

Parte 7

Puntatori a funzione



[S. Dalí – Sleep, 1937]

Puntatori a funzione

- I puntatori in C/C++ possono puntare qualsiasi oggetto, incluse le funzioni!
- Esempio di dichiarazione di un puntatore a funzione

```
int (* puntafun) (double, int) ;
```

puntatore a funzione con due parametri in input (un double e un int) e un int di ritorno

- Parentesi necessarie
 - cosa sarebbe senza?

Assegnamento

- Possono essere assegnati con il nome di una funzione “compatibile” precedentemente dichiarata
- Una funzione è compatibile se ha gli stessi tipi di parametri di input e output
- Es:

```
int (* puntafun) (double, int) ;  
int f1(double , int) ;  
int f2(double , int) ;  
if ( ..... ) puntafun = f1 ;  
else puntafun = f2 ;
```

Solo nome
senza argomenti

Assegnamento

- L'assegnamento corretto sarebbe
`puntafun = &f1 ;`
`puntafun = &f2 ;`
- Ma le cose funzionano ugualmente con
`puntafun = f1 ;`
`puntafun = f2 ;`
- Il compilatore ha una certa flessibilità di interpretazione

Chiamata

- Per chiamare la funzione puntata, occorre dereferenziare il puntatore.
- Esempio:
`(*puntafun) (45.76, 5) ;`
- In realtà, si può anche chiamare direttamente il puntatore stesso.
- Esempio:
`puntafun (45.76, 5) ;`
- In entrambi i casi il compilatore capisce che è l'indirizzo di una funzione.

Array di puntatori a funzione

- Si possono creare vettori anche di puntatori a funzione

- Esempio:

```
int f1 (double, int) ;
```

```
int f2 (double, int) ;
```

```
int (* puntafun[2]) (double, int) = {f1, f2} ;
```

- I singoli elementi possono anche essere assegnati con l'operatore di selezione:

```
puntafun[0] = f1 ;
```

```
puntafun[1] = f2 ;
```

Array di puntatori a funzione

- Per la chiamata dell'elemento desiderato:
`puntafun[0](56.67, 5) ;`
`puntafun[1](64.7, 8) ;`
- Possono essere utili per l'implementazione di un menu nel quale la funzione da eseguire dipende da un indice i , senza usare costrutti *if* e *switch*

Esercizio

- Scrivere un programma che:
 - Richieda se effettuare un'operazione di addizione, sottrazione, moltiplicazione o divisione tra due interi
 - A seconda dell'operazione scelta, invochi una funzione che fornisca il risultato corretto
 - La selezione della funzione avvenga senza costrutti if, switch, etc.
- Soluzione in *operazioni.cc*

Passaggio di puntatori a funzione

- Una funzione può dichiarare tra i suoi argomenti un elemento di tipo puntatore a funzione

- Es:

```
void sel_fun(int (*pfun) (double))  
{int n ;  
  ...  
  n = pfun(64.7) ;  
  ... }
```

- Funzione che prende come argomento una funzione che prende in ingresso un double e ritorna un intero

Passaggio di puntatori a funzione

- Nella dichiarazione di una funzione che prende in ingresso il nome di una funzione si può omettere il nome dell'argomento:

- Es:

```
void sel_fun(int(*) (double)) ;
```

Passaggio di puntatori a funzione

- Nell'invocazione della funzione è necessario indicare come parametro il nome di una funzione precedentemente dichiarata

- Es:

```
int f1 (double) ;
```

```
int f2 (double) ;
```

```
...
```

```
sel_fun (f1) ;
```

```
sel_fun (f2) ;
```

- Notare che la funzione argomento è specificata senza i suoi parametri e senza parentesi tonde

Passaggio di puntatori a funzione

- Nell'invocazione non si possono passare altri parametri attraverso la funzione invocata

- Es. errato:

```
sel_fun(f2(67.89)) ; //invocazione errata!!
```

- Se volessi passare altri parametri alla funzione dovrei aggiungerli nella dichiarazione della funzione stessa:

```
void sel_fun(int(*pfun)(double), double r) ;
```

...

```
sel_fun(f2, 67.89) ;
```

Esercizio

- Programma che implementa una funzione per il calcolo della somma dei primi n valori di una generica funzione **int f(int)**:
somma(n,f) = f(1) + f(2) + f(3) + + f(n)
- Soluzione in *somma_funz.cc*