Funzioni e procedure

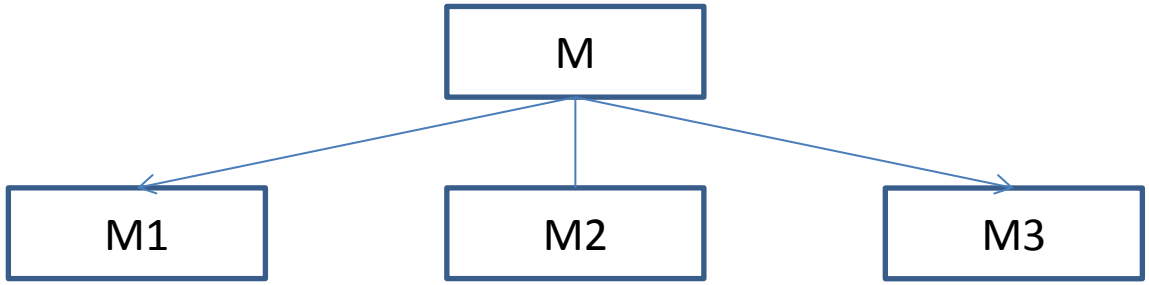
Contenuti rieditati delle slide della Prof. L. Caponetti

Metodologia Top-Down

- Mediante la metodologia top-down la risoluzione di un problema si basa sulla decomposizione del problema in **sotto-problemi**
- Si considera il problema iniziale e si riduce in piccole «parti» più facili da gestire
- La soluzione di un sotto-problema può essere codificata in un **sottoprogramma**

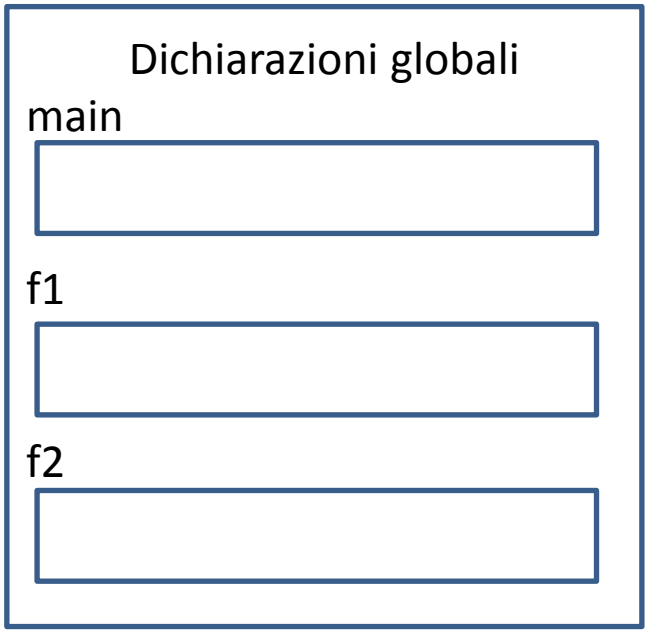
Struttura di un programma

- Un programma si puo' organizzare in un programma principale e un insieme di sottoprogrammi
- Ad esempio M è il programma principale e M1, M2, M3 sono 3 sottoprogrammi



Struttura di un programma in C

Programma costituito da una sezione di dichiarazione globale, dalla function main e dalle funzioni f1 e f2



Struttura di un programma

- C’è una analogia tra la **scomposizione di un problema e la struttura di un programma**.
- Intuitivamente si può affermare che la soluzione di ciascun sotto-problema può essere codificata in un sottoprogramma

Sottoprogrammi

- I linguaggi di programmazione forniscono dei costrutti sintattici per:
  - **Creare** delle unità di programma – **sottoprogrammi** – dando un nome ad un gruppo di istruzioni
  - **Attivare** tali unità con **modalità di comunicazione stabilite** con l'unità chiamante

Caratteristiche di un sottoprogramma

- Un sottoprogramma è identificato da un **nome**
- Un **sottoprogramma** è costituito da istruzioni – generalmente gli stessi costrutti di un programma **dichiarazioni** e **blocco delle istruzioni eseguibili**

Attivazione o chiamata di un sottoprogramma

- Un sottoprogramma va in esecuzione solo se è richiesta la sua **attivazione** da parte di un'altra unità di programma (per la quale il sottoprogramma risulti visibile)
- L'attivazione viene richiesta utilizzando il **nome del sottoprogramma**; tale nome è utilizzato ogni volta che si vuole che il blocco delle istruzioni sia eseguito
- I sottoprogrammi possono essere **interni** o **esterni** all'unità che li utilizza

