Il Linguaggio C

Cap. 5 – Bellini Guidi

Prof. Salvatore Venticinque

Prof. Mario Magliulo

Il Linguaggio C

- Sviluppato tra il 1969 ed il 1973 presso gli AT&T Bell Laboratories

  - Per uso interno
  - Legato allo sviluppo del sistema operativo Unix

Nel 1978 viene pubblicato “The C Programming Language”, prima specifica ufficiale del linguaggio - Detto “K&R”

Il linguaggio C

Il C è un linguaggio:

- Imperativo ad alto livello
  - ma anche poco astratto
- Strutturato
  - … ma con eccezioni
- Tipizzato
  - Ogni oggetto ha un tipo
- Elementare
  - Poche keyword
- Case sensitive
  - Maiuscolo diverso da minuscolo negli identificatori!
- Portabile
- Standard ANSI

Storia

- Sviluppo
  - 1969-1973
  - Ken Thompson e Dennis Ritchie
  - AT&T Bell Labs
- Versioni del C e Standard
  - K&R (1978)
  - C89 (ANSI X3.159:1989)
  - C90 (ISO/IEC 9899:1990)
  - C99 (ANSI/ISO/IEC 9899:1999, INCITS/ISO/IEC 9899:1999)
- Non tutti i compilatori sono standard!
  - GCC: Quasi C99, con alcune mancanze ed estensioni
  - Borland & Microsoft: Abbastanza C89/C90

Esempio

```
#include<stdio.h>
int main()
{
    int a,b,r;
    a = 5;
    b = 5;
    r = a+b;
    printf("il risultato è %d \n",r);
}
```

Elementi del Linguaggio

Essendo il linguaggio un'astrazione, esistono alcuni fondamentali elementi sintattici essenziali per l'uso del linguaggio stesso:

- commenti
- parole chiave (keyword)
- dati
- identificatori
- Istruzioni
- direttive di compilazione

Gli elementi sintattici definiscono la struttura formale di tutti i linguaggi di programmazione

Commenti

- Testo libero inserito all'interno del programma
- Non viene considerato dal compilatore
- Serve al programmatore, non al sistema!

- Formato:
  - Racchiuso tra /* */
  - Non è possibile annidarli
- Da // fino alla fine della linea

Esempi:

- /* Questo è un commento ! */
- /* Questo /* risulterà in un */ errore */
- // Questo è un altro commento

Esempio

```
#include<stdio.h>
/* questo programma
* esegue la somma di 2 numeri
*/
int main()
{
    int a,b,r;
    a = 5;
    b = 5;
    r = a+b; //risultato
    printf("il risultato è %d \n",r);
}
```

Parole chiave

- Sono vocaboli “riservati” al traduttore per riconoscere elementi del linguaggio
  - Per esempio, le istruzioni sono tutte identificate da una keyword.
  - Ad esempio le parole **int, while, for, if, repeat, else, switch, case**
- Non possono essere usate per altri scopi
- Costituiscono i “mattoni” della sintassi del linguaggio

- Riservate!
- Nel C standard sono 32:

auto double int struct break else long
switch case enum register typedef char
extern return union const float short
unsigned continue for signed void default
goto sizeof volatile do if static while

Keywords

```
int main()
{
    int a,b,r;
    a = 5;
    b = 6;
    r = a+b;
}
```

Keywords

```
int main()
{
    int a,b,r;
    a = 5;
    b = 6;
    if (a >b)
      r = a + b;
    else
      r = a - b
}
```

Keywords

Dati

Dalla parte del calcolatore:

- un dato è un insieme di bit memorizzato in memoria centrale

Dalla parte dell'utente:

