

Tipi elementari, costanti

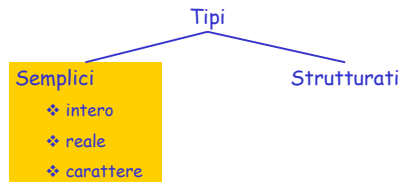
Antonella Santone

Anno Accademico 2008/2009

1

Tipi di dati

- ❖ VALORI: un insieme dei valori del tipo
- ❖ OPERAZIONI: per operare su tali valori



2

... vedremo per ogni tipo

- ❖ Campo di variabilità
- ❖ Notazione per i valori costanti
- ❖ Operazioni
- ❖ Metodi di input/output

3

Tipo intero

Gli interi servono per contare, e contare è uno dei compiti più tipici per i computer

Il tipo intero viene utilizzato per tutte le grandezze che possono essere rappresentate come numeri interi, come per es.: età, numero di figli, ecc.

4

Campo di variabilità

Intervallo finito

Tipo	Dimensione (byte)	Valore minimo	Valore massimo
short int	2	-32768	+32767
int	4	-2^{31}	$2^{31} - 1$
long int	4	-2^{31}	$2^{31} - 1$

5

Notazione per i valori costanti

Sequenza di cifre, preceduta eventualmente dal segno + o -

```
int x;
```

```
x = 356;
```

```
x = -987;
```

6

Operazioni

- + somma
- meno unario
- differenza
- * prodotto
- / divisione intera
- % resto della divisione intera (modulo)

7

Esempi

```
int x=9;      x/y  ==> 1
int y=6;      x%y  ==> 3
```

Se $y=0 \rightarrow$ errore perché un computer non è in grado di eseguire una divisione per zero

Per i numeri negativi, la direzione di troncamento del / ed il segno del risultato di %, dipendono dalla macchina

Se $x\%y$ restituisce 0 \rightarrow x è multiplo di y

8

Metodi di input/output

```
int x;
```

Output

```
printf("%d", x);
```

Input

```
scanf("%d", &x);
```

9

Esempio

```
#include<stdio.h>
main()
{
    int x=9;
    int y=6;
    printf("x/y=%d\n", x/y);
    printf("x%y=%d\n", x%y);
}
```



```
Santone@PC ~
$ gcc pp01a.c
$ ./a.exe
x/y=1
x%y=3
Santone@PC ~
$
```

10

Un altro esempio

```
#include<stdio.h>
```

```
main()
```

```
{ short int x;
  printf("Dammi uno short intero:");
  scanf("%d", &x);
  printf("%d", x);
}
```

11

Cywin



```
Santone@PC ~
$ gcc dim_interi.c -o dim_interi.exe
Santone@PC ~
$ ./dim_interi.exe
Dammi uno short intero:32755
32755
Santone@PC ~
$ ./dim_interi.exe
Dammi uno short intero:40000
-25536
Santone@PC ~
$
```

12

Dettagli sul tipo intero

Possibile aggiungere il qualificatore **unsigned** alla definizione di tipo, che consente alla variabile di contenere solamente numeri positivi

Tipo	Dimensione (byte)	Valore minimo	Valore massimo
unsigned short int	2	0	65535
unsigned int	4	0	$+2^{32} - 1$
unsigned long int	4	0	$+2^{32} - 1$

13

Differenza tra tipo intero ed interi

proprietà valide per i numeri interi non lo sono sempre per il tipo intero

Esempio

$$x + (y - z) = (x + y) - z$$

Ipotesi

tipo intero è $[-10, 10]$, $x = 8$, $y = 7$, $z = 6$

9

indefinito

concetto di **OVERFLOW** (... tipico del finito ...)