Parametri di un sottoprogramma

- Un sottoprogramma comunica i dati con l'unità chiamante attraverso i **parametri**
- Il nome del sottoprogramma e l'elenco dei parametri sono dichiarati esplicitamente nella **intestazione dei sottoprogrammi – cioè la prima istruzione del sottoprogramma**

Parametri di un sottoprogramma

- Consentono di rappresentare i dati di comunicazione tra il sottoprogramma e il programma che lo attiva, cioè i **dati di ingresso e di uscita** del sottoprogramma
- Ogni **parametro** deve essere individuato da un **nome** e deve avere un **tipo**
- La **lista dei parametri** è definita in modo esplicito nella intestazione del sottoprogramma

Attivazione di un sottoprogramma

Al momento dell'attivazione:

- **Viene sospesa** l'esecuzione dell'unità che contiene la richiesta di attivazione ed il **controllo** passa al sottoprogramma attivato
- **Alla fine dell'esecuzione** l'attivazione termina ed il controllo ritorna all'unità chiamante

Astrazione funzionale e procedurale

- Mediante i sottoprogrammi è possibile ampliare l'insieme degli **operatori** e delle **istruzioni** disponibili in un linguaggio di programmazione
- I sottoprogrammi sono di due tipi:
  - **Funzioni** mediante le quali è possibile introdurre nuovi operatori (**astrazione funzionale** o **astrazione della nozione di operatore)**
  - **Procedure** mediante le quali è possibile introdurre nuove istruzioni (**astrazione procedurale** o **astrazione della nozione di istruzione**)

Funzioni matematiche

- Un sottoprogramma di tipo funzione consente di costruire una funzione matematica
- Una funzione matematica f associa d un valore del dominio D dell'argomento della funzione un valore del dominio del risultato – **codominio** – C
- Ad esempio sia f la funzione radice quadrata

  f: insieme dei reali → insieme dei reali

  Dominio Codominio

Sottoprogrammi - funzioni

- In un linguaggio di programmazione una funzione f può essere vista come un operatore che opera su **operandi** – parametri in input e produce un **risultato** o valore di output

Ai parametri di input e al valore di output deve essere associato un **tipo**. Il valore di output è necessariamente **scalare**

Attivazione di una funzione

- Una funzione viene attivata o chiamata inserendo il nome della funzione e la lista dei parametri in una espressione dello stesso tipo della funzione

- Per attivare una funzione è sufficiente conosce il nome della funzione e la lista di parametri – cioè una funzione si può vedere come una scatola nera della quale si conoscono solo i dati di ingresso e il risultato

Sottoprogrammi – astrazione funzionale

- Una funzione f può essere vista come un **operatore** che opera su tipi di dati primitivi o definiti dal programmatore
- Con l'introduzione di una funzione viene ampliato l'insieme degli operatori del linguaggio utilizzato

Sottoprogrammi – Linguaggio C

- Il linguaggio C fornisce il costrutto **function** per realizzare sia le funzioni che le procedure
- Le differenze consistono essenzialmente nel:
  - Modo di comunicare i dati di uscita
  - Modo di chiamare il sottoprogramma

Funzioni in C

- Per poter utilizzare una funzione è necessario:
  - **definire** il sottoprogramma, cioè scrivere in C tutte le istruzioni necessarie
  - **dichiarare** il sottoprogramma nella parte dichiarativa del programma

Definizione di una funzione

- La definizione di una funzione è costituita da:
  - Una **intestazione**
  - Un **blocco di dichiarazioni ed istruzioni**, cioè:
    - Una sezione di dichiarazioni di costanti, tipi e variabili
    - Una sezione di istruzioni

Definizione di una funzione - sintassi

- **Una intestazione**
- **Un blocco di dichiarazioni ed istruzioni**

<tipo> <identificatore> (<lista parametri>)
{ <dichiarazioni locali>
    < istruzioni>
}