- un dato è una quantità associata ad un certo significato

Il linguaggio di programmazione supporta la vista utente

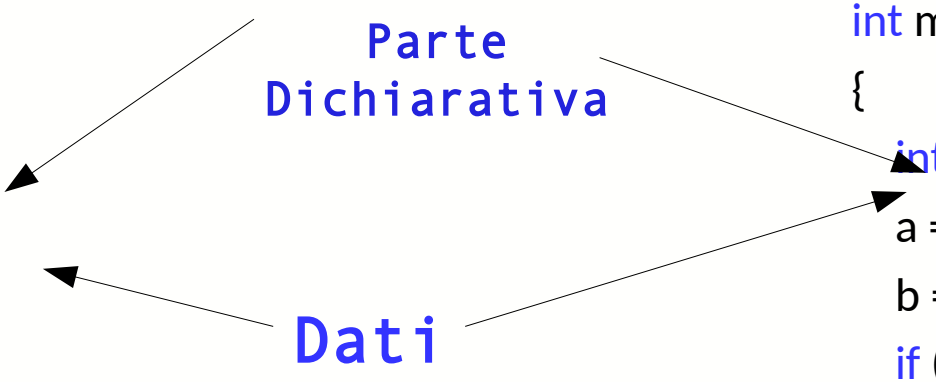
Un dato è individuato da:

- nome (***identificatore***)
- rappresentazione (***tipo***)
- modalità di accesso (***variabile, costante***)

Il dati devono essere dichiarati !!!

I dati

```
int main()
{
   int a,b,r;
   a = 5;
   b = 6;
   r = a+b;
}
```




```
int main()
{
   int a,b,r;
   a = 5;
   b = 6;
   if (a >b)
      r = a + b;
   else
     r = a - b
}
```

a, b, c sono gli **identificatori**.
int è il **Tipo**

Tipi

- Sono quelli forniti direttamente dal C
- Identificati da parole chiave!
  - **char** caratteri ASCII
  - **int** interi (complemento a 2)
  - **float** reali (floating point singola precisione)
  - **double** reali (floating point doppia precisione)

- La dimensione precisa di questi tipi dipende dall'architettura (non definita dal linguaggio)

  |char| = 8 bit = 1 Byte sempre

Dichiarazione di una variabile

- Locazioni di memoria destinate alla memorizzazione di dati il cui valore è modificabile

- Sintassi:

  <tipo> <variabile> ;

  <variabile>: Identificatore che indica il nome della variabile

- Sintassi alternativa (dichiarazioni multiple):

  - <tipo> <lista di variabili>;
  - <lista di variabili >: Lista di identificatori separati da ‘,’

