

## Lista zadań nr 7

### Tablice dwuwymiarowe.

### Arytmetyka wskaźnikowa i dynamiczna alokacja pamięci dla tablic dwuwymiarowych.

### Zadania podstawowe:

**Zadanie 1** Napisz, program, w którym deklarowana jest tablica o wymiarach 10 na 15 typu `int`. Wykorzystaj funkcję `complete_array()` do wypełnienia tablicy losowymi liczbami całkowitymi z przedziału od  $-500$  do  $500$ . Zaprojektuj funkcję `print_array()`, która zajmie się wyświetleniem tablicy – funkcja powinna wyświetlać 10 wierszy, w których jest po 15 liczb.

**Zadanie 2** Uzupełnij program z zadania 1 o deklarację drugiej tablicy tego samego rozmiaru i typu oraz funkcję `new_array()`, która uzupełni drugą tablicę kolejnymi kwadratami elementów pierwszej tablicy.

**Zadanie 3** Uzupełnij program, z zadań 1 i 2 o funkcję `find_max()`, która zwraca maksymalny element przechowywany w tablicy. Wypisz maksymalne elementy obu tablic.

**Zadanie 4** Napisz funkcję, która otrzymuje jako argumenty tablicę dwuwymiarową o elementach typu `int` oraz jej wymiary i odwraca kolejność elementów we wszystkich wierszach otrzymanej tablicy. Przetestuj funkcję w prostym programie.

**Zadanie 5** Napisz, program, w którym deklarowana jest tablica o wymiarach  $N \times N$  typu `double`. Wykorzystaj funkcję `complete_array()` do wypełnienia liczbami pseudolosowymi. Zaprojektuj funkcję `print_array()`, która zajmie się wyświetleniem tablicy – funkcja powinna wyświetlać  $N$  wierszy, w których jest po  $N$  liczb. Napisz:

- funkcję, która wyzeruje liczby umieszczone na głównej przekątnej tablicy;
- funkcję, która wyznaczy sumę liczb umieszczonych na głównej przekątnej tablicy oraz zwróci tę sumę;
- funkcję, która wyznaczy sumę liczb umieszczonych w  $i$ -tym wierszu tablicy (numer wiersza podany jako argument funkcji) i zwróci tę sumę (za pomocą tej funkcji wyświetl sumy liczb kolejnych wierszy tablicy);
- funkcję, która wyznaczy sumę liczb umieszczonych w  $i$ -tej kolumnie tablicy (numer kolumny podany jako argument funkcji) i zwróci tę sumę (za pomocą tej funkcji wyświetl sumy liczb kolejnych kolumn tablicy);
- funkcję, która w tablicy zamieni elementy  $i$ -tego wiersza z  $j$ -tym (numery wierszy podane jako argumenty funkcji);

- funkcję, która w tablicy znajdzie element najmniejszy (największy) i wyzeruje kolumnę i wiersz w którym on się znajduje;
- funkcję, która transponuje tablicę (zamieni wiersze z kolumnami);
- funkcję, która wyzeruje w tablicy wszystkie elementy leżące pod (lub nad) główną przekątną.

Każda z powyższych funkcji powinna m.in. jako argumenty przyjmować tablicę i jej wymiar.

**Zadanie 6** Napisz program w którym zadeklarowane są trzy tablice typu `int` `tab1`, `tab2` i `tab3` o wymiarach  $M \times N$  oraz jedna tablica `tab4` typu `int` o wymiarach  $N \times M$ . Wykorzystaj funkcję `complete_array()` do wypełnienia tablicy `tab1` losowymi liczbami całkowitymi z przedziału od  $-100$  do  $100$ . Napisz:

- funkcję, za pomocą której przemnożysz elementy tablicy `tab1` przez daną liczbę ( $k$ ), a wynik operacji zostanie zapisany w tablicy `tab2` (funkcja powinna jako argumenty przyjmować dwie tablice, wymiary tablic i liczbę  $k$ ) - mnożenie macierzy przez skalar;
- funkcję, za pomocą której zsumujesz odpowiadające sobie elementy dwóch tablic `tab1` i `tab2`, a wynik operacji zostanie zapisany w tablicy `tab3` (funkcja powinna jako argumenty przyjmować trzy tablice oraz wymiary tablic) - sumowanie macierzy;
- funkcję, za pomocą której przetransponujesz (zamiana wierszy z kolumnami) tablicę `tab3`, a wynik operacji zostanie zapisany w tablicy `tab4` (funkcja powinna jako argumenty przyjmować dwie tablice oraz wymiary tablic) - transpozycja macierzy.

**Zadanie 7** Napisz i przetestuj działanie poniższych funkcji w programie. Alokacji pamięci dla tablicy należy dokonać w oparciu o deklarację wskaźnika: `int **m`.

