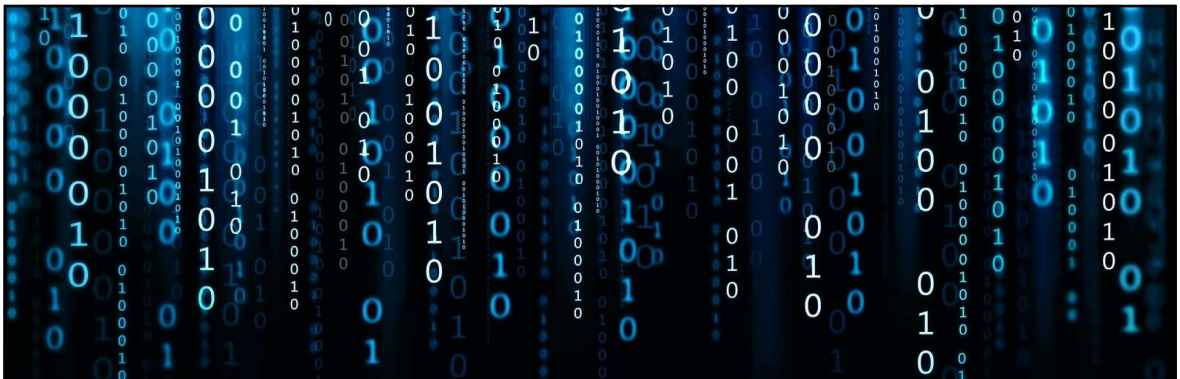


LABORATORIO DI PROGRAMMAZIONE
Corso di laurea in matematica

16 – IL LINGUAGGIO C

Marco Lapegna
Dipartimento di Matematica e Applicazioni
Universita' degli Studi di Napoli Federico II

wpage.unina.it/lapegna

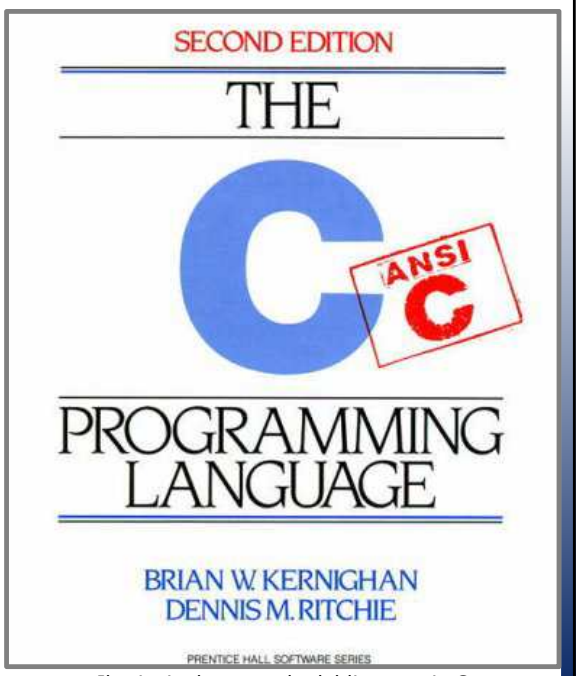
Evoluzione del C

il C e' un linguaggio di programmazione ad alto livello sviluppato agli inizi degli anni '70 del XX sec. presso i BELL Laboratories

E' estremamente potente (i sistemi operativi Unix e Linux sono scritti in C) e versatile (e' possibile sviluppare applicazioni in molti campi: industriale, scientifico, telecomunicazioni,…)

L' ANSI C (1990) e' la principale versione standard del C, e garantisce una ampia portabilita' dei codici

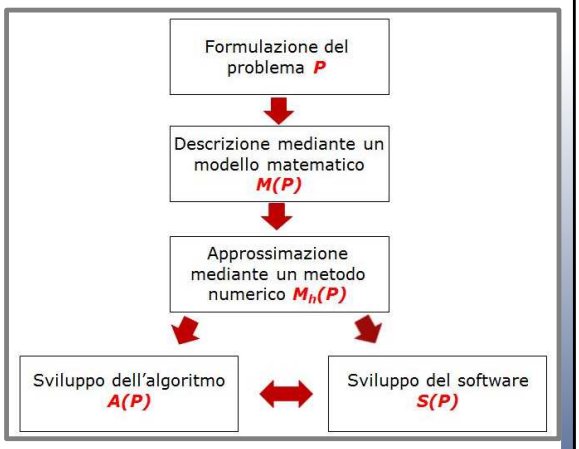
Dal C sono derivati numerosi linguaggi di nuova generazione come C++, Java