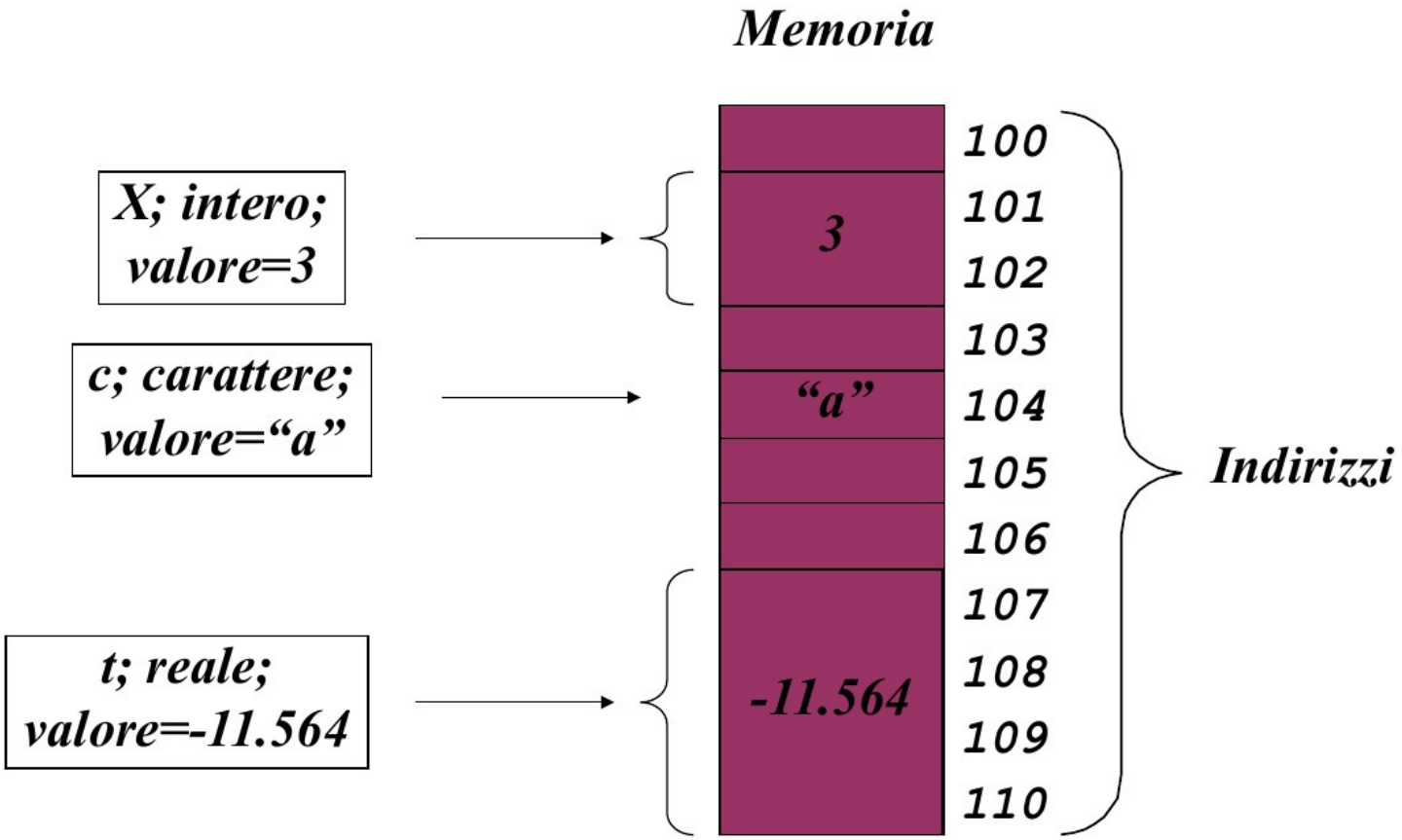
Esempi dichiarazione

- Esempi:
  - int x;
  - char ch;
  - long int x1, x2, x3;
  - double pi;
  - int stipendio;
  - long y, z;
- Usiamo nomi significativi!

Esempi:

- int x0a11; /* NO */
- int valore; /* SI */
- float raggio; /* SI */

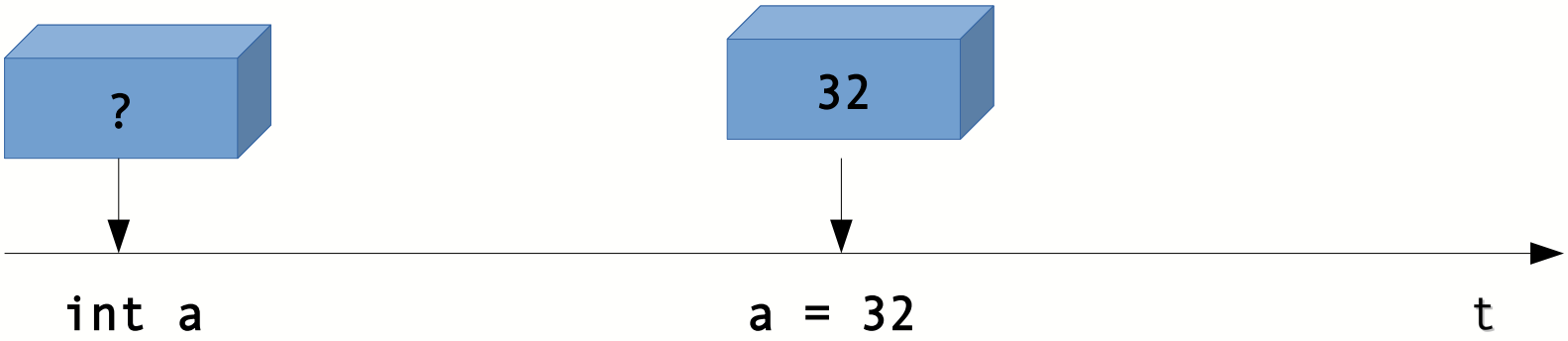
Una variabile nel calcolatore



- Ogni variabile, in ogni istante di tempo, possiede un certo valore
- Le variabili appena definite hanno valore ignoto
  - Variabili non inizializzate
- In momenti diversi il valore può cambiare

