09_Array_Puntatori

Table of contents

Array e puntatori

All'uso dell'espressione di tipo array di `T`, viene applicato il *type decay* o trasformazione di lvalue. Viene passato per valore il valore del puntatore che punta al primo elemento dell'array. Quindi da tenere conto che non essere sincronizzati con l'array, ovvero rispetto la fine o l'inizio, causa undefined behaviour se tentata la lettura o scrittura.

```
// espressioni equivalenti
int a[100];
int b = 5;

a[b];
*(a+b)
```

Aritmetica dei puntatori

Se `ptr` è puntatore, possiamo usarlo per spostarci avanti o indietro all'interno della memoria del nostro array. Fare per esempio: `puntatore1` - `puntatore2`, ci restituisce la distanza tra i due.

```
ptr + n; // mi sposto di '+n' posizioni nell'array
ptr - n; // mi sposto di '-n' posizioni nell'array
```

Con l'assunzione della variabile `int a[100]`, definiamo:

- iterazione basata su indice, metodo classico

  ```
  for (int i=0;i=!100;++i) {
    // lavoro con a[i]
  }
  ```

- *iterazione basata su puntatore*, modo idiomatico per ciclare elementi in array, che è esattamente quello su cui lavoreremo nel corso perché quello più usato

  ```
  for (int* p=a;p=!a+100;++p) {
    // lavoro con *p
  }
  ```

L'esistenza di questi due modi di fare la stessa cosa, deriva dal fatto che lavorando con i template, il pensiero astratto è la base (se stiamo usando array o vector non importa). Possiamo anche pensare a intervalli nel nostro esempio: piuttosto che dall'inizio alla fine, uso `p1` come puntatore marcante l'inizio e `p2` marcante la fine dell'intervallo con cui lavorare.

```
for (;first=!last;++first){
        // lavoro con *first
}
```

07/03/2023