

FONDAMENTI DI PROGRAMMAZIONE

VARIABILI E OPERATORI IN C

Dott. Federico Concone

federico.concone@unipa.it

Informazioni utili

- Ricevimento:
 - Edificio 6, 3° Piano, Laboratorio Intelligenza Artificiale e Sistemi Distribuiti;
- Orario ricevimento:
 - Da concordare tramite e-mail;
- Testo consigliato:
 - Deitel, Deitel, Il linguaggio C – Fondamenti e tecniche di programmazione 8 Ed., Pearson, Italia, 2016
- Link materiale:
 - goo.gl/mrcWL7

Sommario

- Un programma d'esempio;
- Le variabili;
- I tipi di dati del C;
- Funzione ***scanf()***;
- Specifiche di conversione;
- Operatori in C;
- Esercizi;

Un programma d'esempio

```
1 // Fig. 2.5: fig02_05.c
2 // Addition program.
3 #include <stdio.h>
4
5 // function main begins program execution
6 int main( void )
7 {
8     int integer1; // first number to be entered by user
9     int integer2; // second number to be entered by user
10    int sum; // variable in which sum will be stored
11
12    printf( "Enter first integer\n" ); // prompt
13    scanf( "%d", &integer1 ); // read an integer
14
15    printf( "Enter second integer\n" ); // prompt
16    scanf( "%d", &integer2 ); // read an integer
17
18    sum = integer1 + integer2; // assign total to sum
19
20    printf( "Sum is %d\n", sum ); // print sum
21 }
```

```
Enter first integer
45
Enter second integer
72
Sum is 117
```

Un programma d'esempio

```
3  #include <stdio.h>
4
5  // function main begins program execution
6  int main( void )
7  {
8      int integer1; // first number to be entered by user
9      int integer2; // second number to be entered by user
10     int sum; // variable in which sum will be stored
11
12     printf( "Enter first integer\n" ); // prompt
13     scanf( "%d", &integer1 ); // read an integer
14
15     printf( "Enter second integer\n" ); // prompt
16     scanf( "%d", &integer2 ); // read an integer
17
18     sum = integer1 + integer2; // assign total to sum
19
20     printf( "Sum is %d\n", sum ); // print sum
21 } // end function main
```

- Che cosa sono ***integer1***, ***integer2*** e ***sum***?
- Cosa fa ***scanf()***?
- A cosa serve “***%d***” utilizzato dalle funzioni ***printf()*** e ***scanf()***?
- Che cosa significa fare l'azione alla *riga 18*?

Le variabili

- Che cosa sono *integer1*, *integer2* e *sum*?

```
8      int integer1; // first number to be entered by user
9      int integer2; // second number to be entered by user
10     int sum; // variable in which sum will be stored
```

- Sono ***variabili***:
 - Una ***variabile*** rappresenta uno spazio di memoria che contiene un valore;
 - Una ***variabile*** è identificata da un ***tipo*** e da un ***nome univoco***:
 - Il ***tipo*** ci dice se la variabile è *intera*, *in virgola mobile*, *un carattere* ecc;
 - Il ***nome univoco*** permette di avere un modo mnemonico per riferirci a quella variabile;

tipo nome_univoco

Le variabili

- Che cosa sono *integer1*, *integer2* e *sum*?

```
8      int integer1; // first number to be entered by user
9      int integer2; // second number to be entered by user
10     int sum; // variable in which sum will be stored
```

- Sono **variabili**:
 - Una **variabile** rappresenta uno spazio di memoria che contiene un valore;
 - Una **variabile** è identificata da un **tipo** e da un **nome univoco**:
 - Il **tipo** ci dice se la variabile è *intera*, *in virgola mobile*, *un carattere* ecc;
 - Il **nome univoco** permette di avere un modo mnemonico per riferirci a quella variabile;

tipo	nome_univoco
int	integer1

Le variabili

- Che cosa sono *integer1*, *integer2* e *sum*?

```
8      int integer1; // first number to be entered by user
9      int integer2; // second number to be entered by user
10     int sum; // variable in which sum will be stored
```

