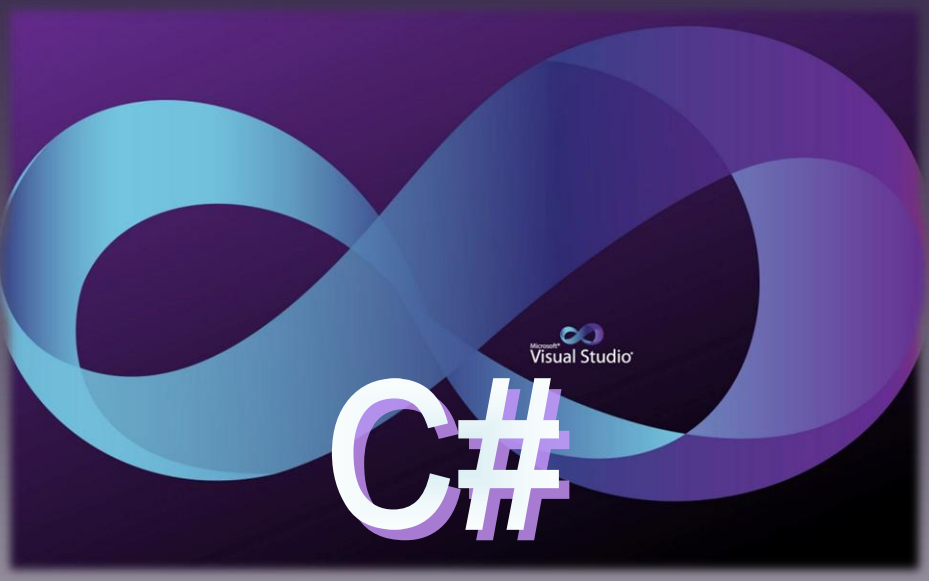
ISTITUTO TECNICO INDUSTRIALE
G. M. ANGIOY
SASSARI



CORSO DI PROGRAMMAZIONE

PROGETTAZIONE DEI METODI

DISPENSA 07.02

07-02_Metodi_Progettazione_[ver_15]

Versione del: 07/11/2015
Revisione numero: 15

Prof. Andrea Zoccheddu
Dipartimento di Informatica

DIPARTIMENTO
INFORMATICA E TELECOMUNICAZIONI

PROGETTAZIONE DI METODI

UNA RIFLESSIONE SU COME PROGETTARE I METODI

STUDIO DI FUNZIONE

In questa dispensa si vuole fornire alcuni elementi di studio per riuscire a costruire più facilmente e efficacemente una funzione. Per farlo si propone una rappresentazione grafica, che chiameremo "scatola".

DISEGNO DELLA SCATOLA DI FUNZIONE

Quando si progetta una funzione occorre porsi le seguenti domande:

1) Quale è lo scopo della funzione e che nome posso scegliere?
2) La funzione ha bisogno di informazioni, prima di iniziare l'elaborazione?
3) Se ci sono informazioni prese dall'esterno, quali vanno modificate al ritorno al chiamante?
4) La funzione deve rendere un valore? Deve rendere molti valori?

Quando si è in grado di rispondere a tutte le domande precedenti si può rappresentare graficamente la funzione con uno schema simile al seguente:

1) Il nome della funzione si scrive dentro il rettangolo, ed esprime lo scopo prefissato.
2) Se la funzione ha necessità di acquisire valori dal chiamante, allora la funzione ha dei parametri
3) Se la funzione deve modificare un valore passato dal chiamante, allora quel parametro è passato per riferimento

4a) Se la funzione deve restituire un solo valore, allora la funzione è tipizzata altrimenti è void

4b) Se la funzione deve restituire più di un solo valore, allora la funzione richiede dei parametri per risultato

Vediamo adesso dei casi concreti per comprendere meglio questo sistema di progettazione.

ESEMPI

CALCOLO DEL VALOR MEDIO

Si desidera definire una funzione che calcoli la media di due numeri interi positivi.

Analisi del problema

1) Lo scopo della funzione è rendere una media, quindi il suo nome può essere Media.

2) La funzione non può lavorare senza sapere su quali valori deve fare il calcolo; quindi ha bisogno di due dati di tipo intero prima di iniziare l'elaborazione: questi sono i parametri.

3) I dati presi dal chiamante saranno usati per il calcolo, ma non vanno modificati: quindi i parametri non sono passati per riferimento, ma per valore.

