Zadania 7

1. Utworzyć plik `zestaw7zad1.cpp`.

   1.1. W funkcji `main` zadeklarować tablicę o nazwie `mojatablica` i rozmiarze `rozmiartablicy`.

   1.2. Napisać funkcję `void wypisz_elementy(int tablica[], int rozmiar)` wypisującą na ekranie elementy tablicy. Użyć jej do wypisania tablicy `mojatablica`.

   1.3. Napisać funkcję `void podaj_elementy(int tablica[], int rozmiar)` pozwalającą użytkownikowi wypełnić liczbami tablicę o rozmiarze n. Z jej pomocą wypełnienić tablicę `mojatablica`, a następnie wypisać zawartość tablicy `mojatablica` funkcją `wypisz_elementy`.

2. Utworzyć plik `zestaw7zad2.cpp`.

   2.1. Przekopiować do niego definicję funkcji `wypisz_elementy`.

   2.2. W funkcji `main` zadeklarować tablicę o nazwie `mojatablica` i rozmiarze `rozmiartablicy`.

   2.3. Napisać funkcję `void wypelnij_tablice(int tablica[], int rozmiar)` wypełniającą tablicę liczbami

   $$1, \ldots, \texttt{rozmiar}.$$

   Z jej pomocą wypełnienić tablicę `mojatablica`, a następnie wypisać zawartość tablicy `mojatablica` funkcją `wypisz_elementy`.

   2.4. Zmienić funkcję `wypelnij_tablice` tak, żeby wypełniała tablicę liczbami:

   $$3 \cdot 2^1,\ 4 \cdot 2^2,\ 5 \cdot 2^3, \ldots$$

3. Użyć pliku `zestaw7zad2.cpp`.

   3.1. Zadeklarować drugą tablicę o nazwie `drugatablica` i tym samym rozmiarze `rozmiartablicy` i przepisać do niej zawartość tablicy `mojatablica`.

   3.2. Odwrócić zawartość tablicy `mojatablica`

   a) za pomocą tablicy pomocniczej (zostawić w pliku `zestaw7zad2.cpp`),

   b) bez tablicy pomocniczej (w nowym pliku `zestaw7zad3.cpp`).

4. Stworzyć plik `zestaw7zad4macierze.cpp`.

   4.1. Zadeklarować globalną zmienną `const int n = 3;`

   4.2. W funkcji `main` zadeklarować dwuwymiarową tablicę typu `double` o nazwie `macierz` i rozmiarze `n`×`n`.

   4.3. Napisać funkcję `void wypisz_macierz(double macierz[n][n])` wyświetlającą wartości macierzy (w formie tablicy). Poprawić wygląd dla liczb zmiennoprzecinkowych używając `printf` (z biblioteki `<iostream>`) zamiast `cout`, np.: `printf("%8.3f ",x);`

   4.4. Napisać funkcję `void transpozycja(double mt[n][n], double m[n][n])` wyświetlającą wartości transponującą macierz `m`.

   4.5. Napisać funkcję `dodaj` dodającą do siebie dwie macierze.

   4.6. Napisać funkcję `iloczyn` mnożącą dwie macierze.

   4.7. Napisać funkcję `double wyznacznik(double m[n][n])` liczącą wyznacznik macierzy `m`.

Opis użycia funkcji `printf`:

1. Modyfikatory:

	opis	przykład użycia	wynik
`d`	liczba całkowita (dziesiętna ze znakiem)	`printf("%d ",-392)`	`-392`
`u`	liczba całkowita (unsigned)	`printf("%u ",35)`	`35`
`f`	dziesiętna zmiennoprzecinkowa	`printf("%f ",392.65)`	`392.65`
`e`	notacja naukowa (mantysa/wykładnik), małe litery	`printf("%e ",392.65)`	`3.9265e+2`
`E`	notacja naukowa (mantysa/wykładnik), wielkie litery	`printf("%E ",392.65)`	`3.9265E+2`
`g`	krótsze spośród `%e` i `%f`	`printf("%g ",392.65)`	`3.9265e+2`
`c`	znak	`printf("%c ",'a')`	`a`
`s`	String of characters	`printf("%s","napis")`	`napis`
`o`	ósemkowa (unsigned)	`printf("%o",200);`	`310`
`x`	szesnastkowa (unsigned) małe litery	`printf("%x",200);`	`c8`
`X`	szesnastkowa (unsigned) wielkie litery	`printf("%X",200);`	`C8`

2. Liczba pomiędzy znakiem `%` a literką oznacza minimalną szerokość pola (napis jest wypełniany spacjami i wyrównany do prawej), np. `"%10d"` wypisuje liczbę w polu o wielkości 10.

`printf("\%10d ",-392)`	`-392`
`printf("\%20d ",-392)`	`-392`
`printf("\%20s ","napis")`	`napis`
`printf("\%20f ",-3.92)`	`-3.920000`

3. Dla formatów liczbowych: jeżeli liczba ta poprzedzona jest zerem (np. `"%010d"`), pole wypełniane jest zerami a nie spacjami:

`printf("\%010d ",392)`	`0000000392`
`printf("\%010d ",-392)`	`-000000392`
`printf("\%20d ",392)`	`00000000000000000392`
`printf("\%20f ",-3.92)`	`-000000000003.920000`

4. Liczba po kropce pomiędzy znakiem `%` a literką (np. `"%.5"`) oznacza:

   - dla liczb całkowitych (`d`, `o`, `x`) — minimalną liczbę cyfr (wypełnione od lewej zerami)

`printf("\%.10d ",392)`	`0000000392`
`printf("\%.10d ",-392)`	`-0000000392`
`printf("\%010d ",-392)`	`-000000392`

   - dla liczb zmiennoprzecinkowych (`f`, `e`, `E`) — liczbę cyfr po przecinku

`printf("\%.4f ",30.925)`	`30.9250`
`printf("\%.2f ",-30.925)`	`30.92`
`printf("\%.4f ",-30.925)`	`-30.9250`

   - dla liczb zmiennoprzecinkowych (`g`, `G`) — liczbę cyfr znaczących

`printf("\%.4g ",30.925)`	`30.93`
`printf("\%.1g ",-30.925)`	`3e+01`

5. Szerokość i precyzję można łączyć:

`printf("\%12f ",30.925)`	`30.925000`
`printf("\%12.2f ",30.925)`	`30.93`
`printf("\%012.2f ",30.925)`	`000000030.93`