

Unità Didattica 3

Linguaggio C

Generalità sulle Funzioni. Variabili locali e globali. Passaggio di parametri per valore.

Funzioni

- Generalizzazione del concetto di funzione algebrica:
 - legge che associa a valori delle variabili in ingresso valori delle variabili in uscita o, più in generale, azioni.
- Raggruppano operazioni che possono essere riutilizzate usando il nome della funzione e i suoi parametri (*chiamata della funzione*), senza preoccuparsi dell'aspetto implementativo;
- I programmi in C combinano funzioni definite dal programmatore con funzioni di libreria standard;
- Una funzione deve essere definita (*definizione*) e dichiarata (*prototipo*), prima della definizione, relativamente al nome e ai tipi dei parametri in ingresso e in uscita.

Parametri alle funzioni

- Una funzione può avere dei parametri in ingresso e un parametro di uscita (tale fatto non rappresenta una limitazione perché si può restituire anche una struttura dati complessa);
- Se non si vuol restituire alcun valore da una funzione è sufficiente dichiararla di tipo *void* ed omettere o meno il `return`;
- E' obbligatorio mettere le parentesi `()` (oppure `() (void)`) dopo il nome della funzione anche se non ci sono parametri

Parametri alle funzioni

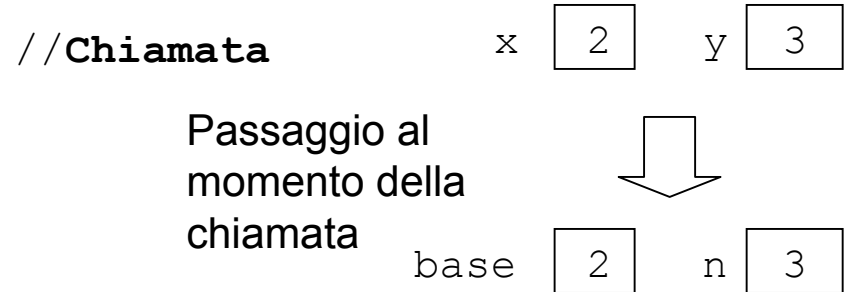
- In C i parametri alle funzioni sono sempre passati per valore;
- Al momento della chiamata, le funzioni allocano (riservano) memoria per le variabili d'ingresso;
- In ciascuna variabile viene copiato il valore che le viene passato al momento della chiamata.

Esempio di passaggio di parametri: Calcolo della potenza di un numero

```
#include <stdio.h>
int potenza(int, int);           //Prototipo
```

```
void main()
{
    int x, y, z;
    x = 2; y = 3;
    z = potenza(x, y);
}
```

```
//Definizione
int potenza(int base, int n)
{
    int i, p = 1;
    for(i = 0; i < n; i++)
        p = p * base;
    return (p);
}
```



Esempio di passaggio di parametri in funzioni ricorsive

```
#include <stdio.h>
long int fattoriale(int);           // Prototipo

void main()
{
    long int fatt;
    int n;
    scanf("%d", &n);
    fatt = fattoriale(n);
}

long int fattoriale(int n)
{
    if (n == 0)
        return 1;
    return (n * fattoriale(n-1));
}
```

Struttura di un programma C

/ commenti: nome programma, descrizione, etc. */*

#istruzioni per il preprocessore

dichiarazione di tipi, variabili, costanti;

prototipi delle funzioni;

tipo_di_ritorno **main** (*lista_argomenti*)

{

dichiarazione variabili locali

sequenza di istruzioni

}

tipo_di_ritorno **funzione_1** (*lista_argomenti*)

