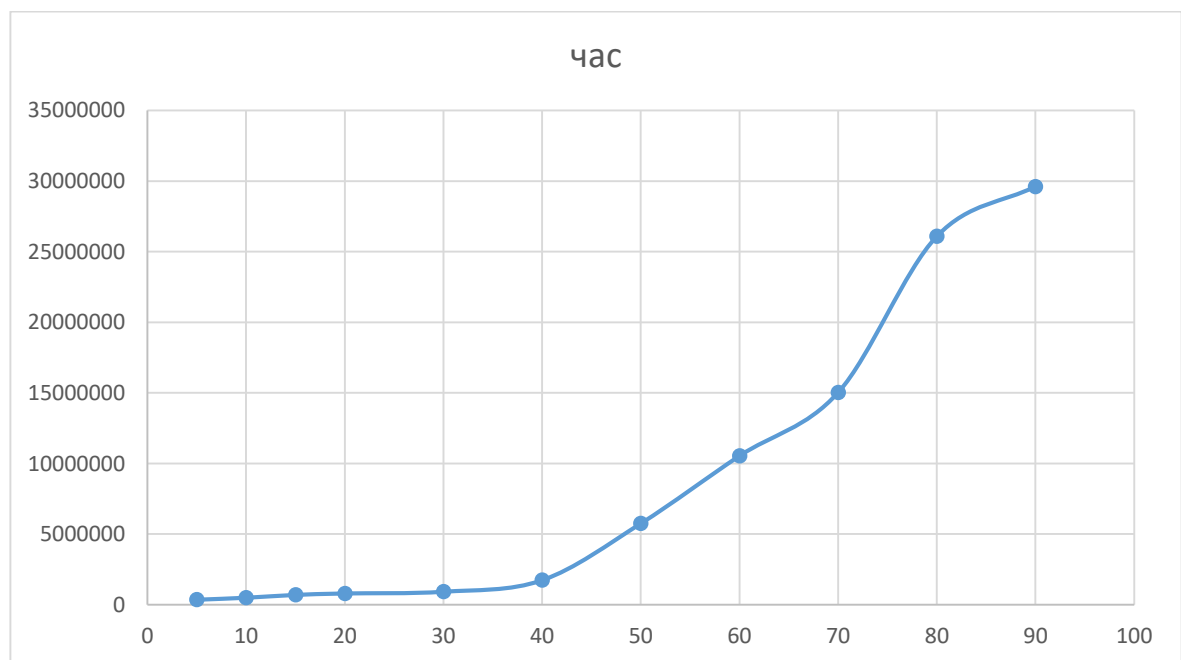


Рисунок 4.16 - Графік часу і вхідних даних

Лістинг програми 2:

```
#include<stdio.h>
#include<windows.h>
#include <iostream>
#include <chrono>
#include <math.h>
#define GETTIME std::chrono::steady_clock:: now
#define CALCTIME std::chrono::duration_cast<std::chrono::nanoseconds>
#define MAX_SIZE 1000
void bubbleSort(double* array, int size)
{
    for (int i = 0; i < size - 1; i++)
    {
        for (int j = (size - 1); j > i; j--)
        {
            if (array[j - 1] < array[j])
            {
                double temp = array[j - 1];
                array[j - 1] = array[j];
                array[j] = temp;
            }
        }
    }
}

int main()
{
    auto begin = GETTIME();
    srand(time(0));
    double array[MAX_SIZE];
    for (int i = 0; i < MAX_SIZE; i++)
    {
        array[i] = rand() + (rand() % RAND_MAX) / 1000.0;
    }
    bubbleSort(array, MAX_SIZE);
    for (int i = 0; i < MAX_SIZE; i++)
    {
        printf("%f ", array[i]);
    }
    auto end = GETTIME();
    auto elapsed_ns = CALCTIME(end - begin);
    printf("\nThe time: %lld ns\n", elapsed_ns.count());
}
```

Аналіз і оцінка часової складності алгоритмів:

$$t(1000) = 39569100 \text{ ns}$$

$$T(n) = Ct \cdot t(n), \text{ де } t(n) = O(n^2)$$

$$T(n) \leq Ct \cdot Cxg(n) = Ct \cdot Cx \cdot n^2$$

$$t_2/t_1 \leq Ct \cdot Cx \cdot n_2^2 / Ct \cdot Cx \cdot n_1^2 = n_2^2/n_1^2.$$

$$t_2 \leq t_1 \cdot n_2^2/n_1^2 = 39569100 \cdot 5000^2/1000^2 = 989227500 \text{ ns}$$

За такий час має відсортуватися масив з 5000 дійсних чисел

Сортування масиву має оцінку складності алгоритму –  $O(n^2)$ , так як бульбашкове сортування має вкладений цикл, кількість операцій буде залежати від розмірності масива.

Практично отримане значення(середнє арифметичне 5 дослідів) - 281980100 ns

		Башиманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

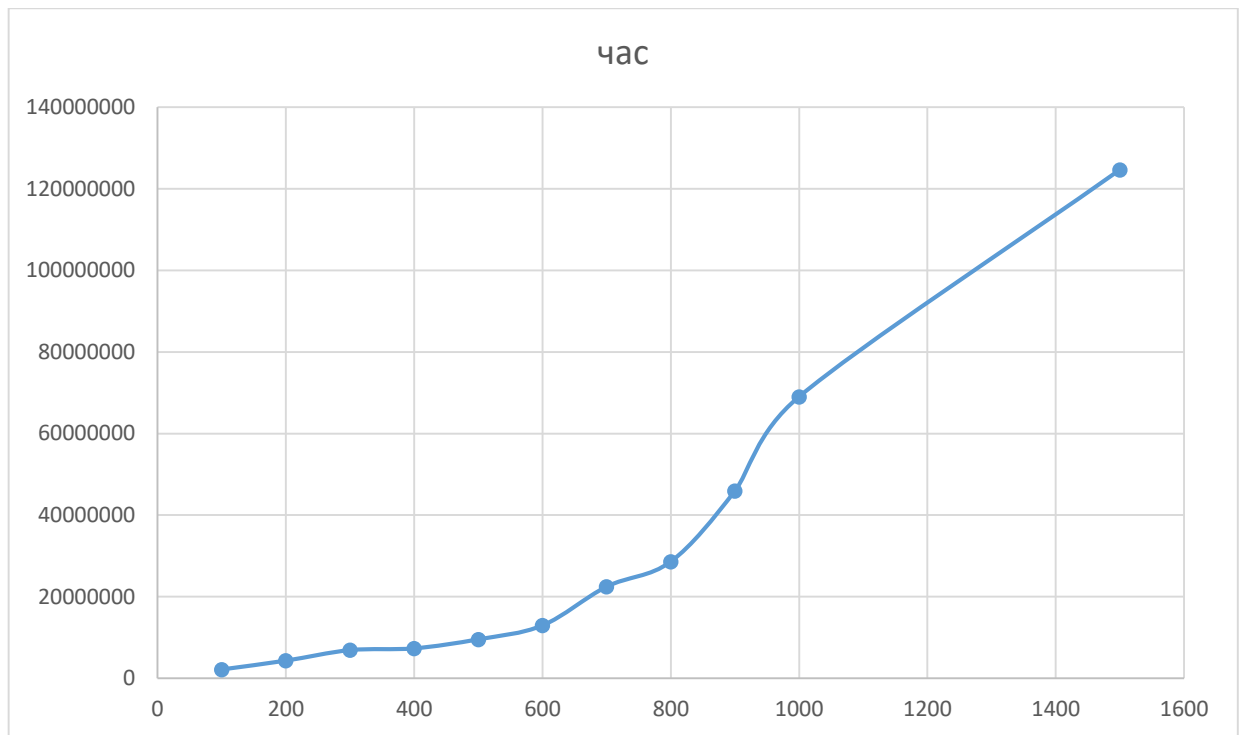


Рисунок 4.17 - Графік часу і вхідних даних

Результат виконання програми:

```

Выбрать Консоль отладки Microsoft Visual Studio
9) (4578.886230) (4575.498047) (4560.570801) (4554.422852) (4526.292969) (4376.4
20898) (4368.478027) (4366.016113) (4364.384766) (4285.520020) (4277.817871) (41
94.649902) (4133.916992) (4127.711914) (4110.328125) (4105.275879) (4017.870117)
(3986.645996) (3958.115967) (3945.384033) (3941.707031) (3806.591064) (3802.2700
20) (3796.373047) (3759.974121) (3748.979004) (3622.044922) (3620.364014) (3611.
558105) (3603.066895) (3583.929932) (3540.952881) (3533.091064) (3529.852051) (3
529.206055) (3491.481934) (3473.749023) (3466.581055) (3464.091064) (3378.372070)
(3326.500977) (3324.093994) (3305.414063) (3244.316895) (3211.899902) (3181.340
088) (3170.331055) (3151.373047) (3138.121094) (3134.933105) (3081.002930) (3053
.239014) (3049.871094) (2997.142090) (2989.008057) (2956.365967) (2934.319092) (
2933.583008) (2913.697021) (2857.267090) (2769.447021) (2740.481934) (2738.668945
) (2699.737061) (2692.854980) (2670.747070) (2663.938965) (2643.163086) (2601.80
9082) (2543.086914) (2540.274902) (2475.456055) (2424.533936) (2415.998047) (239
8.989990) (2377.579102) (2361.544922) (2340.351074) (2245.459961) (2242.344971)
(2229.436035) (2212.635010) (2206.978027) (2193.498047) (2028.020020) (2023.24694
8) (1948.727051) (1945.631958) (1933.136963) (1881.071045) (1665.712036) (1600.4
79004) (1563.396973) (1514.984985) (1461.399048) (1438.391968) (1437.442993) (13
72.680054) (1356.922974) (1340.609009) (1323.186035) (1256.630981) (1231.937988)
(1213.578979) (1138.618042) (1101.655029) (1067.186035) (1013.585999) (998.46502
7) (987.679993) (982.703979) (936.067017) (923.077026) (914.435974) (884.607971)
(882.817017) (865.981995) (843.684021) (712.778015) (711.005981) (654.872009)
(620.653015) (605.513977) (487.488007) (482.080994) (410.338989) (376.618011) (2
54.453003) (253.701004) (240.977005) (112.651001) (104.066002) (74.288002) (66.0
19997) (64.140999) (51.424999)
The time: 48924100 ns
C:\Users\madma\source\repos\rof1ALG\Debug\rof1ALG.exe (процесс 8252) завершил работу с
кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно...

