8 Implementacje macierzy / tablic 2D w ciągłym obszarze pamięci

8.1 Macierz zapisywana jako zlinearyzowana tablica dwuwymiarowa

8.1.1 Macierz prostokątna zapisywana w tablica definiowanej jako 2D. ale przekazywana do funkcji jak tablica 1D

Szablon programu należy uzupełnić o definicje funkcji wykonujących operacje na macierzach prostokątnych o dowolnych wymiarach:

1. `copy_mat()`, która korzysta z bibliotecznej funkcji `memcpy()`,
2. `sum_mat()`, która realizuje operację sumowania macierzy przy użyciu tylko jednej instrukcji pętli,
3. `prod_mat()`, która oblicza iloczyn macierzy korzystając z dwóch pomocniczych funkcji `get()` i `set()`, które obliczają adres każdego elementu macierzy na podstawie numerów wiersza i kolumny oraz liczby kolumn.

Informacje o adresach macierzy (dokładniej — początkowych ich elementów) jest przekazywana do każdej z tych 3 funkcji jako parametr typu `double*`. Jest to wersja tolerowana przez każdy kompilator standardów C90 i wcześniejszych.

Ze względów metodycznych (dydaktycznych, a nie merytorycznych) należy zdefiniować 2 inne wersje funkcji `prod_mat()` różniące się typem parametru informującego o adresie początku tablicy:

- W `prod_mat_v2()` z określeniem typu 3. parametru jako tablicy 2D za pomocą operatora jawnego rzutowania.
- W `prod_mat_v3()` z określeniem typu 3. parametru (tablicy 2D) w nagłówku funkcji.

Zadnie 8.1.1

- **Wejście** 1
  rows cols (liczba wierszy i kolumn 1. macierzy),
  elementy 1. macierzy,
  rows cols (liczba wierszy i kolumn 2. macierzy),
  elementy 2. macierzy.
- **Wyjście** elementy iloczynu macierzy.
- **Przykład** Wejście:
  1
  2 3
  1 2 3
  4 5 6
  3 2

10 20
30 40
50 60

Wyjście:
220.00 280.00 490.00 640.00

8.2 Tablica 2D o wierszach różnej długości – implementacja za pomocą tablicy wskaźników do początkowych elementów wierszy umieszczonych w ciągłym obszarze pamięci

Wiersze tablicy mają być wczytywane ze strumienia wejściowego – dla uproszczenia – z klawiatury. Założenia dotyczące danych w strumieniu wejściowym:

- Każdy ciąg liczbowy jest w linii (rekordzie) zakończonym znakiem nowej linii.
- W każdej linii są tylko liczby - liczba liczb jest dowolna, ale nie mniejsza niż 1.
- Liczby (w systemie dziesiętnym) są w postaci stałych stało- lub zmiennoprzecinkowych, oddzielone spacjami.

8.2.1 Tablica z wierszami typu numerycznego

Szablon programu należy uzupełnić o definicję funkcji

1. `read_dbl_lines_v1()`, która wczytuje linie (rekordy) zawierające ciągi liczb (ciągi o różnej liczebności) i zapisuje (w postaci numerycznej (nie znakowej) do ciągłego obszaru pamięci. Adresy początkowego elementu każdego wiersza zapisuje w tablicy wskaźników przesyłanej pierwszym parametrem. Funkcja zwraca liczbę wczytanych linii.
2. `write_dbl_line_v1()`, która wypisuje wybrany wiersz tablicy.

Zadnie 8.1.2

- **Wejście**
  2
  numer wiersza, który ma być wyprowadzony
  liczby wiersza 1.
  liczby wiersza 2.
  ...
  znak końca pliku (Ctrl D lub Ctrl Z)
- **Wyjście**
  elementy wskazanego wiersza tablicy.

- **Przykład**
  Wejście:
  2
  3
  2 3. 1e-1 -5
  1 2 3
  0 -5
  Ctrl-D

  Wyjście:
  1 2 3

8.3 Tablica 2D o wierszach różnej długości – implementacja za pomocą tablicy wskaźników do początkowych elementów wierszy umieszczonych w odrębnych obszarach pamięci

Funkcje korzystają z dynamicznego przydziału pamięci.

8.3.1 Tablica z wierszami typu znakowego

Linie zawierające ciągi znaków ASCII są wczytywane ze ze strumienia wejściowego i zapisywane do obszarów pamięci przydzielanych dynamicznie funkcją `malloc()`.