{

dichiarazione variabili locali

sequenza di istruzioni

}

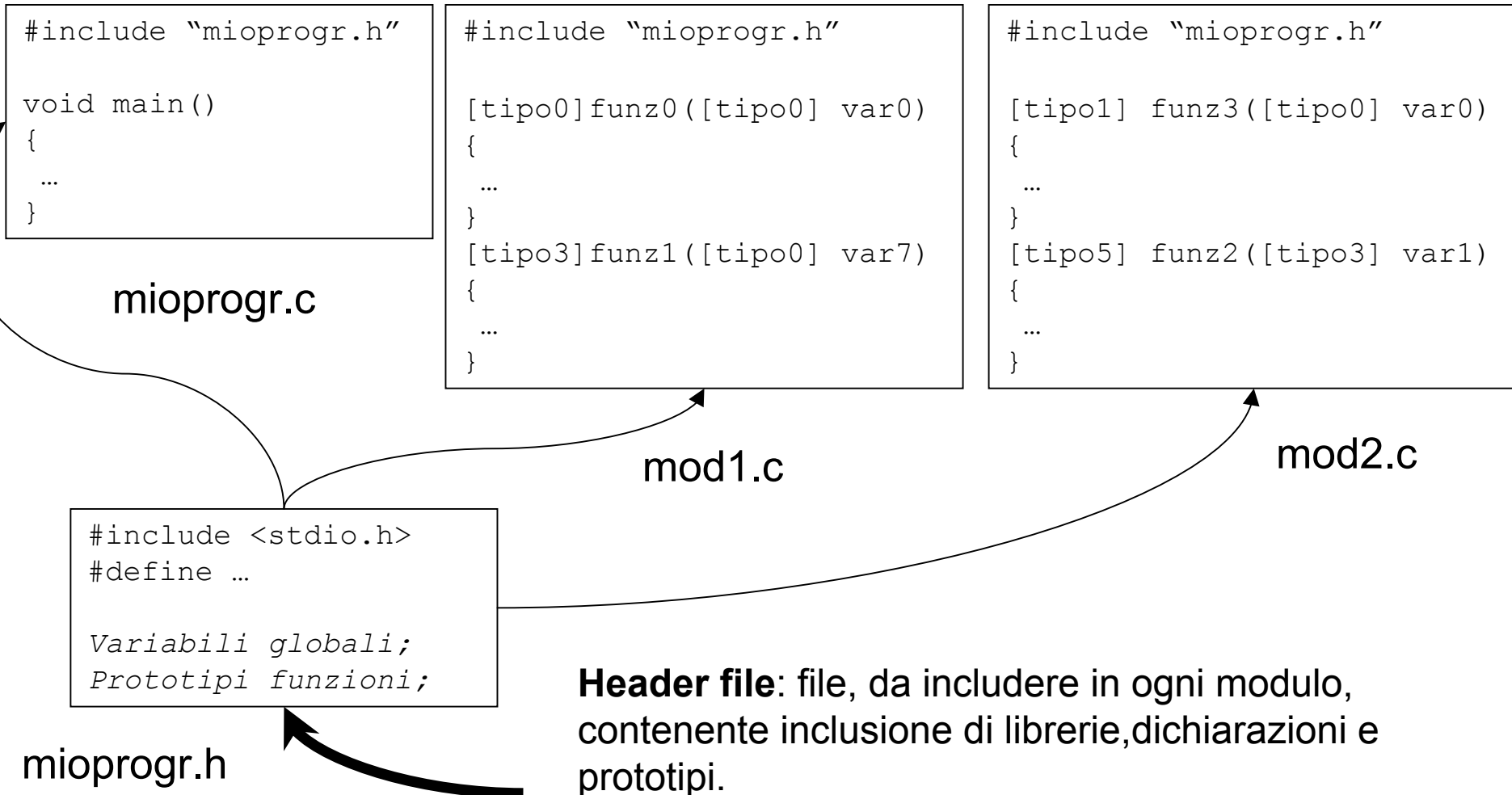
tipo_di_ritorno **funzione_n** (*lista_argomenti*)

{ ... }

Scrittura di un programma C su più moduli (file)

