



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Array

Stefano Ghidoni



DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE



Agenda

- Array come variabili (named variable)
- Array e puntatori
- Aritmetica dei puntatori
- Un esempio



Array

- Gli std::vector sono vettori dinamici, intelligenti, di alto livello
- Gli array "stile C" sono strutture dati più semplici, *più antiche*
- Gli array sono necessari in alcune circostanze (più avanti nel corso)



- Array: sequenza omogenea di oggetti allocati in spazi di memoria contigui
- Stesso tipo, e nessuno spazio vuoto tra elementi
- Indicizzati con []
- Accesso casuale
- Nessun controllo su lettura/scrittura fuori range
 - Facile fare errori difficili da trovare

Definizione di vettore come struttura dati, infatti sia gli array che i vector sono coerenti con questa definizione.





Dimensione di un array

- Forte limitazione sulla dimensione di un array
- Deve essere una **costante** (literal o const) conosciuta a tempo di compilazione
- Esistono anche VLA (Variable Length Array)
 - Fanno parte dello standard C99
 - **Non sono standard C++**
 - GCC li accetta



Array come named variable

- Gli array sono istanziabili come variabili
 - Variabili globali
 - Variabili locali
 - Attenzione alla memoria nello stack!
 - Argomenti di funzioni
 - Trasformati in puntatori
 - Membri di una classe



- Un po' di codice:

```
const int max = 100;
int gai[max];           // array globale, sempre disponibile

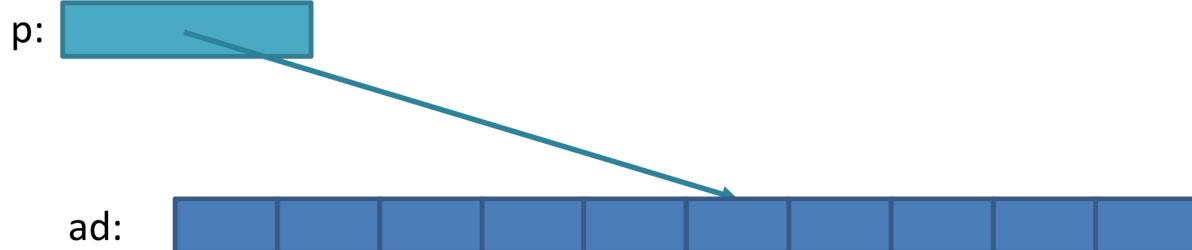
void f(int n)
{
    char lac[20];        // array locale: vive fino all'uscita
                          // dallo scope
    int lai[60];
    double lad[n];        // errore: dimensione non costante
                          // (se n non è nota a tempo di
                          // compilazione)
}
```



Puntatori a elementi di un array

- Possiamo definire puntatori a elementi di un array

```
double ad[10];
double* p = &ad[5];
```

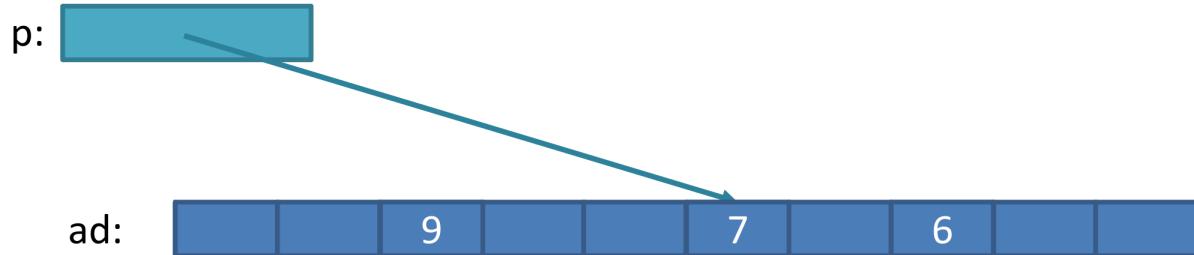




Puntatori a elementi di un array

- Possiamo usare subscript e dereference su p

```
*p = 7;  
p[2] = 6;  
p[-3] = 9;
```



Ho usato e indicizzato il puntatore, come se lavorassi con un vettore.



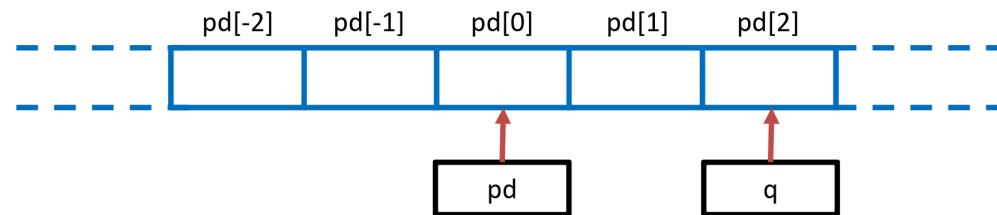
Aritmetica dei puntatori

- **Aritmetica dei puntatori:** i puntatori sono tipi che supportano la somma e la sottrazione con interi
- Sommare o sottrarre un intero n da un puntatore significa spostare il puntatore di n slot a destra o a sinistra
 - Operatori: $+$, $-$, $+ =$, $- =$
- Lo slot dipende dal dato puntato
 - L'aritmetica è sensibile al contesto
- Un'ulteriore dimostrazione che ciò che fanno gli operatori dipende dagli operandi



Esempio

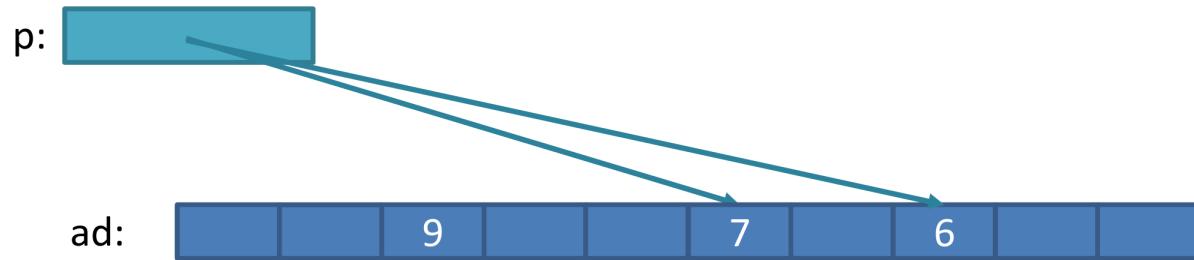
```
double* q = pd + 2;
```





Esempio

```
p += 2;
```





Stringhe

- Le stringhe "stile C" sono simili agli array
 - Array di char *Perché rappresentano stringhe alfanumeriche*
 - Terminati dal carattere terminatore '\0'
 - Manipolabili con una serie di funzioni dedicate
 - Assumono la presenza del carattere terminatore
- Hanno le stesse limitazioni degli array
- Le stringhe sono un retaggio del C
- Il C++ offre una versione moderna e di alto livello
 - std::string

Array e puntatori



Decadimento di un array

- Il nome di un array è un puntatore const al suo primo elemento
- È un rvalue! Cioè un valore, non una variabile
- Non possiamo usarlo per la copia

```
int x[100];
int y[100];
// ...
x = y;           // errore
int z[100] = y; // errore
```



Decadimento di un array

- Passare un array a una funzione significa passare il puntatore – **decadimento**
 - Perdita di informazione sulla dimensione dell'array
- Lo vediamo con un esempio basato sulle stringhe



Decadimento di una stringa

```
int strlen(const char a[])
{
    int count = 0;
    while (a[count]) { ++count; }
    return count;
}

char lots[100000];

void f()
{
    int nchar = strlen(lots);
    // ...
}
```

- Sono copiati 100k char?



