

LINGUAGGI DI PROGRAMMAZIONE IL LINGUAGGIO C

Fondamenti di Programmazione 2021/2022

Francesco Tortorella




Linguaggi di programmazione

- Abbiamo visto come, all'interno dell'esecutore automatico, le informazioni si possono codificare come sequenze di bit memorizzate nei registri.
- Una cosa simile avviene anche per le **istruzioni**, cioè le singole azioni elementari che l'unità centrale può eseguire.



Linguaggi di programmazione

- Nello specificare un'istruzione, bisogna precisare l'**operazione** da compiere e i **dati** coinvolti nell'operazione.

- Esempio:


- Come rappresentare le operazioni ?
- L'insieme delle diverse operazioni che l'unità centrale è in grado di eseguire è **finito** e quindi è possibile codificarlo con un certo numero di bit (**codice operativo**).

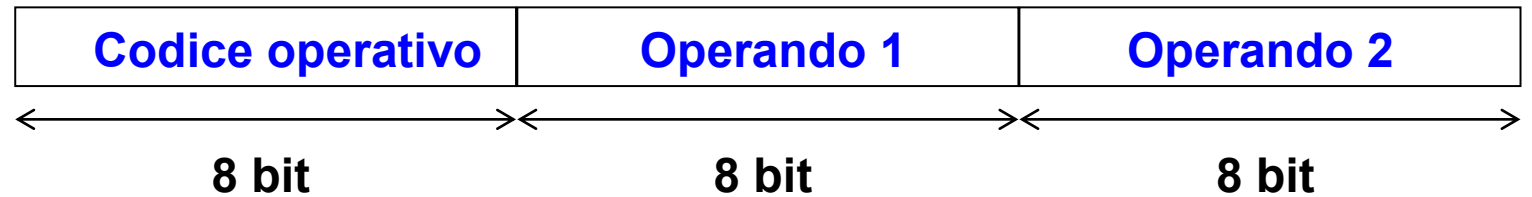
somma	0000
sottrai	0001
moltiplica	0010
dividi	0011
...	...



Linguaggi di programmazione

- Una istruzione sarà quindi rappresentabile da una sequenza di bit divisa in due parti:

- un **codice operativo**
- uno o più **operandi**



- In questo modo, un esecutore automatico basato sul modello di von Neumann permette la memorizzazione e l'esecuzione di un **programma**, cioè di una sequenza di istruzioni che realizzano un particolare algoritmo e che sono descritte nel linguaggio interpretabile dal calcolatore, ma ...



Linguaggio macchina

- ... ma quali sono le caratteristiche di tale linguaggio ?
 - è codificato tramite sequenze di bit
 - ogni istruzione può compiere solo azioni molto semplici
 - non gestisce direttamente i tipi di dati di interesse
 - è strettamente legato alla particolare macchina su cui è definito
- Non a caso viene definito **linguaggio macchina**



Scrivere un programma ...

- Se si volesse implementare un dato algoritmo scrivendo un programma in linguaggio macchina sarebbe quindi necessario:
 - conoscere dettagliatamente tutti i codici operativi e la loro codifica
 - decidere in quali registri vadano memorizzati i dati
 - determinare, per ogni singola operazione richiesta dall'algoritmo, la sequenza di istruzioni in linguaggio macchina che la realizzano
 - definire un'opportuna tecnica di codifica per ogni tipo di dati considerato
 - limitarsi a utilizzare solo i calcolatori per cui esista una tale competenza, tenendo comunque presente che il programma scritto per un certo calcolatore non è eseguibile su altre macchine

Impresa difficile, ma non impossibile ...



Il gap semantico

Esecutore umano

- linguaggio naturale
- gestione completa dei tipi
- istruzioni semanticamente ricche

Calcolatore

- linguaggio rigido e complicato
- gestione dei tipi quasi nulla
- istruzioni estremamente semplici



Il gap semantico

Esecutore umano

- linguaggio naturale
- gestione completa dei tipi
- istruzioni semanticamente ricche



orientato al problema

Linguaggio di programmazione

- linguaggio formale, con costrutti precisi per la definizione dei dati e delle operazioni
- gestione completa dei tipi fondamentali; possibilità di definire tipi strutturati
- istruzioni che realizzano le principali azioni elaborative richieste

orientato alla macchina



Calcolatore

- linguaggio rigido e complicato
- gestione dei tipi quasi nulla
- istruzioni estremamente semplici



