* **3** Napisz program, który wczyta opis skierowanego grafu nieważonego z pliku tekstowego otrzymanego od nauczyciela (np. *graf_2.txt*) oraz numer wierzchołka z klawiatury. Liczby w pierwszym wierszu pliku odpowiadają liczbie wierzchołków i liczbie krawędzi grafu. Każdy kolejny wiersz zawiera numery wierzchołków będących początkiem i końcem krawędzi. Program ma sprawdzić, czy istnieje cykl zawierający ten wierzchołek.
Uwaga: Z cyklem w grafie mamy do czynienia, gdy istnieje ścieżka przejścia, w której wierzchołek początkowy i końcowy jest ten sam. Na przykład w grafie na rysunku 4.3 ze s. 63 jest cykl 0→1→2→0.

** **4** Napisz program, który wczyta opis skierowanego grafu nieważonego z pliku tekstowego otrzymanego od nauczyciela (np. *graf_2.txt*) i sprawdzi, czy w grafie istnieje cykl. Liczby w pierwszym wierszu pliku odpowiadają liczbie wierzchołków i liczbie krawędzi grafu. Każdy kolejny wiersz pliku zawiera numery wierzchołków będących początkiem i końcem krawędzi.
Uwaga: Z cyklem w grafie mamy do czynienia, gdy istnieje ścieżka przejścia, w której wierzchołek początkowy i końcowy jest ten sam. Na przykład w grafie na rysunku 4.3 ze s. 63 jest cykl 0→1→2→0.

** **5** Napisz program, który wczyta opis skierowanego grafu nieważonego z pliku tekstowego otrzymanego od nauczyciela (np. *graf_2.txt*) i sprawdzi, czy graf jest spójny. Liczby w pierwszym wierszu pliku określają liczbę wierzchołków i liczbę krawędzi grafu, każdy kolejny wiersz zawiera numery wierzchołków będących początkiem i końcem krawędzi.

** **6** Napisz program, który wczyta opis skierowanego grafu nieważonego z pliku tekstowego otrzymanego od nauczyciela (np. *graf_2.txt*) i przejrzy wierzchołki grafu w głąb bez użycia rekurencji, z wykorzystaniem stosu. Liczby w pierwszym wierszu pliku odpowiadają liczbie wierzchołków i liczbie krawędzi grafu. Każdy kolejny wiersz pliku zawiera numery wierzchołków będących początkiem i końcem krawędzi.

*** **7** Napisz program, który wczyta opis skierowanego grafu ważonego z pliku tekstowego otrzymanego od nauczyciela (np. *graf_3.txt*) oraz numery wierzchołków początkowego i końcowego z klawiatury. W pierwszym wierszu pliku znajdują się dwie liczby: pierwsza odpowiada liczbie wierzchołków grafu, a druga liczbie krawędzi. Każdy następny wiersz zawiera cztery liczby odpowiadające kolejno: numerowi wierzchołka będącego początkiem krawędzi, numerowi wierzchołka będącego końcem krawędzi, odległości między wierzchołkami, czasowi przejścia między wierzchołkami. Program ma znaleźć najkrótszą oraz najszybszą drogę pomiędzy wczytanymi wierzchołkami.

*** **8** Napisz program, który wczyta opis skierowanego grafu ważonego z pliku tekstowego otrzymanego od nauczyciela (np. *graf_4.txt*) oraz numery wierzchołków początkowego i końcowego z klawiatury. Liczby w pierwszym wierszu pliku odpowiadają liczbie wierzchołków i liczbie krawędzi grafu. Każdy kolejny wiersz pliku zawiera: numer wierzchołka będącego początkiem krawędzi, numer wierzchołka będącego końcem krawędzi, wagę krawędzi. Program ma znaleźć najkrótszą drogę pomiędzy wczytanymi wierzchołkami w czasie $O(n \cdot \log n)$.
Wskazówka: Wykorzystaj szablon typu `priority_queue` z biblioteki STL. Typ ten reprezentuje kolejkę priorytetową.

Sposób na zadania

Zadanie 1

W pliku, który otrzymasz od nauczyciela (np. *WUZ1_zad1_dane_liczby.txt*), w kolejnych 10 wierszach jest zapisanych po 20 liczb całkowitych z zakresu od 0 do 100 000 oddzielonych spacjami. Wśród tych 200 liczb są liczby zakończone każdą z cyfr od 0 do 9. Oto zawartość dwóch początkowych wierszy pliku:

```
820 21292 12637 322 21013 17178 16445 14757 9026 5161 25709 17961 14093 14598
12379 8458 8103 29640 426 2723
820 21292 12637 322 6736 6851 12166 8597 8781 9130 15482 25861 30047 15757 29632
30114 20605 29853 10139 14390
```

Napisz program lub programy rozwiązujące poniższe zadania. Odpowiedzi zapisz w sposób wskazany przez nauczyciela.

