Πίνακες

Δημήτρης Φωτάκης

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

Πανεπιστήμιο Αιγαίου

Πίνακες (arrays)

- Αποθήκευση πολλών αντικειμένων
  - ίδιου τύπου δεδομένων,
  - σε συνεχόμενες θέσεις μνήμης.

```
int A[100]; double D[50];
struct info { ... } B[N]; // N σταθερά
```

  - Η αρίθμηση αρχίζει από το 0 !

```
for (i = 0; i < 100; i++)
        A[i] = 0;
```


Πίνακας ως ΑΤΔ

- Αποθήκευση **διατεταγμένων ζευγαριών** *(index, value)* με **διαφορετικό index** για κάθε αντικείμενο.
  {(Α, 9.5), (Β, 8), (Γ, 4), (Δ, 3.5), (Ε, 10), (Ζ, 9), (Η, 7.5) }
- Συχνά index **δεν έχει σημασία**. Μόνο ότι είναι **διαφορετικό**.
- Λειτουργίες:
  - **Δημιουργία** (στατική, δυναμική).
    `int A[100]; int *A = new int[size];`
  - **Αποθήκευση** (index, value): τιμή value στη θέση index.
    `A[index] = value;`
  - **Προσπέλαση** (index): επιστρέφει value στη θέση index.
    `return(A[index]);`
- Εξαιρετικά εύκολη υλοποίηση. Ελάχιστος χώρος.
- Χρόνος: δημιουργία: Ο(*n*). αποθήκευση – προσπέλαση: Ο(1).

Πολυδιάστατοι Πίνακες (matrices)

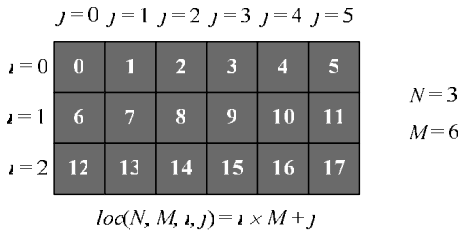
- Αποθήκευση αριθμητικών δεδομένων

```
int A[N][M]; double D[N][M]; // N, M σταθερές
for (i = 0; i < N; i++)
   for (j = 0; j < M; j++)
          A[i][j] = 0;
```

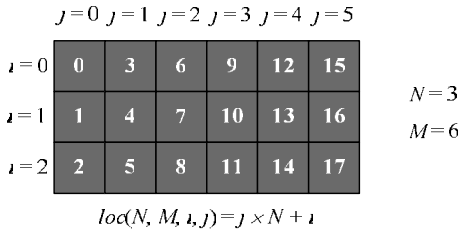
- Μονοδιάστατοι πίνακες (arrays) αρκούν για υλοποίηση πολυδιάστατων (matrices).

Αρίθμηση κατά Γραμμές - Στήλες

- Αρίθμηση κατά **γραμμές** (C, C++).
  loc(*N*, *M*, *i*, *j*) = *i* × *M* + *j*



- Αρίθμηση κατά **στήλες**.
  loc(*N*, *M*, *i*, *j*) = *j* × *N* + *i*



Άθροισμα Πινάκων

- **Άθροισμα** $C[i][j] = A[i][j] + B[i][j]$

```
int A[n][m], B[n][m], C[n][m];
for (i = 0; i < n; i++)
   for (j = 0; j < m; j++)
         C[i][j] = A[i][j] + B[i][j];
```

- Γραμμικός χρόνος O(*n m*) (βέλτιστος).

Γινόμενο Πινάκων

- **Γινόμενο** $C[i][j] = \sum_{k=0}^{p-1} A[i][k] \times B[k][j]$

```
int A[n][p], B[p][m], C[n][m];
for (i = 0; i < n; i++)
   for (j = 0; j < m; j++) {
         C[i][j] = 0;
         for (k = 0; k < p; k++)
               C[i][j] += A[i][k] * B[k][j];
   }
```

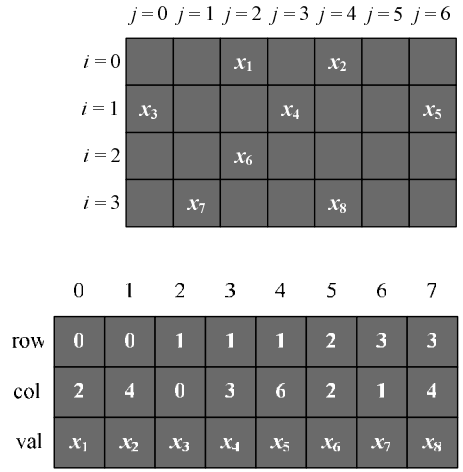
- Χρόνος O(*n p m*).
- Υπάρχουν ταχύτεροι αλγόριθμοι!

