

Лабораторна робота

Тема роботи: Робота із файлами

Мета роботи: Формування навиків та умінь при роботі із файлами засобами C++.

Для виконання роботи необхідно знати:

- правила опису файлів;
- правила доступу до файлових даних;
- ввід-вивід інформації з використанням файлових змінних;
- алгоритми роботи із файлами.

Теоретичні відомості

В C++ можлива робота як з текстовими, так і з двійковими (бінарними) файлами. Про роботу із двійковими файлами в рамках даного практикуму мова йти не буде. Текстовий файл – файл, в якому кожний символ зберігається у вигляді окремого байту. Текстовий файл розбивається на рядки спеціальним символом «кінець рядка \n» та завершується символом «кінець файлу **EOF**». **EOF** (end of file) - відємним числом, як правило -1, що гарантує неспівпадіння із кодом будь-якого іншого символу. Якщо в текстовому файлі знаходяться числові дані, то після останнього числа не можуть стояти ні пробіли, ні символ кінця рядка. Два числа в текстовому файлі вважаються окремими, якщо вони розділені між собою хоча-б одним із наступних символів: пробіл; символ табуляції; символ кінці рядка.

Операції з файлами в C++ можна здійснювати за допомогою файлових вказівників та потокових операцій.

Стиль мови програмування C (файлові вказівники).

Для роботи із файлом необхідно:

1. – описати вказівник на файлову змінну
FILE **<імя файлової змінної>*
2. - відкрити файл за наступним синтаксисом:
FILE * *fopen(filename, mode)*

де:

filename – ім'я файлу, що відкривається. Файл повинен знаходитися у папці, в якій міститься сама програма на C++, інакше необхідно задавати повне ім'я файлу;

mode – режим відкриття файлу (див. додаток №7).

При роботі із текстовими файлами в кінці специфікатора **mode** додається символ **t** – текстовий файл. Наприклад:

```
FILE *f;  
if ((f=fopen("dani.in", "rt"))==NULL)  
{  
printf("Не можливо відкрити файл"\n);  
return;  
}
```

Тут файлова змінна **f** зв'язується із текстовим файлом **dani.in**, який відкривається тільки для читання. В логічному операторі **if** виконується перевірка успішності відкриття файлу. Якщо файл відкрити не вдалося (**==NULL**), то на екран видається відповідне повідомлення і робота програми завершується. По завершенню роботи із файлом він повинен бути закритий функцією **fclose()**. Так, файл, відкритий у попередньому прикладі, буде закритий наступною командою:

```
fclose(f)
```

Інформацію із текстового файлу можна читати декількома способами:

1. Посимвольно за допомогою функції **putc()** за наступним синтаксисом (див приклад №1):

```
fgetc(<імя файлової змінної>)
```

2. По рядках за допомогою функції **fgets()** за наступним синтаксисом (див приклад №2.):

```
fgets(char s, int n, <імя файлової змінної>)
```

де:

char s – рядкова змінна, якій передається **n** зчитаних символів. Зчитується або **n-1** символ (n-тий символ – нульовий символ **\0**), або до появи кінця рядка (**\n**).

3. Читання текстового файлу за форматом здійснюється за допомогою функції **fscanf()** (див приклад №3.):

```
fscanf(<імя файлової змінної>, format,  
address)
```

де:

format – перелік форматів аргументів;

address char – перелік ідентифікаторів змінних, заданих своїми адресами.

При успішному читанні із файлу функція **fscanf()** повертає кількість прочитаних полів, тому організація перевірки правильності зчитування може виглядати наступним чином (при читанні двох даних із файлу **f**):

```
if (fscanf(f, "%i%d", &i, &x) != 2)  
{  
printf("Помилка читання із файлу!");  
};
```

Для перевірки умови досягнення кінця файлу служить функція **feof()**. Якщо зчитано символ кінця файлу (**EOF**), то **feof()** повертає ненульове значення.

Для запису даних у файл використовуються функції **fprintf()**, **fputc()** та **fputs()** (див приклад №4):

fprintf() аналогічна функції **printf()**, тільки першим аргументом у ній є посилання на файл, у який виконується запис;

fputc() та **fputs()** аналогічні функціям **putc()** та **puts()** відповідно, тільки першими аргументами у них є посилання на файл, у який виконується запис.

Стиль мови програмування C++ (потоків операцій).