Zadanie 1.1 (0–3)

Utwórz plik wynikowy (np. *WUZ1_zad1_1_dane_liczby_wynik.txt*), w którym będzie 10 wierszy składających się z liczb oddzielonych spacjami. W poszczególnych wierszach znajdą się liczby z pliku z danymi, przy czym w pierwszym wierszu będą liczby zakończone cyfrą 0, w drugim – cyfrą 1 itd. Liczby w danym wierszu pliku wynikowego powinny wystąpić w tej samej kolejności co w pliku z danymi. Na przykład dla dwóch początkowych wierszy pliku z danymi wynik będzie następujący:

```
820 29640 820 9130 14390
5161 17961 6851 8781 25861
21292 322 21292 322 15482 29632
21013 14093 8103 2723 29853
30114
16445 20605
9026 426 6736 12166
12637 14757 12637 8597 30047 15757
17178 14598 8458
25709 12379 10139
```

Rozwiązanie

Tagi: tablica, dynamiczna struktura danych, kolejka

Sposób I. Wielokrotny odczyt danych z pliku bez użycia złożonych struktur danych

Krok 1

Liczby z pliku z danymi należy podzielić na 10 grup. O przynależności do grupy decyduje ostatnia cyfra liczby, czyli reszta z dzielenia liczby przez 10. Przejrzymy cały plik z danymi 10 razy. Za każdym razem zapiszemy do pliku wynikowego w oddzielnym wierszu liczby należące do jednej grupy.

Krok 2

Liczby z pliku z danymi odczytamy za pomocą zmiennej plikowej `we` typu `ifstream`. Wyniki zapiszemy z wykorzystaniem zmiennej plikowej `wy` typu `ofstream`.

Krok 3

Dla każdej grupy liczb będziemy otwierać i zamykać plik z danymi – otworzymy go za pomocą metody `open`, zamkniemy przy użyciu metody `close`. Po wypisaniu liczby dodamy spację, a po wypisaniu grupy liczb – znacznik końca wiersza `endl`. Na koniec zamkniemy plik wynikowy.

Odpowiedź: Kod źródłowy programu rozwiązującego to zadanie może być następujący:

```
1.  #include <fstream>
2.
3.  using namespace std;
4.
5.  int main()
6.  {
7.      int x;
8.      ifstream we;
9.      ofstream wy("WUZ1_zad1_1_dane_liczby_wynik.txt");
10.     for (int i=0;i<10;i++)
11.     {
12.         we.open("WUZ1_zad1_dane_liczby.txt");
13.         for (int j=0;j<200;j++)
14.         {
15.             we>>x;
16.             if (x%10==i)
17.                 wy<<x<<" ";
18.         }
19.         wy<<endl;
20.         we.close();
21.     }
22.     wy.close();
23.     return 0;
24. }
```

Sposób II. Z wykorzystaniem tablicy, której elementami są liczby całkowite

Krok 1

Po otwarciu pliku z danymi odczytamy kolejne liczby i zapiszemy je w tablicy. Dzięki temu plik wejściowy przejrzymy tylko raz. Do przechowania liczb możemy wykorzystać np. tablicę dwuwymiarową o 10 wierszach i 20 kolumnach lub jednowymiarową o 200 elementach. Użyjemy tablicy jednowymiarowej.

Krok 2

Liczby zapisane w tablicy należy podzielić na 10 grup. O przynależności do grupy decyduje ostatnia cyfra liczby, czyli reszta z dzielenia liczby przez 10. Przejrzymy tablicę 10 razy, za każdym razem zapiszemy do pliku wynikowego liczby należące do jednej grupy.

Krok 3

Podczas wypisywania liczb z grupy po każdej liczbie dodamy spację, a po wypisaniu wszystkich liczb z grupy – znacznik końca wiersza `endl`.

