Funzioni: approfondimento

Lorenzo Porcelli

void

Una funzione che non ha parametri di attivazione ha void come parametro.

Una funzione che svolge un compito senza dover restituire nulla è di tipo void.

- int leggi(void) // legge un numero e lo restituisce
- void stampa(int x) // stampa x con messaggio
- void stampaTestata(void)

return

Una funzione restituisce il controllo al chiamante in tre modi diversi:

- return <valore>
- return;
- Quando si incontra la parentesi di chiusura della funzione

Gli ultimi due modi si possono utilizzare nel caso di funzione di tipo void

prototipo

Ogni funzione deve essere dichiarata prima dell'utilizzo.

La dichiarazione avviene implicitamente quando la funzione viene definita,

Oppure

Si può fare inserendo in testa al file tutte le dichiarazioni delle funzioni che verranno utilizzate in seguito.

prototipo

Un prototipo di funzione è la dichiarazione che indica al compilatore il tipo del dato restituito dalla funzione, il numero dei parametri di attivazione, il tipo dei parametri e l'ordine in cui questi sono attesi.

Il compilatore utilizzerà i prototipi per convalidare le chiamate di funzione

prototipo

- Associato ad ogni libreria c'è un file di testata, header file, <nome>.h, che contiene i prototipi di tutte le funzioni contenute nella libreria.

- #include <nomefile.h> è un comando per il precompilatore, che ordina di inserire il file indicato nel programma sorgente. Tutti i prototipi delle funzioni vengono così inseriti nel sorgente permettendo al compilatore di operare correttamente.

Regole di visibilità: esempio

```
void f( int x ) {
    int a;
    a=x;
}

int main() {
    int a=2, b=3;
    f(b);
    printf("%d", a);
    return 0;
}
```

Cosa viene stampato?

Se ci si ricorda che le variabili dichiarate internamente ad una funzione sono locali, la risposta è semplice …

Regole di visibilità: esempio

```
void f( int x ) {
    a=x;
}

int main() {
    int a=2, b=3;
    f(b);
    printf("%d", a);
    return 0;
}
```

Cosa viene stampato?

Se ci si ricorda che le variabili dichiarate internamente ad una funzione sono locali, a=x cosa significa?

visibilità

Le variabili dichiarate internamente ad un blocco si dicono **variabili locali**

Sono accessibili solo internamente al blocco in cui sono dichiarate.

```
int main() {
    int i=5;
    {
        int i=3, a=5;
        printf("%d", i);
    }
    printf("%d", i);
    printf("%d", a);
    return 0;
}
```

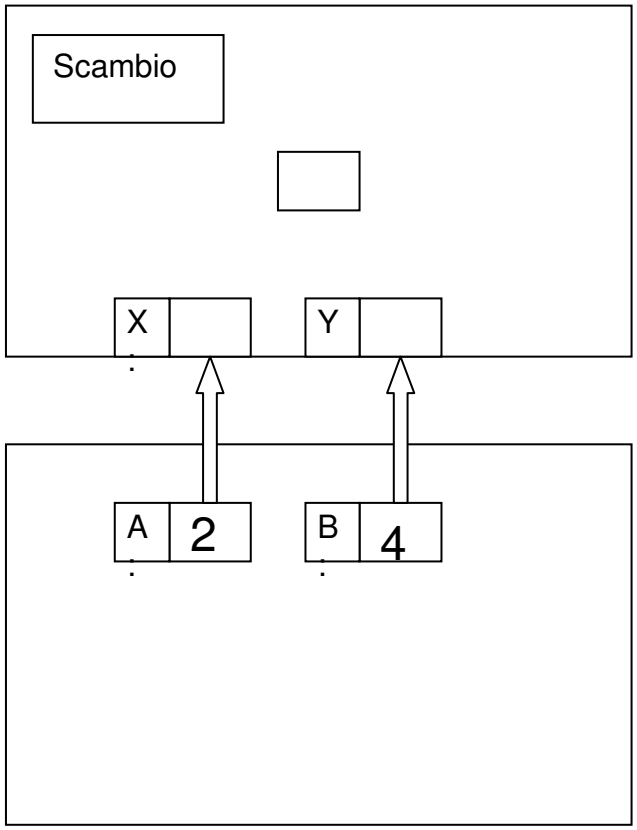
parametri

```
void scambio(int x, int y) {
    int z;
    z=x; x=y; y=z;
}
int main() {
    int x=2, y=5;
    scambio(x, y);
    printf("%d %d", x, y);
    return 0;
}
```

La funzione scambio scambia il contenuto delle due variabili x, y. Lo scambio avviene correttamente all'interno della funzione.

Cosa viene stampato?

parametri



Al momento della chiamata il contenuto di a e di b viene copiato nel parametro x e y.

Lo scambio avviene per x e y, mentre a e b rimangono inalterati.

Chiamata per valore

Si ha quando una funzione viene invocata passando una copia dei dati che servono alla funzione per essere attivata.

Ogni operazione che la funzione effettua sui parametri in ingresso non modifica il valore dei parametri originali.

Chiamata per valore: esempio

```
void f(int a, int b) {
    a=5; b=2;
}

int main( ) {
    int a=2; b=5;
    f(a, b);
    printf("%d %d", a, b);
    return 0;
}
```

Cosa viene stampato?

Ricordarsi che in c la chiamata di funzione avviene sempre per valore.

Chiamata per riferimento.

Una funzione può avere come parametro di ingresso l'indirizzo di memoria da utilizzare per eventuali operazioni.

Questo meccanismo di passaggio di parametri è la chiamata per riferimento.

E' utilizzato ad esempio nella funzione scanf, alla quale viene indicato l'indirizzo della variabile da utilizzare.

Risoluzione di un'equazione di 2° grado

Ob: determinare le soluzioni reali

Ris: x1, x2

Dati: a, b, c

Risolvere il problema utilizzando una funzione.

Come può una funzione restituire due valori?

E come può segnalare il fatto che l'equazione non ha soluzioni reali?

Equazione 2° grado

```
int radice( float *x1, float *x2, float a, float b, float c)  {
        // x1 e x2 contengono l'indirizzo di float.
   float delta = b*b - 4*a*c;

   if( delta < 0 ) return -2;
   if( a == 0 ) return -1;
   *x1 = (-b – sqrt(delta)) / (2*a);
   *x2 = (-b + sqrt(delta)) / (2*a);
   return 1;
}
```

Equazione 2° grado

```
int main( ) {
   float a, b, c, x1, x2;
   int err;

   scanf(“%f%f%f”, &a, &b, &c);
   err = radice(&x1, &x2, a, b, c);
   if( err == -2 ) printf(“Soluzioni complesse”);
   else if( err == -1) printf(“Non è eq 2° gr”);
   else printf(“x1 = %f x2 = %f”, x1, x2);
   return 0;
}
```

scambio

```
void scambio(int *x, int *y) {
    int z;
    z = *x; *x = *y; *y = z;
}

int main() {
    int x=2, y=5;
    scambio(&x, &y);
    printf("%d %d", x, y);

    return 0;
}
```

La funzione scambio è attivata con due indirizzi, quello di x e di y.

Operando sugli originali scambia i due valori.

Quando avviene la stampa i dati di x e y risulteranno scambiati.