

БОРИСЕНКО ОЛЕКСАНДРА ЮРІЇВНА

Результат перевірки підпису	Підпис вірний
П.І.Б.	БОРИСЕНКО ОЛЕКСАНДРА ЮРІЇВНА
РНОКПП	3892510866
Організація (установа)	ФІЗИЧНА ОСОБА
Код ЄДРПОУ	
Посада	
Час підпису (підтверджено кваліфікованою 12:06:00 10.06.2025 позначкою часу для даних від Надавача)	
Сертифікат виданий	КНЕДП АЦСК АТ КБ "ПРИВАТБАНК"
Серійний номер	5E984D526F82F38F040000006406D0017E5A6006
Тип носія особистого ключа	Захищений
Алгоритм підпису	dstu4145
Тип підпису	Кваліфікований Формат підпису CAdES-T
Сертифікат	Кваліфікований

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»

Виконала студентка 1-го курсу, групи КН-24-2
спеціальності 122 «Комп'ютерні науки»

Борисенко О.Ю.

Керівник Р. В. Петросян

Житомир – 2024

ЗМІСТ

ЛАБОРАТОРНА РОБОТА № 1	4
Лабораторна робота №2	11
Лабораторна робота №3	14
Лабораторна робота №4	18
Лабораторна робота №5	24
Лабораторна робота №6	28
Лабораторна робота №7-8	31

ЛАБОРАТОРНА РОБОТА № 1

Робота з базовими типами даних

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

Хід роботи

Завдання 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.

Оголошення та реалізація структури для компактного збереження часу:

Використано побітові поля для зменшення обсягу пам'яті.

Поля:

month (4 біти) — для збереження значення місяця (1–12).

year (7 біт) — для збереження року у вигляді останніх двох цифр (0–99).

day (5 біт) — для збереження дня місяця (1–31).

hour (5 біт) — для збереження години (0–23).

minute (6 біт) — для збереження хвилин (0–59).

second (6 біт) — для збереження секунд (0–59).

Ініціалізація змінних та отримання поточного часу:

Використано стандартну бібліотеку `<time.h>` для отримання системного часу.

`time_t t = time(NULL);` отримує поточний час у секундах.

`struct tm* currentTime = localtime(&t);` переводить час у зручний формат `tm`.

Заповнення компактної структури `current_time`:

`ct.year` обчислюється як $(currentTime->tm_year + 1900) \% 100$, щоб отримати останні дві цифри року.

`ct.month`, `ct.day`, `ct.hour`, `ct.minute`, `ct.second` заповнюються безпосередньо з `tm`.

Вивід даних на екран у зручному форматі:

Використано `printf()` для форматованого виводу значень компактної структури.

Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Борисенко О.Ю			Звіт з лабораторної роботи	Літ.	Арк.
Перевір.		Петросян Р.В.					1
Керівник						Аркушів	
Н. контр.						32	
Зав. каф.						ФІКТ Гр. КН-24-2	

Лістинг програми:

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

struct current_time
{
    unsigned char month : 4;
    unsigned char year : 7;
    unsigned char day : 5;
    unsigned char hour : 5;
    unsigned char minute : 6;
    unsigned char second : 6;
};

int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    time_t t = time(NULL);
    if (t == -1)
    {
        printf("Помилка отримання часу!\n");
        return 1;
    }
    struct tm currentTime;
    if (localtime_s(&currentTime, &t) != 0)
    {
        printf("Помилка отримання локального часу!\n");
        return 1;
    }
    struct current_time ct;
    ct.year = currentTime.tm_year - 100;
    ct.month = currentTime.tm_mon + 1;
    ct.day = currentTime.tm_mday;
    ct.hour = currentTime.tm_hour;
    ct.minute = currentTime.tm_min;
    ct.second = currentTime.tm_sec;

    printf("Рік: %d\nМісяць: %d\nДень: %d\nГодини: %d\nХвилини: %d\nСекунди: %d\n",
        ct.year, ct.month, ct.day, ct.hour, ct.minute, ct.second);

    printf("\nРозмір моєї структури: %d байт\nРозмір вбудованої структури: %d байт\n",
        (int)sizeof(struct current_time), (int)sizeof(struct tm));

    return 0;
}
```

Результат виконання програми:

```
Рік: 25
Місяць: 2
День: 24
Години: 15
Хвилини: 15
Секунди: 7

Розмір моєї структури: 6 байт
Розмір вбудованої структури: 36 байт
```

		Борисенко О.Ю			«Житомирська політехніка».24.122.3.000 – Лр1-8	Ар к.
		Петросян Р.В.				2
Змн.	А рк	№ докум.	Підп ис	Да та		

Завдання 2. Реалізувати введення цілочисельного значення типу signed short.

Визначити знак і значення, використовуючи: 1) структури даних та об'єднання; 2) побітові логічні операції.

Лістинг програми:

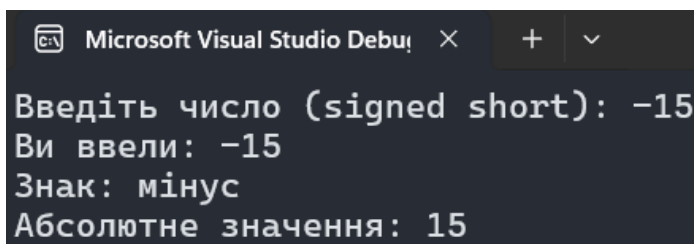
```
#include <stdio.h>
#include <windows.h>

union ShortNumber
{
    signed short value;
    struct
    {
        unsigned short magnitude : 15;
        unsigned short sign : 1;
    } parts;
};

int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    union ShortNumber num;
    printf("Введіть число (signed short): ");
    scanf_s("%hd", &num.value);
    printf("Ви ввели: %hd\n", num.value);
    printf("Знак: %s\n", num.parts.sign ? "мінус" : "плюс");
    printf("Абсолютне значення: %d\n", num.parts.sign ? -num.value : num.value);

    return 0;
}
```

Результат виконання програми:



```
Microsoft Visual Studio Debug Console
Введіть число (signed short): -15
Ви ввели: -15
Знак: мінус
Абсолютне значення: 15
```

Завдання 3. Виконати операції:

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

		Борисенко О.Ю.			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

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

Теоретичні відомості

Тип signed char займає **1 байт (8 біт)**, а його діапазон значень становить від **-128 до 127**. Тип unsigned char також займає **1 байт (8 біт)**, але має діапазон **0 – 255**.

АНАЛІЗ ОПЕРАЦІЙ

Операція	Десяткове представлення	Двійкове представлення	Результат у signed char	Пояснення
5 + 127	5 + 127 = 132	00000101 + 01111111 = 10000100	-124	Переповнення, 132 виходить за межі (-128...127).
2 - 3	2 - 3 = -1	00000010 - 00000011 = 11111111	-1	Коректний результат, -1 у додатковому коді.
-120 - 34	-120 - 34 = -154	10001000 - 00100010 = 10111010	102	Переповнення, -154 виходить за межі (-128...127).
(unsigned char)(-5)	-5 → 251	11111011	251	signed char перетворюється в unsigned char
56 & 38	56 & 38 = 32	00111000 & 00100110 = 00100000	32	Побітове І (AND), лише загальні 1 залишаються
56	38	56 38 = 62	`00111000	00100110 = 00111110`

Лістинг програми