```

Рисунок 4.18 – Результат виконання програми 2

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

## Лабораторна робота № 4

«Зв'язний список, стек, черга. Зворотній польський запис»

**Мета роботи:** 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

Завдання 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".).

1. Створюю заголовочний файл clist.h для збереження функцій та полів потрібних для реалізації списку.

Лістинг файлу .h:

```
#ifndef _CLIST_H_
#define _CLIST_H_

#include <stdlib.h>
#include <stdbool.h>

typedef int elemtype;          // Тип елемента списка

struct elem {
    elemtype* value;           // Значение переменной
    struct elem* next;         // Ссылка на следующий элемент списка
    struct elem* prev;         // Ссылка на предыдущий элемент списка
};

struct myList {
    struct elem* head;         // Первый элемент списка
    struct elem* tail;         // Последний элемент списка
    int size;                  // Количество элементов в списке
};

typedef struct elem cNode;
typedef struct myList cList;

cList* createList(void);       // створення списку
void deletelist(cList* list);   // видалення списку
bool isEmptyList(cList* list); // перевірка на пустоту
int pushFront(cList* list, elemtype* data); //елемент спочатку
int popFront(cList* list, elemtype* data); //видалення першого елемента
int pushBack(cList* list, elemtype* data); // елемент в кінці
int popBack(cList* list, elemtype* data); // видалення елемента в кінці
cNode* getNode(cList* list, int index); // отримання елемента
void printList(cList* list, void (*fun)(elemtype*)); // вивід списку
int insertNode(cList* list, size_t index, elemtype* data); // додавання елемента по індексу
int deleteNode(cList* list, size_t index); // видалення по індексу

#endif // _CLIST_H_
```

2. Реалізую функції списку у файлі clist.cpp

Динамічно виділяю пам'ять для списку.

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

```
// Создание пустого списка
cList* createList(void) {
    cList* list = (cList*)malloc(sizeof(cList));
    if (list) {
        list->size = 0;
        list->head = list->tail = NULL;
    }
    return list;
}
```

Звільняю пам'ять в циклі для кожного елемента.

```
// Удаление списка
void deletelist(cList* list) {
    cNode* head = list->head;
    cNode* next = NULL;
    while (head) {
        next = head->next;
        free(head);
        head = next;
    }
    free(list);
    list = NULL;
}
```

```
C:\Users\madma\source\repos\ALG_lab4\Debug\ALG_lab4.exe
1.Додати елементи на початок списку
2.Видалити елемент з кінця списку
3.Додати елементи в кінець списку
4.Видалити елемент спочатку
5.Весь список
6.Видалити список
7.Створити список
8.Додати елемент за індексом
9.Видалити елемент за індексом
10.Перевірка на пустоту списку
0.Закрити програму
6
Список видалено!
```

Рисунок 3.1 – Видалення списку

Повертає true або false , якщо наявні елементи в вершині або в кінці списку

```
// Проверка списка на пустоту
bool isEmptyList(cList* list) {
    return ((list->head == NULL) || (list->tail == NULL));
}
```

```
5
222
111
1.Додати елементи на початок списку
2.Видалити елемент з кінця списку
3.Додати елементи в кінець списку
4.Видалити елемент спочатку
5.Весь список
6.Видалити список
7.Створити список
8.Додати елемент за індексом
9.Видалити елемент за індексом
10.Перевірка на пустоту списку
0.Закрити програму
10
У списку наявні елементи
```

Рисунок 3.2 – Перевірка на наявність вузлів

Виділяю динамічно пам'ять для нового вузла , елемент стає в вершині списку

		Башиманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				31
Змн.	Арк.	№ докум.	Підпис	Дата		



```
// Додавання нового вузла в начало списку
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);
}
```

```
1.Додати елементи на початок списку
2.Видалити елемент з кінця списку
3.Додати елементи в кінець списку
4.Видалити елемент спочатку
5.Весь список
6.Видалити список
7.Створити список
8.Додати елемент за індексом
9.Видалити елемент за індексом
10.Перевірка на пустоту списку
0.Закрити програму
1
Введіть скільки значень ви хочете ввести:
2
Введіть значення:
111
Введіть значення:
222
```

Рисунок 3.3 – Новий вузол спочатку

#### Видалення верхнього елемента

```
// Извлечение узла из начала списка
int popFront(cList* list, elemtype* data) {
    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;
    }

    data = node->value;
    list->size--;
}
```

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				32
Змн.	Арк.	№ докум.	Підпис	Дата		

```

free(node);

return(0);
}

```

```

111
1.Додати елементи на початок списку
2.Видалити елемент з кінця списку
3.Додати елементи в кінець списку
4.Видалити елемент спочатку
5.Весь список
6.Видалити список
7.Створити список
8.Додати елемент за індексом
9.Видалити елемент за індексом
10.Перевірка на пустоту списку
0.Закрити програму

```

Рисунок 3.4 – Видалення елементу , верхнього = 222

Виділяю пам'ять та через показник вставляю елемент в кінець списку.

```

// Добавление нового узла в конец списка
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);
}

```

```

3
Введіть скільки значень ви хочете ввести:
2
Введіть значення:
777
Введіть значення:
444
1.Додати елементи на початок списку
2.Видалити елемент з кінця списку
3.Додати елементи в кінець списку
4.Видалити елемент спочатку
5.Весь список
6.Видалити список
7.Створити список
8.Додати елемент за індексом
9.Видалити елемент за індексом
10.Перевірка на пустоту списку
0.Закрити програму
5
111
777
444

```

Рисунок 3.5 – Додав 2 елементи в кінець

Видаляю останній елемент.

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

```

// Извлечение узла из конца списка
int popBack(cList* list, elemtype* data) {
    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;
    }

    data = node->value;
    list->size--;
    free(node);

    return(0);
}

```

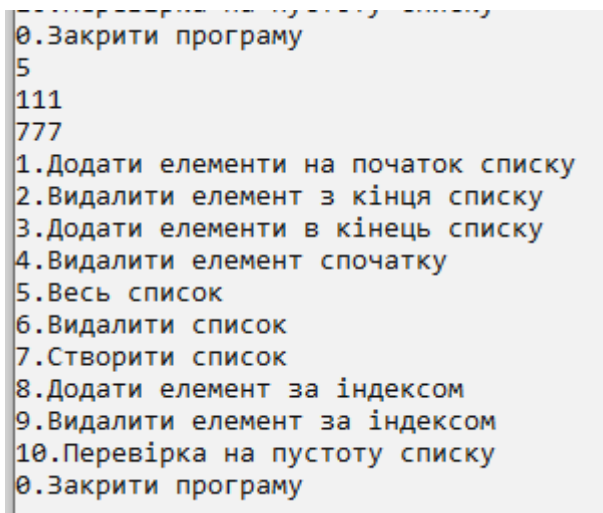


Рисунок 3.6 – Видалив останній елемент = 444

Шукаю потрібний індекс в циклі ділячи список

```

// Чтение произвольного узла списка
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;
    }
}

```

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				34
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        while (node && i > index) {
            node = node->prev;
            i--;
        }
    }

    return node;
}

```

Через показник звертаюсь до значення кожного елемента.

// Вывод списка в консоль

```

void printList(cList* list, void (*func)(element*)) {
    cNode* node = list->head;

```

```

    if (isEmptyList(list)) {
        return;
    }

```

```

    while (node) {
        func(node->value);
        node = node->next;
    }
}

```

//Добавить узел

```

int insertNode(cList* list, size_t index, element* data)
{

```

```

    if (index == 0) {
        pushFront(list, data);
        return(0);
    }

```

```

    cNode* node = getNode(list, index - 1); // елемент після якого потрібно вставити вузол
    cNode* insertNode = (cNode*)malloc(sizeof(cNode)); // виділення пам'яті під вузол

```

```

    if (!node) {
        return(-5);
    }

```

```

    if (!insertNode) {
        return(-6);
    }

```

```

    insertNode->value = data; // присвоюю значення через показник

```

```

    insertNode->prev = node;

```

```

    insertNode->next = node->next; //вставляю новий елемент

```

```

    if (node->next) {
        node->next->prev = insertNode;
    }

```

```

    node->next = insertNode;

```

```

    if (!node->prev) {
        list->head = node;
    }

```

```

    if (!node->next) {
        list->tail = node;
    }

```

```

    list->size++; //збільшую розмірність

```

```

    return(0);
}

```

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				35
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Після якого елемента ви хочете ввести значення:1
Введіть значення:000
1.Додати елементи на початок списку
2.Видалити елемент з кінця списку
3.Додати елементи в кінець списку
4.Видалити елемент спочатку
5.Весь список
6.Видалити список
7.Створити список
8.Додати елемент за індексом
9.Видалити елемент за індексом
10.Перевірка на пустоту списку
0.Закрити програму
5
111
0
777

