

12 Operacje na liście dwukierunkowej tablic z iteratorem

Uwagi ogólne

Celem ćwiczenia jest zaimplementowanie struktury opisującej listę dwukierunkową elementów będących tablicami (alokowanymi dynamicznie) oraz funkcji realizujących operacje na tej liście. Do niektórych operacji będzie wykorzystywana struktura iteratora.

12.1 Dwukierunkowa lista tablic (z iteratorem): zadania do wykonania

Dane wejściowe do programu: część wspólna

W każdym z kolejnych podpunktów dane zawierają:

`to_do` – numer zadania do wykonania
`n` – liczba węzłów
`n` ciągów liczb całkowitych postaci:
`m` – liczba elementów w tablicy węzła,
`m` elementów tablicy rozdzielonych spacją
... ewentualne dodatkowe dane

12.1.1 Wstawianie elementów do dwukierunkowej listy tablic

Szablon programu należy uzupełnić o definicję funkcji `push_back()`. Pozwala ona na budowę dwukierunkowej listy tablic poprzez wstawianie na koniec listy kolejnych węzłów zawierających tablice.

- **Wejście**

1
wymiary i wartości elementów tablic jak opisano w części ogólnej

- **Wyjście**

Lista tablic, gdzie `->` oznacza węzeł, a elementy tablicy rozdzielone są spacjami.

- **Przykład:**

Wejście:

1
3
3 6 7 9
2 4 8
5 9 7 3 5 2

Wyjście:

-> 6 7 9
-> 4 8
-> 9 7 3 5 2

12.1.2 Iterowanie po strukturze dwukierunkowej listy tablic do przodu

Szablon programu należy uzupełnić o definicję funkcji `skip_forward()` i `get_forward()`, pozwalające poprzez iterowanie po liście tablic do przodu, na wypisanie zawartości wybranych komórek na standardowe wyjście.

- **Wejście**

2

wymiary i wartości elementów tablic jak opisano w części ogólnej
liczba komórek do przejścia i numery wybranych komórek

- **Wyjście**

Wartości zawarte w wybranych komórkach listy tablic (komórki liczone od początku listy, pierwszy element pierwszego węzła ma numer 1).

- **Przykład:**

Wejście:

2

3

3 6 7 9

2 4 8

5 9 7 3 5 2

3 5 4 1

Wyjście:

8 4 6

12.1.3 Iterowanie po strukturze dwukierunkowej listy tablic do tyłu

Szablon programu należy uzupełnić o definicję funkcji `skip_backward()` i `get_backward()`, pozwalające poprzez iterowanie po liście tablic do tyłu, na wypisanie zawartości wybranych komórek na standardowe wyjście.

- **Wejście**

3

wymiary i wartości elementów tablic jak opisano w części ogólnej
liczba komórek do przejścia i numery wybranych komórek

- **Wyjście**

Wartości zawarte w wybranych komórkach listy tablic (komórki liczone od końca listy, ostatni element ostatniego węzła ma numer 1).

- **Przykład:**

Wejście:

3

3

3 6 7 9

2 4 8

5 9 7 3 5 2

3 5 4 10

Wyjście:

9 7 6

12.1.4 Usuwanie wybranych komórek z listy tablic

Szablon programu należy uzupełnić o definicję funkcji `remove_at()`. Funkcja ta usuwa wskazane komórki dwukierunkowej listy tablic. Jeśli usuwana komórka jest jedyną komórką tablicy w węźle (tablica jednoelementowa) to usuwany jest cały węzeł.

Uwaga: komórka o danym numerze jest wyznaczana w każdym kroku licząc od początku listy, tzn. jeżeli w pierwszym kroku usuniemy komórkę o numerze 1 to w kolejnym komórka następna będzie miała numer 1 (a nie 2 jak przed wykonaniem poprzedniego kroku).

- **Wejście**

4

wymiary i wartości elementów tablic jak opisano w części ogólnej
liczba komórek do usunięcia i numery wybranych komórek

- **Wyjście**

Lista tablic po usunięciu wskazanych komórek (komórki liczone od przodu)
w formacie jak w punkcie 1.

- **Przykład:**

Wejście:

4

3

3 6 7 9

2 4 8

5 9 7 3 5 2

3 5 4 1

Wyjście:

-> 7 9

-> 9 7 3 5 2

12.1.5 Budowa listy na podstawie liczby cyfr elementów tablic

Niech $d(n)$ oznacza liczbę cyfr liczby n .

Zadanie polega na zbudowaniu listy, której każdy węzeł będzie zawierał posortowaną rosnąco tablicę liczb o tej samej długości (liczbie cyfr).

Szablon programu należy uzupełnić o definicję funkcji `insert_in_order()`. Funkcja ta czyta kolejne liczby całkowite a następnie:

1. Jeżeli w liście znajduje się już węzeł zawierający liczby o długości równej $d(n)$ to funkcja dodaje liczbę n do tej tablicy zachowując jej rosnące uporządkowanie.

2. W przeciwnym przypadku funkcja dodaje do listy nowy węzeł (tak, by lista była uporządkowana rosnąco względem długości liczb w kolejnych węzłach).

Po dodaniu do listy wszystkich wczytanych liczb, program wypisuje listę.

- **Wejście**

5

n – liczba liczb do umieszczenia w węzłach listy

n liczb całkowitych (wartości komórek listy)

- **Wyjście**

Lista tablic po wpisaniu wszystkich elementów z wejścia w formacie jak w punkcie 1. Węzły listy powinny być posortowane rosnąco względem długości (liczby cyfr) liczb, które przechowują. Tablice w węzłach powinny być posortowane rosnąco.

- **Przykład:**

Wejście:

5

9

999 14 733 29 22222 334 0 -12 -856

Wyjście:

-> 0

-> -12 14 29

-> -856 334 733 999

-> 22222