

Strutturare il codice: sottoprogrammi

Andrea Marin

Università Ca' Foscari Venezia
Laurea in Informatica
Corso di Programmazione part-time

a.a. 2011/2012

Motivazioni

- ▶ Alcuni problemi si presentano frequentemente durante lo sviluppo di un programma
- ▶ Esempi:
 - ▶ Un programma che consente di fare operazioni con le frazioni userà di frequente le funzioni `mcm` e `mcd`
 - ▶ Riscrivere ogni volta il codice per calcolarlo ha molti inconvenienti:
 - ▶ Scarsa leggibilità
 - ▶ Difficoltà nel mantenere il codice
 - ▶ Cattiva progettazione (monolitica vs. modulare)



Divide et Impera

- ▶ Progettazione top-down
 - ▶ la progettazione inizia specificando parti complesse e suddividendole successivamente in parti pi piccole
 - ▶ esempio: scrivere un programma che sommi due frazioni
 - ▶ Calcola il minimo comun denominatore
 - ▶ Aggiusta i numeratori
 - ▶ Somma i numeratori
 - ▶ Riduci ai minimi termini
 - ▶ I problemi di calcolo del denominatore comune e di riduzione ai minimi termini sono da risolvere a parte
- ▶ L'uso dei sottoprogrammi facilita questo approccio
 - ▶ Informalmente un sottoprogramma è una porzione di codice che può essere eseguito a seguito di una chiamata da qualsiasi punto del programma. Al termine della sua esecuzione il programma prosegue a partire dall'istruzione successiva alla chiamata

Ambienti

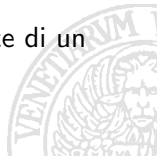
Ambiente

L'ambiente è l'insieme di tutte le associazioni tra identificatori e locazioni di memoria.

Visibilità

La visibilità di un identificatore è l'insieme delle posizioni nel codice dove quell'identificatore può essere utilizzato.

- ▶ Le dichiarazioni hanno lo scopo di modificare l'ambiente di un programma



Variabili globali e locali

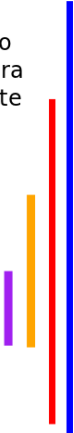
- ▶ In C gli identificatori sono visibili dal punto in cui sono dichiarate, in poi
- ▶ Le variabili dichiarate al di fuori di qualsiasi blocco prendono il nome di **variabili globali**
- ▶ Le variabili dichiarate all'interno di un blocco prendono il nome di **variabili locali**
 - ▶ Le variabili locali vengono deallocate non appena termina il blocco in cui sono state dichiarate
 - ▶ Se una variabile locale prende lo stesso nome di una variabile precedentemente dichiarata, l'identificatore farà riferimento all'ultima dichiarazione



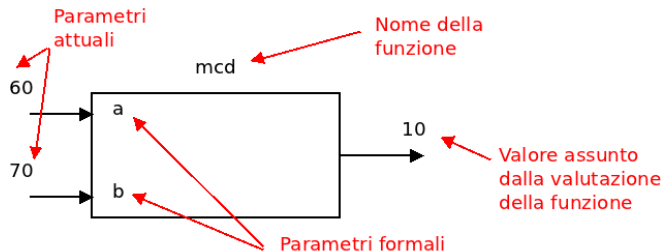
Esempio: cosa stampa?

```
int x;  
int main(){  
    x=3;  
    int y;  
    y=x+3;  
    {  
        int a;  
        a=6;  
        int y;  
        y=2*a;  
        x = y;  
    }  
    printf("%d %d", x, y);  
    return 0;  
}
```

y fa riferimento
a quello dichiara
to più di recente



Funzioni come black-box



```
int mcd(int a, int b);
```

← Firma della funzione



Massimo comun divisore di tre numeri

```
#include <stdio.h>

/*Dichiarazione della funzione*/
int mcd(int a, int b);

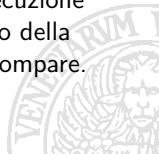
int x, y, z;
int m1, m2;

int main() {
    scanf("%d", &x);
    scanf("%d", &y);
    scanf("%d", &z);
    m1 = mcd(x, y);
    m2 = mcd(m1, z);
    printf("Il mcd e %d", m2);
    return 0;
}
```



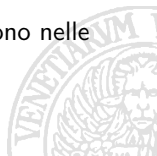
Chiamata ad una funzione

- ▶ Quando in un'espressione compare il nome della funzione essa può essere valutata
 1. Si istanzia l'ambiente della funzione
 - ▶ Parametri formali
 - ▶ Variabili locali
 2. I **valori** dei parametri attuali (che sono **espressioni**) vengono **copiati** nei parametri formali appena allocati
 3. Si esegue il corpo della funzione
 4. L'istruzione `return exp` causa la terminazione dell'esecuzione della funzione e il valore di `exp` viene sostituito al posto della chiamata a funzione nell'espressione dove la funzione compare.



Alcune osservazioni

- ▶ Tipi dei parametri attuali e formali (condizioni sufficienti per la compilazione)
 - ▶ L'espressione che fa da parametro attuale ha lo stesso tipo del parametro formale
 - ▶ Il valore restituito dal `return` ha lo stesso tipo specificato come tipo restituito dalla funzione
 - ▶ L'abbinamento tra parametri formali ed attuali avviene **per posizione** e non per nome
 - ▶ Le modifiche ai parametri formali fatte all'interno della funzione non si ripercuotono sulle variabili che compaiono nelle espressioni corrispondenti ai parametri attuali



Esempio di definizione di una funzione

```
/*calcola mcd tra a e b*/  
/*Condizioni: a>0 e b>0*/  
int mcd(int a, int b) {  
    while (a != b) {  
        if (a > b) {  
            a = a - b;  
        }  
        else {  
            b = b - a;  
        }  
    }  
    return a;  
}
```



Esercizi

1. Scrivere una funzione che calcoli la radice quadrata intera di un numero intero positivo approssimata per difetto (per i calcoli, usare solo gli operatori di $+$ e $*$)
2. Scrivere una funzione che calcoli il minimo comune multiplo tra due valori interi positivi
3. Scrivere una funzione che dato un intero positivo **decida** se è primo (le funzioni di decisione restituiscono un booleano (intero) che è true se la proprietà è verificata, false altrimenti).
4. Per ciascuno dei precedenti punti, scrivere un programma principale di prova