```

Рисунок 3.7 – Додав елемент = 0 , після першого елемента

```

// Видалення вузла за індексом з списку
int deleteNode(cList* list, size_t index) {
    cNode* node = getNode(list, index); // знаходжу елемент який потрібно видалити
    if (!node) {
        return(-7);
    }
    if (node->prev) {
        node->prev->next = node->next; // змінюю порядок елементів
    }
    if (node->next) {
        node->next->prev = node->prev;
    }
    if (!node->prev) {
        list->head = node->next;
    }
    if (!node->next) {
        list->tail = node->prev;
    }
    free(node); // звільняю пам'ять цього вузла
    list->size--;
    return(0);
}

```

```

Після якого елемента ви хочете видалити елемент:2
1.Додати елементи на початок списку
2.Видалити елемент з кінця списку
3.Додати елементи в кінець списку
4.Видалити елемент спочатку
5.Весь список
6.Видалити список
7.Створити список
8.Додати елемент за індексом
9.Видалити елемент за індексом
10.Перевірка на пустоту списку
0.Закрити програму
5
111
0
1.Додати елементи на початок списку

```

Рисунок 3.8 – Видалив елемент = 777

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				36
Змн.	Арк.	№ докум.	Підпис	Дата		

### 3. Реалізую меню для демонстрації функцій

```
#include <stdio.h>
#include <stdlib.h>
#include "clist.h"
#include "windows.h"
#define h 30
void printNode(elemtype* value) {
    printf("%d\n", *((int*)value));
}
int pushtoStack(int znach)
{
    int f;
    f = znach;
    return f;
}
int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    elemtype tmp;
    int g = 1;
    elemtype tmp1 = 0;
    elemtype tmp3;
    int tmp2;
    int menu;
    int mass[h];
    int mass1[h];
    int znach;
    int znach1;
    int j=0;
    int l = 0;

    cList* mylist = createList();
    do {
        printf("1.Додати елементи на початок списку\n2.Видалити елемент з кінця\n3.Додати елементи в кінець списку\n4.Видалити елемент спочатку\n5.Весь список\n6.Видалити список\n7.Створити список\n8.Додати елемент за індексом\n9.Видалити елемент за індексом\n10.Перевірка на пустоту списку\n0.Закрити програму\n");
        scanf_s("%d", &menu);
        switch (menu)
        {
            case 1:printf("Введіть скільки значень ви хочете ввести:\n");
                scanf_s("%d", &g);

                for (int i = 0; i < g; i++)
                {
                    printf("Введіть значення:\n");
                    scanf_s("%d", &znach);
                    mass[i] = znach;
                    pushFront(mylist, &mass[i]);
                }
                break;
            case 2:popBack(mylist, &tmp);
                break;
            case 3:printf("Введіть скільки значень ви хочете ввести:\n");
                scanf_s("%d", &g);

                for (int i = 0; i < g; i++)
                {
                    printf("Введіть значення:\n");
                    scanf_s("%d", &znach1);
                    mass1[i] = znach1;
                    pushBack(mylist, &mass1[i]);
                }
        }
    }
```

		Башиманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				37
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        break;
    case 4: popFront(mylist, &tmp);
        break;
    case 5:
        printList(mylist, printNode);
        break;
    case 6: deleteList(mylist); printf("Список видалено!\n");
        break;
    case 7: {cList* mylist = createList(); }
        break;
    case 8: printf("\nПісля якого елемента ви хочете ввести значення:"); scanf_s("%d",
&j);
        printf("\nВведіть значення:"); scanf_s("%d", &l); insertNode(mylist,
j, &l);
        break;
    case 9: printf("\nПісля якого елемента ви хочете видалити елемент:"); scanf_s("%d",
&j);
        deleteNode(mylist, j);
        break;
    case 10: if (isEmptyList(mylist) == true)
    {
        printf("\nСписок пустий!!!\n");
    }
    else printf("\nУ списку наявні елементи\n");
        break;
    }
} while (menu != 0);
return 0;
}

```

Завдання 3: Розробити програму обчислення арифметичного виразу (використати зворотну 6 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.

Реалізую функцію для результату виконаних дій:

```

static int Counts(string str)
{
    string num = "";
    int n;
    int[] arr = new int[20];
    int arri = 0;
    foreach (char c in str)
    {
        if (int.TryParse(c.ToString(), out n))
        {
            num += c;
            continue;
        }
        if (num != "")
        {
            arr[arri] = Convert.ToInt32(num);
            num = "";
            arri++;
        }
        if (!int.TryParse(c.ToString(), out n) && c != ' ')
        {
            switch (c)
            {
                case '+':
                    arr[arri - 2] = arr[arri - 1] + arr[arri - 2];
                    break;
            }
        }
    }
}

```

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				38
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        case '-':
            arr[arri - 2] = arr[arri - 2] - arr[arri - 1];
            break;
        case '*':
            arr[arri - 2] = arr[arri - 1] * arr[arri - 2];
            break;
        case '/':
            arr[arri - 2] = arr[arri - 1] / arr[arri - 2];
            break;
        case '^':
            arr[arri - 2] = (int)Math.Pow(arr[arri - 2], arr[arri - 1]);
            break;
        case 's':
            arr[arri - 1] = (int)Math.Sqrt(arr[arri - 1]);
            arri++;
            break;
    }
    arri--;
}
}
return arr[arri - 1];
}

```

Функція для запису в оберненому польському записі:

```

static string ConvertToPoly(string str)
{
    string result = "";
    string massSIM = "";
    string num = "";
    int n;
    foreach (char c in str)
    {
        if (int.TryParse(c.ToString(), out n))
        {
            num += n.ToString();
        }
        if (!int.TryParse(c.ToString(), out n))
        {
            result += num;
            if (c != ' ')
            {
                result += " ";
            }
            num = "";
            if (c == 's')
            {
                massSIM += c;
            }
            else if (c == '+' || c == '-' || c == '*' || c == '/')
            {
                if (massSIM.Length != 0)
                {
                    while (massSIM[massSIM.Length - 1] == '*' ||
massSIM[massSIM.Length - 1] == '/' || massSIM[massSIM.Length - 1] == '^' ||
massSIM[massSIM.Length - 1] == 's')
                    {
                        result += massSIM[massSIM.Length - 1];
                        result += " ";
                        massSIM = massSIM.Remove(massSIM.Length - 1);
                        if (massSIM.Length == 0)
                        {
                            break;
                        }
                    }
                }
            }
        }
    }
}

```

		Бащманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				39
Змн.	Арк.	№ докум.	Підпис	Дата		



```

        }
        massSIM += c;
    }
    else if (c == '^')
    {
        massSIM += c;
    }
}
}
if (num != "")
{
    result += num;
    result += " ";
}
while (massSIM.Length != 0)
{
    result += massSIM[massSIM.Length - 1];
    result += " ";
    massSIM = massSIM.Remove(massSIM.Length - 1);
}
return result;
}

```

Виклик реалізованих функцій:

```

static void Main(string[] args)
{
    Console.WriteLine($"{Counts(ConvertToPoly("23 + 2 + 1 + 5 + 6 - 2"))}");
    Console.WriteLine($"{ConvertToPoly("23 + 2 + 1 + 5 + 6 - 2")}");
}

```

Результат роботи програми:

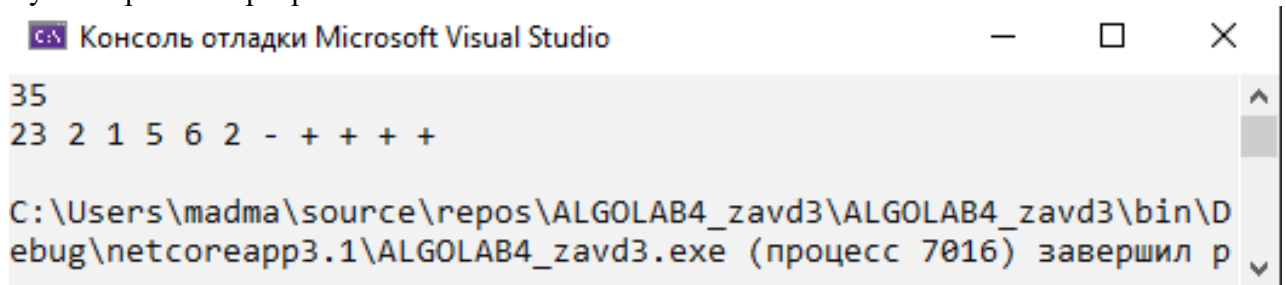


Рисунок 3.9 – Обернений польський запис

## Лабораторна робота № 5

### Прості методи сортування

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

### 5.1 Хід роботи

Завдання 1: Реалізувати алгоритми сортування:

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

#### 1(A):

Сортування вибором(структура даних – двусвязний список);

**Алгоритм працює таким чином:**

1. Знаходить у списку найменше значення
2. Міняє його місцями із першим значеннями у списку
3. Повторює два попередніх кроки, доки список не завершиться (починаючи з наступної позиції)

Фактично, таким чином ми поділили список на дві частини: перша (ліва) — повністю відсортована, а друга (права) — ні.