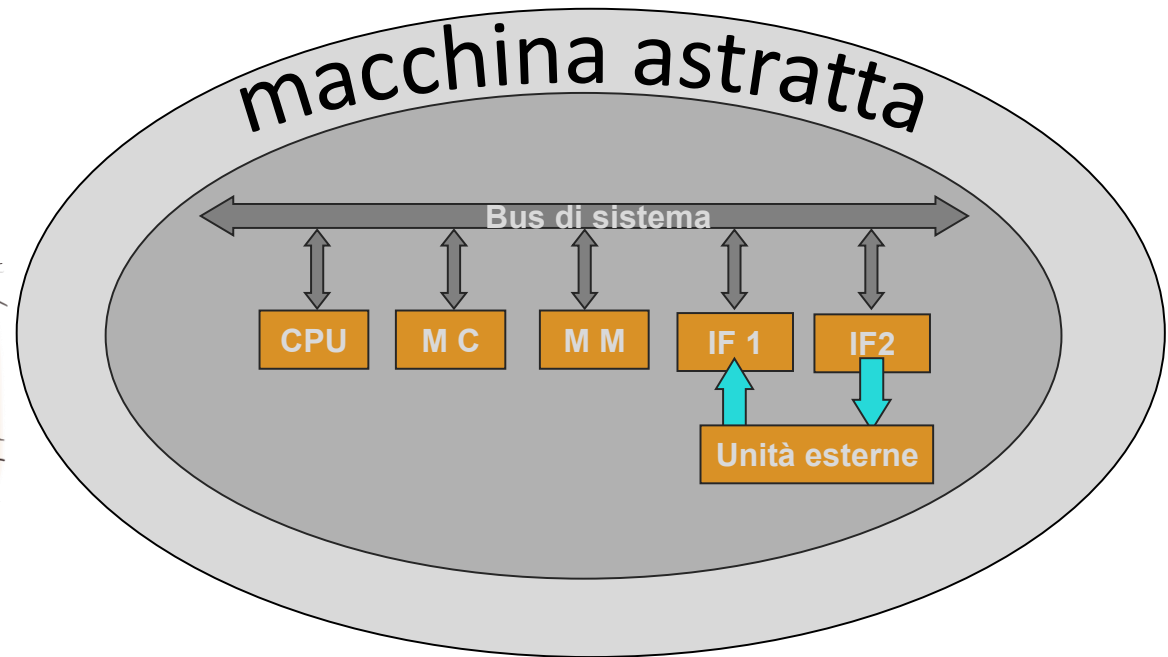
Vantaggi

- L'uso di un linguaggio di programmazione permette di :
 - realizzare un programma che implementa l'algoritmo in maniera precisa ed in un linguaggio *ad alto livello*
 - trascurare tutti i dettagli relativi alla rappresentazione dei dati nei registri
 - definire un programma che non dipende dal particolare calcolatore su cui è stato realizzato



Linguaggio=macchina virtuale

In effetti, l'utente non deve interagire con la macchina reale e le sue limitazioni, ma “vede” una **macchina astratta** che nasconde le particolarità della macchina reale e con la quale è molto più agevole interagire



Dal linguaggio ad alto livello al linguaggio macchina

- I linguaggi di programmazione ad alto livello sono **linguaggi formali**, in cui la forma delle frasi, cioè la **sintassi**, e il loro significato, la **semantica**, sono definiti sulla base di regole rigide e precise.
- In tal modo viene eliminata l'ambiguità e le ridondanze tipiche del linguaggio naturale ed **è possibile realizzare in modo automatico l'analisi di un programma scritto in un linguaggio ad alto livello e la sua traduzione in linguaggio macchina.**
- I programmi che svolgono il compito di tradurre un programma in linguaggio ad alto livello in un programma in linguaggio macchina sono detti **traduttori** (*compilatori o interpreti*).



Il linguaggio C

```
/* Semplice programma in C */  
#include <stdio.h>  
  
int main()  
{  
    printf("Salve, mondo!\n");  
    return (0);  
}
```



Cominciamo con i tipi...