4) La funzione deve rendere un solo valore, di tipo double: quindi la funzione è di tipo double.

Il disegno dello schema sarà il seguente:

Definizione di funzione

```
public double Media ( int a , int b )
{
      //corpo della funzione, un return è obbligatorio
}
```

FUNZIONE CHE CREA UN VETTORE DI INTERI CASUALI

Si desidera definire una funzione che crea un vettore di interi con una dimensione stabilita dal chiamante e che inizializzi i valori delle sue celle, ponendo in esse numeri casuali compresi tra 10 e 100.

Analisi del problema

1) Lo scopo della funzione è rendere un vettore, quindi il suo nome può essere CreaVettore.
2) La funzione ha necessità della dimensione decisa dal chiamante; quindi ha bisogno di un numero intero prima di iniziare l'elaborazione: questo è il parametro.
3) Il dato passato dal chiamante indica la dimensione, ma non deve essere modificato: quindi il parametro della dimensione non è passato per riferimento, ma per valore.
4) La funzione deve rendere un solo valore, un vettore di interi: quindi la funzione è di tipo int[ ].

Il disegno dello schema sarà il seguente:

Definizione di funzione

```
public int [ ] CreaVettore ( int a )
{
      //corpo della funzione, un return è obbligatorio
}
```

Analisi alternativa del problema

Lo scopo della funzione è rendere un vettore, ma un valore restituito può essere reso anche mediante un parametro per risultato, quindi posso evitare di rendere un tipo nella funzione e usare un parametro out. Il disegno dello schema sarà il seguente:

Definizione alternativa di funzione

```
public void CreaVettore ( int a , out int[] v )
{
      //corpo della funzione, è obbligatorio inizializzare con una new il
vettore v
}
```

FUNZIONE CHE ORDINA UN VETTORE DI INTERI

Si vuole definire una funzione che, ricevuto un vettore di interi, lo ordina in modo crescente.

Analisi del problema

1) Lo scopo della funzione è ordinare un vettore, quindi il suo nome può essere OrdinaVettore.
2) La funzione deve ricevere il vettore dal chiamante.
3) Il vettore passato dal chiamante deve essere ordinato, quindi modificato: è un parametro per riferimento.
4) La funzione non deve rendere alcun valore: quindi la funzione è di tipo void.

Il disegno dello schema sarà il seguente:

Definizione di funzione

```
public void OrdinaVettore ( ref int[] v )
{
      //corpo della funzione, un return NON è obbligatorio
}
```

FUNZIONE CHE CALCOLA IL VALORE MINIMO IN UN VETTORE DI INTERI

Si vuole definire una funzione che, ricevuto un vettore di interi, ricerca e rende il valore minimo tra i suoi valori.

Analisi del problema

1) Lo scopo della funzione è cercare il valore minimo, quindi il suo nome può essere Minimo.
2) La funzione deve ricevere dal chiamante il vettore in cui compiere la ricerca.
3) Il vettore passato dal chiamante non deve essere modificato: è un parametro per valore.
4) La funzione deve rendere un intero: quindi la funzione è di tipo int.

Il disegno dello schema sarà il seguente:

Definizione di funzione

```
public int Minimo ( int[] v )
{
      //corpo della funzione, un return è obbligatorio
}
```

FUNZIONE CHE VERIFICA SE DUE VETTORI DI INTERI SONO UGUALI

Si vuole definire una funzione che, ricevuti due vettori di interi, controlla se hanno la stessa dimensione e lo stesso contenuto nello stesso ordine; quindi rende un valore booleano true se sono uguali, altrimenti false

Analisi del problema

1) Lo scopo della funzione è controllare se due vettori sono uguali, quindi il suo nome può essere Uguali.

2) La funzione deve ricevere dal chiamante i due vettori da comparare.

3) Entrambi i vettori non vanno modificati, quindi sono passati per valore.

4) La funzione deve rendere un booleano.

Il disegno dello schema sarà il seguente:

Definizione di funzione

```
public bool Ugual( int[] v , int[] w )
{
     //corpo della funzione, un return è obbligatorio
}
```

VARIABILI LOCALI E LORO VITA

VISIBILITÀ DELLA VARIABILE E CICLO DI VITA

È importante considerare adesso le similitudini e le differenze che passano tra i parametri e le variabili locali.