Blocco delle dichiarazioni ed istruzioni

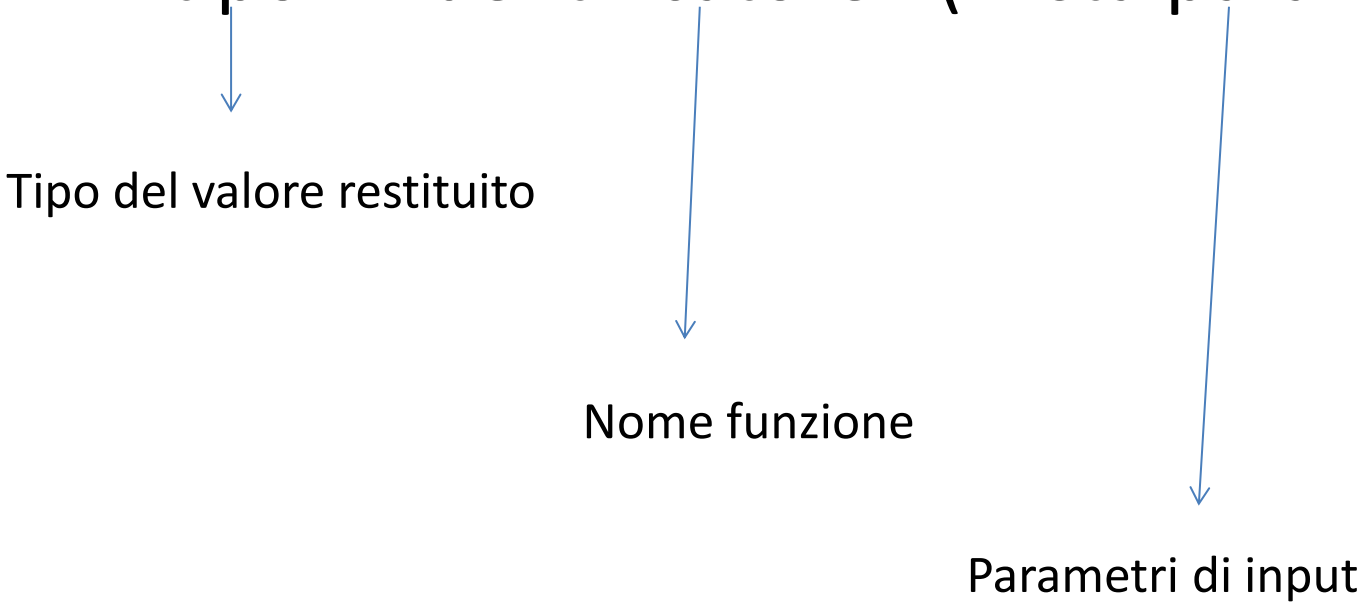
- Dopo l'intestazione di una funzione vi sono:
  - Le **dichiarazioni locali** cioè le dichiarazioni di tutte le risorse (costanti, tipi, variabili, funzioni) necessarie
  - Il **blocco delle istruzioni**

- L'istruzione **return** per restituire il valore calcolato da una espressione

return <espressione>

Intestazione di una funzione

<tipo> <identificatore> (<lista parametri>)



Lista parametri formali

- Ogni parametro della lista è definito come tipo e identificatore

  <tipo> <identificatore>

- I parametri sono separati da virgole
- Esempi di intestazione

  int abs(int x)

  float somma(int a, int b)

Attivazione di una funzione

- Una funzione viene attivata o chiamata **inserendo il nome della funzione e la lista dei parametri in una espressione** dello stesso tipo della funzione

<identificatore funzione>(<lista parametri attuali>)

Lista parametri attuali

- I **parametri attuali** indicano i **valori** degli argomenti rispetto ai quali la funzione deve essere calcolata
- Un parametro attuale può essere:
  - Una costante
  - Una variabile
  - Una espressione
  - Una chiamata ad una funzione

Esempi di attivazione

float somma(int a, int b)

- Esempi di chiamata alla funzione somma

  z = somma(a, b)

  z = somma(2, 5*b)

  z = c + somma(a, b)

  y = somma(x, abs(z))

  printf(”%f”, somma(a,b))

Attivazione di una funzione

- L’attivazione di una funzione può essere inserita dovunque possa essere inserito un **operatore** sul tipo del parametro della funzione

Dichiarazione di una funzione - prototipi

- Una funzione deve essere dichiarata prima del suo utilizzo. La dichiarazione è costituita dal suo **prototipo**
- Il prototipo è uguale alla intestazione della funzione

  <tipo><identificatore>(<lista parametri>)
- Nel prototipo gli identificatori dei parametri sono opzionali, si può quindi avere

  <tipo><identificatore>(<lista tipi>)

Prototipi

- Il **prototipo** di una funzione deve essere inserito nella parte dichiarativa del programma, prima del codice che utilizza la funzione stessa
- In tal modo il compilatore può facilmente controllare il numero ed il tipo dei parametri di input ed il tipo del valore restituito