Szablon programu należy uzupełnić o definicję funkcji

1. `read_char_lines()`, która wczytuje linie (rekordy) zawierające ciągi znaków. Każdy ciąg jest uzupełniany znakiem końca łańcucha i jest zapisywany do przydzielonej pamięci. Adresy początkowego elementu każdego wiersza zapisuje w tablicy wskaźników przesyłanej pierwszym parametrem. Funkcja zwraca liczbę wczytanych linii.

2. `write_char_line()`, która wypisuje wybrany wiersz tablicy.

Zadnie 8.1.3

- **Wejście**
  3
  numer wiersza, który ma być wyprowadzony
  znaki wiersza 1.
  znaki wiersza 2.
  ...
  znak końca pliku

- **Wyjście**
  elementy wskazanego wiersza tablicy.

- **Przykład**
  Wejście:
  3
  2
  To jest wiersz 1,
  a to drugi.
  Trzeciego nie ma.
  Ctrl-D

  Wyjście:
  a to drugi.

8.3.2 Tablica z wierszami typu numerycznego

W programie jest deklaracja struktury `line` opisującej pojedynczy wiersz tablicy: adres pierwszego elementu wiersza, liczbę elementów wiersza oraz średnią arytmetyczną tych elementów. Definicja tablicy tych struktur jest zdefiniowana w segmencie głównym.

Szablon programu należy uzupełnić o definicję funkcji

1. `read_dbl_lines()`, która wczytuje linie (rekordy) zawierające ciągi liczb (ciągi o różnej liczebności), każdy ciąg jest zapisywany w postaci numerycznej (nie znakowej) do obszaru pamięci przydzielanej funkcją `malloc()`. Adresy przydzielanej pamięci, liczbę elementów wiersza oraz średnią arytmetyczną jego elementów zapisuje w tablicy struktur przesyłanej pierwszym parametrem. Funkcja zwraca liczbę wczytanych linii.

2. `sort_by_average()`, która przy użyciu funkcji `qsort()` sortuje wiersze tablicy wg rosnącej średniej elementów.

3. `write_dbl_line()`, która wypisuje wybrany posortowanej wiersz tablicy.

Zadnie 8.1.4

- **Wejście**
  4
  numer wiersza (w kolejności rosnącej średniej), który ma być wyprowadzony
  liczby wiersza 1.
  liczby wiersza 2.
  ...
  znak końca pliku

- **Wyjście**
  elementy wskazanego wiersza tablicy
  średnia arytmetyczna elementów tego wiersza

- **Przykład**
  Wejście:
  4
  2
  1 2 3 4 5
  -1.0 2
  8
  12 3 1
  Ctrl-D

  Wyjście:
  1.00 2.00 3.00 4.00 5.00
  3.00

8.4 Zadania dodatkowe

8.4.1 Funkcje uniwersalne dla macierzy kwadratowych, w tym symetrycznych lub trójkątnych

W szczególnych przypadkach macierzy kwadratowych – symetrycznych lub trójkątnych[1] – można zaoszczędzić prawie połowę pamięci przeznaczanej dla macierzy w postaci ogólnej. W pamięci są przechowywane tylko elementy leżące na przekątnej głównej oraz pod (a dla trójkątnej górnej – nad) przekątną. Powstaje tablica, której długości kolejnych wierszy wzrastają od 1 do n, n – rozmiar macierzy, a w przypadku trójkątnej górnej – maleją od n do 1. Zakładamy, że macierz symetryczna jest zapisywana w pamięci w formie takiej jak trójkątna dolna. W przypadku ogólnej macierzy kwadratowej wszystkie wiersze mają długość n.

Dla przypomnienia:

1. Suma, iloczyn, odwrotność macierzy macierzy trójkątnych dolnych (górnych) jest macierzą trójkątną dolną (górną).
2. Iloczyn macierzy symetrycznych nie jest (na ogół) macierzą symetryczną.

Należy uzupełnić szablon programu o definicje funkcji

1. `s_mat_copy()`,
2. `s_mat_read()`,
3. `s_mat_print()`,
4. `s_mat_prod()`,
5. `s_get()`,
6. `s_set()`.

[1] `https://en.wikipedia.org/wiki/Triangular_matrix`

Funkcje te powinny być uniwersalne, tzn. obsługiwać macierze kwadratowe o elementach typu `double`, ale z uwzględnieniem szczególnych przypadków – symetrycznych, trójkątnych dolnych lub górnych.