В C++ робота із файлом здійснюється за допомогою функцій, що знаходяться у заготовочному файлі **<fstream.h>**:

Для запису даних у файл необхідно:

1. описати змінну типу **ofstream** за наступним синтаксисом:

ofstream <імя файлової змінної>

2. відкрити файл за допомогою функції **open**:

<імя файлової змінної>.open ("<імя файлу>", mode)

тут **mode** – задає режим відкриття файлу.

3. Записати інформацію у файл за наступним синтаксисом:

<імя файлової змінної> << <змінна>

тут **<змінна>** – ідентифікатор змінної, яка записується у файл.

4. Закрити файл командою **<імя файлової змінної>.close()**.

Для читання даних з файлу необхідно:

1. описати змінну типу **ifstream** за наступним синтаксисом:

ifstream <імя файлової змінної>

2. відкрити файл за допомогою функції **open**:

<імя файлової змінної>.open ("<імя файлу>", mode)

тут **mode** – задає режим відкриття файлу.

3. Зчитати інформацію з файл за наступним синтаксисом:

<імя файлової змінної> >> <змінна>

тут **<змінна>** – ідентифікатор змінної, якій присвоюється значення зчитане із файлу.

4. Закрити файл командою **<імя файлової змінної>.close()**.

Основні режими відкриття файлу **mode**:

ios::out – файл відкрито для запису даних (по замовчуванню для **ofstream**);

ios::in – файл відкрито для читання даних (по замовчуванню для **ifstream**);

ios::app – файл відкрито для запису даних в кінець файлу.

Для перевірки досягнення кінця файлу при вводі даних служить функція **eof()** із наступним синтаксисом:

<імя файлової змінної>.eof()

Якщо кінець файлу не досягнуто, то вона повертає **false**, а якщо досягнуто – **true**. Наприклад, цикл зчитування даних (до досягнення кінця файлу) та їх сумування матиме наступний вигляд:

```
int s=0;
while (!F.eof())
{
//Дані читаються із файлу F та присвоюються змінній a
F>>a;
S=s+a;
}
```

Інші додаткові можливості потокових операцій роботи із файлами розглянуто в прикладах №5 та №6).

Приклади виконання завдання

Приклад №1

Програма посимвольного зчитування даних із файлу до досягнення кінця файлу та виводу зчитаних даних на екран. Підраховується також кількість букв **a** у зчитаному тексті та виводиться відповідне повідомлення.

```
#include<stdio.h>
#include<string.h>
#include <windows.h>
void main()
{
```

```

SetConsoleOutputCP(1251);
SetConsoleCP(1251);
char ch;
int n=0;
FILE *f1;
/* Перевірка, чи вдалося успішно відкрити файл
Text1.in */
if ((f1=fopen("Text1.in", "rt"))==NULL) {
printf("Файл відкрити не вдалося\n");
return;
}
int i=0;
// Перевірка, чи вже досягнуто кінець файлу
while (!feof(f1))
{
i++;
// Читання символу із файлу
ch=fgetc(f1);
/*Якщо досягнуто кінець файлу, то символ кінця файлу
не виводиться */
if (!feof(f1))
{
printf("%c",ch);
if (ch=='a') n++;
}
};
// Закриття файлу
fclose(f1);
printf("\n");
printf("\nВ тексті зустрічається %i букв а\n",n);
printf("\n");
}

```

Результат роботи програми:

```

C:\WINDOWS.0\system32\cmd.exe
Програма посимвольного зчитування даних
із файлу до досягнення кінця файлу та
виводу зчитаних даних на екран.
Підраховується також кількість букв
а у зчитаному тексті та виводиться
відповідне повідомлення.

В тексті зустрічається 16  букв а

Для продовження натисніть будь-яку клавішу . . . _

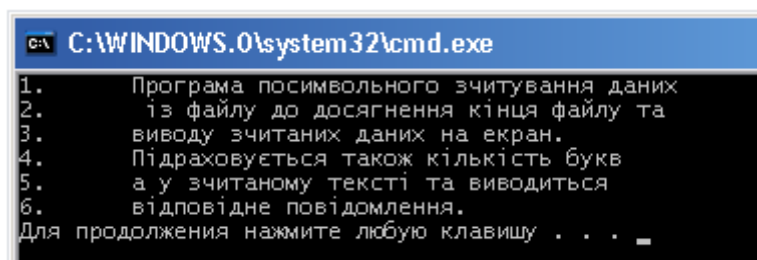
```