Entrambi questi elementi sono locali alla funzione, cioè esistono e sono utilizzabili solo all'interno della funzione; è indispensabile osservare che questi nomi non esistono e non sono utilizzabili all'esterno della funzione.

Tuttavia lo scopo dei due elementi è completamente diversa.

Lo scopo dei parametri è realizzare una comunicazione tra la funzione e il programma chiamante. I parametri passati per valore solitamente realizzano una comunicazione dal chiamante verso la funzione (nel disegno è una freccia in entrata nel rettangolo) ovvero è un dato che la funzione deve richiedere al programma chiamante. I parametri passati per risultato solitamente realizzano una comunicazione dalla funzione verso il programma chiamante (nel disegno è una freccia in uscita dal rettangolo) ovvero è un dato che la funzione deve rendere al programma chiamante. I parametri passati per riferimento solitamente realizzano una comunicazione bidirezionale tra la funzione e il programma chiamante (nel disegno è una freccia con due sensi) ovvero è un dato che il programma chiamante cede alla funzione e quest'ultima lo rende modificato al programma chiamante.

Lo scopo delle variabili locali è solo quello di ospitare dati temporanei per l'elaborazione all'interno della funzione; quando la funzione termina, il loro scopo è esaurito e possono essere distrutti (ovvero si può liberare la memoria del sistema occupata da queste variabili).

Consideriamo il seguente esempio:

```
public void Scambio (ref int a, ref
int b)
{
      int tmp = a;
      a = b;
      b = tmp;
}
```

```
//button1 → gestore dell'evento
Click
int x = 3;
int y = 5;
Scambio ( x , y );
label1.Text = Convert.ToString (x);
label2.Text = Convert.ToString (y);
```

E facciamo le alcune riflessioni:

- **Parametri a e b**: questi nomi sono dichiarati come interni alla funzione Scambio; se usassi i nomi a e b nello spazio del chiamante (il gestore Click del button1) avrei un errore; potrei anche dichiarare identificatori omonimi nello spazio del chiamante, ma sarebbero locazioni diverse e distinte, ciascuna col suo spazio di memoria, coi suoi dati, col suo tipo associato, ecc.
- **Parametri a e b**: questi parametri consentono una comunicazione tra il chiamante e la funzione; nell'esempio proposto la comunicazione avviene proponendo le variabili x ed y come argomenti nella invocazione della funzione Scambio; in questo modo a e b permettono al chiamante di informare la funzione su quali siano i valori concreti con cui lavorare e eseguire il suo compito.
- **Variabile locale tmp**: anche questo nome è dichiarato interno alla funzione Scambio; se usassi il nome tmp nello spazio del chiamante (il gestore Click del button1) avrei comunque un errore; potrei anche dichiarare identificatori omonimi nello spazio del chiamante, ma sarebbero locazioni diverse e distinte, ciascuna col suo spazio di memoria, coi suoi dati, col suo tipo associato, ecc.
- **Variabile locale tmp**: la variabile serve per ospitare temporaneamente i dati mentre la funzione esegue la sua elaborazione; la variabile in questo esempio serve solo come appoggio per consentire lo scambio tra a e b, ma non serve che il suo valore sia reso noto all'esterno. Spesso le variabili temporanee servono come contatori, accumulatori, scambi o indici di cicli, o per memorizzare risposte ancora non complete.

ESERCIZI

ESERCIZI DI TEORIA

I seguenti esercizi prevedono che lo studente

1) disegni lo schema della funzione "a scatola" con le frecce in entrata e in uscita;
2) proponga la firma della funzione che indica tipo della funzione, nome e parametri, inclusi i loro modi di passaggio;
3) realizzi il corpo della funzione, indicando con chiarezza se e quali siano le variabili locali della funzione

ESERCIZI

A) Dati tre numeri rendere il valore minore

B) Dato un vettore di interi, restituirne una copia

C) Dati due vettori, restituire il vettore somma

D) Dato un numero intero ed un vettore di interi, restituire un valore booleano che indica se il numero è contenuto nel vettore

E) Dato un numero intero ed un vettore di interi, restituire l'indice della posizione in cui si trova (se c'è) altrimenti rendere -1 (se non c'è)

F) Creare un vettore con valori casuali (parametro out)

G) Creare un vettore con valori casuali (restituito come funzione)

H) Creare una matrice con valori casuali (parametro out)