Il principale manuale del linguaggio C

Passi per risolvere un problema con il calcolatore

1. Formulazione del problema
2. Formulazione del modello matematico
3. Formulazione del modello discreto
4. Sviluppo dell'algoritmo
5. sviluppo del programma, cioe' traduzione dell'algoritmo in un linguaggio di programmazione ad alto livello



passi per la risoluzione di un problema scientifico con il calcolatore

- Di nostra competenza i passi 4 e 5
- Molto lavoro per il passo 4
- Passo 5 semplice ed automatico !!
- Importante solo rispettare la sintassi

Inizio e fine programma

- Ogni programma C e' compreso le parentesi graffe { } che seguono l'istruzione main
- main e' sempre il nome del programma
- Ogni istruzione del programma termina con ;
- Analogo di begin ed end per il Pascal Like

Pascal Like

```
begin nomealgoritmo

.....
.....
.....
.....

end nomealgoritmo
```

Inizio e fine in Pascal-like

C

```
main ( ) {

...
...

}
```

Equivalente C

Dichiarazione delle variabili

- 5 tipi di dati fondamentali
- Il C e' Case Sensitive (fa differenza tra maiuscole e minuscole)
- Obbligatorio dichiarare le variabili
- Altri tipi meno usati: unsigned, long ,…
- L'utilizzo del tipo logico (bool) richiede #include<stdbool.h>

Pascal Like

```
.....
var: M, N: integer
var: X, Y: real
var: Z, W: double
var: A, B: character
var: R, T: boolean
.....
```

Dichiarazione variabili in Pascal-like

C

```
int  M, N ;
float  X, Y ;
double Z, W;
char   A, B ;
bool  R, T;
```

Equivalente C

Lettura, definizione e stampa delle variabili

- scanf(…) definisce una o piu' variabili mediante l'unita' di input (es. la tastiera)
- printf(…) stampa il contenuto di una variabile sull'unita' di output (es. il monitor)
- scanf e printf sono funzioni intrinseche del C che vanno dichiarate. La dichiarazione e' presente nel file stdio.h che va "incluso" nel programma con l'istruzione #include<…>
- l'operatore % specifica il tipo di variabile da leggere/stampare:
  d : interi f : reali e d.p. c : carateri
- gli operatori & nell'istruzione scanf saranno chiari piu' avanti …..
- L'operatore = indica una operazione di assegnazione di un valore ad una variabile

Pascal Like

```
.....
Read A, B, C
X = A+B+C
Print X
.....
```

Lettura, scrittura e assegnazione in P-L

C

```
#include<stdio.h>
main() {
int A, B, C, X;
...
scanf (" %d %d %d ", &A, &B, &C);
X = A + B + C;
printf("somma = %d \n", X);
...
```

Equivalente C

commenti

I commenti sono linee di testo inserite nel codice che non vengono tradotte dal compilatore

Hanno lo scopo di migliorare la leggibilita' dei programmi, descrivendo le sezioni del programma o anche gruppi di istruzioni

E' buona norma inserire numerose linee di commento (anche fino al 50% delle linee dl programma)

In C le linee di commento iniziano con //

Pascal-like

```
Begin prova
.....
(*  Questo e' un commento *)
...

End prova
```

Commenti in P-L

C

```
main( ){
.....
//   Questo e' un commento
...

}
```

Equivalente C

Struttura di selezione

- Esatta corrispondenza con il Pascal Like
- le istruzioni che compongono i rami della struttura sono racchiusi tra parentesi graffe
- { … }
- Disponibile anche senza il ramo ELSE

Pascal Like

```
.....
If (condizione)  then
                 .....
                 .....
Else
                 .....
                 .....
Endif
.....
```

If-then-else in pascal-like

C

```
if ( condizione ) {

....

} else {

....

}
```

Equivalente in C

Operatori logici e relazionali

- Vengono eseguiti prima gli operatori relazionali
  > < == != >= <=
- Poi quelli logici
  ! && ||
- ! (not)
- && (and)
- || (or)

Pascal Like

```
Boolean L
.....
L = (A > B) or (C >= D)
L = (X == Y) and (Z /= W)
L = not L
```

Esempi di operatori logici e relazionali in P-L

C

```
#include<stbool.h>
main() {
bool L;
.....
L = (A > B) || (C >= D);
L = (X == Y) && (Z != W);
L = !L

}
```