- Tutte le variabili ***PRIMA DI ESSERE UTILIZZATE DEVONO ESSERE DICHIARATE;***
- La *dichiarazione di una variabile* equivale a *riservare un'area di memoria*:
 - Per dichiarare una variabile intera -> ***int*** pippo;
 - Per inizializzare una variabile -> pippo = 10;


Le variabili

- Che cosa sono *integer1*, *integer2* e *sum*?

```
8      int integer1; // first number to be entered by user
9      int integer2; // second number to be entered by user
10     int sum; // variable in which sum will be stored
```

- Tutte le variabili ***PRIMA DI ESSERE UTILIZZATE DEVONO ESSERE DICHIARATE;***
- La *dichiarazione di una variabile* equivale a *riservare un'area di memoria*:
 - Per dichiarare una variabile intera -> ***int*** pippo;
 - Per inizializzare una variabile -> pippo = 10;
 - Per dichiarare e inizializzare ***in-line*** -> ***int*** pippo = 10;

I tipi di dati del C



N. Di byte	Data type
	<i>Floating-point types</i>
	long double
8 (64 bit)	double
4 (32 bit)	float
	<i>Integer types</i>
	unsigned long long int
	long long int
	unsigned long int
	long int
	unsigned int
4 (32 bit)	int
	unsigned short
2 (16 bit)	short
1 (8 bit)	char

← Vengono utilizzati per rappresentare numeri in virgola mobile (numeri reali);

← Vengono utilizzati per rappresentare numeri interi;

← Vengono utilizzati per rappresentare caratteri definiti dallo standard ASCII;

I tipi di dati del C

N. Di byte	Data type
<i>Floating-point types</i>	
	long double
8 (64 bit)	double
4 (32 bit)	float
<i>Integer types</i>	
	unsigned long long int
	long long int
	unsigned long int
	long int
	unsigned int
4 (32 bit)	int
	unsigned short
2 (16 bit)	short
1 (8 bit)	char

- L'uso dei bit varia a seconda del fatto che specifichiamo se il tipo è **signed** (con segno) o **unsigned** (senza segno);
- Per esempio $(114)_{10} = (?)_2$
 - 7 bit -> *signed char* è sufficiente;
 - posso usare il **bit più significativo** come **segno del numero**:
 - '+' se il bit è 0;
 - '-' se il bit è 1;
 - $(114)_{10} = (01110010)_2$
 - $(-114)_{10} = (\underline{10001101})_2$

I tipi di dati del C

N. Di byte	Data type
<i>Floating-point types</i>	
	long double
8 (64 bit)	double
4 (32 bit)	float
<i>Integer types</i>	
	unsigned long long int
	long long int
	unsigned long int
	long int
	unsigned int
4 (32 bit)	int
	unsigned short
2 (16 bit)	short
1 (8 bit)	char

- L'uso dei bit varia a seconda del fatto che specifichiamo se il tipo è **signed** (con segno) o **unsigned** (senza segno);

- In generale:


- **signed:**

$$-2^{(n-1)} \leq \text{valori} \leq 2^{(n-1)}-1$$

- **unsigned:**

$$0 \leq \text{valori} \leq 2^n-1$$

I tipi di dati del C



N. Di byte	Data type
<i>Floating-point types</i>	
	long double
8 (64 bit)	double
4 (32 bit)	float
<i>Integer types</i>	
	unsigned long long int
	long long int
	unsigned long int
	long int
	unsigned int
4 (32 bit)	int
	unsigned short
2 (16 bit)	short
1 (8 bit)	char

- Il numero di bytes necessari per rappresentare un tipo cambia da macchina a macchina;
- Per chiarire facciamo un esempio:
 - tipi_primitivi.c

Funzione *scanf()*

- Cosa fa *scanf()*?

```
13 scanf( "%d", &integer1 ); // read an integer
```

- La funzione *scanf()* è inclusa nella libreria **<stdio.h>** e permette di leggere dallo *standard input* (di solito la tastiera) un generico valore.
- Per poter utilizzare correttamente la funzione, si deve **PRIMA DICHIARARE** la variabile che conterrà il valore e dopo invocare la *scanf()*:

```
int pippo;  
scanf( "%d" , &pippo);
```