Приклад №2

Програма порядкового зчитування із файлу текстових даних до досягнення кінця файлу та їх виводу на екран (перед кожним зчитаним рядком також виводиться його номер):

```
#include<stdio.h>
#include<string.h>
#include <windows.h>
void main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    char ch[80];
    FILE *f1;
    /* Перевірка, чи вдалося успішно відкрити файл
    Text1.in */
    if ((f1=fopen("Text1.in", "rt"))==NULL)
    {
        printf("Файл відкрити не вдалося\n");
        return;
    }
    int i=0;
    do {
        i++;
        // Зчитування рядка довжиною 79 символів
        fgets(ch,80,f1);
        printf("%i.\t%s",i,ch);
        // Перевірка, чи вже досягнуто кінець файлу
    } while (!feof(f1));
    // Закриття файлу
    fclose(f1);
    printf("\n");
}
```

Результат роботи програми:



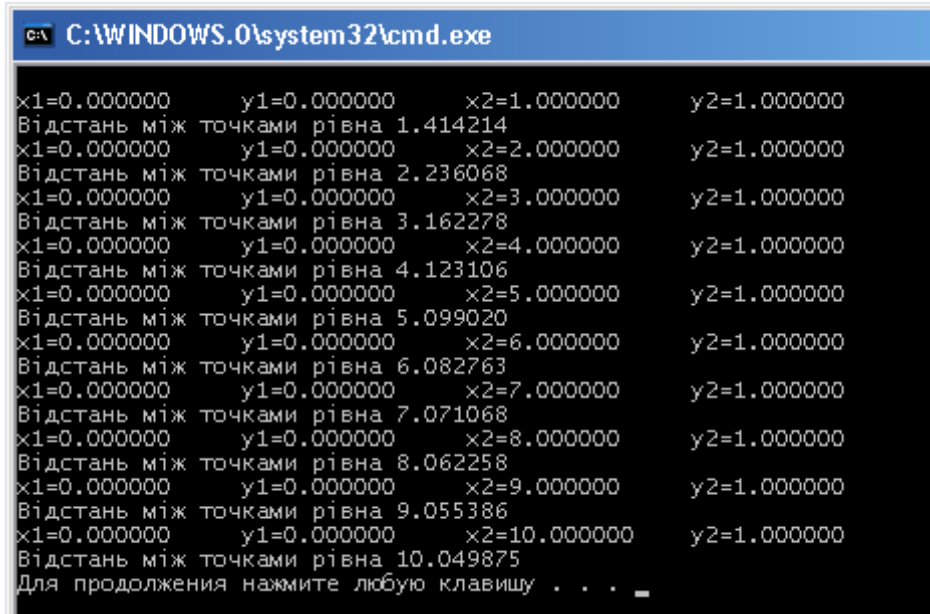
```
C:\WINDOWS.0\system32\cmd.exe
1.    Програма посимвольного зчитування даних
2.    із файлу до досягнення кінця файлу та
3.    виводу зчитаних даних на екран.
4.    Підраховується також кількість букв
5.    а у зчитаному тексті та виводиться
6.    відповідне повідомлення.
Для продовження натисніть будь-яку клавішу . . . _
```

Приклад №3

Програма зчитування із файлу 10 координат точок на площині, обчислення відстані між ними та виводу результату на екран:

```
#include<stdio.h>
#include <windows.h>
#include <math.h>
void main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    float x1,x2,y1,y2;
    float r;
    FILE *f1;
    /* Перевірка, чи вдалося успішно відкрити файл
    Text1.in */
    if ((f1=fopen("Text1.in", "rt"))==NULL)
    {
        printf("Файл відкрити не вдалося\n");
        return;
    }
    for(int i=0;i<10;i++)
    {
        /* Перевірка, чи вдалося успішно зчитати 4 дані із
        файлу */
        if (fscanf(f1,"%f%f%f%f",&x1,&y1,&x2,&y2)!=4)
        {
            printf("\nПомилка читання із файлу!");
            break;
        };
        printf("\nx1=%f\ty1=%f\tx2=%f\ty2=%f\t",x1,y1,x2,y2);
        r=sqrt(pow(x1-x2,2)+pow(y1-y2,2));
        printf("\nВідстань між точками рівна %f\t",r);
    };
    // Закриття файлу
    fclose(f1);
    printf("\n");
}
```