1. Funkcja `create_array()` - tworzy dynamicznie tablicę dwuwymiarową liczb typu `int` o rozmiarach  $n \times m$ , przekazywanych do funkcji przez argumenty. Funkcja zwraca wskaźnik do zaalokowanego bloku pamięci.
2. Funkcja `complete_array()` - wypełnia tablicę dwuwymiarową przekazaną jej jako argument liczbami losowymi z przedziału od  $0$  do  $99$ . Tablica jest pierwszym argumentem funkcji, drugim i trzecim argumentem wywołania funkcji powinny być wymiary tej tablicy.
3. Funkcja `print_array()` - wyświetla elementy tablicy dwuwymiarowej, o takich samych parametrach jak funkcja `complete_array()`.

**Zadanie 8** Napisz funkcję, która otrzymuje jako argumenty dwuwymiarową prostokątną tablicę tablic tab1 o wymiarach  $n \times m$  i elementach typu int oraz jej wymiary i zwraca jako wartość wskaźnik do nowo utworzonej dwuwymiarowej tablicy tab2 o wymiarach  $m \times n$ , zawierającej transponowaną macierz przechowywaną w tablicy tab1 (czyli dla dowolnych  $k$  i  $j$  zachodzi  $tab1[k][j] = tab2[j][k]$ ). Przetestuj funkcję w programie.

**Zadanie 9** Napisz funkcję, która otrzymuje w argumentach dwie kwadratowe tablice dwuwymiarowe elementów typu int oraz ich wspólny wymiar i zwraca jako wartość wynik dodawania macierzy przechowywanych w przekazanych argumentach. Wynik powinien zostać zwrócony w nowo utworzonej tablicy dwuwymiarowej.

**Zadanie 10** Napisz funkcję, która otrzymuje w argumentach dwie kwadratowe tablice dwuwymiarowe elementów typu int oraz ich wspólny wymiar i zwraca jako wartość wynik mnożenia macierzy przechowywanych w przekazanych argumentach. Wynik powinien zostać zwrócony w nowo utworzonej tablicy dwuwymiarowej. Przetestuj funkcję w prostym programie.

**Zadanie 11** Napisz program, w którym osobna funkcja alokuje pamięć dla dwuwymiarowej tablicy (tab) typu int o rozmiarze  $n \times m$ . Funkcja powinna działać dla dowolnych  $m, n > 0$  i wypełniać tablicę liczbami pseudolosowymi z zakresu od 2 do 100. Wartości  $n$  oraz  $m$  są podane przez użytkownika, ale takie, że  $n \cdot m = 1000$ .

Napisz kolejną funkcję, która jako argumenty przyjmuje tablicę (tab) (i jej wymiary) alokowaną przez pierwszą funkcję oraz liczbę całkowitą  $2 \leq k \leq 100$  i wskaźnik do int. Funkcja powinna sprawdzać, ile jest liczb w tablicy tab podzielnych przez  $k$ , tworzyć dynamicznie tablicę jednowymiarową o takim rozmiarze i zapisywać tam te liczby. Funkcja powinna zwracać wskaźnik do tak zaalokowanej tablicy oraz zapisywać jej rozmiar w miejscu wskazywanym przez przekazany jej wskaźnik. Jeżeli w tablicy tab nie ma liczb podzielnych przez  $k$ , funkcja powinna zwracać NULL.

Napisz osobną funkcję do wyświetlania liczb zapisanych w takiej tablicy (tj. tych podzielnych przez  $k$ ) - pamiętaj, że jej rozmiar będzie zapisany w zmiennej, na którą wskazuje argument wskaźnikowy poprzedniej funkcji. Przetestuj działanie wszystkich funkcji w programie.

**Zadanie 12** Napisz program, w którym definiujesz funkcję, która dynamicznie alokuje pamięć dla tablicy jednowymiarowej 200 liczb całkowitych. Funkcja powinna wypełniać tablicę losowymi liczbami z zakresu od 0 do 5000 i zwracać wskaźnik do tak stworzonej tablicy (tab).

Następnie, zdefiniuj funkcję, która jako argument będzie przyjmowała tak stworzoną tablicę i jej rozmiar oraz „zwykłą” tablicę typu int o rozmiarze 10. Wspomniana funkcja powinna stworzyć tablicę postrzępioną o 10 wierszach - w wierszu pierwszym powinny być zapisane wszystkie liczby z tablicy tab, których ostatnią cyfrą jest 0, w wierszu drugim - te, których

ostatnią cyfrą jest 1, itd. Liczność poszczególnych wierszy (tablic alokowanych dynamicznie) powinna być zapisywana w kolejnych elementach tablicy przekazanej funkcji - tej o rozmiarze 10.

Napisz osobną funkcję, która wyświetli taką tablicę postrzępioną - pamiętaj, że rozmiar każdego wiersza jest zapisany w osobnej tablicy! Przetestuj działanie wszystkich funkcji w programie.