- Il linguaggio C prevede che siano esplicitamente dichiarato i tipi di tutte le variabili.
- In C sono disponibili vari tipi di dato.
- **Tipi numerici:**
 - `int`
 - `float`
 - `double`
- **Tipi non numerici:**
 - `char`
 - `bool`
 - `void`



Il tipo `int`

- È costituito da un sottoinsieme limitato dei numeri interi
- Caratteristiche:
 - Dimensione: 4 bytes
 - Valore minimo: -2147483648
 - Valore massimo: +2147483647
- Operazioni ammesse:
 - Assegnazione =
 - Somma +
 - Sottrazione -
 - Moltiplicazione *
 - Divisione /
 - Modulo %
 - Confronto > < >= <= == !=



Il tipo `float`

- È costituito da un sottoinsieme limitato e discreto dei numeri reali
- Caratteristiche:
 - Dimensione: 4 bytes
 - Valore minimo (abs): $3.4E-38$
 - Valore massimo (abs): $3.4E+38$
- Operazioni ammesse:
 - Assegnazione =
 - Somma +
 - Sottrazione -
 - Moltiplicazione *
 - Divisione /
 - Confronto > < >= <= == !=



Il tipo double

- È costituito da un sottoinsieme limitato e discreto dei numeri reali, ma con range e precisione maggiore rispetto a float (doppia precisione)
- Caratteristiche:
 - Dimensione: 8 bytes
 - Valore minimo (abs): $1.7E-308$
 - Valore massimo (abs): $1.7E+308$
- Operazioni ammesse:
 - Assegnazione =
 - Somma +
 - Sottrazione -
 - Moltiplicazione *
 - Divisione /
 - Confronto > < >= <= == !=



Il tipo `char`

- Consiste in un insieme di caratteri, alcuni stampabili (caratteri alfabetici, cifre, caratteri di punteggiatura, ecc.) ed altri non stampabili tramite i quali si gestisce il formato dell'input/output (**caratteri di controllo**).
- I sottoinsiemi delle lettere e delle cifre sono ordinati e coerenti.
- Per la rappresentazione interna, viene tipicamente usato il codice ASCII, che mette in corrispondenza ogni carattere con un numero intero compreso tra 0 e 255.



ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]



Il tipo `char`

- Di fatto anche `char` è un tipo numerico
- Caratteristiche:
 - Dimensione: 1 byte
 - Valore minimo: -128
 - Valore massimo: +127
- Sono permesse le operazioni aritmetiche tipiche degli interi



Il tipo `bool`

- È un tipo costituito dai due soli valori **false** e **true**, corrispondenti a falso e vero e rappresentati da 0 e 1. Il tipo rappresenta le informazioni di tipo logico (es. il risultato di un confronto, il verificarsi di una situazione).
- Caratteristiche:
 - Dimensione: 1 byte
 - Valore minimo: **false**
 - Valore massimo: **true**
- Operazioni ammesse
 - assegnazione =
 - disgiunzione ||
 - congiunzione &&
 - negazione !

Per utilizzare il tipo `bool`, è necessario inserire la direttiva `#include <stdbool.h>`



Modificatori di tipo

- Sono usati per creare nuovi tipi modificando i tipi base.
- **signed** e **unsigned**: senza ulteriori specificazioni (default), i tipi sono signed. Con il modificatore **unsigned**, il tipo è in grado di contenere soltanto valori non negativi
- **unsigned char**: 0...255
- **unsigned int**: 0...4294967295



Modificatori di tipo

- **short** e **long**: modificano l'estensione del tipo
 - `short int`: 2 byte
 - `long int`: 4 byte
 - `long double`: 12 byte
- I modificatori possono combinarsi:
 - `unsigned short int`
 - `unsigned long int`



Variabili

- Per usare una variabile, questa deve essere dapprima **definita**.
- La definizione rende disponibile (da qualche parte in memoria) la variabile che mantiene il tipo assegnato nella definizione fino al termine del programma.
- La sintassi è `<tipo> nome_variabile;`
- Esempi:
 - `int a;`
 - `int a,b,c;`
 - `float x,y,z;`



Definizione di variabili

- Il nome della variabile non può coincidere con una delle parole chiave riservate del C:
`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 _Bool _Imaginary restrict _Complex`
`inline`



Definizione di variabili

- I caratteri ammessi sono lettere, cifre e carattere di sottolineatura (underscore `_`), messi in qualunque ordine, purché il primo carattere del nome sia una lettera o l'underscore (sconsigliato).
- C'è differenza tra caratteri minuscoli e maiuscoli (case sensitive), per cui **a** e **A** sono due variabili diverse.
- Nello scegliere il nome per le variabili, è consigliabile orientarsi verso nomi significativi del ruolo della variabile nel programma.



