Struttura di un file sorgente in C

Il seguente programma scrive ciao ciao sullo schermo

```
#include <stdio.h>      /* Include la libreria stdio.h che contiene funzioni
                           necessarie per le operazioni di Input/Output */

int main()              /* Definiamo la funzione main, che non riceve nessun valore come argomento.
                           Il programma inizierà l'esecuzione a partire da questa  funzione,
                           che deve essere presente in ogni programma */
{
Definizione delle Variabili
                        /* Le istruzioni della funzione sono racchiuse tra le {,} */

  printf("Ciao Ciao!\n");   /* Chiamiamo la funzione di libreria printf che si occupa di stampare.
                               \n indica il NewLine,  ossia serve per andare a capo.
                               si deve mettere un punto e virgola alla fine di ogni istruzione. */

  return(0);            /* Istruzione per terminare la funzione main.

}                       /* chiudiamo la funzione principale main() */
```

In questo primo programma si notano già alcune cose fondamentali:

- La prima istruzione scritta #include <stdio.h> è una direttiva per il precompilatore C, ***questa istruzione (non termina con ;)*** corrisponde alla richiesta di caricamento del file di nome stdio.h, il quale a sua volta contiene le funzioni necessarie alla gestione delle operazioni di ingresso e uscita. Questi file sono detti header, infatti terminano con il suffisso .h

- **La funzione principale main**(), deve essere sempre presente, in quanto è quella che viene eseguita per prima. Gli argomenti da passare alle funzioni vanno posti tra le parentesi ( ) che si trovano dopo il nome della funzione stessa.

- La funzione printf() ha come argomento una stringa di caratteri (il testo da stampare) tra " ". Alla fine della stringa di caratteri abbiamo inserito \n, per indicare di andare a capo. (Sono disponibili altri ***caratteri speciali*** non editabili, ad esempio \t , \" , ..)

- I commenti possono essere inseriti tra i caratteri /* e */ (se interessano una sola riga con //

Nota

Quando il programma riceve degli argomenti sulla linea di comando, allora la definzione della funzione `main` deve essere modificata come:

```
int main(int argc, char *argv[])
```

← Questo argomento lo affronteremo in seguito

Definizione delle variabili

Definizione di variabili intere	`int i, j ;`	`int i = 0,j=7;`
Definizione di variabili reali	`float r ;`	`float r = 4.67;`

Le variabili possono essere inizializzate quando sono definite

Librerie principali

Lettura/scrittura su terminale e su file	`#include <stdio.h>`
Interazione con sistema operativo	`#include <stdlib.h>`
Funzioni matematiche	`#include <math.h>`
Elaborazione di testi e stringhe	`#include <string.h>`
Analisi del tipo di caratteri	`#include <ctype.h>`
Valori minimi e massimi	`#include <limits.h>`

Istruzioni eseguibili

Assegnazione a variabile

```
a = 0 ;
b = 17 ;
c = a + b ;
b = b + 1 ;
d = b * b - 4 * a * c ;
e = b * (b - 4 * a) * c ;
x1 = ( -b + sqrt(d) ) / ( 2*a ) ;
nomevariabile = espressione  ;
```

Lettura (input) di numeri interi	`scanf("%d", &i) ;` ⇐ ricordare il '&'
Lettura (input) di numeri reali	`scanf("%f", &r) ;` ⇐ ricordare il '&'
Stampa (output) di messaggi e numeri	`printf("Numero␣%d,␣valore␣%f\n", i, r) ;`
Vai a capo nella stampa	`printf("\n") ;`

Espressioni aritmetiche

Le 4 operazioni	`+ - * /`
Le parentesi	`( (a + b) * (c / (d - e) ) )`
Resto della divisione	`%`

Funzioni definite in `math.h`

Valore assoluto	$y \leftarrow \lvert x \rvert$	`y = fabs(x) ;`
Radice quadrata	$y \leftarrow \sqrt{x}$	`y = sqrt(x) ;`
Radice cubica	$y \leftarrow \sqrt[3]{x}$	`y = cbrt (x) ;`
Elevamento a potenza	$y \leftarrow x^z$	`y = pow(x, z) ;`
Ipotenusa	$y \leftarrow \sqrt{x^2 + z^2}$	`y = hypot(x, z) ;`
Ceiling	$y \leftarrow \lceil x \rceil$	`y = ceil(x) ;`
Floor	$y \leftarrow \lfloor x \rfloor$	`y = floor(x) ;`
Arrotondamento	$y \leftarrow \lfloor x + 1/2 \rfloor$	`y = round(x) ;`
Troncamento verso 0	$y \leftarrow sign(x)\ \lfloor \lvert x \rvert \rfloor$	`y = trunc(x) ;`
Resto della divisione	$y \leftarrow Resto(x/z)$	`y = fmod(x, z) ;`
Esponenziale	$y \leftarrow e^x$	`y = exp(x) ;`
	$y \leftarrow 2^x$	`y = exp2(x) ;`
Logaritmo	$y \leftarrow \ln x$	`y = log(x) ;`
	$y \leftarrow \log_2 x$	`y = log2(x) ;`
	$y \leftarrow \log_{10} x$	`y = log10(x) ;`
Funzioni trigonometriche	$y \leftarrow \sin x$	`y = sin(x) ;`
	$y \leftarrow \cos x$	`y = cos(x) ;`
	$y \leftarrow \tan x$	`y = tan(x) ;`
Funzioni trigonometriche inverse	$y \leftarrow \arcsin x$	`y = asin(x) ;`
	$y \leftarrow \arccos x$	`y = acos(x) ;`
	$y \leftarrow \arctan x$	`y = atan(x) ;`
	$y \leftarrow \arctan(x/z)$	`y = atan2(x, z) ;`
Funzioni iperboliche	$y \leftarrow \sinh x$	`y = sinh(x) ;`
	$y \leftarrow \cosh x$	`y = cosh(x) ;`
	$y \leftarrow \tanh x$	`y = tanh(x) ;`
Funzioni iperboliche inverse	$y \leftarrow \sinh^{-1} x$	`y = asinh(x) ;`
	$y \leftarrow \cosh^{-1} x$	`y = acosh(x) ;`
	$y \leftarrow \tanh^{-1} x$	`y = atanh(x) ;`

Le funzioni dichiarate in **stdio.h** possono generalmente essere divise in due categorie: le funzioni per la manipolazione di file e quelle per la manipolazione dell'input/output.

Nome	Descrizione
Funzioni per la manipolazione di file	
fclose	Chiude il file associato al valore `FILE *` passatole.
fopen, freopen, fdopen*	Apre un file in lettura o scrittura.
remove	Rimuove un file.
rename	Rinomina un file.
rewind	Agisce come se fosse stata chiamata la funzione `fseek(stream, 0L,`
tmpfile	Crea ed apre un file temporaneo, che viene poi cancellato e chiuso con `fclose()`.
Funzioni per la manipolazione dell'input/output	
clearerr	Cancella l'indicatore di fine file e quello d'errore per un dato stream.
feof	Controlla se l'indicatore di fine file è stato settato per un dato stream.
ferror	Controlla se l'indicatore d'errore è stato settato per un dato stream.
fflush	Forza lo svuotamento del buffer output per un dato stream, provocando l'immediata
fgetpos	Salva la posizione corrente associata allo stream passato come primo argomento
fgetc	Restituisce un carattere da un file.
fgets	Restituisce una stringa presa da un file, terminata da un carattere di nuova riga ('\n') o
fputc	Scrive un carattere su un file.
fputs	Scrive una stringa su un file.
ftell	Restituisce un indicatore di posizione sul file che può essere passato alla
fseek	Si sposta attraverso un file.
fsetpos	Imposta l'indicatore di posizionamento del file di uno stream associato al primo
fread	Legge da file dei dati di diverse dimensioni.
fwrite	Scrive su file dei dati di diverse dimensioni.
getc	Legge e restituisce un carattere dallo stream passatole ed incrementa l'indicatore di
ungetc	Restituisce allo stream almeno un carattere. Alla successiva chiamata di getc verrà
getchar	Ha gli stessi effetti di getc(stdin). È lo Strea associato alla tastiera
gets	Legge caratteri da stdin finché non incontra un carattere di nuova riga (newline) o un
printf, fprintf,	Utilizzato per stampare vari tipi di dato su stdout. È lo stream associato al monitor
vprintf	Anch'essa utilizzata per stampare su stdout.
perror	Scrive un messaggio di errore su stderr.
puts	Scrive una stringa su stdout.

Le funzioni di **stdlib.h** possono essere classificate nelle seguenti categorie: conversione tra tipi, gestione della memoria, controllo dei processi, ricerca ed ordinamento, matematica semplice.

Nome	Descrizione
Conversione tra tipi	
`atof`	Converte una stringa in un numero in virgola mobile. Equivalente astrtod(s,
`atoi`	Converte una stringa in un numero intero. Equivalente a (int)strtol(s, (char**)NULL, 10).
`atol`	Converte una stringa in un numero intero lungo (long int). Equivalente astrtol(s,
`strtod`	Converte una stringa in un double (numero a virgola mobile), effettuando dei controlli
`strtol`	Converte una stringa, che rappresenta un numero in una base arbitraria compresa tra 2
`strtoul`	Equivalente a strtol() tranne per il tipo del risultato, che è unsigned long.
Generazione di numeri pseudocasuali	
`rand`	Restituisce un numero intero pseudocasuale compreso tra 0 e RAND_MAX.
`srand`	Inizializza il seme per la sequenza di numeri pseudocasuali della funzione rand().
Allocazione e deallocazione di memoria	
`calloc, malloc, realloc`	Funzioni che si occupano dell'allocazione dinamica della memoria.
`free`	Libera la memoria allocata dinamicamente dalla famiglia di funzionimalloc().
Controllo dei processi	
`abort`	Causa la terminazione immediata ed anormale del programma, come se fosse stato
`atexit`	Registra una funzione, della quale le viene passato il puntatore, affinché venga eseguita
`exit`	Causa la normale terminazione del programma. Tutte le funzioni registrate
`getenv`	Restituisce la stringa che nell'ambiente di lavoro del programma è associata al nome
`system`	Passa la stringa fornitale all'ambiente di lavoro per l'esecuzione e restituisce il codice
Ricerca ed ordinamento	
`bsearch`	Implementa in maniera generica l'algoritmo di ricerca dicotomica.
`qsort`	Implementa in maniera generica l'algoritmo di ordinamento quicksort.
Matematica semplice - presenti anche in math.h	
`abs, labs`	Calcola il valore assoluto dell'argomento.
`div, ldiv`	Calcola il quoziente ed il resto della divisione intera tra il dividendo ed il divisore forniti.