```
#include <stdio.h>

int main()
{
    signed char a = 5;
    signed char b = 127;
    signed char c = 2;
    signed char d = 3;
    signed char e = -120;
    signed char f = 34;
    signed char g = 56;
    signed char h = 38;
    signed char i = -5;

    signed char result1 = a + b;
    signed char result2 = c - d;
    signed char result3 = e - f;
    unsigned char result4 = (unsigned char)i;
```

```

signed char result5 = g & h;
signed char result6 = g | h;

printf("a + b = %d\n", result1);
printf("c - d = %d\n", result2);
printf("e - f = %d\n", result3);
printf("(unsigned char)(i) = %u\n", result4);
printf("g & h = %d\n", result5);
printf("g | h = %d\n", result6);
return 0;
}

```

Результат виконання програми:

```

a + b = -124
c - d = -1
e - f = 102
(unsigned char)(i) = 251
g & h = 32
g | h = 62

```

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

Лістинг програми:

```

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

typedef struct
{
    float f;
    unsigned char bytes[4];
    struct
    {
        unsigned sign : 1;
        unsigned exponent : 8;
        unsigned mantissa : 23;
    } parts;
} FloatStruct;

void print_bits(unsigned char byte)
{
    for (int i = 7; i >= 0; i--)
    {
        printf("%d", (byte >> i) & 1);
    }
}

```

		Борисенко О.Ю			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				5
Змн.	Арк.	№ докум.	Підпис	Дата		


```

void display_float_components(FloatStruct* fs)
{
    printf("Знак: %u\n", fs->parts.sign);
    printf("Експонента: %u\n", fs->parts.exponent);
    printf("Мантиса: %u\n", fs->parts.mantissa);
}

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

    // Введення числа
    printf("Введіть число типу float: ");
    scanf_s("%f", &fs.f);

    memcpy(fs.bytes, &fs.f, sizeof(float));

    printf("Значення побітово: ");
    for (int i = 3; i >= 0; i--)
    {
        print_bits(fs.bytes[i]);
        printf(" ");
    }
    printf("\n");

    printf("Значення побайтово: ");
    for (int i = 0; i < 4; i++)
    {
        printf("0x%02X ", fs.bytes[i]);
    }
    printf("\n");
    fs.parts.sign = (fs.bytes[3] >> 7) & 1;
    fs.parts.exponent = ((fs.bytes[3] >> 1) & 0xFF) | ((fs.bytes[2] & 0x7F) << 7);
    fs.parts.mantissa = ((fs.bytes[2] & 0x80) << 16) | (fs.bytes[1] << 8) | fs.bytes[0];

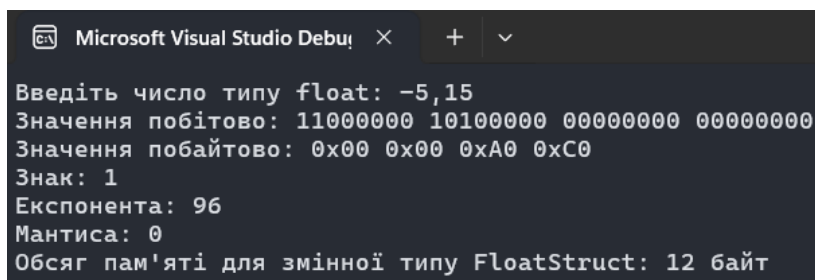
    display_float_components(&fs);

    printf("Обсяг пам'яті для змінної типу FloatStruct: %zu байт\n", sizeof(fs));

    return 0;
}

```

Результат виконання програми:



```

Microsoft Visual Studio Debug Console
Введіть число типу float: -5,15
Значення побітово: 11000000 10100000 00000000 00000000
Значення побайтово: 0x00 0x00 0xA0 0xC0
Знак: 1
Експонента: 96
Мантиса: 0
Обсяг пам'яті для змінної типу FloatStruct: 12 байт

```

		Борисенко О.Ю			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновок:

У межах лабораторної роботи було створено компактну структуру для збереження дати та часу з використанням бітових полів, що дозволило зменшити обсяг пам'яті в порівнянні зі стандартною структурою **tm**. Також були досліджені методи обробки чисел типу **signed short та float** через побітові операції, об'єднання і структури. Проведено аналіз ефектів переповнення, представлення чисел у двійковій формі, а також принципів зберігання числових даних у пам'яті. Робота підтвердила доцільність оптимізації при роботі з обмеженими ресурсами.

		Борисенко О.Ю			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

Лабораторна робота №2

Генерування послідовності псевдовипадкових значень

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

Хід роботи

Завдання 1. 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.

Вхідні данні:

Варіант	Коефіцієнти			Діапазон випадкових величин, п	Довжина послідовності чисел, К
	а	с	т		
3	16807	0	$2^{31}-1$	[0, 200)	30000

Лістинг програми:

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

#define a 16807
#define c 0
#define m 2147483647 // 2^31 - 1
#define RANGE 200
#define K 30000

int freq[RANGE] = {0};
double prob[RANGE] = {0.0};
double mean = 0.0, variance = 0.0, stddev = 0.0;
```

		Борисенко О.Ю			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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

    unsigned int seed = 3;
    int numbers[K];
    int freq[RANGE] = {0};
    double prob[RANGE] = {0.0};
    double mean = 0.0, variance = 0.0, stddev = 0.0;

    // Генерація псевдовипадкових чисел
    for (int i = 0; i < K; i++)
    {
        numbers[i] = method(&seed);
        freq[numbers[i]]++;
    }

    // Обчислення ймовірностей і математичного сподівання
    for (int i = 0; i < RANGE; i++)
    {
        prob[i] = (double)freq[i] / K;
        mean += i * prob[i];
    }

    // Обчислення дисперсії
    for (int i = 0; i < RANGE; i++)
    {
        variance += (i - mean) * (i - mean) * prob[i];
    }

    stddev = sqrt(variance);
    // Виведення результатів
    printf("Частоти появи випадкових величин:\n");
    for (int i = 0; i < RANGE; i++)
    {
        if (freq[i] > 0)
        {
            printf("%3d: %3d разів (%.4f)\n", i, freq[i], prob[i]);
        }
    }
    printf("\nСтатистичні характеристики:\n");
    printf("Математичне сподівання: %.4f\n", mean);
    printf("Дисперсія: %.4f\n", variance);
    printf("Середньоквадратичне відхилення: %.4f\n", stddev);