Результат роботи програми:



```
C:\WINDOWS.0\system32\cmd.exe
x1=0.000000      y1=0.000000      x2=1.000000      y2=1.000000
Відстань між точками рівна 1.414214
x1=0.000000      y1=0.000000      x2=2.000000      y2=1.000000
Відстань між точками рівна 2.236068
x1=0.000000      y1=0.000000      x2=3.000000      y2=1.000000
Відстань між точками рівна 3.162278
x1=0.000000      y1=0.000000      x2=4.000000      y2=1.000000
Відстань між точками рівна 4.123106
x1=0.000000      y1=0.000000      x2=5.000000      y2=1.000000
Відстань між точками рівна 5.099020
x1=0.000000      y1=0.000000      x2=6.000000      y2=1.000000
Відстань між точками рівна 6.082763
x1=0.000000      y1=0.000000      x2=7.000000      y2=1.000000
Відстань між точками рівна 7.071068
x1=0.000000      y1=0.000000      x2=8.000000      y2=1.000000
Відстань між точками рівна 8.062258
x1=0.000000      y1=0.000000      x2=9.000000      y2=1.000000
Відстань між точками рівна 9.055386
x1=0.000000      y1=0.000000      x2=10.000000     y2=1.000000
Відстань між точками рівна 10.049875
Для продовження натисніть будь-яку клавішу . . . _
```

Приклад №4

Програма обчислення степені двійки від 2^0 до 2^9 та виводу результату у файл *Text1.out*

```
#include<stdio.h>
#include <windows.h>
#include <math.h>
void main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    FILE *f1;
    /* Перевірка, чи вдалося успішно відкрити файл
    Text1.out*/
    if ((f1=fopen("Text1.out", "wt"))==NULL)
    {
        printf("Файл відкрити не вдалося\n");
        return;
    }
    for(int i=0;i<10;i++)
    {
        // Вивід даних у файл Text1.out
        fprintf(f1,"\n    2 в степені %i =
        %4.0f",i,pow(2.,i));
    }
```

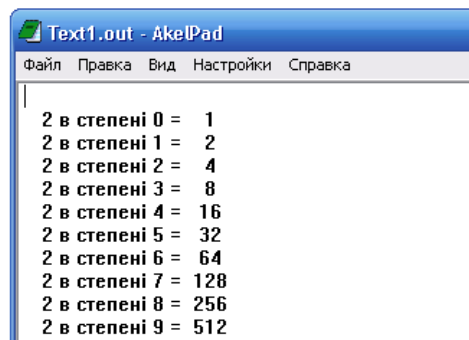


```

        printf("\n");
// Закриття файлу Text1.out
        fclose(f1);
    }

```

В результаті роботи програми буде створено текстовий файл **Text1.out**:



Приклад №5

Програма обчислення степенів двійки та трійки та виводу результатів у файли **Stepin** та **Stepin1** відповідно (з використанням потокових операцій):

```

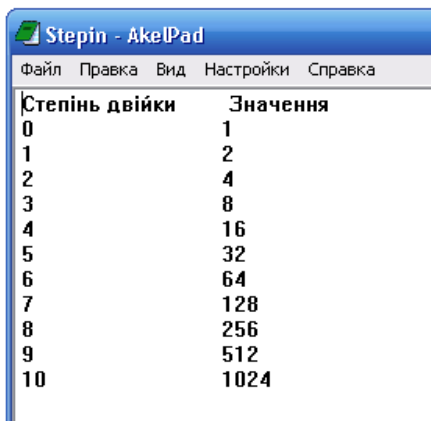
#include<iostream>
#include<fstream>
#include <windows.h>
#include<stdlib.h>
#include <math.h>
using namespace std;
void main ()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    ofstream out;
    out.open ("Stepin",ios::out );
// Перевірка, чи вдалося успішно відкрити файл Stepin
    if ( ! out )
    {
        cout<< " Неможливо відкрити  файл \n ";
        exit (1);
    }
    ofstream out1;

```

```
// Відкриття файлу Stepin1
    out1.open("Stepin1");
// Перевірка, чи вдалося успішно відкрити файл
    if ( ! out1 )
    {
        cout<< " Неможливо відкрити файл \n ";
        exit (2);
    }
    out<<"Степінь двійки\t "<<"Значення\n";
    for(int i=0;i<11;i++)
        out<<i<<"\t\t"<<pow(2.,i)<<"\n";
    out.close();
    out1<<"Степінь трійки\t "<<"Значення\n";
    for(int i=0;i<11;i++)
        out1<<i<<"\t\t"<<pow(3.,i)<<"\n";
    out1.close();
}
```

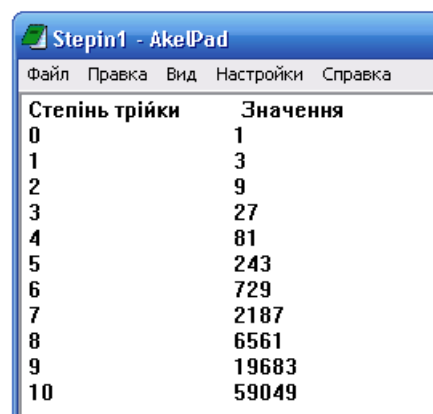
В результаті роботи програми буде створено два текстові файли:

Stepin



Степінь двійки	Значення
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024

Stepin1



Степінь трійки	Значення
0	1
1	3
2	9
3	27
4	81
5	243
6	729
7	2187
8	6561
9	19683
10	59049

Приклад №6

Програма із одного файлу зчитує інформацію про початкову точку, крок та кількість точок табуляції, а в другий виводить результат табулювання функції $f(x) = \sin(x) + \cos(x)$.

```
#include<iostream>
#include<fstream>
#include<stdlib.h>
#include <math.h>
```

```

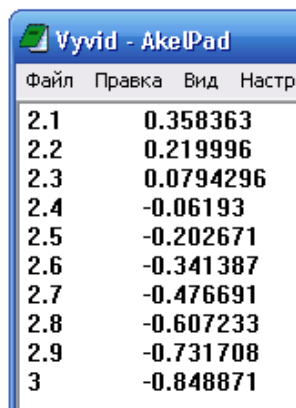
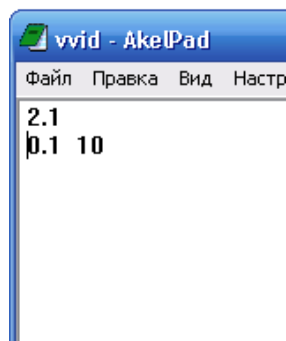
#include <windows.h>
using namespace std;
void main ()
{
    float x0;
    float h;
    int n;
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    ifstream in( "vvid" );
    ofstream out( "Vyvid" );
    // Перевірка, чи вдалося успішно відкрити файл Vyvid
    if ( ! out )
    {
        cout<< " Неможливо відкрити файл \n ";
        exit (1);
    }
    // Перевірка, чи вдалося успішно відкрити файл vvid
    if ( ! in ) {
        cout<< " Неможливо відкрити файл \n ";
        exit (1);
    }
    in>>x0;
    in>>h>>n;
    for(int i=0;i<n;i++,x0=x0+h)
    // Вивід даних у файл Vyvid
    out<<x0<<"\t"<<sin(x0)+cos(x0)<<"\n";
    out.close();
    in.close();
}

```

Результат роботи програми:

Вхідний файл **vvid**

Результат роботи програми – файл **Vyvid**



Завдання для виконання

Написати програму розв'язку задач двома способами: із використанням файлових вказівників та використовуючи потокові операції. Перевірити правильність роботи програми за допомогою контрольних прикладів. Результат вивести з використанням пояснювальної текстової інформації.

1. Файл містить 100 цілих чисел. Підрахувати, скільки серед них є трійок та скільки чисел більші десяти.
2. Файл містить 50 дійсних чисел. Підрахувати, скільки серед них є від'ємних чисел (вивести на екран). Вивести у інший файл номери позицій, в яких знаходяться від'ємні числа та їх значення.
3. Написати програму, що зчитує із файлу результати деякого експерименту, обчислює середнє значення, абсолютну та відносну похибки вимірювання і результат дописує у даний файл.
4. Написати програму, яка із даного файлі із дійсними числами формує два нові. Один із них містить всі від'ємні числа та їх кількість, другий - всі додатні та їх кількість.
5. Написати програму, яка із даного файлу із цілими числами формує два нові. Один із них містить позиції всіх чисел, більших 3, другий – позиції всіх нульових елементів.
6. Написати програму, яка обчислює середнє арифметичне чисел, що записані у файлі. Якщо середнє арифметичне більше нуля, то отриманий результат записується у файл plus.rez, а якщо менше нуля – minus.rez.
7. Написати програму, яка із даного файлу дійсних чисел видаляє всі, які більші деякого числа A , але менші B та підраховує кількість таких елементів.
8. Написати програму, що створює копію файлу, але записує числові дані задом наперед (першим стоїть останній елемент і т. д.).
9. Написати програму, що створює копію файлу, який містить дійсні числа, але спочатку ідуть додатні елементи, потім нульові, а в кінці від'ємні.
10. Написати програму, що створює копію файлу із даними цілого типу, вирізавши із нього всі елементи, рівні нулю.
11. Написати програму, яка сортує дійсні числа, зчитані із файлу по зростанню та виводить їх в інший файл.