Funzione *scanf()*

- Cosa fa *scanf()*?

```
13 scanf( "%d", &integer1 ); // read an integer
```

- La funzione *scanf()* è inclusa nella libreria **<stdio.h>** e permette di leggere dallo *standard input* (di solito la tastiera) un generico valore.
- Per poter utilizzare correttamente la funzione, si deve **PRIMA DICHIARARE** la variabile che conterrà il valore e dopo invocare la *scanf()*:

```
int pippo;  
scanf( "%d" , &pippo);
```

Approfondiremo
successivamente il
significato del
simbolo '&'

Specifiche di conversione

- A cosa serve “%d”, utilizzato dalle funzioni **printf()** e **scanf()**?

```
13 scanf( "%d", &integer1 ); // read an integer
20 printf( "Sum is %d\n", sum ); // print sum
```

Data type	printf conversion specification	scanf conversion specification
<i>Floating-point types</i>		
long double	%Lf	%Lf
double	%f	%lf
float	%f	%f
<i>Integer types</i>		
unsigned long long int	%llu	%llu
long long int	%lld	%lld
unsigned long int	%lu	%lu
long int	%ld	%ld
unsigned int	%u	%u
int	%d	%d
unsigned short	%hu	%hu
short	%hd	%hd
char	%c	%c

Operatori in C

- Cos'è l'istruzione alla riga 18?

```
18    sum = integer1 + integer2; // assign total to sum
```

- In C esistono 4 classi principali di operatori:
 - assegnamento;
 - aritmetici;
 - relazionali;
 - logici;

Operatori in C

- Cos'è l'istruzione alla riga 18?

18 `sum = integer1 + integer2; // assign total to sum`

- In C esistono 4 classi principali di operatori:

- ***assegnamento;***
- aritmetici;
- relazionali;
- logici;



La forma più generale in cui si usa questo operatore è:

nome_variabile = espressione;

Con l'operatore assegnamento:


il valore dell'espressione viene immagazzinato nella posizione di memoria associata ad una variabile;

Operatori in C

- Cos'è l'istruzione alla riga 18?

18 `sum = integer1 + integer2; // assign total to sum`

- In C esistono 4 classi principali di operatori:
 - assegnamento
 - **aritmetici**;
 - relazionali;
 - logici;



C operation	Arithmetic operator	Algebraic expression	C expression
Addition	+	$f + 7$	<code>f + 7</code>
Subtraction	-	$p - c$	<code>p - c</code>
Multiplication	*	bm	<code>b * m</code>
Division	/	x / y or $\frac{x}{y}$ or $x \div y$	<code>x / y</code>
Remainder	%	$r \bmod s$	<code>r % s</code>


Operatori in C

- Cos'è l'istruzione alla riga 18?

18 `sum = integer1 + integer2; // assign total to sum`

- In C esistono 4 classi principali di operatori:

- assegnamento
- aritmetici;
- **relazionali;**
- logici;



C equality or relational operator	Example of C condition	Meaning of C condition
<code>==</code>	<code>x == y</code>	x is equal to y
<code>!=</code>	<code>x != y</code>	x is not equal to y
<code>></code>	<code>x > y</code>	x is greater than y
<code><</code>	<code>x < y</code>	x is less than y
<code>>=</code>	<code>x >= y</code>	x is greater than or equal to y
<code><=</code>	<code>x <= y</code>	x is less than or equal to y

Operatori in C

- Cos'è l'istruzione alla riga 18?

