9 Implementacje macierzy / tablic 2D

9.1 Macierz zapisywana jako zlinearyzowana tablica dwuwymiarowa

9.1.1 Macierz prostokątna zapisywana w tablicy definiowanej jako 2D, ale przekazywana do funkcji jak tablica 1D

Szablon programu należy uzupełnić o definicję funkcji `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 funkcji jako parametr typu `int*`.

- **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  280
490  640
```

9.2 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 całkowitych, oddzielone spacjami.

9.2.1 Tablica z wierszami typu numerycznego

Szablon programu należy uzupełnić o definicję funkcji

1. `read_int_lines_cont()`, 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_int_line_cont()`, która wypisuje wybrany wiersz tablicy.

- **Wejście**
  2
  numer wiersza, który ma być wyprowadzony (licząc od 1)
  liczby wiersza 1.
  liczby wiersza 2.
  ...
- **Wyjście**
  elementy wskazanego wiersza tablicy.
- **Przykład**
  Wejście:

  ```
  2
  2
  2 3 1 -5
  1 2 3
  0 -5
  ```

  Wyjście:

  ```
  1 2 3
  ```

9.3 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.

9.3.1 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.

- **Wejście**

  3
  numer wiersza, który ma być wyprowadzony (licząc od 1)
  znaki wiersza 1.
  znaki wiersza 2.
  ...

- **Wyjście**

  elementy wskazanego wiersza tablicy.

- **Przykład**

  Wejście:

  ```
  3
  2
  To jest wiersz 1,
  a to drugi.
  Trzeciego nie ma.
  ```

  Wyjście:

  ```
  a to drugi.
  ```

9.4 Tablica z wierszami typu numerycznego

W szablonie programu znajduje się 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_int_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_int_line()`, która wypisuje wybrany posortowanej wiersz tablicy.

- **Wejście**

  4
  numer wiersza, (w kolejności rosnącej średniej) który ma być wyprowadzony (licząc od 1)
  liczby wiersza 1
  liczby wiersza 2
  ...

- **Wyjście**

  elementy wskazanego wiersza tablicy
  średnia arytmetyczna elementów tego wiersza

- **Przykład**

  Wejście:

  ```
  4
  2
  1 2 3 4 5
  -1 2
  8
  12 3 1
  ```

  Wyjście:

  ```
  1 2 3 4 5
  3.00
  ```

9.5 Macierze rzadkie: format skompresowanych wierszy (*Compressed sparse row, CSR*)

Użyteczny link: `https://en.wikipedia.org/wiki/Sparse_matrix`

9.5.1 Format CSR

W formacie CSR macierz rzadka $M_{m\times n}$ jest reprezentowana w pamięci przy pomocy trzech wektorów: V, C (o długości N - liczba niezerowych elementów macierzy), oraz R (o długości $m+1$).

Wektory te zawierają odpowiednio:

1. Wartości niezerowych elementów macierzy ułożone wierszami (wektor V).
2. Indeksy kolumn (wektor C, C_i jest indeksem kolumny, w której znajduje się element V_i).
3. Zakresy wierszy (wektor R, R_j jest całkowitą liczbą niezerowych elementów powyżej wiersza j, oznacza to, że zawsze $R_0 = 0$ i $R_m = N$).

Przykład:
Macierz $M_{4\times 4}$ z pięcioma niezerowymi elementami

$$\begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 5 & 0 & 7 \end{bmatrix}$$

można zapisać używając 3 tablic:
$V = [\ 5\ 8\ 3\ 5\ 7\ ]$
$C = [\ 0\ 1\ 2\ 1\ 3\ ]$
$R = [\ 0\ 1\ 2\ 3\ 5\ ]$

Aby "rozpakować" i-ty wiersz, definiujemy $b_i = R_i$ i $e_i = R_{i+1}$ a następnie dla wszystkich elementów j, $b_i \leqslant j < e_i$ wstawiamy wartość V_j do kolumny C_j.

Aby rozpakować wiersz $i = 1$ (drugi wiersz), bierzemy $b_1 = R_1 = 1$ i $e_1 = R_2 = 2$. Czyli dla $j = 1$ mamy $V_1 = 8$ i $C_1 = 1$ co oznacza, że w wierszu o indeksie 1 mamy jeden niezerowy element (j przyjmuje tylko jedną wartość) o wartości 8 w kolumnie 1.

Format CSR jest bardzo wygodny do implementacji algorytmu mnożenia macierzy rzadkiej przez wektor x. Ponieważ w mnożeniu uwzględniamy tylko elementy o niezerowych wartościach, algorytm można zapisać w postaci:

for $i \in \{1, 2, \ldots, m-1\}$ **do**
  $y_i = 0$
  for $j \in \{R_i, \ldots, R_{i+1} - 1\}$ **do**
    $y_i = y_i + V_j \times x_{C_j}$
  end for
end for

Szablon programu należy uzupełnić o definicję funkcji:

1. `read_sparse()`, która wczytuje linie zawierające trójki liczb całkowitych. Pierwsze dwie liczby są odpowiednio numerem wiersza i kolumny elementu macierzy, a trzecia stanowi jego wartość. Funkcja zwraca liczbę wczytanych trójek.
2. `make_CSR()`, która na podstawie wczytanych trójek generuje trzy wektory odpowiadające formatowi CSR.
3. `multiply_by_vector()`, która mnoży macierz rzadką w formacie CSR przez zadany wektor x.
4. `write_vector()`, która wypisuje wektor liczb całkowitych.

- **Wejście**

  5
  m n – liczba wierszy i kolumn macierzy
  N – liczba elementów macierzy o niezerowych wartościach
  N trójek wiersz, kolumna, wartość
  i_0 j_0 v_0
  i_1 j_1 v_1
  ...
  n liczb całkowitych - wartości elementów wektora x

- **Wyjście**

  elementy wektora V
  elementy wektora C
  elementy wektora R
  elementy wektora $y = Mx$

- **Przykład**

  Wejście:

  ```
  5
  4 4
  5
  2 2 3
  3 1 5
  0 0 5
  3 3 7
  1 1 8
  1 2 3 4
  ```

  Wyjście:

  ```
  5 8 3 5 7
  0 1 2 1 3
  0 1 2 3 5
  5 16 9 38
  ```