Operatore Assegnazione

- E' un'istruzione
- Copia un valore in una variabile



Valori Costanti

Esempi di valori attribuibili ad una variabile:

- Costanti di tipo char: **'f'**
- Costanti di tipo *stringa*: **“hallo world”**
- Costanti di tipo *int*, *short*, *long*
  - ***26***
  - **0x1a,0X1a**
- Costanti di tipo *float*, *double*
  - **212.6**
  - **2.126e2, -2.126E2, -212.6f**

Operatore di indirizzo &

L'**operatore unario** di **indirizzo** **&** restituisce l'indirizzo della locazione di memoria dell'**operando**

Es.:

&a ammesso

&(a+1) non ammesso

&a = b non ammesso

L'indirizzo di memoria di una variabile non può essere modificato in un'istruzione, ma solo usato come riferimento in quanto è predeterminato e non modificabile.

Il puntatore come tipo

Il puntatore è un tipo di variabile il cui valore è un indirizzo

Es:

int* pointer;

dichiara la variabile **pointer**, **puntatore** *ad una qualunque variabile di tipo* **int**

Un dato **puntatore** può puntare solo a un determinato **tipo** di variabili, quello specificato nella dichiarazione

Esistono puntatori per ogni tipo di variabile puntata

Area del triangolo

```
#include <stdio.h>

int main()
{
	//parte dichiarativa
	int base;
	int altezza;
	int area;

	//parte esecutiva
	base = 3;
	altezza = 7;
	area = base * altezza;
	area = area/2;

	printf("area= %d\n",area);
}
```

Scambio di due variabili

```
#include <stdio.h>
int main()
{
	//parte dicharativa
	int a;
	int b;
	int temp;
	//parte esecutiva
	a = 3;
	b = 7;
	temp = a;
	a = b;
	b = temp;

	printf("a= %d b= %d\n",a,b);

}
```

Strutture a blocchi

- In C, è possibile raccogliere istruzioni in blocchi racchiudendole tra parentesi graffe
- Significato: Delimitazione di un ambiente di visibilità di “oggetti” (variabili, costanti)
- Corrispondente ad una “sequenza” di istruzioni

Esempio:

```
{
    int a;
    int b ;
    a = 2;
    b = 2*a
}
```

a e b sono definite solo all’interno del blocco!

Direttive di compilazione

- Sono comandi al compilatore
- Non vengono tradotte in istruzioni in linguaggio macchina

  Esempi:

  #define

  #include

#define

- Viene utilizzato per dichiarare una costante
- Non c'è allocazione di memoria
- E' solo un'etichetta che viene sostituita dal valore corrispondente

Area del Cerchio

```
#include <stdio.h>
#define pi 3.14

int main()
{
    //parte dicharativa
    int raggio;
    int area;

    //parte esecutiva
    raggio = 3;
    area = raggio*raggio*pi;

    printf("area= %d\n",area);

}
```

Include

- Dice al compilatore quali librerie utilizzare
- Il compilatore cerca nel file specificato dalla direttiva l'esistenza e il formato delle funzioni di libreria
- Solo in fase di linking viene collegato al programma il codice che realizza le funzioni.

Es: #include<stdio.h>

#include<math.h>

I/O Formattato

- Output
- istruzione printf()

- L'utilizzo di queste istruzioni richiede l'inserimento di una direttiva

  #include <stdio.h>

  all'inizio del file sorgente

- Significato: "includi il file stdio.h"

- Contiene alcune dichiarazioni

- Sintassi: printf(<formato>,<arg1>,…,<argn >);
-

<formato>: Sequenza di caratteri che determina il formato di stampa di ognuno dei vari argomenti

Può contenere:

- Caratteri (stampati come appaiono)
- Direttive di formato nella forma %<carattere>
  - %d intero
  - %u unsigned
  - %s stringa
  - %c carattere
  - %x esadecimale
  - %o ottale
  - %f float
  - %g double

<arg1>,…,<argn>: Le quantità (espressioni) che si vogliono stampare associati alle direttive di formato nello stesso ordine!

Esempi printf

```
int x=2;
float z=0.5;
char c='a';

printf("%d %f  %c\n",x,z,c);

printf("%f***%c***%d\n",z,c,x);
```

output

```
2  0.5  a
```

output

```
0.5***a***2
```

Flusso di controllo

- L'**ordine** di esecuzione delle operazioni elementari è determinante per la soluzione del problema
- Le operazioni elementari(passi di algoritmi) vengono chiamate **istruzioni** nel linguaggio dei calcolatori e possono essere classificate in :
  - *istruzioni non condizionate*
  - *istruzioni condizionate*: l'esecuzione dipende da una condizione
  - *istruzioni di controllo*: esprimono le condizioni da cui dipende l'esecuzione delle istruzioni condizionate(dette *pseudo-istruzioni* perché controllano solo il flusso delle operazioni)
- Le istruzioni possono essere composte in ***blocchi*** o ***sequenze*** che risolvono sottoproblemi del problema principale e sono viste come un'istruzione elementare

Costrutto

- Insieme di istruzioni di controllo e controllate:

  - costrutto condizionale: insieme di condizione e istruzioni condizionate

  - costrutto iterativo: insieme di istruzioni la cui esecuzione viene ripetuta sotto il controllo di opportune istruzioni di controllo

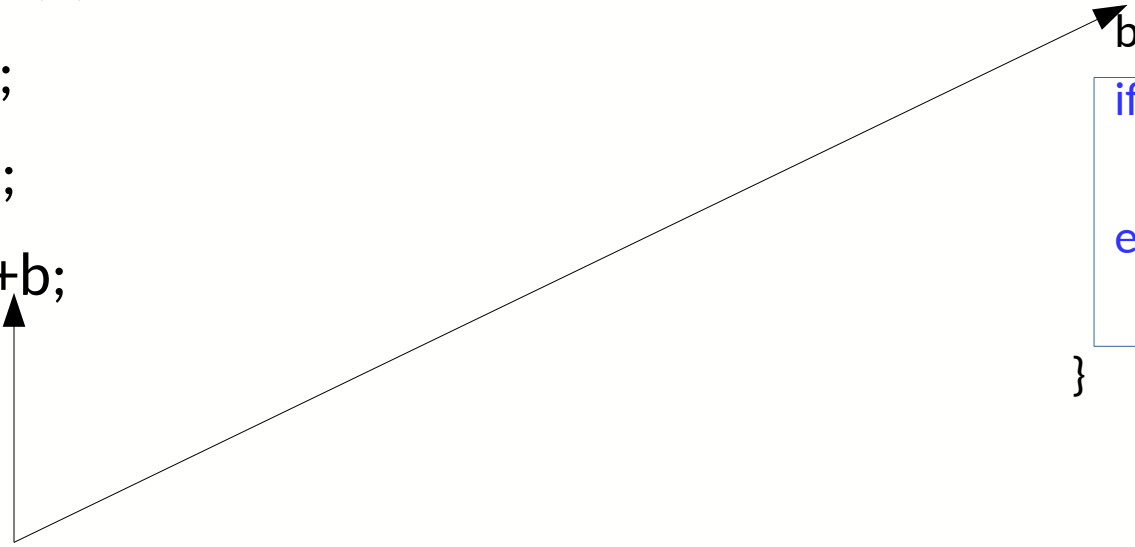
Costrutti

```
int main()
{
    int a,b,r;
    a = 5;
    b = 6;
    r = a+b;
}
```