I) Creare una matrice con valori casuali (restituito come funzione)

J) Creare una matrice identità

K) Verificare se due vettori sono uguali

L) Controllare se due vettori hanno almeno un valore in comune (anche in posizioni diverse)

M) Controllare se un vettore ha un valore ripetuto

N) Verificare se due matrici 100x100 sono uguali

O) Controllare se due matrici 100x100 hanno almeno un valore in comune (anche in posizioni diverse)

P) Controllare se una matrice 100x100 ha un valore ripetuto

Q) Trasporre una matrice

R) Creare la copia di un vettore

S) Creare la copia di una matrice

ESERCIZI PRATICI

Definire le seguenti funzioni e predisporre un interfaccia visuale per invocarle e eventualmente visualizzare il risultato:

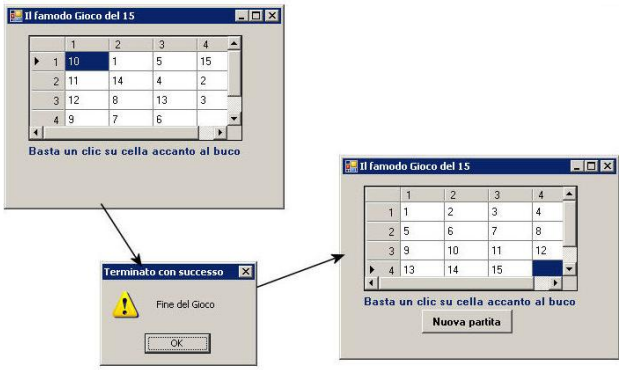
1) Dati tre numeri restituire la media dei due maggiori
2) Dati tre numeri restituire quello "in mezzo"
3) Un numero è perfetto se è uguale alla somma dei suoi divisori (es. 6 è perfetto): definire una funzione che dice se un numero è perfetto
4) Un numero è primo se è divisibile solo per sé stesso e per 1: definire una funzione che dice se un numero è primo
5) Il fattoriale di un numero è quel numero moltiplicato per i suoi predecessori: definire una funzione che restituisce il fattoriale di un numero
6) Definire una funzione che crea e rende un vettore di numeri con la virgola; la dimensione del vettore è stabilita dal chiamante; il vettore è inizializzato con valori compresi tra 0 e 1
7) Definire una funzione che "fonde" insieme due vettori di interi, restituendo un nuovo vettore ottenuto da tutti i valori dei primi due
8) Definire una funzione che "inverte" un vettore di interi, scambiando il primo valore con l'ultimo, il secondo col penultimo e così via
9) Definire una (sola) funzione che dato un vettore, rende i valori minimo e massimo contenuti in esso
10) Definire una funzione che data una matrice e un numero rende il vettore formato dalla colonna corrispondente al numero dato
11) Definire una funzione che data una matrice, rende le coordinate della cella che contiene il suo valore minimo
12) Definire una funzione che data una matrice, rende le coordinate delle celle che contengono i suoi valori minimo e massimo
13) Definire una funzione che data una matrice, verifica se contiene due valori identici ma in celle diverse

ESERCIZIO 1. GIOCO DEL QUINDICI

Il gioco riguarda (un solo) giocatore contro il computer. Il programma coinsente di spostare un numero nella cella vuota; la cella dove c'era il numero diventa vuota

Il gioco è vinto quando tutti i numeri sono in fila.

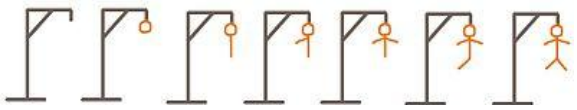
Il gioco è perso quando il giocatore rinuncia e chiude il programma.



ESERCIZIO 2. GIOCO DELL'IMPICCATO

Il computer crea una forca e una griglia di 12 etichette e una casella a discesa con le 26 lettere inglesi; il computer sceglie una parola nascosta di 12 lettere. Il giocatore tenta la scelta di una lettera mediante la casella a discesa: la lettera viene eliminata dall'elenco disponibile. Se la lettera è presente nella parola nascosta allora la si mostra nelle etichette e nelle opportune posizioni; altrimenti sotto la forca si aggiunge un elemento di disegno all'omino impiccato.

Il gioco termina con successo se il giocatore indovina tutte le lettere che formano la parola; il gioco termina con fallimento se si completa l'omino.



SOMMARIO