Esempi

- Definizioni corrette:
 - `int Pippo, a31;`
 - `float pippo, radice_equazione;`
 - `double Vercingetorige;`
- Definizioni errate:
 - `double 27pluto;`
 - `int conta num;`
 - `Float x,y,z;`



Costanti

- Il C prevede tre modalità per definire delle costanti:
 - **Costanti letterali** (literals)
 - **Costanti definite** (`#define`)
 - **Costanti dichiarate** (`const`)



Costanti letterali

- **Il valore della costante è rappresentato direttamente.**
- **Costanti di tipo `int`**
 - Sono definite come sequenze di cifre decimali, eventualmente precedute da un segno (+ o -):
0 -1 3256 +34 12L 33U 5321UL 0713 0X12FF 0XFUL
- **Costanti di tipo `float`**
 - Sono definite come sequenze di cifre decimali, eventualmente precedute da un segno (+ o -), strutturate in virgola fissa o in virgola mobile (floating point):
0.1 -3.7 0.0001 1.0E-4 -7.6E12 4.



Costanti letterali

■ Costanti di tipo **char**

- sono definite come caratteri racchiusi tra singoli apici ('):
'x' 'A' '\n' '\t' '2'

■ Costanti di tipo stringa di caratteri

- sono definite come sequenze di caratteri racchiusi tra doppi apici ("):
"Pippo" "Valore di x: " "x"

■ Costanti di tipo **bool**

- sono solo due: **false** e **true**



Costanti definite

- Viene utilizzata la direttiva **#define** che permette di associare **testualmente** un valore ad un identificatore

```
#define MAX 12
```

```
#define PI 3.14
```

- All'atto della compilazione il **preprocessore** sostituisce ogni occorrenza dell'identificatore con il valore corrispondente



Costanti definite

```
#define MAX 10

int main(){
    int a,b,i;

    a = MAX;
    b = MAX/2 + 1;
    i = 0;

    while(i < MAX)
        b = b+1;
}
```



preprocessore

```
int main(){
    int a,b,i;

    a = 10;
    b = 10/2 + 1;
    i = 0;

    while(i < 10)
        b = b+1;
}
```

Costanti dichiarate

- Il valore viene associato ad un identificatore e ne viene specificato anche il tipo:

```
const int MIN=0;
```

```
const float PI=3.14;
```

- In questo caso l'identificatore è a tutti gli effetti una variabile non modificabile.

- Qual è la differenza?

```
const int pippo=1;
```

```
#define pippo 1
```



Operatori

- Un **operatore** specifica un'operazione da eseguire su uno o due **operandi** definendo un'**espressione**.
- L'ordine con cui sono valutati definisce la precedenza degli operatori. Può essere alterata con l'uso delle parentesi ().

precedenza

Alcuni operatori	Associazione
! - (unario) + (unario)	destra
* (moltiplicazione) / %	sinistra
+ (binario) - (binario)	sinistra
< <= > >=	sinistra
== !=	sinistra
&&	sinistra
	sinistra
=	destra



Espressioni

- Un'espressione consiste in un operando o in una combinazione di operandi e operatori.
- La valutazione di un'espressione porta al calcolo di un valore appartenente ad un tipo specifico.

- Esempi:

`2+3` `2.1*3.5+pi greco` `a>=2`

`(7+2) / 3` `(a>2) && (b==0)` `! trovato`



Espressioni

- Nel caso ci siano operandi di tipo diverso, l'espressione assume il tipo "più ampio" tra quelli presenti.
- Esempio:
 $\text{int op float} \rightarrow \text{float}$
 $7+5*2.4 \rightarrow 7+12.0 \rightarrow 19.0$
- **Achtung!** Qual è il valore dell'espressione?
 $7.5+1/2$
- Questo effetto va sotto il nome di **casting implicito**.



Espressioni

- **Alcuni casi di casting implicito**

`int op float → float`

`char op int → int`

`float op double → double`

`int op double → double`



Espressioni

- È possibile modificare il **valore del tipo** di un'espressione, indicando **esplicitamente** il tipo da impiegare.
- Esempio:
 $7.5 + (\text{float}) 1/2 \quad \rightarrow \quad 7.5 + 1.0/2$
- Questa operazione va sotto il nome di **casting esplicito**.