Decadimento di una stringa

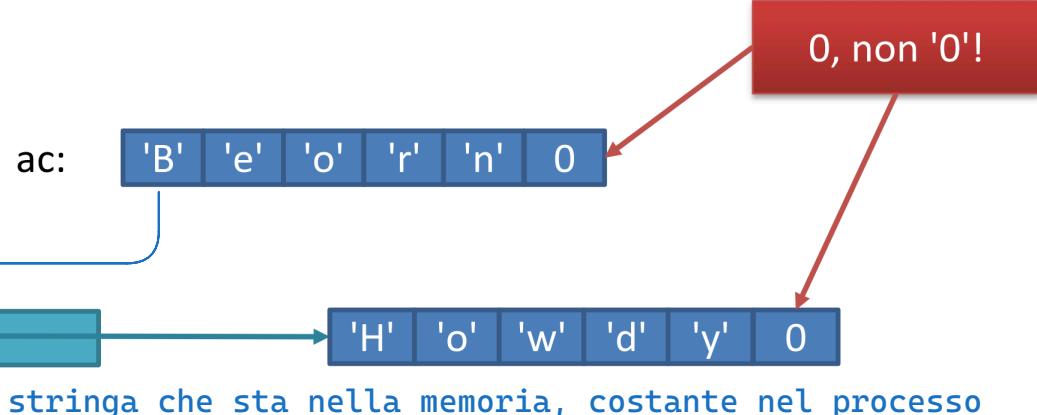
- `strlen(lots)` è considerato equivalente a `strlen(&lots[0]);`
- Il decadimento serve per evitare copie non desiderate
- Un comportamento un po' antiquato
 - Deriva da C, C++ lo mantiene per retrocompatibilità



Inizializzazione di una stringa

- Inizializzazione di una stringa: string literal

```
char ac[] = "Beorn";  
char* pc = "Howdy";
```



Ricordo che gli apici singoli si riferiscono a uno specifico char, mentre gli apici doppi si riferiscono a una stringa



Inizializzazione di un vettore

- Inizializzazione di un array: lista di valori

```
int ai[] = { 1, 2, 3, 4, 5, 6 };      // dimensione dedotta: 6

int ai2[100] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 }; // gli altri
                                                // inizializzati a 0

double ad[100] = {}; // tutti inizializzati a 0.0

char chars[] = { 'a', 'b', 'c' }; // nessuno 0 terminatore
```

Problemi legati a puntatori e array



Problemi legati a puntatori e array

- Alcuni problemi classici:
 - Accesso usando un nullptr

```
int* p = nullptr;  
*p = 7;           // errore!
```



Problemi legati a puntatori e array

- Alcuni problemi classici:
 - Accesso usando un puntatore non inizializzato
 - In particolare: puntatori membro (facilmente ci si dimentica di inizializzarli)

```
int* p;  
*p = 9; // errore!
```



- Alcuni problemi classici:
 - Accessi al di fuori dei limiti di un array
 - In particolare: primo e ultimo elemento

```
int a[10];
int* p = &a[10];
*p = 11;           // errore!
a[10] = 11;        // errore!
```



Problemi legati a puntatori e array

- Alcuni problemi classici:
 - Accesso a un oggetto uscito dallo scope

```
int* f()
{
    int x = 7;
    // ...
    return &x;
}

int* p = f();           // a cosa punta?
*p = 15;
```

Finita la chiamata alla funzione, viene rimossa dallo stack. Non viene eliminata immediatamente, ma i suoi valori possono essere sovrascritti, quindi non è sicuro a cosa punti quel puntatore.



Problemi legati a puntatori e array

- Un caso logicamente analogo

```
vector& ff()
{
    vector x(7);
    return x;
}      // x è distrutto qui!

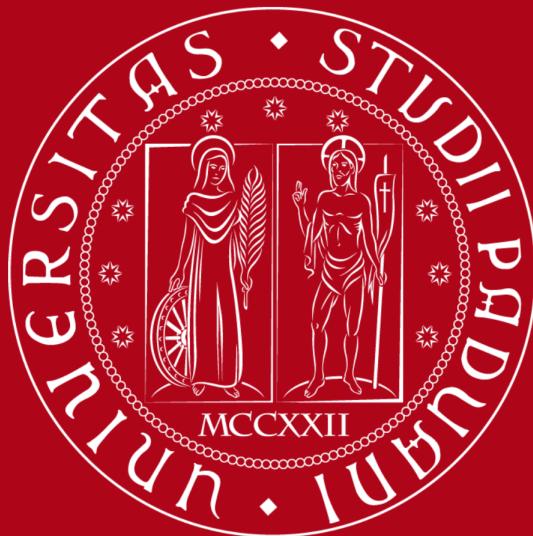
vector& p = ff();
p[4] = 15;           // ouch! (BS)
```

- A cosa si riferisce p?



Recap

- Array "stile C"
- Dimensione (costante) di un array
- Puntatori applicati a elementi di un array
 - E relativi problemi
- Stringhe "stile C"
- Nome di un array come puntatore costante



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Array

Stefano Ghidoni



DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE