Programma al Maggio 2003

- Rappresentazione della informazione
- Algoritmi in pseudocodice e flow charts
- Dati semplici (tutto).... *no POINTER*
- Variabili ed Espressioni
- Dati strutturati (tutto)
- Strutture di programma selezioni e cicli
- Funzioni elementari e struttura del programma (dati globali, include, librerie, funzioni esterne ..)

Rappresentazione della informazione

- Solo cifre 1 e 0 in unità di memoria (byte, word)
- Quanti “pattern” diversi con n bit? (2^n)
- Associazione dell’informazione ai “pattern”
  - Algoritmi (numeri $x_1*2^{n-1}+x_2*2^{n-2}+ \ldots$)
  - Tabelle (caratteri - Codice Ascii)
- Scrittura dei valori (‘X’ “Abc” 32 032 0x3F)

Tabella ASCII

bit meno significativi

bit più significativi

	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP 32	0 48	@ 64	P 80	` 96	p 112
0001	SOH	DC1	! 33	1 49	A 65	Q 81	a 97	q 113
0010	STX	DC2	« 34	2 50	B 66	R 82	b 98	r 114
0011	ETX	DC3	# 35	3 51	C 67	S 83	c 99	s 115
0100	EOT	DC4	$ 36	4 52	D 68	T 84	d 100	t 116
0101	ENQ	NAK	% 37	5 53	E 69	U 85	e 101	u 117
0110	ACK	SYN	& 38	6 54	F 70	V 86	f 102	v 118
0111	BEL	ETB	' 39	7 55	G 71	W 87	g 103	w 119
1000	BS	CAN	(40	8 56	H 72	X 88	h 104	x 120
1001	HT	EM	) 41	9 57	I 73	Y 89	i 105	y 121
1010	LF	SUB	* 42	: 58	J 74	Z 90	j 106	z 122
1011	VT	ESC	+ 43	; 59	K 75	[91	k 107	{ 123
1100	FF	FS	, 44	< 60	L 76	\ 92	l 108	| 124
1101	CR	GS	– 45	= 61	M 77	] 93	m 109	} 125
1110	SO	RS	. 46	> 62	N 78	^ 94	n 110	~ 126
1111	SI	US	/ 47	? 63	O 79	_ 95	o 111	DEL

Scrittura dei valori

Costanti

Numeri interi

Rappresentano numeri relativi (quindi con segno):

	2 byte	**4 byte**
base decimale	12	70000, 12**L**
base ottale	**0**14	**0**210560
base esadecimale	**0x**FF	**0x**11170

Numeri reali

Varie notazioni:

24.0 2.4E1 240.0E-1

Suffissi: l, L, u, U (interi-long, unsigned)
f, F (reali - floating)

Prefissi: 0 (ottale) 0x, OX(esadecimale)

Dati semplici

- Una unità manipolabile dotata di TIPO
  - Codifica usata per rappresentare i valori
  - Quali valori sono validi
  - Quali operazioni sono lecite e come agiscono tali operazioni

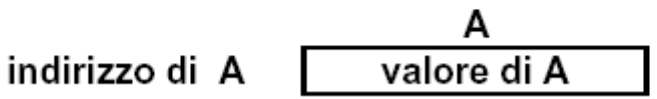
  - Es. `int` numeri compl a 2, -32.768 a +32.767

    `+ - * / % -- ++ == != < > <= >=`

Variabili

- Sono contenitori di informazioni
- Associano un nome simbolico a una locazione di memoria ….
  `int A;`
- Associano un TIPO al nome

Rappresentazione in memoria della variabile **A**



- Con &A si indica l'indirizzo di A

Dati strutturati

- Composizioni di dati semplici
- Regole di composizione
  - `[n]` n oggetti uguali e dello stesso tipo **(0↔n-1)**
  - `struct {` *dich di var;*
    dich di var; ….. }
- Regole di accesso
  - `A[12]` dodicesimo elemento di A
  - `A.B` componente B del dato A

Espressioni

- Variabili e Costanti cui si applicano Operatori secondo la loro precedenza
- Se le variabili sono strutturate si deve accedere ai componenti per cui sono definiti gli operatori

  - `(V<<1 | 7>>1 & 0xB)`
  - `S.CO[2] + B / 015 - V + S.Index`
  - `((i<S.CO[5]) && (i > S.NO[5] - S.CO[4]))`

Assegnamento

- E’ la frase principale con cui si fanno cambiare i valori delle variabili
  - VAR = <espressione>;
- Ma anche in uso le varianti

  += -= /= ecc.

Struttura del programma

```
main ()

{

  parte dichiarativa
  locale


  parte esecutiva

}
```

parola riservata (identificatore di funzione)
appare una e una sola volta nel programma
definisce l'inizio dell'esecuzione
è (formalmente) una funzione

definisce l'insieme di «oggetti» usati dal programma principale per l'esecuzione. sono oggetti visibili (locali) a main.

insieme di istruzioni che costituiscono il programma principale

Controllo della esecuzione

- Le istruzioni del programma possono essere eseguite
  - In sequenza
  - Alternativamente
  - Iterativamente
- Ogni frase del C, anche se composta di più frasi, è equivalente ad una sola frase.

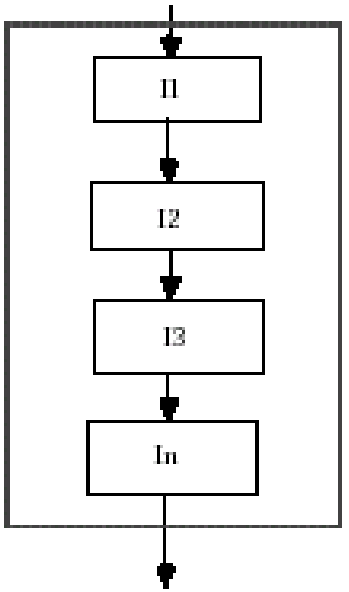
Istruzione composta { }

Determina l'esecuzione nell'ordine testuale delle istruzioni componenti.

Sintassi:

```
{
   <Dichiarazioni e Definizioni>
   <Sequenza di Istruzioni>
}
```

```
<sequenza-istruzioni> ::=
        <istruzione> {; <istruzione> }
```



Sintatticamente equivalente a una singola istruzione (strutturata).

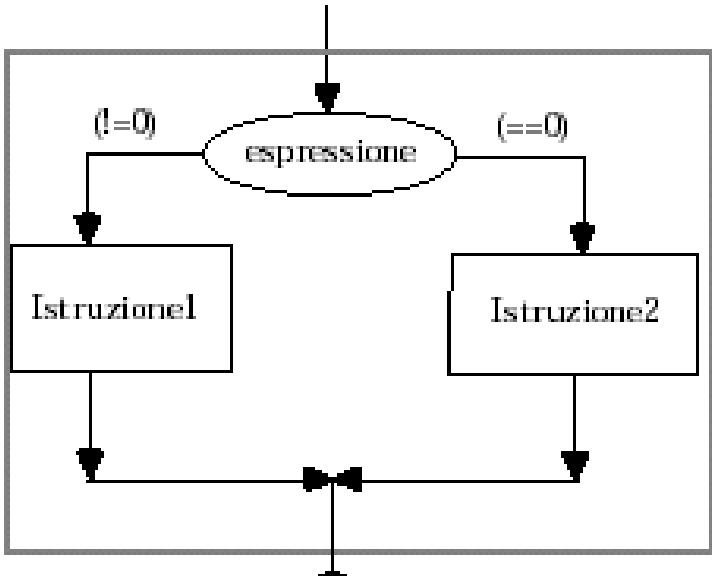
Istruzioni di alternativa

Istruzione if:

seleziona l'esecuzione di una sola tra le istruzioni componenti in base al risultato di un'espressione logica (detta selettore).

```
if (<espressione>)
    <istruzione1>
[else
    <istruzione2>]
```

se il risultato di <espressione> è *vero* (cioe`, diverso da zero) viene eseguita <istruzione1>, altrimenti viene eseguita <istruzione2>.