Pierwsze 4 ww. funkcje wywołują w tym celu funkcje `s_get(), s_set()`, które pobierają z (wstawiają do) tablicy wskazane elementy macierzy. Informacja o rodzaju macierzy jest przekazywana za pomocą typu wyliczeniowego `square, symmetric, lower_triangular, upper_triangular`. Dla nadania nazwy nowego typu tych stałych należy użyć słowa kluczowego `typedef` (w linii 12 szablonu programu).

Zadanie 8.2.1 Iloczyn macierzy

- **Wejście**
  1
  rozmiar macierzy A
  typ macierzy A (0-square,1-symmetric,2-lt,3-ut)
  elementy macierzy A
  typ macierzy B (0-square,1-symmetric,2-lt,3-ut)
  elementy macierzy B
  typ iloczynu AB (0-square,1-symmetric,2-lt,3-ut)

- **Wyjście**
  elementy iloczynu macierzy

- **Przykład**
  Wejście:
  1
  3
  2
  1 2 3 4 5 6
  2
  27 8 9 10 11 12
  0

  Wyjście:
  7.00 0.00 0.00
  38.00 27.00 0.00
  128.00 111.00 72.00

8.4.2 Odwracanie macierzy trójkątnej na przykładzie macierzy trójkątnej dolnej

Wyznaczanie macierzy B odwrotnej do A.

Najpierw obliczamy elementy na przekątnej głównej

$$b_{ii} := \frac{1}{a_{ii}}, \quad i = 1, 2, \ldots, n,$$

a później elementy pod przekątną

$$b_{ij} := -b_{ii} \sum_{q=j}^{i-1} a_{iq} b_{qj}, \quad i = 2, 3, \ldots, n, \quad j = 1, 2, \ldots, i-1.$$

Należy przestrzegać kolejności obliczeń: kolumny od lewej do prawej, a w każdej kolumnie wiersze od górnego w dół.

Wersja *in situ* algorytmu wykorzystuje pamięć tylko jednej tablicy (macierzy). Różnica polega na tym, że obliczone elementy są zapisywane bezpośrednio do macierzy odwracanej (nie wymaga innej, np. roboczej tablicy). Cała modyfikacja algorytmu sprowadza się do wstawienia w powyższych wzorach elementów a w miejsce elementów b.

8.4.3 Zadania

Zadanie 8.2.2 Odwrotność macierzy trójkątnej dolnej w wersji $B := A^{-1}$

Szablon programu należy uzupełnić o definicję funkcji `s_mat_inv`, która oblicza macierz odwrotną B do zadanej macierzy trójkątnej dolnej. Jako test funkcji obliczającej odwrotność można sprawdzić, czy iloczyn zadanej macierzy i jej odwrotności jest macierzą jednostkową (z błędem wynikającym z ograniczonej dokładności obliczeń numerycznych).

- **Wejście**
  2
  rozmiar macierzy A
  elementy macierzy A
- **Wyjście**
  elementy macierzy odwrotnej
  elementy iloczynu macierzy A
- **Przykład**
  Wejście:
  2
  3
  1 2 3 4 5 6
  Wyjście:
  1.00 0.00 0.00
  -0.67 0.33 0.00
  -0.11 -0.28 0.17
  1.00 0.00 0.00
  0.00 1.00 0.00
  0.00 0.00 1.00

Zadanie 8.2.3 Odwrotność macierzy trójkątnej dolnej w wersji *in situ* $A := A^{-1}$

Szablon programu należy uzupełnić o definicję funkcji `s_mat_inv_in_situ` – kopią funkcji `s_mat_inv` z wyżej opisaną modyfikacją. Sprawdzenie czy $AA^{-1} == I$ może wymagać zastosowania funkcji `s_mat_copy()`.

- **Wejście**
  3
  rozmiar macierzy A
  elementy macierzy A
- **Wyjście**
  elementy macierzy odwrotnej
  elementy iloczynu macierzy A
- **Przykład**
  Wejście:
  3
  3
  1 2 3 4 5 6
  Wyjście:
  1.00 0.00 0.00
  -0.67 0.33 0.00
  -0.11 -0.28 0.17
  1.00 0.00 0.00
  0.00 1.00 0.00
  0.00 0.00 1.00