Виконаний алгоритм:

```
#include <iostream>
#include <iomanip>
#include <locale.h>
#include "windows.h"
#include <cstdlib>
#include <iostream>
#include <chrono>
#define GETTIME std::chrono::steady_clock::now
#define CALCTIME std::chrono::duration_cast<std::chrono::nanoseconds>

using namespace std;

struct List {
    int info;
    List* pred, * next;
};

// Вставка елемента в двузв'язний список після останнього елемента
List* InsertElementInList(List* last, List* p)
{
    if (last && p)
    {
        p->pred = last;
        p->next = last->next;
        last->next = p;
        p->next->pred = p;
        return p; // Повертається адреса елемента
    }
    else
        return NULL;
}
```

		Бащманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

```

//Створення списку та запис елементів
void CreateList(List*& head, List*& tail)
{
    head = new List;
    tail = new List;
    head->next = tail;
    tail->pred = head;
    int k;
    List* last = head;
    for (int count = 0; count < 40000; ++count)
    {
        k = rand();
        List* p = new List;
        p->info = k;
        last = InsertElementInList(last, p);
    }
    return;
}

void PrintList(List* head, List* tail) // вивід списку
{
    List* p = head->next;
    while (p != tail)
    {
        cout << "\t" << p->info;
        p = p->next;
    }
    cout << endl;
    return;
}

void SortListVYB(List*& head, List*& tail)
{
    List* last = tail;

    tail->next = head;
    head->pred = tail;
    while (head->next != tail) // поки залишилися елементи шукаємо мінімальний та видаляємо
    {
        List* min = head->next, * p = head->next;
        while (p != tail)
        {
            if (p->info < min->info)
                min = p;
            p = p->next;
        }

        min->next->pred = min->pred; //видалення min елемента із списку
        min->pred->next = min->next;

        last = InsertElementInList(last, min); // додаємо знайдений елемент
    }
    swap(head, tail);
}

int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    auto begin = GETTIME();

    auto end = GETTIME();
}

```

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

```

srand(time(NULL));

List* head, * tail = NULL;
Createlist(head, tail);
cout << "\nСписок\n" << endl;
PrintList(head, tail);
cout << "\nВідсортований список\n" << endl;
SortListVYB(head, tail);
PrintList(head, tail);

auto elapsed_ns = CALCTIME(end - begin);
printf("The time: %lld ns\n", elapsed_ns.count());
}

```

Консоль отладки Microsoft Visual Studio

32093 32094 32095 32096 32097 32098 32099 32100 32101 32102 32103 32104 32105 32106 32107 32108 32109 32110 32111 32112 32113 32114 32115 32116 32117 32118 32119 32120 32121 32122 32123 32124 32125 32126 32127 32128 32129 32130 32131 32132 32133 32134 32135 32136 32137 32138 32139 32140 32141 32142 32143 32144 32145 32146 32147 32148 32149 32150 32151 32152 32153 32154 32155 32156 32157 32158 32159 32160 32161 32162 32163 32164 32165 32166 32167 32168 32169 32170 32171 32172 32173 32174 32175 32176 32177 32178 32179 32180 32181 32182 32183 32184 32185 32186 32187 32188 32189 32190 32191 32192 32193 32194 32195 32196 32197 32198 32199 32200 32201 32202 32203 32204 32205 32206 32207 32208 32209 32210 32211 32212 32213 32214 32215 32216 32217 32218 32219 32220 32221 32222 32223 32224 32225 32226 32227 32228 32229 32230 32231 32232 32233 32234 32235 32236 32237 32238 32239 32240 32241 32242 32243 32244 32245 32246 32247 32248 32249 32250 32251 32252 32253 32254 32255 32256 32257 32258 32259 32260 32261 32262 32263 32264 32265 32266 32267 32268 32269 32270 32271 32272 32273 32274 32275 32276 32277 32278 32279 32280 32281 32282 32283 32284 32285 32286 32287 32288 32289 32290 32291 32292 32293 32294 32295 32296 32297 32298 32299 32300 32301 32302 32303 32304 32305 32306 32307 32308 32309 32310 32311 32312 32313 32314 32315 32316 32317 32318 32319 32320 32321 32322 32323 32324 32325 32326 32327 32328 32329 32330 32331 32332 32333 32334 32335 32336 32337 32338 32339 32340 32341 32342 32343 32344 32345 32346 32347 32348 32349 32350 32351 32352 32353 32354 32355 32356 32357 32358 32359 32360 32361 32362 32363 32364 32365 32366 32367 32368 32369 32370 32371 32372 32373 32374 32375 32376 32377 32378 32379 32380 32381 32382 32383 32384 32385 32386 32387 32388 32389 32390 32391 32392 32393 32394 32395 32396 32397 32398 32399 32400 32401 32402 32403 32404 32405 32406 32407 32408 32409 32410 32411 32412 32413 32414 32415 32416 32417 32418 32419 32420 32421 32422 32423 32424 32425 32426 32427 32428 32429 32430 32431 32432 32433 32434 32435 32436 32437 32438 32439 32440 32441 32442 32443 32444 32445 32446 32447 32448 32449 32450 32451 32452 32453 32454 32455 32456 32457 32458 32459 32460 32461 32462 32463 32464 32465 32466 32467 32468 32469 32470 32471 32472 32473 32474 32475 32476 32477 32478 32479 32480 32481 32482 32483 32484 32485 32486 32487 32488 32489 32490 32491 32492 32493 32494 32495 32496 32497 32498 32499 32500 32501 32502 32503 32504 32505 32506 32507 32508 32509 32510 32511 32512 32513 32514 32515 32516 32517 32518 32519 32520 32521 32522 32523 32524 32525 32526 32527 32528 32529 32530 32531 32532 32533 32534 32535 32536 32537 32538 32539 32540 32541 32542 32543 32544 32545 32546 32547 32548 32549 32550 32551 32552 32553 32554 32555 32556 32557 32558 32559 32560 32561 32562 32563 32564 32565 32566 32567 32568 32569 32570 32571 32572 32573 32574 32575 32576 32577 32578 32579 32580 32581 32582 32583 32584 32585 32586 32587 32588 32589 32590 32591 32592 32593 32594 32595 32596 32597 32598 32599 32600 32601 32602 32603 32604 32605 32606 32607 32608 32609 32610 32611 32612 32613 32614 32615 32616 32617 32618 32619 32620 32621 32622 32623 32624 32625 32626 32627 32628 32629 32630 32631 32632 32633 32634 32635 32636 32637 32638 32639 32640 32641 32642 32643 32644 32645 32646 32647 32648 32649 32650 32651 32652 32653 32654 32655 32656 32657 32658 32659 32660 32661 32662 32663 32664 32665 32666 32667 32668 32669 32670 32671 32672 32673 32674 32675 32676 32677 32678 32679 32680 32681 32682 32683 32684 32685 32686 32687 32688 32689 32690 32691 32692 32693 32694 32695 32696 32697 32698 32699 32700 32701 32702 32703 32704 32705 32706 32707 32708 32709 32710 32711 32712 32713 32714 32715 32716 32717 32718 32719 32720 32721 32722 32723 32724 32725 32726 32727 32728 32729 32730 32731 32732 32733 32734 32735 32736 32737 32738 32739 32740 32741 32742 32743 32744 32745 32746 32747 32748 32749 32750 32751 32752 32753 32754 32755 32756 32757 32758 32759 32760 32761 32762 32763 32764 32765 32766 32767

The time: 400 ns

C:\Users\madma\source\repos\ALG\_lab5\Debug\ALG\_lab5.exe (процесс 5016) завершил работу с кодом 0.  
Нажмите любую клавишу, чтобы закрыть это окно...

Рисунок 5.1 – Результат сортування вибором(списку)

## 1(В): сортування вставками (структура даних – масив);

### Алгоритм працює таким чином:

На кожному кроці алгоритму ми вибираємо один з елементів вхідних даних і вставляємо його на потрібну позицію у вже відсортованому списку доти, доки набір вхідних даних не буде вичерпано. Метод вибору чергового елемента з початкового масиву довільний; може використовуватися практично будь-який алгоритм вибору. Зазвичай (і з метою отримання стійкого алгоритму сортування), елементи вставляються за порядком їх появи у вхідному масиві.

### Виконаний алгоритм:

```

#include <stdio.h>
#include <iostream>
#include "windows.h"
#include "stdlib.h"
#include "time.h"
#include <ctime>
#include <cstdlib>
#include "math.h"
#include <chrono>

```

		Башиманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

```

#define MAX_SIZE 10000

#define GETTIME std::chrono::steady_clock::now
#define CALCTIME std::chrono::duration_cast<std::chrono::nanoseconds>

void InsertSort(int* array, int size)
{
    int c;
    for (int i = 1; i < size; i++) //Проходимо по масиву знаходячи мінімальний елемент ,
    вставляємо його у відсортовану частину
    {
        c = array[i];
        for (int j = i - 1; j >= 0 && array[j] > c; j--)
        {
            array[j + 1] = array[j];
            array[j] = c;
        }
    }
}

int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    auto begin = GETTIME();
    auto end = GETTIME();
    srand(time(NULL));

    int array[MAX_SIZE];
    for (int i = 0; i < MAX_SIZE; i++)
    {
        array[i] = rand();
    }
    InsertSort(array, MAX_SIZE);
    for (int i = 0; i < MAX_SIZE; i++)
    {
        printf("%d ", array[i]);
    }

    auto elapsed_ns = CALCTIME(end - begin);
    printf("The time: %lld ns\n", elapsed_ns.count());
    return 0;
}

