STRUTTURA DI UN PROGRAMMA

File `prova1.c`

Area globale

```
#include <stdio.h>
...
```
Direttive

```
int m;
int f(int);
```
Dichiarazioni globali e prototipi di funzioni

```
int g(int x){
.../*ambiente locale a g*/}
```
Definizioni di funzioni

```
main(){
...}
```

```
int f(int x){
.../*ambiente locale a f*/}
```
Definizioni di funzioni

STRUTTURA DI UN PROGRAMMA

- Il ***main*** è l'unica parte ***obbligatoria***
- Le ***direttive*** sono gestite dal ***preprocessore***
- Le ***variabili globali*** sono visibili in ***tutti gli ambienti*** del programma
- Esistono delle regole di visibilità per gli identificatori (nomi di variabili, di funzioni, costanti) che definiscono in quali parti del programma tali identificatori possono essere usati

AMBIENTI

In un programma esistono diversi ambienti:

- area globale
- il main
- ogni singola funzione
- ogni blocco

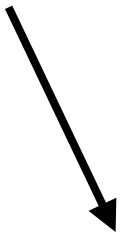
FUNZIONI COME COMPONENTI SOFTWARE

- Una funzione è un *componente software (servitore) riutilizzabile*

- che costituisce una *unità di traduzione*:
  - può essere definita in un unico file e compilata per proprio conto
  - pronta per essere usata da chiunque

DICHIARAZIONE DI FUNZIONE

La *dichiarazione* di una funzione è costituita dalla sola interfaccia, *senza corpo* (sostituito da un ;)



<dichiarazione-di-funzione> ::=
<tipoValore> **<*nome*>(**<parametri>**)** **;**

DICHIARAZIONE DI FUNZIONI

Dunque,

- per usare una funzione non occorre conoscere tutta la *definizione*
- *basta conoscere la dichiarazione*, perché essa specifica proprio il *contratto di servizio*

DEFINIZIONE vs. DICHIARAZIONE

La *definizione* di una funzione costituisce l'effettiva realizzazione del componente

- Non può essere duplicata
- Ogni applicazione deve contenere una e una sola definizione per ogni funzione utilizzata
- La compilazione della definizione genera il codice macchina che verrà eseguito ogni volta che la funzione verrà chiamata

DICHIARAZIONE DI FUNZIONI

La *dichiarazione* specifica:

- il nome della funzione
- numero e tipo dei parametri (non necessariamente *il nome*)
- il tipo del risultato

} ***interfaccia***

La *dichiarazione* di una funzione costituisce solo una specifica delle proprietà del componente:

- Può essere duplicata senza danni
- Un'applicazione può contenerne più di una
- La compilazione di una dichiarazione non genera codice macchina

DICHIARAZIONE vs. DEFINIZIONE

- La definizione è *molto più* di una dichiarazione

definizione = dichiarazione + *corpo*

La definizione funge anche da dichiarazione (ma non viceversa)

FUNZIONI E FILE

- Un programma C è, in prima battuta, una collezione di funzioni
  - una delle quali è il *main*
- Il testo del programma deve essere scritto in uno o più *file di testo*
  - il file è un concetto *del sistema operativo*, non del linguaggio C

Quali regole osservare?

FUNZIONI E FILE

- Il *main* può essere scritto *dove si vuole* nel file
  - viene chiamato dal sistema operativo, il quale sa come identificarlo
- Una funzione, invece, deve rispettare una *regola fondamentale di visibilità*
  - prima che qualcuno possa *chiamarla*, la funzione deve essere stata dichiarata
  - altrimenti si ha errore di compilazione

ESEMPIO SCORRETTO (SINGOLO FILE)

File `prova1.c`

```
main() {
  int y = fact(3);
}

int fact(int n) {
   if(n<=1)  return 1 ;
         else  return n*fact(n-1);
}
```

NOTA: all'atto della chiamata **`fact`** non è ancora stata definita

ERRORE DI COMPILAZIONE

ESEMPIO CORRETTO (SINGOLO FILE)

File `prova1.c`

```
int fact(int n) {
   if(n<=1)  return 1 ;
        else  return n*fact(n-1);
}

main() {
  int y = fact(3);
}
```

Prima definisco **`fact`** poi la uso

ESEMPIO CORRETTO (SINGOLO FILE)

File `prova1.c`

```
int fact(int);

main() {
  int y = fact(3);
}

int fact(int n) {
   if(n<=1)  return 1 ;
        else  return n*fact(n-1);
}
```

OPPURE prima dichiaro **`fact`** tramite un **PROTOTIPO** poi la uso e dopo la definisco

ALTRO ESEMPIO SCORRETTO

File **prova1.c**

```
int f(int x){
  if (x > 0) return g(x);
  else return x;
}
int g(int x) {
  return f(x-2);
}

main() {
  int y = f(3);
}
```

ATTENZIONE:
f chiama **g** e **g** chiama **f**
Quale definisco prima?

UTILITÀ DEI PROTOTIPI

File **prova1.c**

```
int g(int);  /* prototipo */
int f(int x){
  if (x > 0) return g(x);
  else return x;
}
int g(int x) {
  return f(x-2);
}

main() {
  int y = f(3);}
```

FUNZIONI E FILE

- Una funzione, ***opportunamente dichiarata***, può essere invocata anche da un'altra funzione contenuta in un ***file sorgente differente***

- Per il progetto di ***programmi C organizzati su più file***, vedere le lezioni corrispondenti del corso di *Laboratorio di Informatica L-A*