1 PANORAMICA SUL LINGUAGGIO C

1.1 Le origini del linguaggio C

Il C fu inventato e implementato da Dennis Ritchie negli anni '70 su una macchina che impiegava il sistema operativo Unix. Il C è il risultato di uno sviluppo che è partito da un linguaggio chiamato BCPL. Il BCPL influenzò un linguaggio chiamato B; il B portò allo sviluppo del C.
Negli anni 90 l'ANSI e l'ISO hanno terminato un processo di standardizzazione che ha portato allo sviluppo dello Standard C ANSI/ISO.
Oggi tutti i principali compilatori C/C++ seguono lo Standard C.

1.2 Il C è un linguaggio strutturato

La caratteristica che distingue un linguaggio strutturato è l'isolabilità del codice e dei dati. Questa è la capacità di suddividere e nascondere dal resto del programma tutte le informazioni e le istruzioni necessarie per eseguire una determinata operazione.

Il principale componente strutturale del C è la *funzione*: una subroutine a sé stante. In C, le funzioni sono i mattoni su cui si basa tutta l'attività di programmazione. Esse consentono di definire e codificare in modo distinto le varie operazioni svolte dal programma e quindi consentono di creare programmi modulari.
Utilizzando variabili locali, è possibile scrivere subroutine realizzate in modo tale che gli eventi che si verificano al suo interno non provochino effetti collaterali in altre parti del programma. Se si sviluppa una funzione ben isolata, tutto quello che si deve sapere sulla funzione è cosa essa faccia e non come lo faccia.

Un altro modo per strutturare e isolare il codice in C prevede l'uso di blocchi di codice. Un *blocco di codice* è formato da un gruppo di istruzioni connesse logicamente che viene considerato come una singola unità. In C, è possibile creare blocco di codice inserendo una sequenza di istruzioni fra una coppia di parentesi graffe. In questo esempio,

```
if (x<10) {
  printf("troppo basso, riprova\n");
  scanf("%d",&x);
}
```

se x è minore di 10 vengono eseguite entrambe le istruzioni che si trovano dopo l'if e fra le parentesi graffe. Queste due istruzioni, insieme alle parentesi graffe, rappresentano un blocco di codice: non è possibile eseguire un'istruzione senza eseguire anche l'altra.
I blocchi di codice consentono di implementare molti algoritmi con chiarezza, eleganza ed efficienza.

1.3 L'aspetto di un programma C

Tutti i programmi C sono formati da una o più funzioni. L'unica funzione che deve essere obbligatoriamente presente si chiama main( ), la prima funzione che viene richiamata quando inizia l'esecuzione del programma. In un programma C ben realizzato, main( ) contiene uno schema dell'intero funzionamento del programma, costituito da una serie di chiamate a funzioni.

Dichiarazioni globali

tipo restituito `main(`*elenco parametri*`)` `{`
  sequenza istruzioni
`}`

tipo restituito f_1`(`*elenco parametri*`)` `{`
  sequenza istruzioni
`}`

tipo restituito f_2`(`*elenco parametri*`)` `{`
  sequenza istruzioni
`}`

…

tipo restituito f_N`(`*elenco parametri*`)` `{`
  sequenza istruzioni
`}`

1.4 Il compilatore ed il linker

La fase di *compilazione* è la fase nella quale il codice sorgente viene convertito in codice oggetto (se non ci sono errori). I file oggetto sono file, con in genere estensione obj (i quali si trovano in Visual C++ nelle directory Debug o Release) e che contengono tutto il codice tradotto in linguaggio pseudo-assembler.
In senso teorico, è possibile creare un funzionale programma C costituito unicamente dalle istruzioni scritte dal programmatore. Tuttavia, tutti i compilatori C sono dotati di una *libreria di funzioni standard* che eseguono le operazioni più comuni. Il risultato è che la maggior parte dei programmi includono chiamate alle varie funzioni contenute nella libreria standard, il cui codice è già espresso in formato oggetto.
A questo punto abbiamo tanti files oggetto che devono essere 'incollati' e a ciò pensa il *linker*, il quale verifica anche che tutti i riferimenti di chiamate a funzioni siano presenti e corretti; il risultato finale del linking è un eseguibile (exe o dll).

2 LE ESPRESSIONI

2.1 I cinque tipi di dati principali

L'elemento di base del linguaggio C è l' *espressione*. Le espressioni sono costituite da *dati* legati da *operatori*. I dati possono essere essere rappresentati da variabili o costanti.
In C ci sono cinque tipi di dati principali:

caratteri	`char`
numeri interi	`int`
numeri in virgola mobile	`float`
numeri in virgola mobile doppi	`double`
non valori	`void`

Se si esclude il tipo void, i tipi di dati principali possono essere dotati di modificatori. L'elenco dei modificatori è il seguente: signed, unsigned, long e short.
La tabella seguente mostra tutte le combinazioni di tipi di dati validi, indicando le dimensioni approssimative in bit e l'intrevallo minimo richiesto.

Tipo	**Dimensioni**	**Intervallo minimo approssimativo**
char	8	Da –127 a 127
unsigned char	8	Da 0 a 255
signed char	8	Da –127 a 127
int	16	Da –32767 a 32767
unsigned int	16	Da 0 a 65535
signed int	16	Come int
short int	16	Come int
unsigned short int	16	Da 0 a 65535
signed short int	16	Come short int
long int	32	Da –2.147.483.647 a 2.147.483.647
signed long int	32	Come long int
unsigned long int	32	Da 0 a 4.294.967.295
float	32	Sei cifre di precisione
double	64	Dieci cifre di precisione
long double	80	Dieci cifre di precisione

2.2 Le variabili

Una *variabile* corrisponde ad un determinato gruppo di celle di memoria utilizzate per contenere un valore che può essere modificato dal programma. Tutte le variabili C prima di essere utilizzate devono essere dichiarate. La forma più generale di una dichiarazione è:

tipo elenco_variabili;

dove *tipo* deve essere un tipo di dati valido in C comprendendo eventuali modificatori, mentre *elenco_variabili* può essere formato da uno o più nomi di identificatori separati da virgole.
Ecco alcune dichiarazioni:

```
int i,j,k;
short int pippo;
double guadagno, ricavo;
```

2.2.1 Dove vengono dichiarate le variabili?

Le variabili possono essere dichiarate in tre luoghi: all'interno delle funzioni, nella definizione dei parametri delle funzioni e all'esterno di tutte le funzioni. Queste tre posizioni corrispondono a tre diversi tipi di variabili:

Tipo	**Dichiarazione**
Variabili locali	All'interno delle funzioni
Parametri formali	Nella definizione dei parametri delle funzioni
Variabili globali	All'esterno di tutte le funzioni

Le *variabili locali* vengono dichiarate all'interno di un blocco { } e possono essere utilizzate solo dalle istruzioni che si trovano all'interno del blocco. In altre parole, le variabili locali non sono note all'esterno del proprio blocco di codice. La vita delle variabili locali è legata all'esecuzione del blocco di codice in cui sono dichiarate: questo significa che una variabile locale viene creata nel momento in cui si entra nel blocco e viene distrutta all'uscita. Il blocco di codice più comune in cui vengono dichiarate le variabili locali è la funzione.
Nell'esempio la variabile `Temp` è locale ed è quindi nota solo alla funzione `Somma`; viene creata quando l'esecuzione entra nel blocco `Somma` e distrutta all'uscita.

I *parametri formali* sono le variabili che devono accogliere i valori passati come argomenti di una funzione. Queste variabili si comportano come qualunque altra variabile locale all'interno della funzione. Come si può vedere nell'esempio, la loro dichiarazione si trova dopo il nome della funzione e tra parentesi.
Nell'esempio le variabili `a`, `b` e `c` sono parametri formali della funzione `Somma` ed hanno il compito di ospitare per tutta la durata dell'esecuzione della funzione i valori passati al momento della chiamata.

A differenza delle variabili locali, le *variabili globali* sono note all'intero programma e possono essere utilizzate in ogni punto del codice. Inoltre esse conservano il proprio valore durante l'intera esecuzione del programma. Le variabili globali devono essere dichiarate all'esterno di ogni funzione. Nell'esempio che segue la variabile `Risultato` è globale e viene vista sia dalla funzione `main` che da `Stampa`.

```
#include <stdio.h>

/* Dichiarazioni globali */
long int Risultato;
long int Somma(int a, int b, int c);
void Stampa(void);

/* Routine principale */
int main(void){
```

```
  Risultato=Somma(10,20,30);
  Stampa();
  return(0);
}

/* Funzione Somma */
long int Somma(int a, int b, int c){
  long int Temp;

  Temp=a+b+c;
  return(Temp);
}

/* Funzione Stampa */
void Stampa(void) {
  printf("La somma dei numeri 10, 20 e 30 è: %d\n",Risultato);
}
```

2.3 Le costanti

Le variabili di tipo const non possono essere modificate dal programma ma è possibile assegnare loro un valore iniziale.
Ad esempio,
`const int a=10;`
crea una variabile intera chiamata `a` con un valore iniziale paria a 10 che il programma non può modificare.

2.4 Gli operatori

Il C è un linguaggio molto ricco di operatori. Vi sono quattro classi principali di operatori: *aritmetici*, *relazionali*, *logici* e *bit-a-bit*.

2.4.1 Operatori aritmetici

OPERATORI ARITMETICI	
Operatore	**Azione**
-	Sottrazione, anche meno unario
+	Addizione
*	Moltiplicazione
/	Divisione
%	Modulo
--	Decremento
++	Incremento

La precedenza degli operatori aritmetici è la seguente:
Alta ++ --
- (meno unario)
* / %
Bassa + -

2.4.2 Operatori relazionali e logici

Nel termine *operatore relazionale*, con "relazionale" si intende la relazione che un valore ha rispetto ad un altro. Nel termine *operatore logico*, l'aggettivo "logico" fa riferimento ai modi in cui queste relazioni possono essere connesse. Questi due operatori vengono discussi insieme in quanto spesso vengono utilizzati congiuntamente.

OPERATORI RELAZIONALI	
Operatore	**Azione**
>	Maggiore
>=	Maggiore o uguale
<	Minore
<=	Minore o uguale
==	Uguale
!=	Diverso

OPERATORI LOGICI	
Operatore	**Azione**
&&	AND
||	OR
!	NOT

I livelli di precedenza relativa esistenti tra gli operatori relazionali e logici sono i seguenti:

Alta !
> >= < <=
== !=
&&
Bassa ||

2.4.3 Operatori bit-a-bit

Le *operazioni bit-a-bit* si occupano di controllare, impostare o spostare i bit che compongono un byte o una word, che corrispondono ai tipi char e int e relative varianti. Quindi non è possibile utilizzare questi operatori sui tipi float, double, long double, void e sui tipi complessi. La tabella elenca tutti gli operatori bit-a-bit.

OPERATORI BIT-A-BIT	
Operatore	**Azione**
&	AND
|	OR
^	OR esclusivo (XOR)
~	Complemento a uno
>>	Scorrimento a destra
<<	Scorrimento a sinistra

2.5 Le espressioni

Gli operatori, le costanti e le variabili sono gli elementi costitutivi delle *espressioni*. Un'espressione C è una combinazione valida di questi elementi.