Istruzioni sequenziali

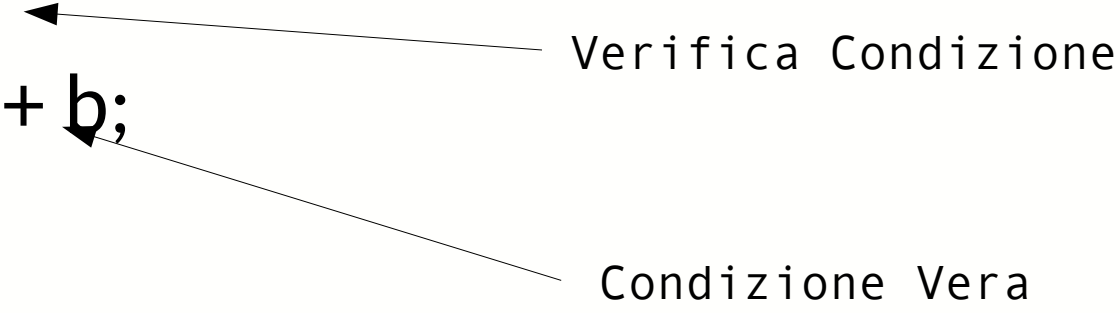
```
int main()
{
    int a,b,r;
    a = 5;
    b = 6;
    if (a >b)
        r = a + b;
    else
        r = a - b
}
```

Costrutto Condizionale



if-then

```
if (a >b)
      r = a + b;
```

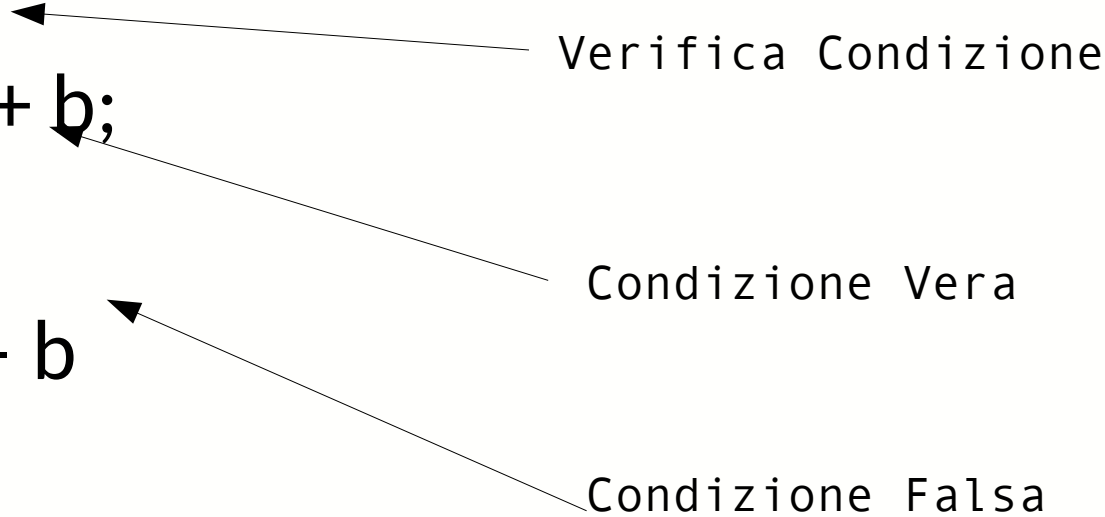


...

[Altre Istruzioni]

if-then-else

```
if (a >b)
      r = a + b;
else
      r = a – b
...
[Altre Istruzioni]
```



While

```
int main()
{
   int a;
   a = 5;
   while (a>0)
      a = a -1;
   printf (“a=%d”,a);
}
```

Condizione Vera

```
int main()
{
   int a;
   a = 5;
   while (a>0)
   {
      a = a -1;
   }
   printf (“a=%d”,a);
}
```

Costrutto Iterativo