Odpowiedź: Kod źródłowy programu rozwiązującego to zadanie może być następujący:

```
1.  #include <fstream>
2.
3.  using namespace std;
4.
5.  int main()
6.  {
7.      int Tab[200];
8.      ifstream we("WUZ1_zad1_dane_liczby.txt");
9.      for (int i=0;i<200;i++)
10.         we>>Tab[i];
11.     we.close();
12.     ofstream wy("WUZ1_zad1_1_dane_liczby_wynik.txt");
13.     for (int i=0;i<10;i++)
14.     {
15.         for (int j=0;j<200;j++)
16.             if (Tab[j]%10==i)
17.                 wy<<Tab[j]<<" ";
18.         wy<<endl;
19.     }
20.     wy.close();
21.     return 0;
22. }
```

Sposób III. Z wykorzystaniem tablicy, której elementami są kolejki złożone z liczb całkowitych

Krok 1

Liczby z pliku z danymi należy podzielić na 10 grup. O przynależności do grupy decyduje ostatnia cyfra liczby, czyli reszta z dzielenia liczby przez 10. Ponieważ liczby w grupie mają zachować kolejność występowania z pliku wejściowego, do ich pamiętania możemy wykorzystać strukturę danych o nazwie kolejka. Utworzymy 10-elementową tablicę, której poszczególne elementy to kolejki przechowujące liczby z poszczególnych grup (zaczynając od grupy liczb zakończonych cyfrą 0, a kończąc na grupie liczb zakończonych cyfrą 9).

Krok 2

Plik z danymi przejrzymy tylko raz. Podczas odczytywania danych z pliku każdą liczbę dołączymy do właściwej kolejki. Liczbę zakończoną cyfrą 0 dołączymy do kolejki pamiętanej w tablicy pod indeksem 0, liczbę zakończoną cyfrą 1 – do kolejki pamiętanej pod indeksem 1 itd.

Krok 3

Przejrzymy wszystkie kolejki, zaczynając od kolejki pamiętającej liczby zakończone cyfrą 0, a kończąc na kolejce pamiętającej liczby zakończone cyfrą 9. Będziemy usuwać liczby z kolejek i zapisywać je w pliku wynikowym. Po wypisaniu każdej liczby dodamy spację, a po przejrzeniu wszystkich elementów z danej kolejki – znacznik końca wiersza `endl`.

Odpowiedź: Kod źródłowy programu rozwiązującego to zadanie może być następujący:

```
#include <fstream>
#include <queue>

using namespace std;

int main()
{
    int x;
    queue<int> Tab[10];
    ifstream we("WUZ1_zad1_dane_liczby.txt");
    for (int i=0;i<200;i++)
    {
        we>>x;
        Tab[x%10].push(x);
    }
    we.close();
    ofstream wy("WUZ1_zad1_1_dane_liczby_wynik.txt");
    for (int i=0;i<10;i++)
    {
        while (!Tab[i].empty())
        {
            wy<<Tab[i].front()<<" ";
            Tab[i].pop();
        }
        wy<<endl;
    }
    wy.close();
    return 0;
}
```

Zadanie 1.2 (0–3)

Utwórz plik wynikowy (np. *WUZ1_zad1_2_dane_liczby_wynik.txt*), w którym będzie 10 wierszy składających się z liczb oddzielonych spacjami. W poszczególnych wierszach znajdą się liczby z pliku z danymi, przy czym w pierwszym wierszu będą liczby zakończone cyfrą 0, w drugim – cyfrą 1 itd. Liczby w danym wierszu pliku wynikowego powinny wystąpić w odwrotnej kolejności niż w pliku z danymi. Na przykład dla dwóch początkowych wierszy pliku z danymi wynik będzie następujący:

```
14390 9130 820 29649 820
25861 8781 6851 17961 5161
29632 15482 322 21292 322 21292
29853 2723 8103 14093 21013
30114
20605 16445
12166 6736 426 9026
15757 30047 8597 12637 14757 12637
8458 14598 17178
10139 12379 25709
```


Rozwiązanie

Tagi: tablica, dynamiczna struktura danych, stos

Sposób I. Z wykorzystaniem tablicy, której elementami są liczby całkowite

Krok 1

Aby plik z danymi przejrzeć tylko raz, po jego otwarciu odczytamy kolejne liczby i zapiszemy je w tablicy. Do przechowania liczb możemy wykorzystać np. tablicę dwuwymiarową o 10 wierszach i 20 kolumnach lub jednowymiarową o 200 elementach. Użyjemy tablicy jednowymiarowej.

Krok 2

Liczby pamiętane w tablicy należy podzielić na 10 grup. O przynależności do grupy decyduje ostatnia cyfra liczby, czyli reszta z dzielenia liczby przez 10. Liczby w pliku wynikowym w danej grupie mają wystąpić w odwrotnej kolejności niż w pliku z danymi, dlatego przejrzymy tablicę 10 razy od ostatniego elementu do pierwszego i za każdym razem zapiszemy do pliku wynikowego liczby należące do jednej grupy.

Krok 3

Po wypisaniu każdej liczby dodamy spację, a po wypisaniu wszystkich liczb z danej grupy – znacznik końca wiersza `endl`.

Odpowiedź: Kod źródłowy programu rozwiązującego to zadanie może być następujący:

```
#include <fstream>

using namespace std;

int main()
{
    int Tab[200];
    ifstream we("WUZ1_zad1_dane_liczby.txt");
    for (int i=0;i<200;i++)
        we>>Tab[i];
    we.close();
    ofstream wy("WUZ1_zad1_2_dane_liczby_wynik.txt");
    for (int i=0;i<10;i++)
    {
        for (int j=199;j>=0;j--)
            if (Tab[j]%10==i)
                wy<<Tab[j]<<" ";
        wy<<endl;
    }
    wy.close();
    return 0;
}
```

Sposób II. Z wykorzystaniem tablicy, której każdy element jest stosem przechowującym liczby całkowite

Krok 1

Liczby z pliku z danymi należy podzielić na 10 grup. O przynależności do grupy decyduje ostatnia cyfra liczby, czyli reszta z dzielenia liczby przez 10. Ponieważ liczby w danej grupie mają wystąpić w odwrotnej kolejności niż w pliku wejściowym, do pamiętania liczb z grupy wykorzystamy strukturę danych o nazwie stos. Utworzymy 10-elementową tablicę, w której każdy z elementów będzie stosem pamiętającym liczby z jednej grupy (zaczynając od grupy liczb zakończonych cyfrą 0, a kończąc na grupie liczb zakończonych cyfrą 9).

Krok 2

Podczas odczytywania danych z pliku wejściowego każdą liczbę włożymy na właściwy stos. Dzięki temu nie będziemy musieli wielokrotnie przeglądać danych.

Krok 3

Przejrzymy wszystkie stosy, zaczynając od stosu pamiętającego liczby zakończone cyfrą 0, a kończąc na stosie pamietającym liczby zakończone cyfrą 9. Będziemy zdejmować liczby ze stosów i zapisywać je w pliku wynikowym. Po wypisaniu każdej liczby dodamy spację, a po zdjęciu wszystkich liczb z danego stosu – znacznik końca wiersza **endl**.

Odpowiedź: Kod źródłowy programu rozwiązującego to zadanie może być następujący:

```
#include <fstream>
#include <stack>

using namespace std;

int main()
{
    int x;
    stack<int> Tab[10];
    ifstream we("WUZ1_zad1_dane_liczby.txt");
    for (int i=0;i<200;i++)
    {
        we>>x; Tab[x%10].push(x);
    }
    we.close();
    ofstream wy("WUZ1_zad1_2_dane_liczby_wynik.txt");
    for (int i=0;i<10;i++)
    {
        while (!Tab[i].empty())
        {
            wy<<Tab[i].top()<<" ";
            Tab[i].pop();
        }
        wy<<endl;
    }
    wy.close();
    return 0;
}
```


Zadanie 1.3 (0–4)

Każdy wiersz pliku z danymi to ciąg liczbowy złożony z 20 wyrazów. Posortuj te ciągi leksykograficznie i zapisz uporządkowane ciągi w pliku wynikowym (np. *WUZ1_zad1_3_dane_liczby_wynik.txt*), każdy w oddzielnym wierszu. Z dwóch ciągów wcześniej powinien być zapisany ten, którego pierwszy wyraz różniący ciągi (patrząc od lewej) jest mniejszą liczbą. Na przykład z dwóch początkowych ciągów liczbowych z pliku z danymi wcześniejszy będzie ciąg drugi. Pierwszy wyraz różniący ciągi występuje na piątej pozycji, w pierwszym ciągu jest równy 21013, a w drugim 6736.

Rozwiązanie

Tagi: tablica, algorytmy sortowania, sortowanie leksykograficzne

Sposób I. Z wykorzystaniem pomocniczej tablicy pamiętającej kolejność ciągów

Krok 1

Po otwarciu pliku z danymi odczytamy kolejne liczby i zapiszemy je w tablicy dwuwymiarowej o 10 wierszach i 20 kolumnach. W każdym wierszu tablicy zapamiętamy jeden ciąg liczbowy. Dodatkowo zadeklarujemy tablicę jednowymiarową o 10 elementach, która będzie pamiętać kolejność ciągów. Wartości początkowe tej tablicy to kolejne liczby całkowite nieujemne, a początkowa kolejność ciągów jest zgodna z kolejnością ciągów w pliku z danymi.

Krok 2

Zdefiniujemy funkcję porównującą dwa ciągi. Jej parametrami będą dwie tablice jednowymiarowe o 20 elementach. Funkcja zwróci wartość **true**, gdy pierwszy ciąg powinien wystąpić wcześniej lub gdy oba ciągi są takie same. W przeciwnym przypadku funkcja zwróci wartość **false**.

Krok 3

Będziemy sortować elementy z pomocniczej tablicy pamiętającej kolejność ciągów. Można wykorzystać dowolną metodę sortowania. My zastosujemy sortowanie przez wybieranie. O ewentualnej zamianie wartości elementów tablicy pomocniczej decyduje wynik porównania dwóch ciągów pamiętanych w tablicy dwuwymiarowej. Numery wierszy, w których są pamiętane porównywane ciągi, są określone przez elementy tablicy pamiętającej kolejność ciągów.

Krok 4

Zapisujemy do pliku wynikowego ciągi we właściwej kolejności. Przeglądamy tablicę pamiętającą kolejność ciągów i wypisujemy elementy z odpowiednich wierszy tablicy dwuwymiarowej – numer wiersza tablicy dwuwymiarowej odpowiada elementowi z tablicy pamiętającej kolejność ciągów.

Odpowiedź: Kod źródłowy programu rozwiązującego to zadanie może być następujący:

```
#include <fstream>
using namespace std;

const int N=10, M=20;

bool Wczesniejszy(int Tab1[], int Tab2[])
{
    int i=0;
    while (i<M && Tab1[i]==Tab2[i]) i++;
    if (i<M) return (Tab1[i]<Tab2[i]);
    else return true;
}
```

```
13.
14. int main()
15. {
16.     int Tab[N][M], W[N];
17.     for (int i=0;i<N;i++)
18.         W[i]=i;
19.     ifstream we("WUZ1_zad1_dane_liczby.txt");
20.     for (int i=0;i<N;i++)
21.         for (int j=0;j<M;j++)
22.             we>>Tab[i][j];
23.     we.close();
24.     for (int i=0;i<N-1;i++)
25.     {
26.         int k=i;
27.         for (int j=i+1;j<N;j++)
28.             if (Wczesniejszy(Tab[W[j]],Tab[W[k]]))
29.                 k=j;
30.         swap(W[i],W[k]);
31.     }
32.     ofstream wy("WUZ1_zad1_3_dane_liczby_wynik.txt");
33.     for (int i=0;i<N;i++)
34.     {
35.         for (int j=0;j<M;j++)
36.             wy<<Tab[W[i]][j]<<" ";
37.         wy<<endl;
38.     }
39.     wy.close();
40.     return 0;
41. }
```

Sposób II. Z wykorzystaniem funkcji **sort** z biblioteki STL

Krok 1

Do pamiętania ciągów wykorzystamy typ **vector** z biblioteki STL. Utworzymy zmienną typu **vector** o 10 elementach. Każdy element tej zmiennej również będzie typu **vector** i będzie reprezentować jeden ciąg liczbowy. Odczytamy dane z pliku i zapamiętamy je w tak utworzonej strukturze danych.

Krok 2

Zdefiniujemy pomocniczą funkcję porównującą dwa ciągi. Jej parametry będą typu **vector** – każdy z nich będzie odpowiadał jednemu ciągowi liczbowemu. Wartością funkcji będzie wartość logiczna **true**, gdy pierwszy ciąg powinien wystąpić wcześniej lub gdy oba ciągi są takie same. W przeciwnym przypadku funkcja zwróci wartość **false**.

Krok 3

Do uporządkowania ciągów we właściwej kolejności użyjemy funkcji **sort** z biblioteki STL. Jej parametrami będą wartości metod **begin** i **end** struktury danych pamiętającej ciągi oraz nazwa funkcji porównującej dwa ciągi.

Krok 4

Przeglądamy wartości zmiennej typu **vector** pamiętającej posortowane leksykograficznie ciągi i wypisujemy je do pliku wynikowego.