Oppure .....

switch

```
switch ( integer expression )
{
        case int-const1:  statement1;
                          statement2;
                          break;

        case int-const2:  statement3;
                          statement4;
                          break;
                :
          default:        statement5;
                          statement6;
                          break;
}
```

Iteration

- 2 Basic types of repetition control
- Counter controlled

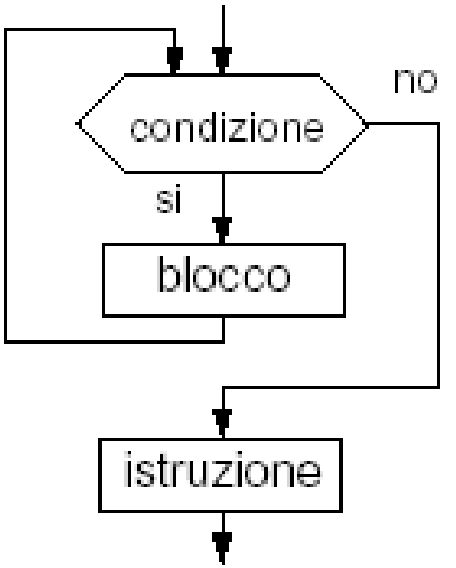
  loop is done until counter reaches a predetermined ending value

  needs: NAME of counter, INITIAL VALUE, INCREMENT amount, and FINAL VALUE

- Sentinel Controlled

  looping continues until some event occurs or some value is encountered

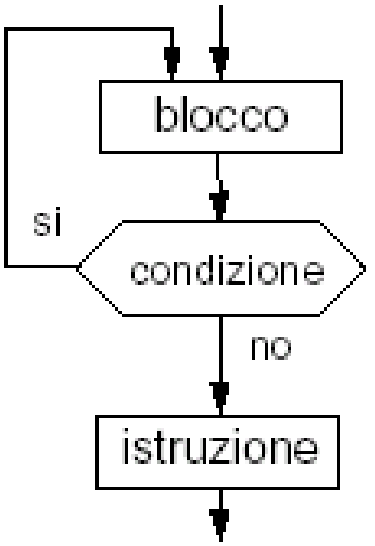
CICLO A CONDIZIONE INIZIALE (Sentinel Controlled)



Sintassi

```
while (<condizione>)
{
    <blocco>
}
<istruzione>
```

CICLO A CONDIZIONE FINALE



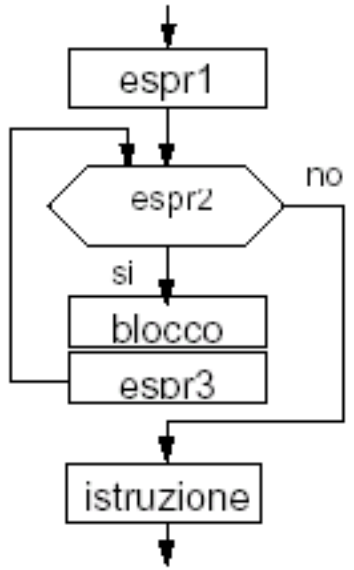
Sintassi

```
do
{
    blocco
} while (<condizione>);
<istruzione>
```

Ciclo con contatore

Il linguaggio C prevede anche un costrutto di ciclo che racchiude **inizializzazione**, **test** e **modifica** di una variabile.

```
for (espr1; espr2; espr3)
    {
      <istr del ciclo>
    }
    <istruzione>
```