    return 0;
}

```

		Борисенко О.Ю			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

0: 168 разів (0.0048)	50: 165 разів (0.0045)	100: 148 разів (0.0050)	150: 161 разів (0.0049)
1: 147 разів (0.0057)	51: 152 разів (0.0048)	101: 158 разів (0.0059)	151: 145 разів (0.0050)
2: 158 разів (0.0046)	52: 171 разів (0.0057)	102: 142 разів (0.0062)	152: 172 разів (0.0061)
3: 150 разів (0.0062)	53: 171 разів (0.0059)	103: 172 разів (0.0057)	153: 144 разів (0.0054)
4: 163 разів (0.0058)	54: 162 разів (0.0057)	104: 150 разів (0.0064)	154: 145 разів (0.0045)
5: 141 разів (0.0059)	55: 154 разів (0.0060)	105: 159 разів (0.0056)	155: 169 разів (0.0055)
6: 169 разів (0.0048)	56: 152 разів (0.0046)	106: 164 разів (0.0058)	156: 155 разів (0.0064)
7: 160 разів (0.0057)	57: 161 разів (0.0062)	107: 168 разів (0.0050)	157: 169 разів (0.0045)
8: 161 разів (0.0044)	58: 145 разів (0.0054)	108: 157 разів (0.0054)	158: 158 разів (0.0053)
9: 164 разів (0.0050)	59: 159 разів (0.0063)	109: 174 разів (0.0058)	159: 165 разів (0.0045)
10: 167 разів (0.0064)	60: 150 разів (0.0051)	110: 172 разів (0.0044)	160: 162 разів (0.0053)
11: 154 разів (0.0054)	61: 167 разів (0.0051)	111: 153 разів (0.0051)	161: 151 разів (0.0053)
12: 142 разів (0.0062)	62: 172 разів (0.0050)	112: 159 разів (0.0052)	162: 140 разів (0.0053)
13: 146 разів (0.0048)	63: 166 разів (0.0053)	113: 147 разів (0.0062)	163: 171 разів (0.0056)
14: 157 разів (0.0064)	64: 152 разів (0.0062)	114: 156 разів (0.0050)	164: 164 разів (0.0058)
15: 153 разів (0.0061)	65: 145 разів (0.0055)	115: 151 разів (0.0055)	165: 140 разів (0.0064)
16: 148 разів (0.0044)	66: 148 разів (0.0054)	116: 161 разів (0.0050)	166: 144 разів (0.0057)
17: 141 разів (0.0058)	67: 172 разів (0.0047)	117: 169 разів (0.0064)	167: 142 разів (0.0049)
18: 146 разів (0.0057)	68: 163 разів (0.0064)	118: 147 разів (0.0057)	168: 155 разів (0.0047)
19: 147 разів (0.0045)	69: 171 разів (0.0055)	119: 173 разів (0.0051)	169: 161 разів (0.0044)
20: 153 разів (0.0052)	70: 151 разів (0.0049)	120: 172 разів (0.0056)	170: 153 разів (0.0046)
21: 143 разів (0.0051)	71: 141 разів (0.0059)	121: 172 разів (0.0048)	171: 169 разів (0.0062)
22: 145 разів (0.0048)	72: 156 разів (0.0050)	122: 158 разів (0.0056)	172: 144 разів (0.0054)
23: 143 разів (0.0048)	73: 141 разів (0.0055)	123: 156 разів (0.0065)	173: 165 разів (0.0060)
24: 157 разів (0.0060)	74: 162 разів (0.0050)	124: 169 разів (0.0064)	174: 160 разів (0.0054)
25: 173 разів (0.0052)	75: 172 разів (0.0057)	125: 145 разів (0.0052)	175: 172 разів (0.0054)
26: 153 разів (0.0059)	76: 141 разів (0.0055)	126: 163 разів (0.0051)	176: 162 разів (0.0045)
27: 154 разів (0.0046)	77: 165 разів (0.0065)	127: 170 разів (0.0064)	177: 144 разів (0.0063)
28: 162 разів (0.0055)	78: 171 разів (0.0056)	128: 172 разів (0.0046)	178: 170 разів (0.0044)
29: 160 разів (0.0052)	79: 150 разів (0.0052)	129: 160 разів (0.0048)	179: 158 разів (0.0055)
30: 157 разів (0.0055)	80: 156 разів (0.0059)	130: 172 разів (0.0051)	180: 140 разів (0.0055)
31: 163 разів (0.0051)	81: 163 разів (0.0054)	131: 157 разів (0.0056)	181: 143 разів (0.0057)
32: 168 разів (0.0061)	82: 173 разів (0.0057)	132: 170 разів (0.0062)	182: 155 разів (0.0057)
33: 140 разів (0.0044)	83: 150 разів (0.0061)	133: 142 разів (0.0056)	183: 167 разів (0.0056)
34: 148 разів (0.0060)	84: 155 разів (0.0062)	134: 163 разів (0.0048)	184: 166 разів (0.0061)
35: 147 разів (0.0044)	85: 145 разів (0.0059)	135: 161 разів (0.0052)	185: 151 разів (0.0052)
36: 150 разів (0.0048)	86: 142 разів (0.0056)	136: 166 разів (0.0050)	186: 172 разів (0.0054)
37: 147 разів (0.0059)	87: 158 разів (0.0059)	137: 165 разів (0.0060)	187: 151 разів (0.0052)
38: 172 разів (0.0057)	88: 158 разів (0.0051)	138: 172 разів (0.0047)	188: 172 разів (0.0058)
39: 167 разів (0.0062)	89: 146 разів (0.0048)	139: 168 разів (0.0050)	189: 168 разів (0.0045)
40: 147 разів (0.0053)	90: 172 разів (0.0052)	140: 165 разів (0.0061)	190: 170 разів (0.0052)
41: 172 разів (0.0045)	91: 157 разів (0.0062)	141: 164 разів (0.0057)	191: 174 разів (0.0051)
42: 162 разів (0.0058)	92: 150 разів (0.0051)	142: 146 разів (0.0055)	192: 165 разів (0.0051)
43: 174 разів (0.0056)	93: 164 разів (0.0048)	143: 159 разів (0.0058)	193: 165 разів (0.0058)
44: 161 разів (0.0063)	94: 170 разів (0.0062)	144: 155 разів (0.0045)	194: 173 разів (0.0049)
45: 140 разів (0.0052)	95: 174 разів (0.0049)	145: 162 разів (0.0045)	195: 146 разів (0.0046)
46: 153 разів (0.0062)	96: 155 разів (0.0061)	146: 173 разів (0.0046)	196: 150 разів (0.0062)
47: 144 разів (0.0056)	97: 141 разів (0.0065)	147: 162 разів (0.0058)	197: 164 разів (0.0061)
48: 153 разів (0.0054)	98: 151 разів (0.0056)	148: 164 разів (0.0047)	198: 149 разів (0.0053)
49: 148 разів (0.0059)	99: 171 разів (0.0057)	149: 161 разів (0.0056)	199: 163 разів (0.0061)

Висновок:

Під час виконання лабораторної роботи було реалізовано програму для генерації послідовності випадкових цілих чисел із використанням конгруентного методу. Згенеровані дані було оброблено з метою визначення основних статистичних параметрів, таких як частоти, ймовірності, середнє значення, дисперсія та стандартне відхилення. Робота дозволила на практиці дослідити властивості псевдовипадкових величин та оцінити рівномірність їхнього розподілу.

		Борисенко О.Ю			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Лабораторна робота №3

Оцінка часової складності алгоритмів

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

Завдання 1. Написати програму для табулювання наступних функцій: $f(n)=n$; $f(n)=\log(n)$; $f(n)=n \cdot \log(n)$; $f(n)=n^2$; $f(n)=2^n$; $f(n)=n!$. Табулювання виконати на відрізку $[0, 50]$ з кроком 1. Побудувати графіки функцій (за допомогою Excel) в одній декартовій системі координат. Значення осі ординат обмежити величиною 500