12. Дано масив 100 цілих чисел. Перевірити, чи впорядкований даний масив по зростанню (спаданню, неупорядкований) та вивести відповідне повідомлення на екран та у файл.
13. Написати програму, яка із даного файлу, що містить дійсні числа формує новий, в якому сусідні елементи поміняно місцями.
14. Файл s.dani містить 25 цілих чисел, записаних у рядок. Вивести ці числа у новий файл, де вони будуть записані у стовпчик.
15. Створити файл, який містить табличку множення чисел від одного до 9, сформовану у вигляді матриці.
16. У файлі ds.dat записано матрицю цілих чисел розміром 5×5 елементів. Транспонувати дану матрицю та вивести її у новий файл.
17. Написати програму табулювання функції $sh(x) = \frac{e^x - e^{-x}}{2}$ із виводом результату у файл. Всі дані для табуляції вводяться із клавіатури (задати самостійно).
18. Сформувати файл, який містить таблицю значень функції $n!$ з використанням форматного виводу (значення n та відповідне йому значення $n!$). Значення n читається із попередньо підготовленого файлу.
19. У файл gjad.rez виводити у вигляді таблички (значення i – відповідне йому значення члену ряду) результат обчислення членів ряду $\sum_{i=1}^{\infty} i^{-3}$ до тих пір, поки член ряду не стане меншим 0.00001. У інший файл вивести кількість членів ряду, яка була обчислена із пояснювальною текстовою інформацією.
20. У одному файлі записано інформацію про координати 10 точок у просторі, а в іншому координати центру кулі та її радіус. У новий файл вивести інформацію про координати точок, що лежать в середині кулі. На екран вивести кількість таких точок
21. Із клавіатури вводиться цілі числа. Числа, кратні 3, записуються у один файл, а кратні 7 – у інший. Якщо сумарна кількість записаних у файл чисел стане рівною 20 – завершити роботу програми.
22. Файл містить дані про результати 25 експериментальних вимірювань. Переписати їх у новий файл, відкинувши найбільший та найменший результат.
23. Файл містить результати 50 вимірювань. Вивести у новий файл найбільший та найменший результат та їх порядкові номери.

24. Катети прямокутного трикутника вводяться із клавіатури. У файл вивести гіпотенузу, площу трикутника та його кути із пояснювальною текстовою інформацією. Передбачити можливість проведення обчислень для кількох різних значень (задається із клавіатури) довжин катетів.
25. Файл містить 100 цілих чисел. Дописати у вихідний файл загальну суму чисел. Знайти суми цифр цих чисел, а результати записати у інший файл.
26. У файлі записана матриця $B_{7 \times 7}$. Обчислити добуток ненульових елементів та дописати у вихідний файл. У новий файл вивести транспоновану матрицю.
27. У двох файлах записано матриці $A_{7 \times 7}$ та $C_{7 \times 7}$. Обчислити суму цих матриць та записати сумарну матрицю у новий файл та вивести її на екран.
28. У файлі записано інформацію про координати вершин 20 прямокутників на площині. У новий файл вивести координати найбільшого та найменшого за площею прямокутників та їх порядкові номери у вихідному файлі.
29. У файлі записано інформацію про 10 варіантів довжин 3 відрізків. У новий файл вивести інформацію про те, чи можна із даних відрізків побудувати трикутник, чи ні.
30. У файлі записано інформацію про координати 100 точок на площині. У новий файл вивести інформацію про відстань від точки до початку координат та те, в якому квадранті системи координат вони знаходяться.