Odpowiedź: Kod źródłowy programu rozwiązującego to zadanie może być następujący:

```
1.  #include <fstream>
2.  #include <vector>
3.  #include <algorithm>
4.
5.  using namespace std;
6.
7.  const int N=10, M=20;
8.
9.  bool Wczesniejszy(vector<int> Tab1, vector<int> Tab2)
10. {
11.     int i=0;
12.     while (i<M && Tab1[i]==Tab2[i])
13.         i++;
14.     if (i<M)
15.         return (Tab1[i]<Tab2[i]);
16.     else return true;
17. }
18.
19. int main()
20. {
21.     vector<vector<int> > Tab;
22.     Tab.resize(N);
23.     for (int i=0;i<N;i++)
24.         Tab[i].resize(M);
25.     ifstream we("WUZ1_zad1_dane_liczby.txt");
26.     for (int i=0;i<N;i++)
27.         for (int j=0;j<M;j++)
28.             we>>Tab[i][j];
29.     we.close();
30.     sort(Tab.begin(),Tab.end(),Wczesniejszy);
31.     ofstream wy("WUZ1_zad1_3_dane_liczby_wynik.txt");
32.     for (int i=0;i<N;i++)
33.     {
34.         for (int j=0;j<M;j++)
35.             wy<<Tab[i][j]<<" ";
36.         wy<<endl;
37.     }
38.     wy.close();
39.     return 0;
40. }
```

Trening

Zadanie 1

W pliku, który otrzymasz od nauczyciela (np. *WUZ1_zad1_ciagi.txt*), zapisanych jest 100 liczb całkowitych z zakresu od 0 do 100 000, każda w oddzielnym wierszu. Ostatnia liczba w pliku jest równa 0. W pliku nie występują bezpośrednio jedna po drugiej dwie liczby 0. Wyrazy z pierwszych wierszy pliku aż do liczby 0 tworzą ciąg liczbowy – liczba 0 nie należy do tego ciągu, pełni jedynie funkcję znacznika końca ciągu. Kolejne ciągi składają się z liczb znajdujących się między liczbami 0. Pierwszych 14 wierszy z pliku z danymi wygląda następująco:

```
29660
3662
6457
2340
26892
21832
23534
0
3153
5304
14659
10064
19277
0
```

Napisz program lub programy rozwiązujące poniższe zadania. Odpowiedzi zapisz w sposób wskazany przez nauczyciela.

Zadanie 1.1 (0–2) ROZWIĄŻ NA KOMPUTERZE

Policz liczbę ciągów występujących w pliku z danymi oraz długość najkrótszego i najdłuższego ciągu. Otrzymane wyniki zapisz w pliku tekstowym (np. *WUZ1_zad1_1_ciagi_wynik.txt*). Na przykład dla pierwszych 14 liczb z pliku z danymi wyniki są następujące:

```
Liczba ciagow: 2, dlugosc najkrotszego ciagu: 5, dlugosc najdluzszego ciagu: 7
```

Zadanie 1.2 (0–3) ROZWIĄŻ NA KOMPUTERZE

Utwórz plik wynikowy (np. *WUZ1_zad1_2_ciagi_wynik.txt*), w którym elementy każdego ciągu będą zapisane w oddzielnym wierszu (oddzielone spacjami) i posortowane niemalejąco. Na przykład dla pierwszych 14 liczb z pliku z danymi wynikiem są ciągi:

```
2340 3662 6457 21832 23534 26892 29660
3153 5304 10064 14659 19277
```

Zadanie 1.3 (0–4) ROZWIĄŻ NA KOMPUTERZE

Utwórz plik wynikowy (np. *WUZ1_zad1_3_ciagi_wynik.txt*), w którym kolejne elementy każdego ciągu będą zapisane w oddzielnym wierszu, a liczby będą oddzielone spacjami. Ciągi w pliku wynikowym mają być uporządkowane od najkrótszego do najdłuższego. Na przykład dla pierwszych 14 liczb z pliku z danymi wynikiem są ciągi:

```
3153 5304 14659 10064 19277
29660 3662 6457 2340 26892 21832 23534
```


2 Algorytmy numeryczne

5. Reprezentacja liczb rzeczywistych w komputerze
6. Błędy w obliczeniach
7. Obliczanie wartości wielomianu
8. Metody obliczeń przybliżonych
9. Algorytmy badające własności geometryczne
10. Fraktale

Wiesz, umiesz, zdasz