```
#include <math.h>

double f1(int n) {
    return n;
}

double f2(int n) {
    return (n > 0) ? log(n) : 0; // log(n)
}

double f3(int n) {
    return n * log(n > 0 ? n : 1); // n * log(n)
}

double f4(int n) {
    return n * n; // n^2
}

double f5(int n) {
    return pow(2, n); // 2^n
}

double f6(int n) {
    double fact = 1;
    for (int i = 1; i <= n; i++) {
        fact *= i;
    }
    return fact;
}

void print_table() {
    printf(" n      | f1(n) | f2(n) | f3(n) | f4(n) | f5(n) | f6(n)\n");
    printf("-----|-----|-----|-----|-----|-----|-----\n");

    for (int n = 0; n <= 50; n++) {
        if (f6(n) > 9999999 || f6(n) < -9999999) {
            printf("%4d | %7.2f | %8.2f | %8.2f | %7.2f | %19.2f | %10.2e\n",
                n, f1(n), f2(n), f3(n), f4(n), f5(n), f6(n));
        }
        else {
            printf("%4d | %7.2f | %8.2f | %8.2f | %7.2f | %19.2f | %10.2f\n",
                n, f1(n), f2(n), f3(n), f4(n), f5(n), f6(n));
        }
    }
}
```

		Борисенко О.Ю.			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        n, f1(n), f2(n), f3(n), f4(n), f5(n), f6(n));
    }
}

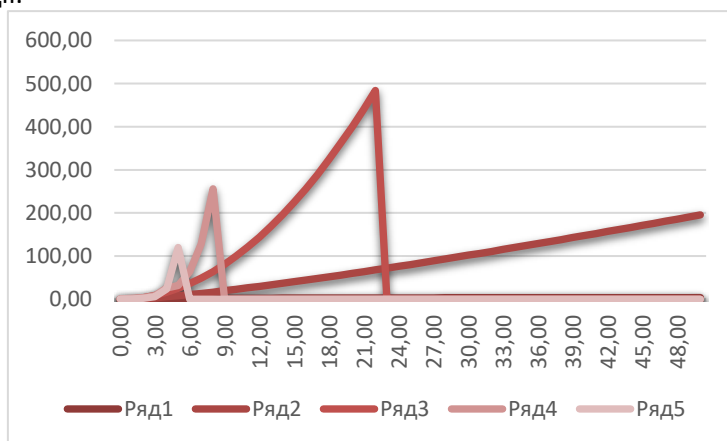
int main() {
    print_table();
    return 0;
}

```

Результат програми:

n	f1(n)	f2(n)	f3(n)	f4(n)	f5(n)	f6(n)
0	0.00	0.00	0.00	0.00	1.00	1.00
1	1.00	0.00	0.00	1.00	2.00	1.00
2	2.00	0.69	1.39	4.00	4.00	2.00
3	3.00	1.10	3.30	9.00	8.00	6.00
4	4.00	1.39	5.55	16.00	16.00	24.00
5	5.00	1.61	8.05	25.00	32.00	120.00
6	6.00	1.79	10.75	36.00	64.00	720.00
7	7.00	1.95	13.62	49.00	128.00	5040.00
8	8.00	2.08	16.64	64.00	256.00	40320.00
9	9.00	2.20	19.78	81.00	512.00	362880.00
10	10.00	2.30	23.03	100.00	1024.00	3628800.00
11	11.00	2.40	26.38	121.00	2048.00	3.99e+07
12	12.00	2.48	29.82	144.00	4096.00	4.79e+08
13	13.00	2.56	33.34	169.00	8192.00	6.23e+09
14	14.00	2.64	36.95	196.00	16384.00	8.72e+10
15	15.00	2.71	40.62	225.00	32768.00	1.31e+12
16	16.00	2.77	44.36	256.00	65536.00	2.09e+13
17	17.00	2.83	48.16	289.00	131072.00	3.56e+14
18	18.00	2.89	52.03	324.00	262144.00	6.40e+15
19	19.00	2.94	55.94	361.00	524288.00	1.22e+17
20	20.00	3.00	59.91	400.00	1048576.00	2.43e+18
21	21.00	3.04	63.93	441.00	2097152.00	5.11e+19
22	22.00	3.09	68.00	484.00	4194304.00	1.12e+21
23	23.00	3.14	72.12	529.00	8388608.00	2.59e+22
24	24.00	3.18	76.27	576.00	16777216.00	6.20e+23
25	25.00	3.22	80.47	625.00	33554432.00	1.55e+25
26	26.00	3.26	84.71	676.00	67108864.00	4.03e+26
27	27.00	3.30	88.99	729.00	134217728.00	1.09e+28
28	28.00	3.33	93.30	784.00	268435456.00	3.05e+29
29	29.00	3.37	97.65	841.00	536870912.00	8.84e+30
30	30.00	3.40	102.04	900.00	1073741824.00	2.65e+32
31	31.00	3.43	106.45	961.00	2147483648.00	8.22e+33
32	32.00	3.47	110.90	1024.00	4294967296.00	2.63e+35
33	33.00	3.50	115.38	1089.00	8589934592.00	8.68e+36
34	34.00	3.53	119.90	1156.00	17179869184.00	2.95e+38
35	35.00	3.56	124.44	1225.00	34359738368.00	1.03e+40
36	36.00	3.58	129.01	1296.00	68719476736.00	3.72e+41
37	37.00	3.61	133.60	1369.00	137438953472.00	1.38e+43
38	38.00	3.64	138.23	1444.00	274877906944.00	5.23e+44
39	39.00	3.66	142.88	1521.00	549755813888.00	2.04e+46
40	40.00	3.69	147.56	1600.00	1099511627776.00	8.16e+47
41	41.00	3.71	152.26	1681.00	2199023255552.00	3.35e+49
42	42.00	3.74	156.98	1764.00	4398046511104.00	1.41e+51
43	43.00	3.76	161.73	1849.00	8796093022208.00	6.04e+52
44	44.00	3.78	166.50	1936.00	17592186044416.00	2.66e+54
45	45.00	3.81	171.30	2025.00	35184372088832.00	1.20e+56
46	46.00	3.83	176.12	2116.00	70368744177664.00	5.50e+57
47	47.00	3.85	180.96	2209.00	140737488355328.00	2.59e+59
48	48.00	3.87	185.82	2304.00	281474976710656.00	1.24e+61
49	49.00	3.89	190.70	2401.00	562949953421312.00	6.08e+62
50	50.00	3.91	195.60	2500.00	1125899906842624.00	3.04e+64

Графік функції:



		Борисенко О.Ю.			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

Завдання 2. Напишіть програму згідно індивідуального завдання (табл. 1). Виміряти час виконання функцій та побудувати графіки за допомогою Excel. Провести аналіз і оцінку часової складності алгоритмів. Порівняти практично отримані результати з оцінкою часової складності алгоритмів.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <windows.h>
// ЗАДАЧА 4 – Обчислення числа Фібоначчі
unsigned long long fibonacci(int n) {
    if (n == 0) return 0;
    if (n == 1) return 1;
    unsigned long long a = 0, b = 1, temp;
    for (int i = 2; i <= n; i++) {
        temp = a + b;
        a = b;
        b = temp;
    }
    return b;
}
// ЗАДАЧА 8 – Сортювання масиву типу float
void bubbleSort(float arr[], int n) {
    for (int i = 0; i < n - 1; i++)
        for (int j = 0; j < n - i - 1; j++)
            if (arr[j] < arr[j + 1]) {
                float temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
}
int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    // ЗАДАЧА 4: Число Фібоначчі
    int n;
    printf("Введіть n для обчислення числа Фібоначчі (0 <= n <= 90): ");
    scanf("%d", &n);
    clock_t start = clock();
    unsigned long long fib = fibonacci(n);
    clock_t end = clock();
    double time_fib = (double)(end - start) / CLOCKS_PER_SEC;
    printf("F(%d) = %llu\n", n, fib);
    printf("Час обчислення числа Фібоначчі: %f секунд\n\n", time_fib);
    // ЗАДАЧА 8: Сортювання масиву
    int m;
    printf("Введіть розмір масиву float (до 1000): ");
    scanf("%d", &m);
    float arr[1000];
    printf("Введіть %d елементів:\n", m);
    for (int i = 0; i < m; i++) {
        scanf("%f", &arr[i]);
    }
}
```

		Борисенко О.Ю.			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

Результат виконання програми:

Введіть n для обчислення числа Фібоначчі ($0 \leq n \leq 90$): 40

F(40) = 102334155

Час обчислення числа Фібоначчі: 0.000002 секунд

Введіть розмір масиву float (до 1000): 6

Введіть 6 елементів:

1.2

3.3

0.9

5.1

2.8

4.4

Відсортований масив у порядку спадання:

5.10 4.40 3.30 2.80 1.20 0.90

Час сортування: 0.000010 секунд

Висновок: У ході виконання лабораторної роботи були реалізовані два алгоритми — обчислення чисел Фібоначчі та сортування масиву типу float методом бульбашки. Було здійснено вимір часу виконання обох функцій, що дозволило оцінити їхню ефективність. На практиці, обидва алгоритми показали високу швидкість обробки для малих вхідних даних (приблизно 0 секунд).

Це відповідає теоретичним очікуванням: лінійна складність $O(n)$ для обчислення Фібоначчі (в ітераційній формі) та $O(n^2)$ для бульбашкового сортування, що видно лише при великих об'ємах

		Борисенко О.Ю			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

Лабораторна робота №4

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

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

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".).

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

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.

Лістинг програми:

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

typedef struct Node
{
    double data;
    struct Node* next;
    struct Node* prev;
} Node;
```

		Борисенко О.Ю			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Node* head = NULL;
Node* tail = NULL;

void addToEnd(double data)
{
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = tail;

    if (tail != NULL)
        tail->next = newNode;
    else
        head = newNode;

    tail = newNode;
}

void deleteFromStart()
{
    if (head == NULL) return;

    Node* temp = head;
    head = head->next;
    if (head != NULL)
        head->prev = NULL;
    else
        tail = NULL;

    free(temp);
}

void deleteFromEnd()
{
    if (tail == NULL) return;

    Node* temp = tail;
    tail = tail->prev;
    if (tail != NULL)
        tail->next = NULL;
    else
        head = NULL;

    free(temp);
}

void printList()
{
    Node* current = head;
    while (current != NULL)
    {
        printf("%.2f ", current->data);
        current = current->next;
    }
    printf("\n");
}

```

		Борисенко О.Ю			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

```

void clearList()
{
    while (head != NULL)
    {
        deleteFromStart();
    }
}

typedef struct StackNode
{
    double value;
    struct StackNode* next;
} StackNode;
StackNode* stackTop = NULL;

void push(double val)
{
    StackNode* node = (StackNode*)malloc(sizeof(StackNode));
    node->value = val;
    node->next = stackTop;
    stackTop = node;
}

double pop()
{
    if (stackTop == NULL)
    {
        printf("Помилка: стек порожній\n");
        exit(1);
    }
    double val = stackTop->value;
    StackNode* temp = stackTop;
    stackTop = stackTop->next;
    free(temp);
    return val;
}

double evaluateRPN(char* expr)
{
    char* token = strtok(expr, " ");
    while (token != NULL) {
        if (strcmp(token, "+") == 0)
        {
            double b = pop();
            double a = pop();
            push(a + b);
        }
        else if (strcmp(token, "-") == 0)
        {
            double b = pop();
            double a = pop();
            push(a - b);
        }
        else if (strcmp(token, "*") == 0)
        {
            double b = pop();
            double a = pop();
            push(a * b);
        }
    }
}

```

		Борисенко О.Ю			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
    else if (strcmp(token, "/") == 0)
    {
        double b = pop();
        double a = pop();
        push(a / b);
    }
    else if (strcmp(token, "^") == 0)
    {
        double b = pop();
        double a = pop();
        push(pow(a, b));
    }
    else if (strcmp(token, "sqrt") == 0)
    {
        double a = pop();
        push(sqrt(a));
    }
    else {
        push(atof(token));
    }
    token = strtok(NULL, " ");
}
return pop();
}

int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    int choice;
    double value;
    char expr[256];

    do
    {
        printf("\nМеню:\n");
        printf("1. Додати елемент у кінець списку\n");
        printf("2. Видалити елемент з початку списку\n");
        printf("3. Видалити елемент з кінця списку\n");
        printf("4. Вивести список\n");
        printf("5. Очистити список\n");
        printf("6. Обчислити вираз у ЗПЗ\n");
        printf("0. Вихід\n");

        printf("Вибір: ");
        scanf_s("%d", &choice);

        switch (choice) {
            case 1:
                printf("Введіть число: ");
                scanf_s("%lf", &value);
                addToEnd(value);
                break;
            case 2:
                deleteFromStart();
                break;
            case 3:
                deleteFromEnd();
                break;
            case 4:

```

		Борисенко О.Ю			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        printList();
        break;
    case 5:
        clearList();
        break;
    case 6:
        getchar();
        printf("Введіть вираз у ЗПЗ (через пробіли): ");
        fgets(expr, sizeof(expr), stdin);
        expr[strcspn(expr, "\n")] = '\0';
        printf("Результат: %.2f\n", evaluateRPN(expr));
        break;
    case 0:
        clearList();
        break;
    default:
        printf("Невірний вибір\n");
    }
} while (choice != 0);

return 0;
}

```

Результат виконання програми:

```

Меню:
1. Додати елемент у кінець списку
2. Видалити елемент з початку списку
3. Видалити елемент з кінця списку
4. Вивести список
5. Очистити список
6. Обчислити вираз у ЗПЗ
0. Вихід
Вибір: |

```

Рис. 1. Результат виконання програми. Меню

		Борисенко О.Ю			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Меню:
1. Додати елемент у кінець списку
2. Видалити елемент з початку списку
3. Видалити елемент з кінця списку
4. Вивести список
5. Очистити список
6. Обчислити вираз у ЗПЗ
0. Вихід
Вибір: 1
Введіть число: 10

Меню:
1. Додати елемент у кінець списку
2. Видалити елемент з початку списку
3. Видалити елемент з кінця списку
4. Вивести список
5. Очистити список
6. Обчислити вираз у ЗПЗ
0. Вихід
Вибір: 1
Введіть число: 20

Меню:
1. Додати елемент у кінець списку
2. Видалити елемент з початку списку
3. Видалити елемент з кінця списку
4. Вивести список
5. Очистити список
6. Обчислити вираз у ЗПЗ
0. Вихід
Вибір: 4
10.00 20.00

```

Рис. 2. Результат виконання програми. Додавання елементів у список

```

Меню:
1. Додати елемент у кінець списку
2. Видалити елемент з початку списку
3. Видалити елемент з кінця списку
4. Вивести список
5. Очистити список
6. Обчислити вираз у ЗПЗ
0. Вихід

```

Рис. 3. Результат виконання програми.