- Quando si scrivono programmi di grosse dimensioni è consigliabile suddividere i programmi in moduli separati. La funzione `main()` sarà contenuta in un solo file (es: “mioprogr.c”);
- E’ buona norma concentrare inclusioni di librerie standard, dichiarazioni e prototipi necessari ad un modulo, in un unico file (*header file*) da includere in tale modulo (es: `#include “mioprogr.h”`);
- Esiste un compromesso fra il desiderio che ogni modulo acceda solo alle informazioni di cui ha bisogno (N moduli => N header file) e la realtà pratica secondo la quale è difficile gestire molti header file;
- Per programmi di dimensioni ridotte è meglio avere un unico header file contenente le informazioni da condividere fra due punti qualunque del programma scritto su più moduli e includere in essi l’unico header file creato;

Scrittura di un programma C su più moduli e un unico header file



Classi di Memorizzazione delle Variabili

- Variabili locali (automatiche);
- Variabili globali (esterne);

Variabili Locali (o automatiche)

- Parola chiave `auto` o nessuna dichiarazione;
- Sono interne ad un blocco individuato da `{...}` (es: interne ad una funzione);
- Scope (ambito di visibilità):
visibili (accessibili) solo all'interno del blocco di definizione;
- Ciclo di vita:
create al momento della dichiarazione, cessano di esistere quando si esce dal blocco;
- Sono inizializzate con valori casuali.

Esempi di dichiarazione di variabili automatiche

```
#include <stdio.h>

void main(void)
{
    int i, j;
    ...
    funz();
}

void funz()
{
    int i, j;    //Sono variabili automatiche, ovvero
                //locali alla funzione. Fuori dalla
                //funz non sono più accessibili, cessano
                //di esistere (per esempio le due
                //variabili "i, j" definite nel "main"
                //sono del tutto scorrelate da queste).
}
```

Modifica del ciclo di vita di variabili locali

- E' possibile modificare il ciclo di vita di una variabile locale in modo che conservi il suo valore fra l'uscita e il successivo rientro nella funzione o nel blocco nel quale è stata dichiarata;
- Tale modifica si ottiene antepoendo alle variabili locali la parola chiave `static`:

```
void funz()  
{  
    static int i, j;  
        //Sono variabili automatiche, ovvero  
        //locali alla funzione. Fuori dalla  
        //funz non sono più accessibili ma al  
        //rientro nella funzione riacquistano  
        //il valore che avevano al momento  
        //dell'uscita dalla funzione stessa.  
}
```

Variabili Globali (o esterne)

- Sono dichiarate una sola volta esternamente a qualsiasi blocco;
- Devono essere rese note alle funzioni o ad altri moduli tramite la parola chiave `extern` (eccetto il caso in cui la dichiarazione preceda la funzione o il blocco in uno stesso modulo (es: all'inizio di un modulo, come avviene nella pratica)).
- Scope (ambito di visibilità):
 - Se rese note ad un blocco con l'uso di `extern`: ovunque, anche in moduli diversi;
 - Se rese note ad un blocco senza l'uso di `extern`: ovunque, nello stesso modulo, a partire dalla loro dichiarazione in avanti;
- Ciclo di vita:

Esistono e conservano il loro valore ovunque siano visibili all'interno del programma (es: uscita e successivo rientro da funzioni o blocchi);

Variabili globali in un unico modulo

```
#include <stdio.h>
int i, j;    //variabili globali inizializzate a 0

void main(void)
{
    extern int i, j; //in questo caso la
                    //dichiarazione viene
                    //generalmente omessa

    int x;
    i = 2;
    j = 3;
    x = i + j;
}
```

Variabile globale condivisa in più moduli

```
#include <stdio.h>
int i;
void main(void)
{
    funz();
}
```

mioprogr.c

```
#include <stdio.h>
extern int i;
void funz(void);

void funz(void)
{
    int j;
    j = i+1;
    ...
}
```

mod1.c

i è globale per entrambi i moduli

Variabile globale condivisa in più moduli con uso di header file

```
#include <stdio.h>
int i;
void funz(void);
...
```

mioprogr.h

```
#include "mioprogr.h"

void main(void)
{
    funz();
}
```

mioprogr.c

```
#include "mioprogr.h"
void funz()
{
    int j;
    j = i+1;
    ...
}
```

mod1.c

i è globale per entrambi i moduli (mioprogr.c, mod1.c)