```

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

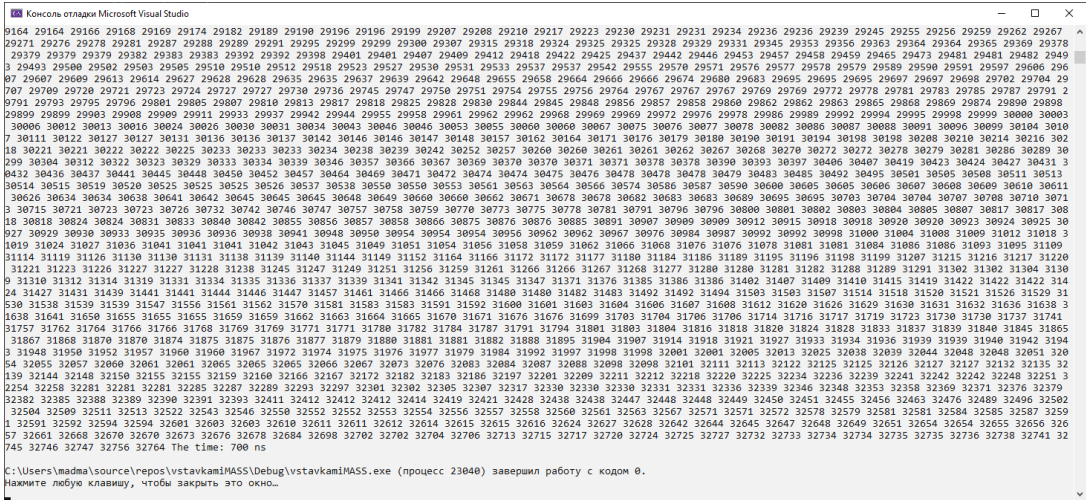


Рисунок 5.2 – Сортунання вставками(масив)

1(С): Сортунання вставками (структура даних – двусвязний список);

Алгоритм працює таким чином:

1)Створити порожній відсортоване (або результат) двусвязний список.

2) Пройдіть заданий двусвязний список, виконайте наступні дії для кожного вузла.

а) Вставити поточний вузол відсортованим способом в відсортоване (або результат) двусвязний список.

3) Замінити заголовки даного пов'язаного списку на заголовки відсортованого (або результату) списку.

```
#include <iostream>
#include <iomanip>
#include <locale.h>
#include "windows.h"
#include <cstdlib>
#include <iostream>
#include <chrono>
#define GETTIME std::chrono::steady_clock::now
#define CALCTIME std::chrono::duration_cast<std::chrono::nanoseconds>
```

```
using namespace std;
```

```
struct List
{
    List() : Next(nullptr), Prev(nullptr) {}
    int info;
    List* Next, * Prev;
};
```

```
// Функция сортирует список методом вставки
void InsertionSort(List* head)
{
```

```
    List* curr = nullptr, * prev = nullptr;
    for (curr = head->Next; curr->Next; curr = curr->Next)
    {
        int tmp = curr->info;
        for (prev = curr->Prev; prev && prev->info > tmp; prev = prev->Prev)
```

		Башманівський				Арк.
		Петросян Р.В.			ДУ «Житомирська політехніка».20.121.3.000 – Лр	
Змн.	Арк.	№ докум.	Підпис	Дата		45

```

        {
            prev->Next->info = prev->info;
        }
        prev->Next->info = tmp;
    }
}

// Функция вставляет элемент p после элемента last
// возвращает адрес на вставленный в список элемент
List* InsertElementIntoList(List* last, List* p)
{
    if (last != NULL || p != NULL)
    {
        p->Prev = last;
        p->Next = last->Next;
        last->Next = p;
        p->Next->Prev = p;
        return p;
    }
    else
        return NULL;
}

// Функция создает двусвязный список из n элементов
// через head и tail возвращает указатели головного и хвостового сторожей
void CreateRandomList(List*& head, List*& tail, int n)
{
    head = new List;
    tail = new List;

    head->Next = tail;
    tail->Prev = head;

    srand(time(NULL));

    List* last; // Последний вставленный элемент в список
    last = head;

    for (int i = 0; i < n; i++)
    {
        List* p = new List;
        p->info = rand();
        last = InsertElementIntoList(last, p);
    }
}

// Функция выводит список на экран
void PrintList(List* head, List* tail)
{
    List* p = head->Next;
    while (p != tail)
    {
        cout << "\t" << p->info;
        p = p->Next;
    }
    cout << endl;
}

int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    auto begin = GETTIME();

```

		Башиманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		46



[illegible]

Рисунок 5.3 – Сортування вставками(список)

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				47
Змн.	Арк.	№ докум.	Підпис	Дата		



Графіки результатів вимірів часу:

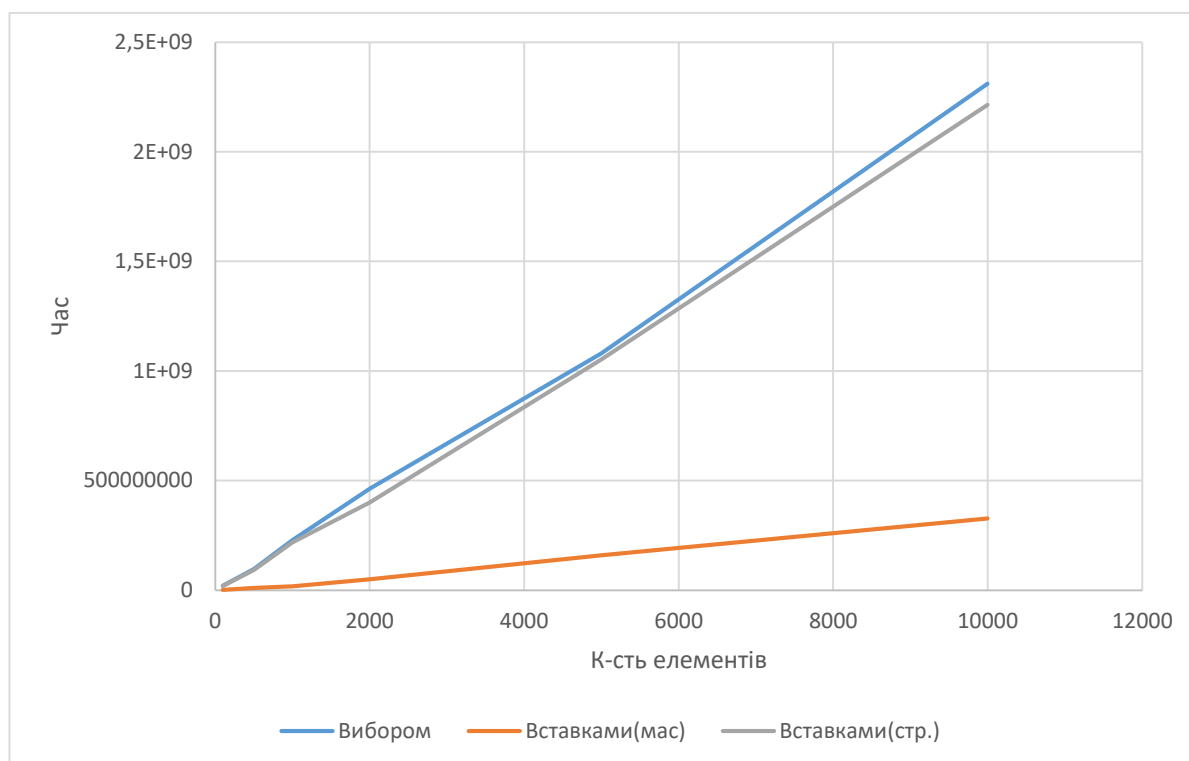


Рисунок 5.4 – графік залежності часу від к-сті елементів

Час	К-сть
19545500	100
92021300	500
2,17E+08	1000
4E+08	2000
1,05E+09	5000
2,21E+09	10000
Вставками(стр.)	

Час	К-сть
20159900	100
97234100	500
2,29E+08	1000
4,64E+08	2000
1,08E+09	5000
2,31E+09	10000
Вибором	

Час	К-сть
1555800	100
10441600	500
18350000	1000
49426500	2000
1,6E+08	5000
3,28E+08	10000
Вставками(мас)	

Висновок: сортування реалізоване з допомогою вставок має кращі показники складності, однак не практична у використанні з елементами до яких потрібен доступ.

## Лабораторна робота № 6

### Швидкі методи сортування

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

#### 6.1 Хід роботи

Завдання 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).

3, 18	int	[0, 100]	float	[0, 300]	Р.Седжвіка	char	[-200, 10]
-------	-----	----------	-------	----------	------------	------	------------

#### 1(А):

Пірамідальне сортування (структура даних – масив);

Складність:  $O(n \log n)$

Додаткова пам'ять:  $O(1)$

#### Суть сортування:

Етап 1: Куча

Організація даних в спеціальну деревоподібну структуру – кучу.

Беремо перший елемент масиву і вважаємо що це корінь дерева – вузол 1 рівня.

Наступні 2 елементи це – вузли 2 рівня, правий і лівий нащадки кореневого елемента. Наступні 4 елементи – це вузли 3 рівня, праві і ліві нащадки 2 та 3 елемента масиву і так далі поки не буде побудоване дерево(куча).