espressione1: deve definire il valore iniziale della variabile di conteggio

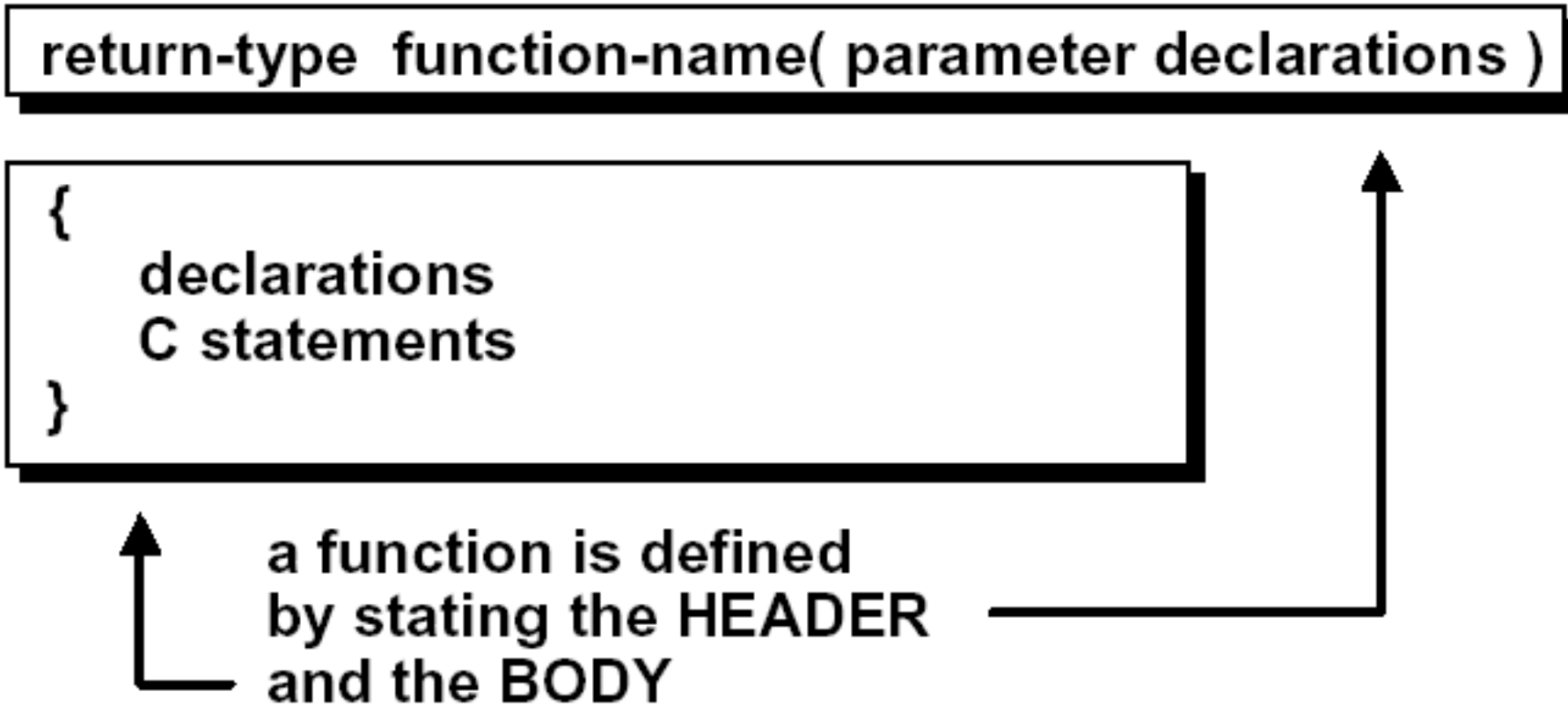
espressione2: deve definire la condizione sul valore finale della variabile di conteggio

espressione3: deve definire la modifica della variabile di conteggio

Funzioni elementari

- Le funzioni sono un modo per comporre le frasi assegnando un NOME alla operazione complessa (funzione) eseguita
- L'operazione complessa può essere esguita in modo parametrico (rispetto ad una lista di parametri formali)
- L'operazione complessa può ritornare un valore (come una $f(x)$)

Function definition general form:



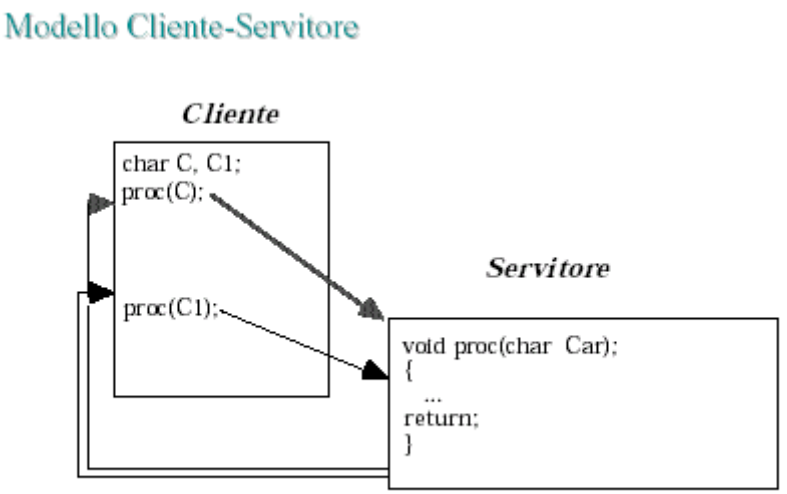
Es. funzione che somma due valori di tipo int e restituisce un int:

```
int somma(int a, int b)
{
        int sum;
        sum = a+b;
        return(sum);
}
```

La chiamata della funzione viene fatta così:

```
void main(void)
{
        int A=23; int B=-31; int risultato;
        risultato = somma(A,B);
        printf("somma= %d\n", risultato);
}
```

Passaggio dei parametri



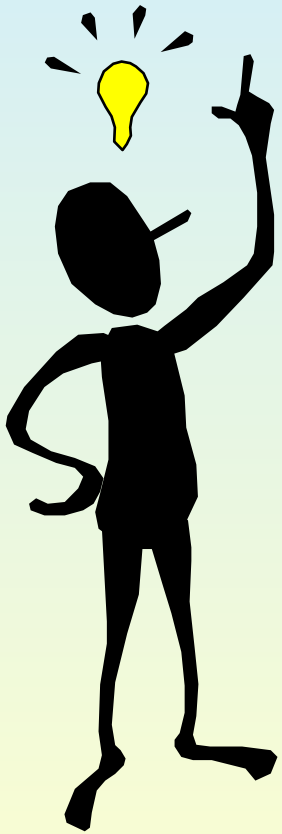
- Il VALORE del parametro ATTUALE viene COPIATO nel parametro FORMALE
- È come se
  - `Car=C` .....
  - `Car=C1`

Per proseguire

- E’ necessario introdurre un nuovo Tipo di variabili
- Un tipo semplice per costruire tipi complessi

Il POINTER

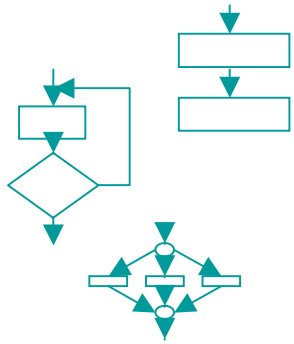
Un cammino tortuoso!



Chiusura dei costruttori dei dati

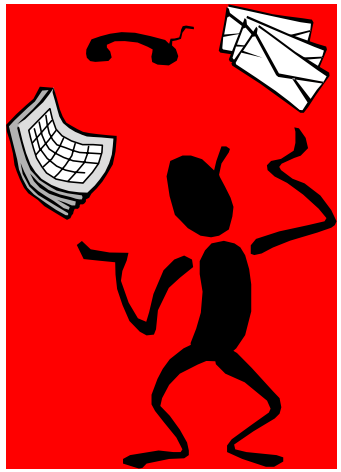
I costruttori

- Tipi strutturati: ***struct***
- Considerazioni sui metodi si strutturazione
  - Sequenza: *struct*
  - Iterazione: *array*
  - Selezione: *union*



- Costruttore di tipi POINTER:

  ☞fuori dagli schemi



- Interazione tra *array, struct e Pointer*