Equivalente C

Struttura di iterazione for

- Tre campi sempre obbligatori
  - Inizializzazione dell'indice
  - Condizione di validita' dell'indice !!
  - Incremento dell'indice
- Istruzioni da ripetere racchiuse da parentesi graffe {… }
- Piu' strutture innestate richiedono indici distinti
- nel caso di incremento di 1 si puo' usare l'operatore i++ , equivalente a i=i+1
- (es. for (i=1; i < N; i++))

Pascal Like

```
for i = 1 to N step K
        .....
        .....
endfor
```

For-endfor in Pascal-like

C

```
for ( i =0; i < N; i=i+k) {

.....
.....

}
```

Equivalente C

Struttura di iterazione while

- Ripete piu' volte le istruzioni tra while e endwhile
- Quanto la condizione risulta **vera** la struttura di iterazione **continua**
- traduzione diretta del pascal-like

Pascal Like

```
while (condizione)
        ......
        ......
endwhile
```

While-endwhile in pascal-like

C

```
......
while (condizione){
...
...
}
```

Equivalente C

Struttura di iterazione repeat

- Ripete piu' volte le istruzioni tra repeat e until
- Quanto la condizione risulta **vera** la struttura di iterazione **termina**
- Non esiste una traduzione "diretta"
- Realizzato mediante do-while
- poiche' il while continua l'esecuzione quando la condizione e' vera, in questo caso e' **necessario negare la condizione di uscita**

Pascal Like

```
repeat
        ......
        ......
until (condizione)
```

Repeat-until in Pascal-like

C

```
do {
......
.....
} while ( ! (condizione) ) ;
```

Equivalente C

Gli array

- in fase di dichiarazione e' necessario specificare la dimensione
- successivamente non e' possibile modificare la dimensione
- e' possibile fare riferimento a ciascuna componente tramite un indice
- La prima componente ha indice 0 (zero)
- in memoria le componenti di un array occupano locazioni di memoria consecutive
- gli array 2-dimensionali sono memorizzati per righe in locazioni consecutive

Pascal Like

```
var: a(100): array of real
var: X(10,10): array of integer

...
...

for i=1 to 10
    read a(i)
endfor
```

Dichiarazione e lettura di un array in P-L

C

```
float A[100] ;
int X[10][10];
...
...
for ( i=0; i<10; i++ ) {
   scanf ( " %f ", &A[i]);
}
```

Equivalente C

Sottoprogrammi FUNCTION

- Unico tipo di procedura disponibile in C
- Invocata attraverso il nome, puo' ritornare direttamente un valore al programma chiamante attraverso il nome
- la function deve essere dichiarata con un tipo, che coincide con il tipo ritornato dalla function (se non ritorna niente il tipo e' void). Nella dichiarazione della funzione si dichiara anche il tipo dei dati di input
- Gli argomenti della function sono tutti argomenti di input
- L'esecuzione del programma chiamante riprende dopo il termine della subroutine

Pascal Like

```
begin mainprogram
var: A, B, C: integer
read A, B

somma (in: A, B; out: C)

print C
end
```

Chiamata di una procedura in Pascal-like

C

```
#include<stdio.h>
main(){
int A, B, C;
int somma(int, int);
scanf("%d %d", &A, &B);
C=somma(A,B);
printf(" %d \n", C);
}
```

Equivalente C

Sottoprogrammi FUNCTION

- il passaggio delle informazioni e' **per valore** (della variabile)
- la function opera su una copia delle locazioni di memoria delle variabili del programma chiamante
- L'istruzione return, ritorna il valore di output al programma chiamante (opzionale se la funzione e' void)
- E' possibile utilizzare nomi diversi nelle function e nel programma chiamante
- Osservazione. Le function ritornano sempre un solo valore (oppure nessuno se la funzione e' void)

Pascal Like

```
procedure somma(in: X, Y; out: Z)
var: X, Y, Z: integer

Z = X + Y

end somma
```

Procedura in Pascal-like

C

```
int somma( int X, int Y){
int Z;
Z = X+Y;
return Z;
}
```

Equivalente C

Function con piu' valori di ritorno

- **Come fare se si vuole una function che ritorna 2 o piu' valori?**
- IDEA: utilizzare lo stesso meccanismo del Fortran e passare l'indirizzo delle variabili. In tal modo le function operano direttamente sulle stesse locazioni di memoria
- Nel main, per passare l'indirizzo delle variabili X e Y si usa l'operatore &
- Nella function, A e B contengono l'indirizzo delle variabili X e Y del main. Per accedere ad esse si usa l'operatore *
- Esempio: function per lo scambio di due variabili A e B
  - Input: indirizzi delle variabili A e B
  - Output: niente (funzione void)
- Ora dovrebbe essere chiara la presenza di & nelle istruzioni scanf..

