Indice

Capitolo 1

Common Library

1.1 Strutture

Array

Gli Array sono una struttura statica, che permette di accedere ad un gruppo di elementi dello stesso tipo attraverso un indice. Una volta creati non è possibile modificarne la dimensione e soprattutto il tipo di dato contenuto. Per questo al momento della creazione è necessario specificare entrambi questi parametri, ad esempio:

```
Array<double, 100> v;
```

costruisce un Array con 100 elementi di tipo `double`. Si accede ad ogni suo elemento specificando l'indice tra delle parentesi quadre, ricordando che il primo elemento si trova nella posizione `0` e non nella `1`. Ad esempio:

```
v[2] = 0.;
v[1] = v[2];
```

il primo comando memorizza il valore `0.` nella componente individuata dall'indice `2`, il secondo invece legge questo valore e lo riscrive nella componente `1`. È anche possibile costruire vettori multipli, per la quale è necessario usare la notazione a più indici. L'esempio seguente mostra come costruire un Array doppio con 5 righe e 6 colonne, poi come accedervi:

```
Array<Array<double, 6>, 5> m;
m[0][0] = 1.;
m[1][0] = 2.;
m[0][1] = m[1][0];
```

Bisogna fare attenzione che questa struttura non esegue nessun controllo sugli indici, un utilizzo scoretto potrebbe portare errori a *run-time*, perché si tenta di accedere ad aree di memoria riservate ad altre strutture o programmi.
Vi è anche la possibilità di copiare un Array in un altro, purché abbiamo la stessa dimensione e contengano elementi dello stesso tipo, questo controllo viene eseguito a *compile-time*. L'utilizzo di questo metodo è preferibile a quello di copiarlo componente per componente, perché molto più efficente, siccome sfrutta la possibilità di copiare blocchi di memoria di dimensioni maggiori.

```
Array<double, 10> v;
Array<double, 10> w;
Array<float, 10> z;
v = w;   // L'Array w viene copiato in v
z = w;   // Errore di compilazione
```

L'utilizzo dei template limita la costruzione di Array la cui dimensione è nota a compile-time, ma è anche possibile definirne la dimensione a run-time, per fare questo è sufficiente definire un Array nel modo seguente:

```
Array<double> v(10);
Array<double> w(10);
Array<float> z(10);
```

In questo caso è sempre possibile copiare un Array in un altro, ma se le due strutture hanno dimensioni differenti, viene copiata la porzione di dimensione minima.

Facciamo notare che questa struttura in modo automatico decide se allocare se stessa nello stack o nel heap, in base alla propria dimensione. Questo per garantire una maggiore velocità di esecuzione per piccole strutture e evitare problemi di stack-overflow su strutture molto grandi.

Capitolo 2

Linear Algebra

2.1 Vettori e Matrici

2.2 Sistemi lineari

2.3 Autovalori e autovettori