Ειδικές Μορφές

- Διαγώνιος: στοιχεία εκτός διαγωνίου 0.
- Τριδιαγώνιος: *M*[*i*, *j*] = 0 αν | *i* – *j* | > 1.
- Τριγωνικός κάτω (άνω): *M*[*i*, *j*]= 0 αν *i* < *j* (*i* > *j*).
- Συμμετρικός: *M*[*i*, *j*] = *M*[*j*, *i*].
- Αποθηκεύουμε μόνο *μη-μηδενικά (διαφορετικά)* στοιχεία.
  - Διαγώνιος: *loc*(*i*, *i*) = *i*.
  - Τριδιαγώνιος: *loc*(*i*, *j*) = 2*i* + *j*.
  - Τριγωνικός άνω: *loc*(*i*, *j*) = *i*(*i*+1)/2+*j*
  - Συμμετρικός: *loc*(*i*, *j*) = *loc*(*i*, *j*) = *i*(*i*+1)/2+*j*

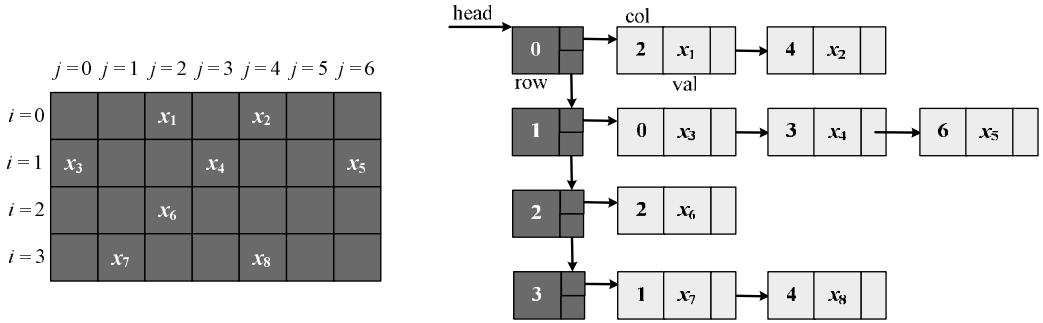
Αραιοί Πίνακες

- **Αραιός πίνακας** $n \times n$ με $k = o(n^2)$ ($< n^2 / 5$) μη-μηδενικά στοιχεία.
- Αναπαράσταση με πίνακες.
- $\approx 3k$ χώρος (δυναμική δομή;).
- Αποθήκευση: χώρος $O(k)$.
- Προσπέλαση: χρόνος $O(\log k)$ (δυαδική αναζήτηση).



Αραιοί Πίνακες

- Αναπαράσταση με λίστες: χώρος $O(k)$.
- Αποθήκευση: χρόνος $O(\min\{n+m, k\})$.
  Προσπέλαση: χρόνος $O(\min\{n+m, k\})$.



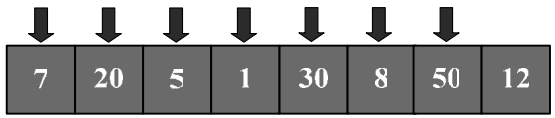
Αναζήτηση σε Πίνακες

- Όχι ταξινόμηση: **γραμμική** αναζήτηση

```
int linearSearch(int A[], int n, int x)
{
   for (int i = 0; i < n; i++)
      if (x == A[i]) return(i);
   return(-1);
}
```

- Χρόνος $O(n)$ (βέλτιστος).

Αναζήτηση **50**:



Αναζήτηση σε Πίνακες

- Ταξινόμηση: **δυαδική** αναζήτηση.
- Χρόνος $O(\log n)$ (βέλτιστος).

```
while (low <= up) {
   mid = (low + up) / 2;
   if (A[mid] == k) return(mid);
   else if (A[mid] > k) up = mid - 1;
        else            low = mid + 1;
}
```

Αναζήτηση **30**: