Array n-dimensionali
Matrici e cenni di Algebra Lineare

Liceo G.B. Brocchi
Classi seconde Scientifico – opzione scienze applicate
Bassano del Grappa, Gennaio 2023
Prof. Giovanni Mazzocchin

Array multidimensionali

- Un array a 2 dimensioni *n x m* («n per m») di tipo *T*, è un array di *n* array, ciascuno di *m* elementi di tipo *T*. Gli array bidimensionali vengono comunemente detti matrici e sono utilizzatissimi nei programmi per il calcolo scientifico/numerico
- Un array a 3 dimensioni *n x m x p* («n per m per p») di tipo *T*, è un array di *n* array, ciascuno dei quali è un array bidimensionale *m x p*, ossia un array di *m* array, ciascuno dei quali di p elementi di tipo T. Visualizzatelo come un parallelepipedo rettangolo di *n* strati. Ogni strato è una matrice *m x p*
- Un array a 4 dimensioni è … teoricamente si può fare, ma è abbastanza inutile

Costruzione ricorsiva

{_, _ ,_}

Un array di 3 «cose» (gli *underscore*), separate da 2 virgole.
Si tratta di un array 1-dimensionale di lunghezza 3

{{_, _},{_, _},{_, _}}

Un array di 3 array.
Ciascun array «interno» contiene 2 «cose»

Costruzione ricorsiva

{ { {_, _, _}, {_, _, _} },{ {_, _, _}, {_, _, _} },{ {_, _, _}, {_, _, _} } }

Un array *A* di 3 array.
Ciascuno dei 3 elementi di *A* è un array bidimensionale 2 x 3.
Quindi, ciascun elemento di *A* è un array di 2 array, ciascuno dei quali di 3 elementi

In C++

```
int ai_one[5];  //1-dimensional 5 int array
double ad[3][4];  //2-dimensional 3x4 double array
double ad_square[4][4];  //2-dimensional 4x4 double array (aka square matrix)
char ac[6][3][2];  //3-dimensional 6x3x2 char array

ad[2][3] = 4.5;  //access to item at row 2 and column 3
ad_square[4][4] = 3;  //illegal access! row 4 and column 4 don't exist
```

```
int mat[3][3] = {{6, 5, 4}, {2, 2, 1}, {7, 6, 2}}; //decl. and initialization
```

In C++

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
#define N_ROWS 5
#define N_COLS 4
int matr[N_ROWS][N_COLS];

void print_matrix(int matr[][N_COLS], int rows) {
 for (int i = 0; i < rows; ++i) {
  for (int j = 0; j < N_COLS; ++j) {
   cout << matr[i][j] << '\t';
  }
  cout << '\n';
 }
}
int main() {
 srand(time(NULL));
 for (int i = 0; i < N_ROWS; ++i) {
  for (int j = 0; j < N_COLS; ++j) {
   matr[i][j] = rand() % 200;
  }
 }

 print_matrix(matr, N_ROWS);
}
```

Algebra lineare – definizioni ed algoritmi

1. Una matrice quadrata è detta *matrice identità* se la diagonale principale è composta da tutti 1, e in tutte le altre posizioni si hanno solo zeri:
   - scrivere una funzione che verifica se una matrice quadrata `12 x 12` è la matrice identità

2. Data la matrice `A` di dimensioni `n x m`, la sua trasposta è una matrice A_T di dimensioni m x n tale che per ogni coppia `(i, j)` si ha che `A_T[i][j] == A[j][i]`
   - scrivere un programma che riceve una matrice 24 x 32 e ne calcola la traposta

Algebra lineare – definizioni ed algoritmi

- Prodotto matrice per scalare: scrivere una funzione che riceve una matrice `A` `n_rows` `x 6` di double, un double-reference `k` e una seconda matrice `B` `n_rows` `x 6`. La funzione deve riempire `B[i][j]` con `A[i][j] * k` per ogni `(i, j)`

- Implementare il prodotto tra matrici (in classe con la guida dell'insegnante)

- Implementare l'eliminazione Gaussiana (in classe con la guida dell'insegnante)