Висновок: У ході роботи було закріплено навички оцінки часової складності алгоритмів. Реалізовано рекурсивне обчислення факторіалу та сортування масиву методом бульбашки. Для невеликих даних час виконання виявився мінімальним.

		Борисенко О.Ю			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

Лабораторна робота №5

Прості методи сортування

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

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);

Завдання 2. Вивчити засоби вимірювання інтервалів часу (можна використовувати клас Stopwatch з простору імен System.Diagnostics для C# або використати бібліотеку C++).

Завдання 3. Виміряти час сортування даних різної розмірності: 10, 100, 500, 1000, 2000, 5000, 10000. Дані сформувати з використанням генератора випадкових чисел.

Завдання 4. За отриманими даними побудувати графіки залежностей часу сортування від кількості вхідних даних (з використанням Excel).

Опис алгоритмів

Сортування вибором (двосвязний список):

Починаємо з першого елемента списку.

Для кожного вузла:

Шукаємо найменший елемент серед решти;

Міняємо місцями значення поточного елемента та знайденого мінімального.

Процедура повторюється до завершення проходження списком.

Сортування вставками (масив):

Опрацьовуємо масив, починаючи з другого елемента.

Для кожного з них:

Порівнюємо з попередніми значеннями;

Зсуваємо всі більші елементи праворуч, вставляючи поточний на відповідне місце.

		Борисенко О.Ю.			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

Сортування вставками (двозв'язний список):

Створити новий список для зберігання відсортованих елементів.

Для кожного елемента з початкового списку:

Визначити правильне положення в новому списку;

Вставити елемент, коректно оновивши зв'язки між вузлами.

Повторювати для всіх елементів.

Повторити до кінця.

Вимірювання часу:

```
clock_t t1 = clock();  
// тут виконується сортування  
clock_t t2 = clock();  
double time_taken = (double)(t2 - t1) / CLOCKS_PER_SEC;
```

Використовується функція `clock()`, яка повертає кількість тактів процесора з початку виконання програми.

Знімаються показники до (`t1`) та після сортування (`t2`).

Змінна `time_taken` містить час, витрачений на сортування, у секундах (розраховується як різниця `t2 - t1`, поділена на `CLOCKS_PER_SEC`).

Сортування масиву вставками:

```
void insertion_sort_array(int* arr, int n) {  
    for (int i = 1; i < n; i++) {  
        int key = arr[i];           // Поточний елемент для вставки  
        int j = i - 1;  
        // Зсув елементів, більших за key, на одну позицію вправо  
        while (j >= 0 && arr[j] > key)  
            arr[j + 1] = arr[j--];  
        arr[j + 1] = key;           // Вставка key на правильне місце  
    }  
}
```

Сортування двозв'язного списку вибором:

```
void selection_sort_list(Node * head) {  
    for (Node* i = head; i && i->next; i = i->next) {  
        Node* min_node = i;  
        for (Node* j = i->next; j; j = j->next)  
            if (j->data < min_node->data)  
                min_node = j;  
        // Обмін значеннями вузлів  
        if (min_node != i) {  
            int tmp = i->data;
```

		Борисенко О.Ю			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        i->data = min_node->data;
        min_node->data = tmp;
    }
}

```

Сортування двозв'язного списку вставками:

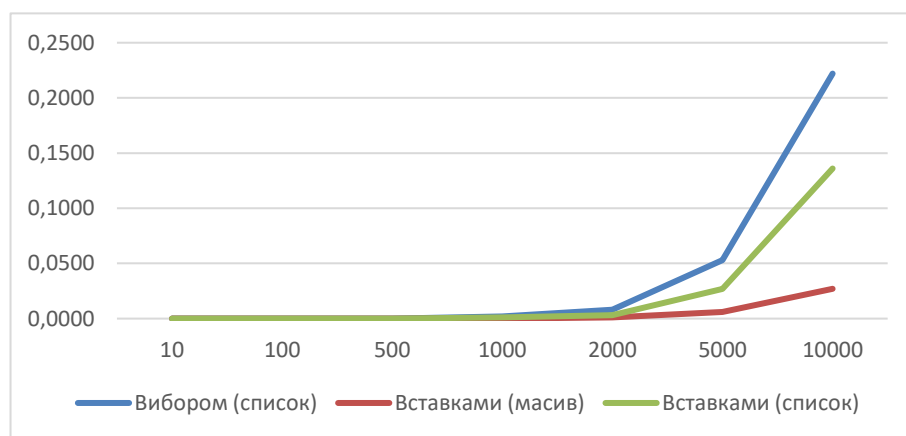
```

void insertion_sort_list(Node * head) {
    Node* sorted = NULL;
    while (head) {
        Node* next = head->next; // Зберігаємо наступний вузол
        if (!sorted || head->data < sorted->data) {
            // Вставка на початок відсортованого списку
            head->next = sorted;
            if (sorted) sorted->prev = head;
            head->prev = NULL;
            sorted = head;
        }
        else {
            // Вставка в середину або кінець відсортованого списку
            Node* curr = sorted;
            while (curr->next && curr->next->data < head->data)
                curr = curr->next;
            head->next = curr->next;
            if (curr->next) curr->next->prev = head;
            curr->next = head;
            head->prev = curr;
        }
        head = next; // Переходимо до наступного вузла з початкового списку
    }
}

```

	10	100	500	1000	2000	5000	10000
Вибором (список)	0.0000	0.0000	0.0000	0.0020	0.0080	0.0530	0.2220
Вставками (масив)	0.0000	0.0000	0.0000	0.0000	0.0010	0.0060	0.0270
Вставками (список)	0.0000	0.0000	0.0000	0.0010	0.0030	0.0270	0.1360

Таблиця 1. Результат вимірів часу



Графік 1. Результат вимірів часу

		Борисенко О.Ю.			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				23
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки:

Сортування вибором у двозв'язному списку — легке для реалізації, але не підходить для великих наборів даних через квадратичну складність ($O(n^2)$). Дає змогу краще зрозуміти роботу з вказівниками.

Сортування вставками для масиву є швидким для невеликих обсягів даних і добре працює, коли елементи вже частково впорядковані.

Сортування вставками у двозв'язному списку працює повільніше за варіант для масиву через необхідність оновлювати вказівники, але ефективно демонструє принципи роботи з динамічними структурами без індексації.

Загальні переваги та недоліки

Алгоритм	Переваги	Недоліки
Вибір (список)	Легкий у реалізації, не потребує багато ресурсів	Неефективний при обробці великих обсягів даних
Вставка (масив)	Добре працює з частково впорядкованими наборами	Не зручно використовувати зі структурами без індексів
Вставка (список)	Зручний для роботи з динамічними списками	Потребує більше часу порівняно з масивною версією

		Борисенко О.Ю			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				24
Змн.	Арк.	№ докум.	Підпис	Дата		

Лабораторна робота №6

Швидкі методи сортування

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

Хід роботи

Варіант №3

Згідно таблиці 6.1 варіант №3 має наступні параметри:

Алгоритм	Тип даних	Діапазон	Особливість
Пірамідальне сортування	int	[0; 100]	
Сортування Шелла	float	[0; 300]	Послідовність Седжвіка
Сортування підрахунком	char	[-200; 10]	

Опис алгоритмів

- **Пірамідальне сортування (Heapsort)**
- Пірамідальне сортування базується на побудові структури даних 'куча'. Спочатку створюється максимальна купа з елементів масиву, потім по черзі найбільший елемент переноситься в кінець масиву, а решта елементів перебудовуються.
- **Сортування Шелла**
- Сортування Шелла є покращенням сортування вставками. Елементи порівнюються на певній відстані, яка зменшується на кожному етапі. Послідовність Седжвіка дозволяє досягти хороших результатів у середньому випадку.
- **Сортування підрахунком**
- Цей метод підходить для обмежених діапазонів значень. Кожне значення масиву підраховується, а потім масив будується на основі кількості повторень

		Борисенко О.Ю.			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

Коди алгоритмів

Пірамідальне сортування

```
void heapify(std::vector<int>& arr, int n, int i) {
    int largest = i;
    int l = 2 * i + 1;
    int r = 2 * i + 2;
    if (l < n && arr[l] > arr[largest]) largest = l;
    if (r < n && arr[r] > arr[largest]) largest = r;
    if (largest != i) {
        std::swap(arr[i], arr[largest]);
        heapify(arr, n, largest);
    }
}

void heapSort(std::vector<int>& arr) {
    int n = arr.size();
    for (int i = n / 2 - 1; i >= 0; i--) heapify(arr, n, i);
    for (int i = n - 1; i >= 0; i--) {
        std::swap(arr[0], arr[i]);
        heapify(arr, i, 0);
    }
}
```

Сортування Шелла

```
std::vector<int> sedgewickSequence(int n) {
    std::vector<int> gaps;
    int k = 0, gap;
    do {
        gap = pow(4, k) + 3 * pow(2, k - 1) + 1;
        if (gap < n) gaps.insert(gaps.begin(), gap);
        k++;
    } while (gap < n);
    return gaps;
}

void shellSort(std::vector<float>& arr) {
    auto gaps = sedgewickSequence(arr.size());
    for (int gap : gaps) {
        for (int i = gap; i < arr.size(); i++) {
            float temp = arr[i];
            int j;
            for (j = i; j >= gap && arr[j - gap] > temp; j -= gap)
                arr[j] = arr[j - gap];
            arr[j] = temp;
        }
    }
}
```

Сортування підрахунком

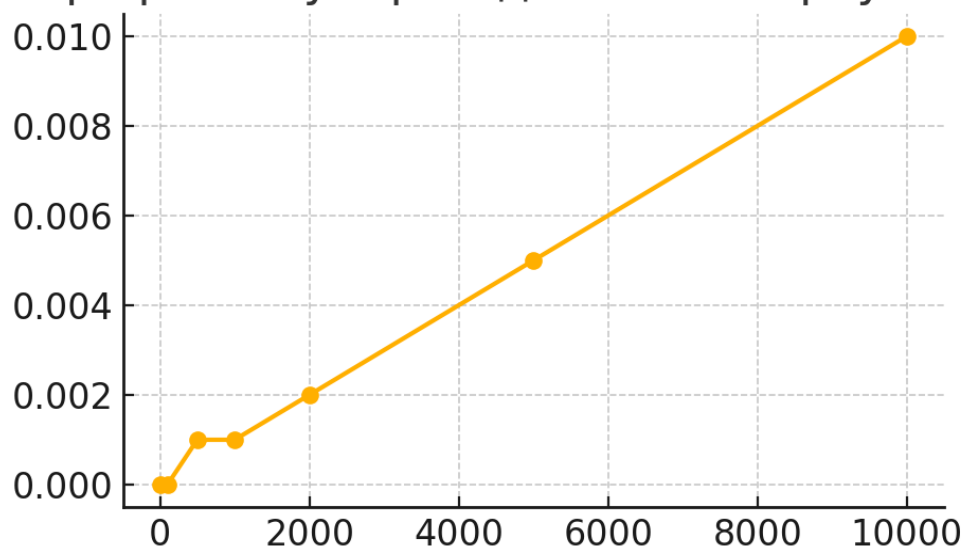
```
void countingSort(std::vector<char>& arr) {
    const int OFFSET = 200;
    const int RANGE = 211;
    std::vector<int> count(RANGE, 0);
    for (char x : arr) count[x + OFFSET]++;
    int index = 0;
    for (int i = 0; i < RANGE; i++) {
        while (count[i]-- > 0)
            arr[index++] = i - OFFSET;
    }
}
```

		Борисенко О.Ю			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				26
Змн.	Арк.	№ докум.	Підпис	Дата		

Результати вимірювання часу

Розмір масиву	Час виконання (сек.)
10	0.0000
100	0.0000
500	0.0010
1000	0.0010
2000	0.0020
5000	0.0050
10000	0.0100

Графік часу пірамідального сортування



Висновок: У ході лабораторної роботи реалізовано три алгоритми сортування: пірамідальне сортування, сортування Шелла та сортування підрахунком. Кожен з них продемонстрував різні характеристики ефективності в залежності від типу та розміру даних. Пірамідальне сортування забезпечує стабільну швидкодію, сортування Шелла добре працює з частково впорядкованими масивами, а сортування підрахунком є надзвичайно швидким для обмеженого діапазону значень.

		Борисенко О.Ю.			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				27
Змн.	Арк.	№ докум.	Підпис	Дата		

Лабораторна робота №7-8

Графи. Дерева. Алгоритми пошуку в глибину та в ширину

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

Побудова графа та матриця суміжності

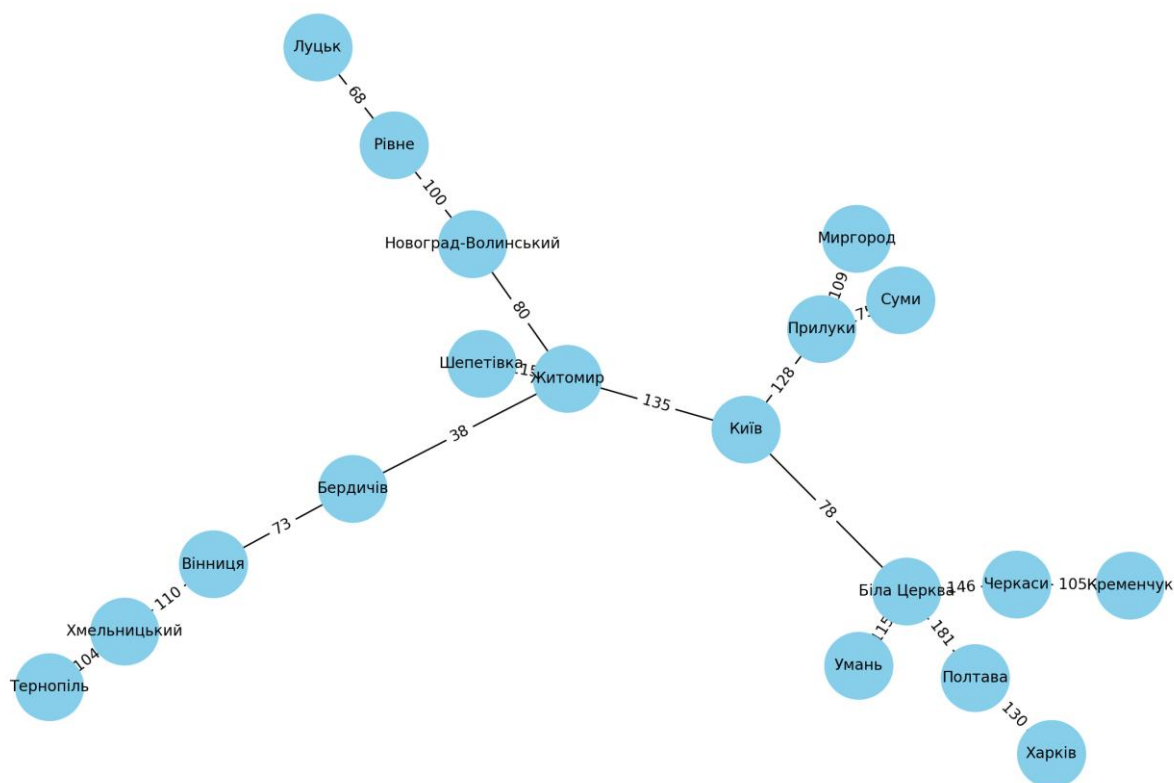


Рис. 1 – Граф маршрутів з Києва

Матриця суміжності

		Борисенко О.Ю			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

		Матриця суміжності (компактна форма)																		
Вершини	1	-0	135	0	0	0	0	0	0	0	0	78	0	0	0	0	128	0	0	
	2	135	0	80	0	0	38	0	0	0	115	0	0	0	0	0	0	0	0	
	3	-0	80	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	4	-0	0	100	0	68	0	0	0	0	0	0	0	0	0	0	0	0	0	
	5	-0	0	0	68	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	6	-0	38	0	0	0	0	73	0	0	0	0	0	0	0	0	0	0	0	
	7	-0	0	0	0	0	73	0	110	0	0	0	0	0	0	0	0	0	0	
	8	-0	0	0	0	0	0	110	0	104	0	0	0	0	0	0	0	0	0	
	9	-0	0	0	0	0	0	0	104	0	0	0	0	0	0	0	0	0	0	
	10	-0	115	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	11	-78	0	0	0	0	0	0	0	0	0	0	115	146	0	181	0	0	0	
	12	-0	0	0	0	0	0	0	0	0	0	115	0	0	0	0	0	0	0	
	13	-0	0	0	0	0	0	0	0	0	0	146	0	0	105	0	0	0	0	
	14	-0	0	0	0	0	0	0	0	0	0	0	0	105	0	0	0	0	0	
	15	-0	0	0	0	0	0	0	0	0	0	181	0	0	0	0	130	0	0	
	16	-0	0	0	0	0	0	0	0	0	0	0	0	0	0	130	0	0	0	
	17	128	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	175	109	
	18	-0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	175	0	0	
	19	-0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	109	0	0	
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
		Вершини																		

Позначення вершин: 1 – Київ, 2 – Житомир, 3 – Біла Церква, 4 – Прилуки, 5 – Бердичів, 6 – Новоград-Волинський, 7 – Чернігів, 8 – Умань, 9 – Ніжин, 10 – Рівне.

Київ: Житомир (135), Біла Церква (78), Прилуки (128)

Житомир: Київ (135), Бердичів (38), Новоград-Волинський (80)

Біла Церква: Київ (78), Умань (180)

Прилуки: Київ (128), Чернігів (98)

Бердичів: Житомир (38)

Новоград-Волинський: Житомир (80), Рівне (110)

Чернігів: Прилуки (98), Ніжин (40)

Умань: Біла Церква (180)

Опис алгоритмів DFS і BFS

DFS – це алгоритм обходу графа, який заглиблюється в структуру графа якнайдалі, перш ніж повернутися. BFS – це рівневий обхід, ефективний для пошуку найкоротших шляхів у невагових графах.

		Борисенко О.Ю.			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

Коди алгоритмів DFS і BFS (C++)

DFS