**Zadanie 13** Napisz program, w którym osobna funkcja alokuje pamięć dla dwuwymiarowej tablicy typu `double` o rozmiarze  $n \times m$  ( $5 \leq n, m \leq 20$ ). Funkcja powinna wypełniać tablicę liczbami losowymi, np. z zakresu od  $-100.0$  do  $100.0$ .

Napisz funkcję, która, stosując np. sortowanie bąbelkowe albo sortowanie przez wybieranie, posortuje tę tablicę względem kolumn. W posortowanej tablicy jako pierwsza (z lewej strony) powinna być kolumna o najmniejszej sumie elementów, a jako ostatnia (z prawej strony) ta o największej sumie elementów.

W celu zamiany danych kolumn w tablicy skorzystaj z osobnej funkcji. Również do sumowania elementów we wskazanej kolumnie wykorzystaj osobną funkcję.

## Zadania dodatkowe:

**Zadanie 1** Napisz program, w którym deklarowana jest tablica o wymiarach 16 na 8 typu double. Zaprojektuj i wykorzystaj następujące funkcje:

- `complete()` – funkcja, która uzupełni tablice dowolnymi liczbami (wykorzystaj funkcję `rand()` i `srand()`);
- `sort()` – funkcja, która posortuje tablicę w kolejności rosnącej (tablica powinna być posortowana od lewej do prawej i od góry w dół) np.:

2.3	-1.5	7.0		-12.5	-1.5	2.0
8.1	-12.5	11.0	$\xrightarrow{\text{po sortowaniu}}$	2.3	6.3	7.0
6.3	8.5	2.0		8.1	8.5	11.0

- `show()` – funkcja, która wyświetli tablice (w 16 wierszach po 8 elementów w każdym w polu o szerokości 15).

**Zadanie 2** Jedną z metod szyfrowania wiadomości jest tzw. *metoda podstawieniowa* - poszczególne fragmenty danych są zastępowane innymi. Jedną z odmian tej metody jest sposób określany jako *podstawienie wieloalfabetowe*, a z kolei w jednej z metod podstawiania wieloalfabetowego korzysta się z siatki alfabetów zwanej *tabula recta* (zobacz rysunek 1). W tej tabeli każdy wiersz i kolumna są poetykietowane literą alfabetu, która rozpoczyna dany wiersz lub kolumnę. Każde oczko w siatce jest lokalizowane za pomocą dwóch liter, na przykład w wierszu D i kolumnie H występuje litera K. Mając dany tekst do szyfrowania - *tekst jawny* korzystamy z tzw. *klucza szyfrowania* - tekstu o takiej samej długości jak tekst jawny<sup>1</sup>. Litery tekstu jawnego identyfikują wiersze w tabeli recie, a litery klucza wskazują kolumny. Załóżmy na przykład, że tekst jawny ma postać SECRET, a kluczem szyfrowania jest słowo TOUGH. Ponieważ pierwszą literą tekstu jawnego jest S, a pierwszą literą klucza jest T, pierwszą literą *kryptogramu* czyli tekstu zaszyfrowanego będzie L. Deszyfrowanie polega na odwróceniu tej procedury. Przykładowo mając literę klucza T, bierzemy pod uwagę kolumnę tej litery i szukamy w którym wierszu znajduje się litera kryptogramy L, następnie dla znalezionej wiersza odczytujemy odpowiednią literę z pierwszej kolumny czyli S - pierwsza litera odszyfrowanej wiadomości.

Napisz program, w którym deklarowana jest tablica o wymiarach 26 na 26 typu char. Zaprojektuj i wykorzystaj następujące funkcje:

- `complete_recta()` – funkcja, która uzupełni tablice ( $26 \times 26$ ) literami, aby powstała tabula recta;

---

<sup>1</sup>Jeżeli klucz ma mniej liter to zawsze możemy go "rozszerzyć" cyklicznie powtarzając litery np. mając trzyliterowy klucz KOT możemy zrobić klucz siedmioliterowy KOTKOTK.

- `encryption.text()` – funkcja, która będzie pobierać jako argumenty cztery tablice typu `char`: trzy tablice jednowymiarowe - reprezentujące tekst jawny, klucz i powstały kryptogram, tablicę dwuwymiarową reprezentującą tabelę *tabula recta*, a także rozmiar tych trzech pierwszych tablic (zakładamy, że tablice są tej samej długości), funkcja ma wypełniać (pustą) tablicę kryptogramu odpowiednimi literami zgodnie z przedstawioną w ćwiczeniu zasadą szyfrowania tekstu;
- `decryption.text()` - funkcja, która pobiera argumenty podobnego typu jak funkcja `encryption.text()` ale służy do deszyfrowania tekstu.

Przetestuj napisane funkcje szyfrując i odszyfrowując tekst PROGRAM. Jako klucza użyj słowa MAJONEZ.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Rys. 1: Tabula recta (czerwona pierwsza kolumna i pierwszy wiersz zawierają etykiety)