Для побудови використовують формули:

$2 \times i + 1$  – лівий потіток

$2 \times i + 2$  – правий потіток

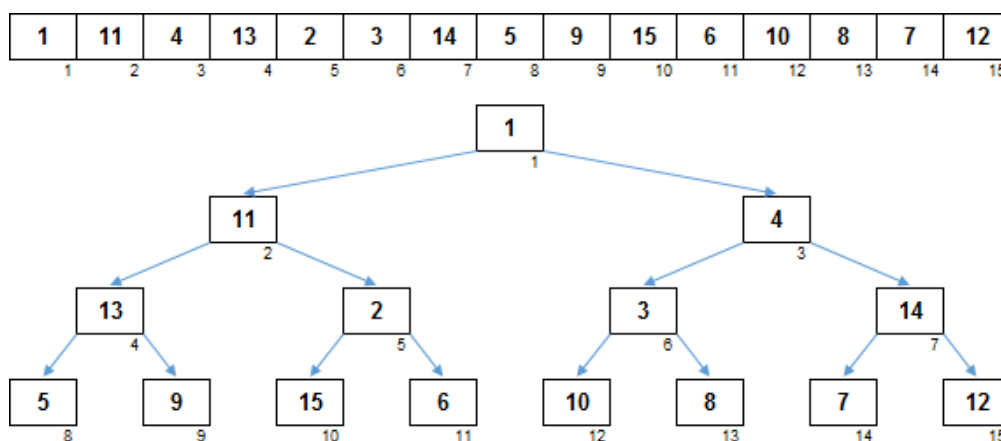


Рисунок 6.1 – Приклад побудованої кучі

## Етап 2: Просейка

Просейка потрібна для того щоб відсортувати кучу.

Якщо елемент менший або більший чим його батьки(залежить від того max чи min куча) , то цей елемент потрібно перемістити наверх на один рівень.

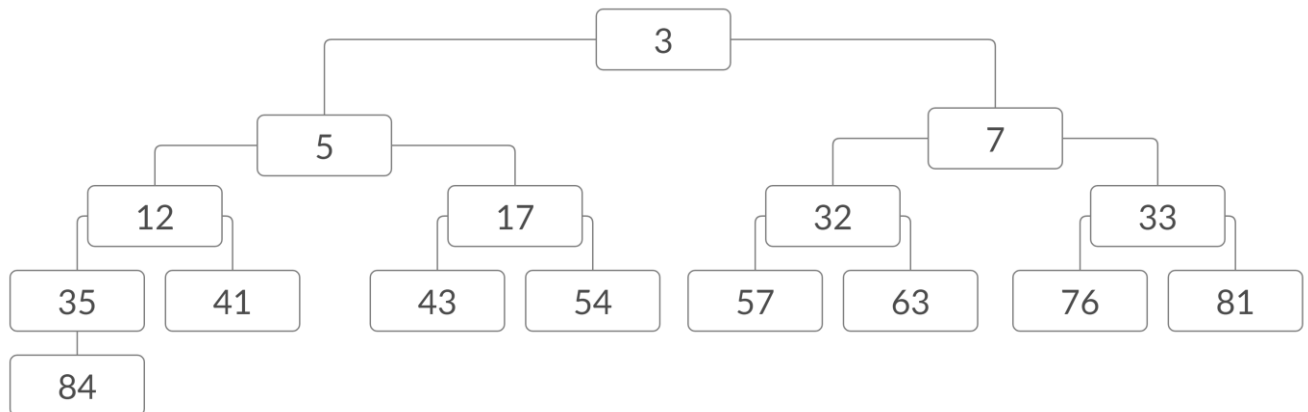


Рисунок 6.2 – Приклад відсортованої кучи(min)

## Етап 3: Heap – sort.

Так як дані в масиві після першого етапу вдають із себе Сортувальне дерево, максимальний елемент знаходиться на першому місці в масиві. Перший елемент (він же максимум) міняємо з останнім елементом невідсортованої частини масиву місцями. Після цього обміну максимум виявився своєму остаточному місці, тобто максимальний елемент відсортований. Несортована частина масиву перестала бути сортувати деревом, але це виправляється одноразової просейкою - в результаті чого на першому місці масиву виявляється попередній за величиною максимальний елемент. Дії цього етапу знову повторюються для залишилася невпорядкованою області, до тих пір поки максимуми по черзі НЕ будуть переміщені на свої остаточні позиції.

## Реалізований алгоритм:

```
#include <iostream>
#include <stdio.h>
#include "windows.h"
#include "stdlib.h"
#include "time.h"
#include <ctime>
#include <cstdlib>
#include "math.h"
#define MAX_SIZE 1000
#include <chrono>
#include <random>
#define GETTIME std::chrono::steady_clock::now
#define CALCTIME std::chrono::duration_cast<std::chrono::nanoseconds>
#define left_child(node) ( (node) * 2 + 1 ) // Лівий потімок
#define right_child(node) ( (node) * 2 + 2 ) // Правий потімок
```

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

```

void swap(int* array, int i, int j) // Функція для зміни порядку елементів , використовується
в сортуванні
{
    int tmp = array[i];
    array[i] = array[j];
    array[j] = tmp;
}

void heap_it(int* array, int length, int root) //Сортування кучі
{
    int leftChild = left_child(root);
    int rightChild = right_child(root);
    int biggest = root;

    if (leftChild < length && array[root] < array[leftChild]) // Переміщення елементів вверх
по кучі
        biggest = leftChild;
    if (rightChild < length && array[biggest] < array[rightChild]) // Переміщення елементів
вверх по кучі
        biggest = rightChild;
    if (biggest != root) // якщо найбільший елемент не у верху кучі , переміщуємо у верх ,
знову виконуємо heap_it тільки вже з найбільшим елементом у верху.
    {
        swap(array, biggest, root);
        heap_it(array, length, biggest);
    }
}

void make_heap(int* array, int length) //Створення кучі
{
    int i = length / 2;

    for (; i >= 0; --i)
        heap_it(array, length, i);
}

void heap_sort(int* array, int count) //Загальний алгоритм сортування
{
    int last;

    make_heap(array, count);
    for (last = count - 1; last > 0; --last)
    {
        swap(array, 0, last);
        heap_it(array, last, 0);
    }
}

#define COUNT (1000)

int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    auto begin = GETTIME();

    auto end = GETTIME();
    srand(time(0));
    int i;
    int array[COUNT];
    for (int f = 0; f < COUNT; f++)
    {
        array[f] = 0 + rand() % (100 + 1); // [0 ; 100]
    }
}

```

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}

printf("unSorted:\n");
for (i = 0; i < COUNT; ++i)
    printf("%d ", array[i]);

heap_sort(array, COUNT); //Сортування
printf("\nSorted:\n");

for (i = 0; i < COUNT; ++i)
    printf("%d ", array[i]);
printf("\n");

return 0;
auto elapsed_ns = CALTIME(end - begin);
printf("The time: %lld ns\n", elapsed_ns.count());
}

```

Рисунок 6.3 – Відсортований масив.

## 1(В):

сортування Шелла (структура даних – масив);

Складність:  $O(n \log^2 n)$

Додаткова пам'ять:  $O(1)$

## Суть сортування:

Модифікований варіант сортування вставками.

Перше значення - це половина довжини сортованого масиву, друге - половина від попереднього і так далі, кожен раз округляючи значення до цілого числа.

З точки зору коду ми просто додаємо до коду сортування вставками ще один цикл із значенням зсуву, що зменшується кожну ітерацію, і починаємо сортування тепер від елемента з індексом, рівним цьому значенню.

		Башиманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

Метод має властивість – приріст. В моєму варіанті розглядається формула Седжвіка.

$$\text{inc}[s] = \begin{cases} 9 \cdot 2^s - 9 \cdot 2^{s/2} + 1, & \text{если } s \text{ чётно} \\ 8 \cdot 2^s - 6 \cdot 2^{(s+1)/2} + 1, & \text{если } s \text{ нечётно} \end{cases}$$

Рисунок 6.4 – Формула Седжвіка

Приріст - відстань між сортованими елементами, в залежності від проходу. При використанні таких приростів середня кількість операцій:  $O(n^7 / 6)$ , в гіршому випадку - порядку  $O(n^4 / 3)$ .

### Реалізований алгоритм:

```
#include <iostream>
#include <stdio.h>
#include "windows.h"
#include "stdlib.h"
#include "time.h"
#include <ctime>
#include <cstdlib>
#include "math.h"
#define MAX_SIZE 1000
#include <chrono>
#include <random>
#define GETTIME std::chrono::steady_clock::now
#define CALCTIME std::chrono::duration_cast<std::chrono::nanoseconds>
#define left_child(node) ( (node) * 2 + 1 ) // Лівий потімок
#define right_child(node) ( (node) * 2 + 2 ) // Правий потімок

void swap(int* array, int i, int j) // Функція для зміни порядку елементів , використовується в сортуванні
{
    int tmp = array[i];
    array[i] = array[j];
    array[j] = tmp;
}

void heap_it(int* array, int length, int root) //Сортування кучі
{
    int leftChild = left_child(root);
    int rightChild = right_child(root);
    int biggest = root;

    if (leftChild < length && array[root] < array[leftChild]) // Переміщення елементів в верх по кучі
        biggest = leftChild;
    if (rightChild < length && array[biggest] < array[rightChild]) // Переміщення елементів в верх по кучі
        biggest = rightChild;
    if (biggest != root) // якщо найбільший елемент не у верху кучі , переміщуємо у верх , знову виконуємо heap_it тільки вже з найбільшим елементом у верху.
    {
        swap(array, biggest, root);
        heap_it(array, length, biggest);
    }
}

void make_heap(int* array, int length) //Створення кучі
{
    int i = length / 2;
```

		Башиманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        for (; i >= 0; --i)
            heap_it(array, length, i);
    }

void heap_sort(int* array, int count) //Загальний алгоритм сортування
{
    int last;

    make_heap(array, count);
    for (last = count - 1; last > 0; --last)
    {
        swap(array, 0, last);
        heap_it(array, last, 0);
    }
}

#define COUNT (1000)

int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    auto begin = GETTIME();

    srand(time(0));
    int i;
    int array[COUNT];
    for (int f = 0; f < COUNT; f++)
    {
        array[f] = 0 + rand() % (100 + 1); // [0 ; 100]
    }

    printf("unSorted:\n");
    for (i = 0; i < COUNT; ++i)
        printf("%d ", array[i]);

    heap_sort(array, COUNT); //Сортування
    printf("\nSorted:\n");

    for (i = 0; i < COUNT; ++i)
        printf("%d ", array[i]);
    printf("\n");

    auto end = GETTIME();
    auto elapsed_ns = CALCTIME(end - begin);
    printf("The time: %lld ns\n", elapsed_ns.count());
}

```



}

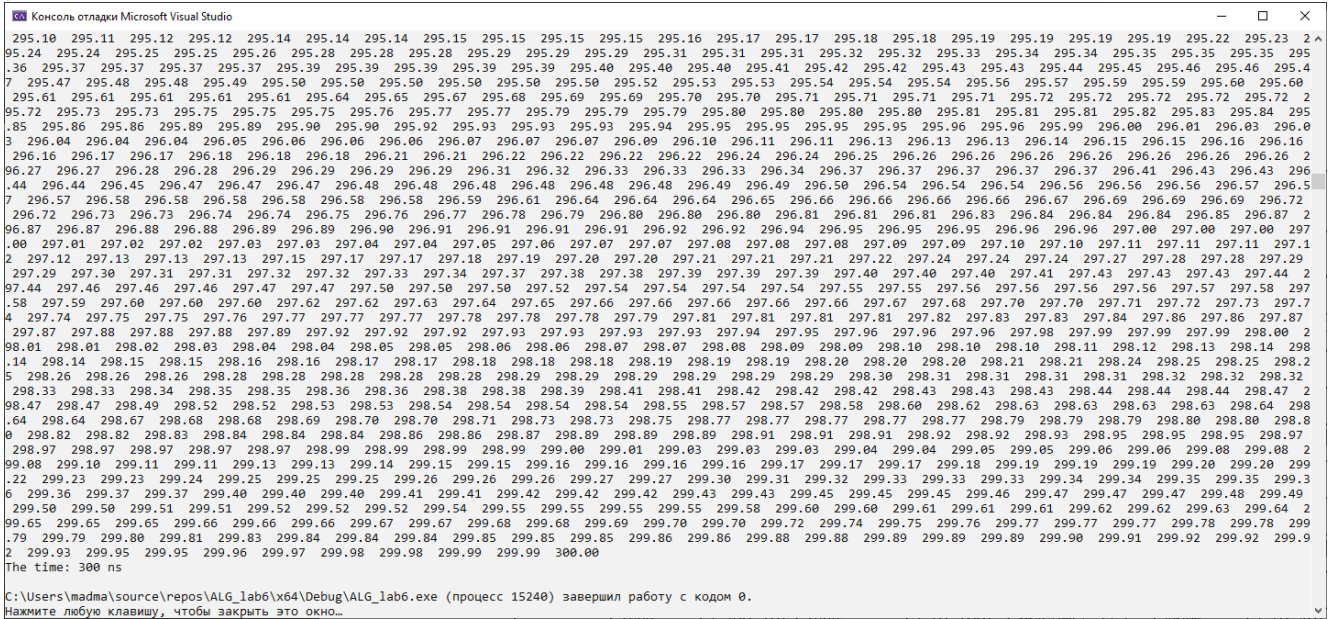


Рисунок 6.5 – Результат сортування

**1(С):**  
Сортування підрахунком (структура даних – масив);  
Складність:  $O(n+k)$  К – ширина діапвзону.

Додаткова пам'ять: В алгоритмі використовуються два додаткових масиви. Тому алгоритм потребує  $O(N+K)$  додаткової пам'яті.

**Суть сортування:** Ідея алгоритму полягає в наступному: спочатку підрахувати скільки разів кожен елемент (ключ) зустрічається в вихідному масиві. Спираючись на ці дані можна одразу вирахувати на якому місці має стояти кожен елемент, а потім за один прохід поставити всі елементи на свої місця.

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		55



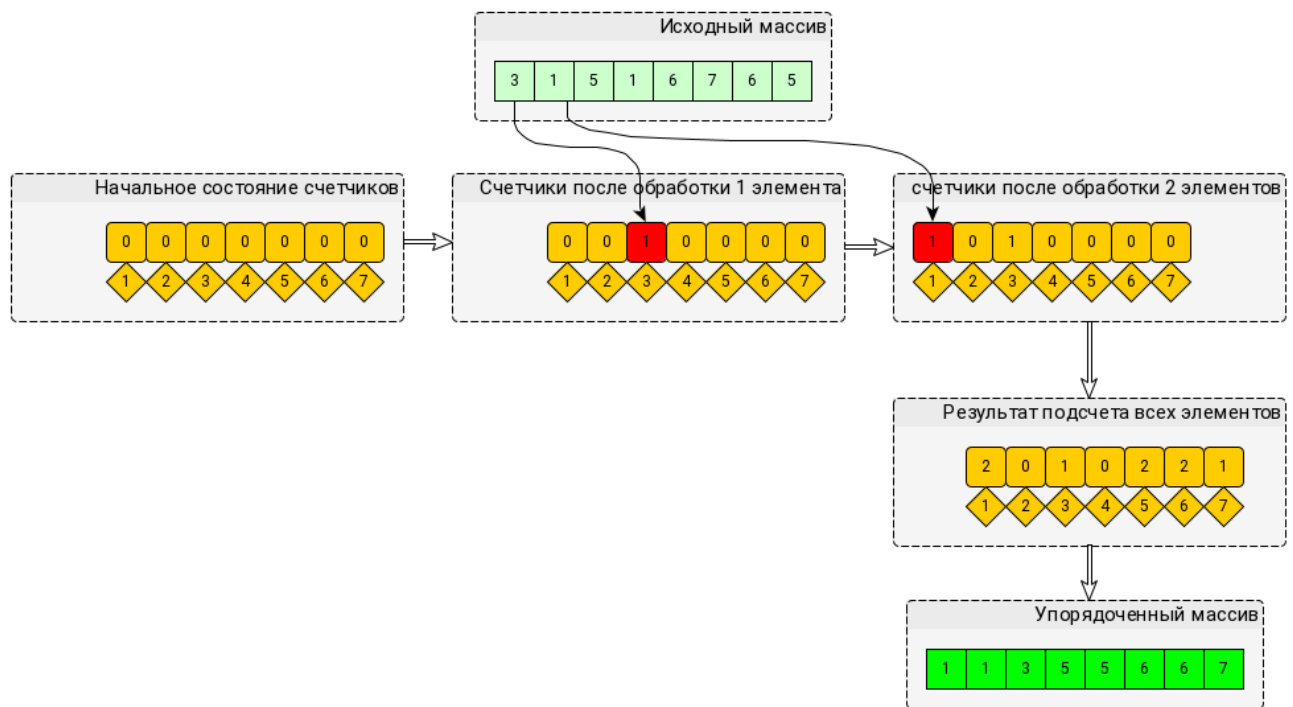


Рисунок 6.6 – алгоритм підрахунку

### Реалізований алгоритм:

```
#include <iostream>
#include <time.h>
#include <chrono>
#include <random>
#include "windows.h"
#define GETTIME std::chrono::steady_clock::now
#define CALCTIME std::chrono::duration_cast<std::chrono::nanoseconds>
#define N 5000
using namespace std;

void CountingSort(short mass[], int n)
{
    int max = INT_MIN, min = INT_MAX;
    for (int i = 0; i < n; i++) {
        if (mass[i] > max)
            max = mass[i];
        if (mass[i] < min)
            min = mass[i];
    }
    int* c = new int[max + 1 - min];
    for (int i = 0; i < max + 1 - min; i++) {
        c[i] = 0;
    }
    for (int i = 0; i < n; i++) {
        c[mass[i] - min] = c[mass[i] - min] + 1;
    }
    int i = 0;
    for (int j = min; j < max + 1; j++) {
        while (c[j - min] != 0) {
            mass[i] = j;
            c[j - min]--;
            i++;
        }
    }
}
```

```

}

int main()
{
    srand(time(NULL));
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    auto begin = GETTIME();

    auto end = GETTIME();
    short mass[N];

    for (int i = 0; i < N; i++)
    {
        mass[i] = -200 + rand() % (10 - (-200) + 1);
        printf(" %d ", mass[i]);
    }

    printf("\n\n\n\n\n");
    CountingSort(mass, N);

    for (int i = 0; i < N; i++)
    {
        printf(" %d ", mass[i]);
    }
    auto elapsed_ns = CALCTIME(end - begin);
    printf("\nThe time: %lld ns\n", elapsed_ns.count());
}

```

Рисунок 6.7 – Результат сортування

		Башиманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

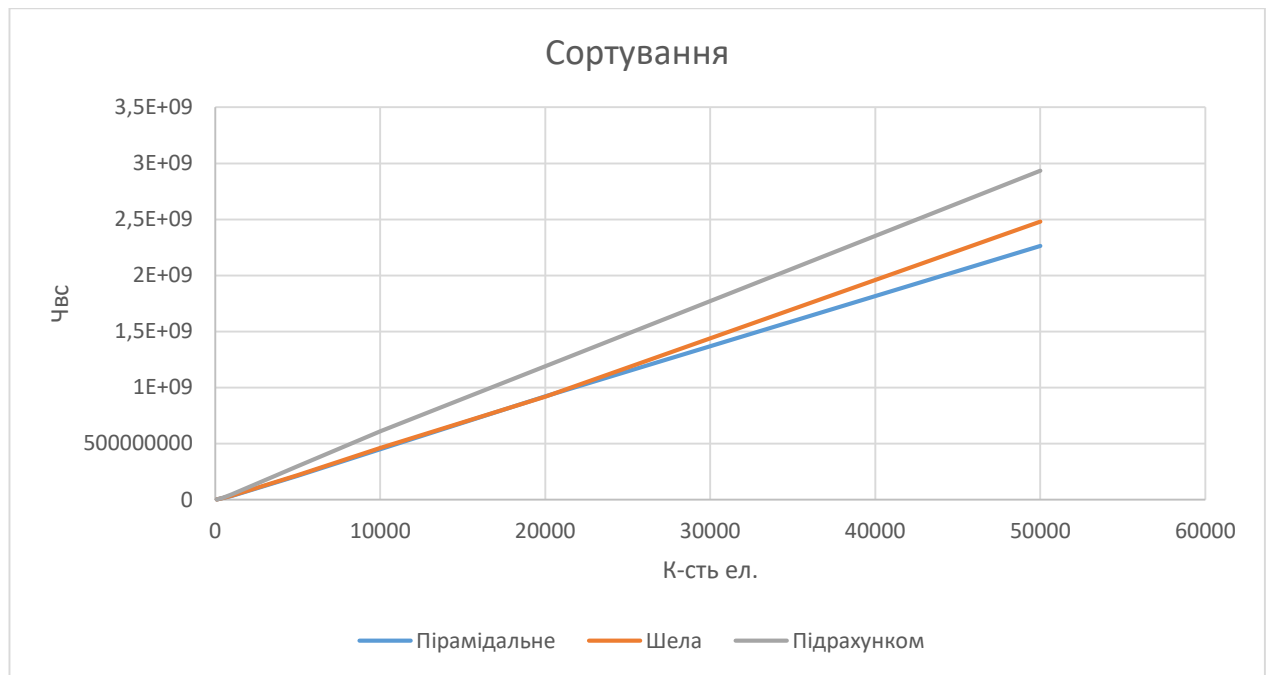


Рисунок 6.8 – Графік сортування

Час	К-сть
3184200	100
15753000	500
33022500	1000
213361500	5000
450564000	10000
920061100	20000
2264474600	50000
Пірамідальне	

Час	К-сть
3018800	100
15435700	500
35924700	1000
218565400	5000
462058600	10000
919604000	20000
2480017000	50000
Шела	

Час	К-сть
2972000	100
19438900	500
47720700	1000
2,99E+08	5000
6,1E+08	10000
1,19E+09	20000
2,94E+09	50000
Підрахунком	

Отже сортування підрахунком ефективне , при сортуванні великої к-сті елементів.  
Метод Шела з формулою приросту Седжвіка ефективніший ніж звичайний метод.

## Лабораторна робота № 7-8

### Швидкі методи сортування

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

### 7.1 Хід роботи

Завдання 1:

Маршрути руху автобусів з Києва;

1. Київ –(135) Житомир –(80) Новоград-Волинський –(100) Рівно –(68)

Луцьк

2. Київ –(135) Житомир –(38) Бердичев –(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

### Графи. Робота з графами.

Графи - це абстрактний спосіб представлення типів відносин, наприклад доріг, що з'єднують міста, і інших видів мереж. Графи складаються з ребер і вершин. Вершина - це точка на графі, а ребро - це те, що з'єднує дві точки на графі.

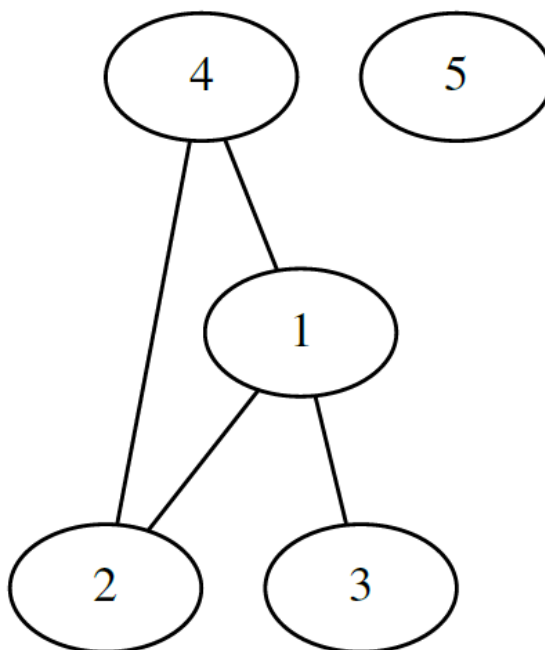


Рисунок 7.1 – Приклад графу

		Багманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				59
Змн.	Арк.	№ докум.	Підпис	Дата		

**Матриця суміжностей** - являє собою граф у вигляді двовимірної матриці з розмірами  $V \times V$ , де  $V$  - кількість вершин графа. Матриці суміжності найкраще застосовувати, коли  $V^2$  приблизно дорівнює  $E$  (числу ребер), тобто коли граф щільний.

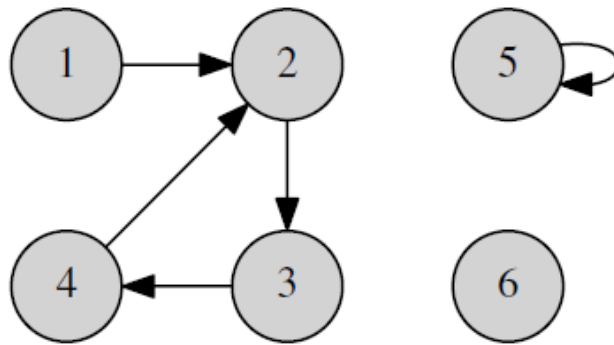


Рисунок 7.2 – Приклад графу який можна представити матрицею

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Рисунок 7.3 – Матриця суміжностей

### Пошук в глибину(DFS):

Пошук в глибину - це один з базових алгоритмів на графах. Він застосовується для пошуку відстані від однієї вершини до інших вершин в графі. Це алгоритм обходу.

Пошук в глибину позначає кожну вершину в графі однієї з двох назв відвіданих або НЕ відвіданих. Алгоритм позначає кожну вершину як відвідану, якщо вдається уникнути циклів. Він працює таким чином:

- 1) Розміщуємо будь-яку з вершин графа в стек.
- 2) Беремо елемент зі стека і додаємо його в список відвіданих.
- 3) Створюємо список сусідів цієї вершини. Додаємо в стек ті, що не знаходяться в списку відвіданих.
- 4) Повторюємо 2 і 3 пункти, поки стік не спорожніє.

### Пошук в ширину(BFS):

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

Пошук в ширину теж поміщає кожну вершину в графі в одну з двох категорій: відвіданих або невідвіданих. І мета у обох алгоритмів одна і та ж: позначати кожну вершину в графі як відвіданих, якщо вдається уникнути циклів. Ось як працює алгоритм пошуку в ширину:

- 1) Розміщуємо будь-яку вершину в графі в кінець черги.
- 2) Беремо елемент на початку черги і додаємо його в список відвіданих.
- 3) Створюємо список сусідів цієї вершини. Додаємо в кінець черги невідвіданих.
- 4) Повторюємо 2 і 3 пункти, поки черга не спорожніє.

Реалізовану матрицю надав в Excel документі репозиторію.

Реалізований алгоритм:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Task1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.OutputEncoding = Encoding.Unicode;
            Console.InputEncoding = Encoding.Unicode;

            for (int i = 0; i < n; i++)
            {
                Marked[i] = false;
            }

            Console.WriteLine("DFS:");
            DFS(0);

            for (int i = 0; i < n; i++)
            {
                Marked[i] = false;
            }

            Console.WriteLine("\nBFS:");
            BFS(0);
        }
        public static string[] Routes = //масив с названиями городов
        {
            "Київ",
            "Житомир",
            "Біла церква",
            "Прилуки",
            "Новоград-Волинський",
            "Бердичів",
            "Шепетівка",
            "Умань",
            "Черкаси",
            "Полтава",
            "Суми",
            "Миргород",
            "Рівне",
            "Вінниця",
        }
    }
}
```

		Башиманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				61
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        "Кременчук",
        "Харків",
        "Луцьк",
        "Хмельницький",
        "Тернопіль",
    };

    public static byte[,] MatrizaSmezhnosti = // Создаем матрицу
    {
        {0, 135, 78, 128, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 80, 38, 115, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 115, 146, 181, 0, 0, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 175, 109, 0, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 100, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 73, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 105, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 130, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 68, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 110, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 104 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
    };

    public const int n = 19; // кол-во вершин матрицы
    public static bool[] Marked = new bool[n]; // массив используемых вершин
    public static List<int> List = new List<int>(); // стек
    public static Queue<int> Queue = new Queue<int>(); // очередь в стеке
    public static List<int> Distance = new List<int>(); // расстояние между городами

    public static void DFS(int f, int sum = 0) //алгоритм DFS
    {
        if (!Marked[f])
        {
            List.Add(f);
            Marked[f] = true;
            for (int i = 0; i < n; i++) //проход всех вершин графа
            {
                if (MatrizaSmezhnosti[f, i] != 0)
                {
                    sum += MatrizaSmezhnosti[f, i]; //добавление вершин в стек
                    DFS(i, sum);
                    foreach (int town in List)
                    {
                        Console.WriteLine($"{Routes[town]} - ");
                    }
                    Console.WriteLine($"{sum}\n");
                    sum -= MatrizaSmezhnosti[f, i]; //убираем со стека верхнии вершины
                    List.RemoveAt(List.Count - 1);
                }
            }
        }
    }

    public static void BFS(int f) //алгоритм BFS
    {
        string start = Routes[f];
    }

```

		Башманівський			ДУ «Житомирська політехніка».20.121.3.000 – Лр	Арк.
		Петросян Р.В.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