```
void DFS(int v, vector<vector<int>>& graph, vector<bool>& visited) {
    visited[v] = true;
    cout << v << " ";
    for (int u : graph[v]) {
        if (!visited[u]) {
            DFS(u, graph, visited);
        }
    }
}
```

BFS

```
void BFS(int start, vector<vector<int>>& graph) {
    vector<bool> visited(graph.size(), false);
    queue<int> q;
    visited[start] = true;
    q.push(start);
    while (!q.empty()) {
        int v = q.front();
        q.pop();
        cout << v << " ";
        for (int u : graph[v]) {
            if (!visited[u]) {
                visited[u] = true;
                q.push(u);
            }
        }
    }
}
```

		Борисенко О.Ю			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

Маршрути DFS

Місто	Маршрут
Київ	Київ
Житомир	Київ → Житомир
Новоград-Волинський	Київ → Житомир → Новоград-Волинський
Рівне	Київ → Житомир → Новоград-Волинський → Рівне
Луцьк	Київ → Житомир → Новоград-Волинський → Рівне → Луцьк
Бердичів	Київ → Житомир → Бердичів
Вінниця	Київ → Житомир → Бердичів → Вінниця
Хмельницький	Київ → Житомир → Бердичів → Вінниця → Хмельницький
Тернопіль	Київ → Житомир → Бердичів → Вінниця → Хмельницький → Тернопіль
Шепетівка	Київ → Житомир → Шепетівка
Біла Церква	Київ → Біла Церква
Умань	Київ → Біла Церква → Умань
Черкаси	Київ → Біла Церква → Черкаси
Кременчук	Київ → Біла Церква → Черкаси → Кременчук
Полтава	Київ → Біла Церква → Полтава
Харків	Київ → Біла Церква → Полтава → Харків
Прилуки	Київ → Прилуки
Суми	Київ → Прилуки → Суми
Миргород	Київ → Прилуки → Миргород

Маршрути BFS

Місто	Маршрут
Київ	Київ
Житомир	Київ → Житомир
Біла Церква	Київ → Біла Церква
Прилуки	Київ → Прилуки
Новоград-Волинський	Київ → Житомир → Новоград-Волинський
Бердичів	Київ → Житомир → Бердичів
Шепетівка	Київ → Житомир → Шепетівка
Умань	Київ → Біла Церква → Умань
Черкаси	Київ → Біла Церква → Черкаси
Полтава	Київ → Біла Церква → Полтава
Суми	Київ → Прилуки → Суми
Миргород	Київ → Прилуки → Миргород
Рівне	Київ → Житомир → Новоград-Волинський → Рівне
Вінниця	Київ → Житомир → Бердичів → Вінниця
Кременчук	Київ → Біла Церква → Черкаси → Кременчук
Харків	Київ → Біла Церква → Полтава → Харків
Луцьк	Київ → Житомир → Новоград-Волинський → Рівне → Луцьк
Хмельницький	Київ → Житомир → Бердичів → Вінниця → Хмельницький
Тернопіль	Київ → Житомир → Бердичів → Вінниця → Хмельницький → Тернопіль

Висновок:

У ході виконання лабораторної роботи були реалізовані та протестовані два алгоритми обходу графа — пошук у глибину (DFS) та пошук у ширину (BFS). Побудовано граф маршрутів, створено його матрицю суміжності та виведено маршрути з заданої вершини.

Алгоритм DFS показав глибокий обхід до максимальної глибини перед поверненням назад, тоді як BFS проходив граф рівнями, що робить його ефективним для пошуку найкоротших шляхів у невагових графах.

Під час виконання було закріплено практичні навички роботи з графами, використання структур даних, а також засвоєно особливості реалізації алгоритмів обходу графів.

		Борисенко О.Ю			«Житомирська політехніка».24.122.3.000 – Лр1-8	Арк.
		Петросян Р.В.				32
Змн.	Арк.	№ докум.	Підпис	Дата		