18 `sum = integer1 + integer2; // assign total to sum`

- In C esistono 4 classi principali di operatori:

- assegnamento
- aritmetici;
- relazionali;
- **logici;**

AND logico &&

expression1	expression2	expression1 && expression2
0	0	0
0	nonzero	0
nonzero	0	0
nonzero	nonzero	1

NOT logico !

expression	!expression
0	1
nonzero	0

OR logico ||

expression1	expression2	expression1 expression2
0	0	0
0	nonzero	1
nonzero	0	1
nonzero	nonzero	1

Un programma d'esempio

```
1 // Fig. 2.5: fig02_05.c
2 // Addition program.
3 #include <stdio.h>
4
5 // function main begins program execution
6 int main( void )
7 {
8     int integer1; // first number to be entered by user
9     int integer2; // second number to be entered by user
10    int sum; // variable in which sum will be stored
11
12    printf( "Enter first integer\n" ); // prompt
13    scanf( "%d", &integer1 ); // read an integer
14
15    printf( "Enter second integer\n" ); // prompt
16    scanf( "%d", &integer2 ); // read an integer
17
18    sum = integer1 + integer2; // assign total to sum
19
20    printf( "Sum is %d\n", sum ); // print sum
21 }
```

```
Enter first integer
45
Enter second integer
72
Sum is 117
```

ESERCIZI

Variabili e operatori in C

FONDAMENTI DI PROGRAMMAZIONE



Dato il seguente programma C:



```
1  #include<stdio.h>
2
3  int main(void) {
4      int a = 2;
5      float b = 1.1;
6
7      int result = a * b;
8      printf("Risultato: %f\n", result);
9
10     return 0;
11 }
```

- 1. Identificare gli errori (c'è almeno un errore);***
- 2. Compilare ed eseguire il codice corretto;***



Il linguaggio C può rappresentare le lettere maiuscole, minuscole e molti altri simboli utilizzando un byte per ogni carattere. E' possibile visualizzare l'intero equivalente della lettera maiuscola A eseguendo l'istruzione:

printf("%d", 'A');

*Scrivere un programma C che visualizzi gli interi equivalenti dei seguenti simboli: A, B, a, b, 1, 2, \$, *, + e il carattere spazio.*

Esercizio 2

ASCII character set

	0	1	2	3	4	5	6	7	8	9
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
1	lf	vt	ff	cr	so	si	dle	dc1	dc2	dc3
2	dc4	nak	syn	etb	can	em	sub	esc	fs	gs
3	rs	us	sp	!	"	#	\$	%	&	'
4	()	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[\]	^	_	'	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{		}	~	del		



Dato il seguente programma C:



```
1  #include <stdio.h>
2
3  int main( void ){
4      unsigned char c1 = 255;
5      unsigned char c2 = 256;
6      unsigned char c3 = 257;
7
8      printf("c1 = %d\n", c1);
9      printf("c2 = %d\n", c2);
10     printf("c3 = %d\n", c3);
11
12     return 0;
13 }
```

1. *Qual è l'output prodotto dalle tre printf()?*



Scrivere un programma C che domandi all'utente di inserire un numero intero e stampi i valori intero del precedente e del successivo.



Scrivere un programma C che domandi all'utente di inserire due numeri interi e li usi per stampare la somma, la differenza, il prodotto, il quoziente e il resto (operatore %).

HOMeworks

FONDAMENTI DI PROGRAMMAZIONE



Scrivere un programma C che chieda all'utente di inserire il peso (Kg) e l'altezza (m) e che calcoli l'indice di massa corporea (BMI) definito come segue:

$$BMI = \frac{\text{peso}}{\text{altezza}^2}$$



Scrivere un programma C che, dato un numero reale R immesso da tastiera, calcoli e stampi:

- 1. l'area del quadrato di lato R ;*
- 2. l'area del cerchio di diametro R ;*
- 3. l'area del triangolo equilatero di lato R ;*



Spiegare come mai il seguente codice stampa il più grande intero rappresentabile sul vostro sistema.

```
unsigned long val = -1;
```

```
printf("Il più grande valore intero: %lu\n", val);
```

- 1. Cosa stampa il sistema (il valore è dipendente dal sistema)?*
- 2. Cercate di spiegarne il motivo.*



Dato il seguente programma C:

```
1  #include <stdio.h>
2
3  int main( void ){
4      int a = 19,
5      int b = 1;
6
7      somma = a + b;
8      printf("Somma: %d\n", somma)
9
10     printf("Prodotto: %f\n", a * b);
11
12     return 0;
13 }
```

1. **Identificare gli errori** (c'è almeno un errore);
2. **Compilare ed eseguire** il codice corretto;