C

```
#include<stdio.h>
main() {
  void scambio (int *, int *);
  int X, Y;
  scanf("%d %d", &X, &Y);
  scambio(&X, &Y);
  printf("%d %d", X, Y);
}
```

Programma chiamante

C

```
void scambio ( int *A, int *B){
int T;
T = *A;
*A = *B;
*B = T;
return ;
}
```

Function per lo scambio

Passaggio di un array ad una function

Il nome di un array e' un indirizzo

Quindi

attraverso il nome, viene passato l'indirizzo della prima componente (analogo del Fortran)

- Allocazione di array in locazioni consecutive
- Passaggio per indirizzo (la function opera sulle stesse locazioni di memoria)

Da queste informazioni la function e' in grado di "ricostruire" tutto l'array

Nella function e' possibile evitare di dichiarare gli array specificando una dimensione fissata

C

```
int  A[9];
void funz( int, int [] );

n = 3;
...
funz( n, A);
...
```

Passaggio di un array ad un function in C

C

```
void funz (int  n,  int a[]){
int   i;
for (i=0; i<n; i++){
   printf(" %d" ,a[i]);
}

}
```

Function C che stampa un array

Array 2-d e function

- Gli array 2-d sono memorizzati per righe
- E' necessario specificare la leading dimension (numero massimo di colonne)
- Nella dichiarazione della funzione nel programma chiamante e' possibile indicare con il carattere * la presenza della leading dimension
- La leading dimension viene specificata nella chiamata della function
- Nella function si usa la leading dimension per specificare il numero di colonne dell'array
- La leading dimension deve precedere il nome dell'array nella lista degli argomenti

C

```
int  A[9][9];
void funz( int, int (*)[ ] , int);

ld =9;
n = 3;
...
funz( 9, A, n);
...
```

Esempio di chiamata di funzione con array 2-d

C

```
void funz(int  ld, int a[][ld], int n){
  int i, j;
  for (i=0; i<n, i++){
  for (j=0; j<n, j++){
     printf("%d ", a[i][j]);
  } printf("\n");  }
}
```

Funzione C che stampa un array 2-d

Funzioni intrinsiche

Il C dispone di una libreria di function intrinsiche per le piu' comuni funzioni matematiche e/o di utilita

- Y = SQRT(X)
- Y = ABS(X)
- Y = TAN(X)

- Y = SIN(X)
- Y = COS(x)

- Y = LOG(X)
- Y = LOG10(X)
- Y = EXP(X)

- E tante altre…

- Necessario includere <math.h>
- Compilare con opzione -lm

```
#include<stdio.h>
#include<math.h>
main () {
float x, y, pi, h;

pi = acos(-1);
h = pi/10.;

for (i=1; i<=10; i++){
   x = 0 + i*h;
   y = sin(x);
   printf(" %f %f \n", x, y);
}

}
```

Semplice programma per il calcolo di 11 valori di sin(x) nell'intervallo [0,3.14]

Un po' di storia (16)

Timothy John Berners-Lee (1955)

- Inglese, laureato in Fisica a Oxford nel 1976, dopo alcuni anni di lavoro in aziende di telecomunicazioni, nel 1980 si trasferisce al CERN

- Il 6 agosto 1991 mette in linea un sistema chiamato World Wide Web, una rete di nodi internet (siti web) collegati tra loro tramite collegamenti logici (link) per la condivisione globale di testi e/o contenuti multimediale

- Il sistema e' basato su un meccanismo di identificazione di risorse (URL), un protocollo di comunicazione (http), un linguaggio per i documenti (html), un programma server (httpd), e un programma per la visualizzazione (browser).

- L'efficienza, la facilita' e l'economicita' del sistema permettono a tutti di diventare editori di contenuti visibili in tutto il mondo.

- Per il suo lavoro riceve numerosi premi ed onoreficenze.

T.J. Berners-Lee (courtesy Computer History Museum)

Alcuni servizi basati sul protocollo http