14

Cosa stampa?

```
#include<stdio.h>
main()
{
    int x;
    x = 3*5+2;
    printf("%d", x);
}
```

17

oppure

~~21~~

15

Precedenze

Il C dispone di un insieme di regole che determinano l'ordine in cui le varie operazioni devono essere eseguite

Per le operazioni aritmetiche le precedenze sono quelle definite in matematica

16

Precedenza degli operatori aritmetici

Operatori	Associatività
Parentesi: ()	dall'interno all'esterno
Operatore unario: -	da destra a sinistra
Operatori binari: * / %	da sinistra a destra
Operatori binari: + -	da sinistra a destra

+ alta

+ bassa

17

Associatività

$2*3*5$



$((2*3)*5)$

da sinistra a destra

18

Regole

1. In base alla precedenza
2. Se tutti gli operandi hanno la stessa precedenza: associatività
3. Priorità degli operatori può essere alterata con le parentesi tonde: vengono valutate per prima le operazioni all'interno delle parentesi tonde più interne

19

Esempio

$3*5+2$ → 17
* precedenza su +

$2*3/2*3$ → 9
* e / stessa precedenza:
sx verso dx: $((2*3)/2)*3$

$3*(5+2)$ → 21

$7+3-15+4*5$ → 15
 $((7+3)-15)+(4*5)$

20

Esercizi

... sul tipo intero

21

Esercizio 1

Scrivere un programma che richiede all'utente un numero che rappresenta un'altezza in centimetri. Il programma converte tale altezza in metri e centimetri e visualizza il risultato in metri e centimetri

Esempio

Utente immette 134cm → 1 m, 34 cm

Utente immette 45cm → 0 m, 45 cm

Utente immette 200cm → 2 m, 0 cm

22

Soluzione: esercizio 1

```
#include<stdio.h>

main()
{
    int numero, metri, centimetri;

    printf("Dammi l'altezza in cm ");
    scanf("%d", &numero);
    metri = numero/100;
    centimetri= numero%100;
    printf("%d m, %d cm", metri, centimetri);
}
```

23

Cywin



```
Santone@PC ~
$ gcc esercizio.c -o esercizio.exe
Santone@PC ~
$ ./esercizio.exe
Dammi l'altezza in cm 134
1 m, 34 cm
Santone@PC ~
$
```

24

Esercizio 2

Scrivere un programma che richiede all'utente un numero positivo di tre cifre. Il programma stampa la cifra centrale.

Esempio:

356 → 5
789 → 8

25

Soluzione: esercizio 2

```
#include<stdio.h>

main()
{
    int n, cfr;
    printf("Introduci un numero positivo di tre cifre\n");
    scanf("%d", &n);
    cfr = (n/10) % 10;
    printf("Cifra centrale: %d", cfr);
}
```

26

Cygwin



```
Santone@PC ~
$ gcc esercizio.c -o esercizio.exe
Santone@PC ~
$ ./esercizio.exe
Introduci un numero positivo di tre cifre
123
Cifra centrale: 2
Santone@PC ~
$ =
```

27

Esercizio 3

Rovesciare un numero positivo di tre cifre

Esempio:

356 diventa 653
789 diventa 987

28

Soluzione: esercizio 3

```
#include<stdio.h>

main()
{
    int n, unita, decine, centinaia;
    printf("Introduci un numero positivo di tre cifre\n");
    scanf("%d", &n);
    unita = n % 10;
    decine = (n/10) % 10;
    centinaia = n/100;
    printf("Numero rovesciato: %d%d%d", unita, decine, centinaia);
}
```

29

Cygwin



```
Santone@PC ~
$ gcc rovescia.c -o rovescia.exe
Santone@PC ~
$ ./rovescia.exe
Introduci un numero positivo di tre cifre
456
Numero rovesciato: 654
Santone@PC ~
$ =
```

30

... oppure

```
#include<stdio.h>
main()
{
    int n, unita, decine, centinaia;
    printf("Introduci un numero positivo di tre cifre\n");
    scanf("%d", &n);
    unita = n % 10;
    decine = (n/10) % 10;
    centinaia = n/100;
    printf("Rov: %d", 100*unita+10*decine+centinaia);
}
```

31

Altri esercizi ...

32

Esercizio

Scrivere un programma che richiede all'utente un numero che rappresenta un periodo di tempo espresso in minuti. Il programma converte tale periodo in ore e minuti e visualizza il risultato in ore e minuti

Esempio

Utente immette 134m → 2 h, 14 m

Utente immette 45m → 0 h, 45 m

Utente immette 180m → 3 h, 0 m

33

Soluzione


```
#include<stdio.h>

main()
{
    int numero, minuti, ore;

    printf("Dammi il tempo in minuti ");
    scanf("%d", &numero);
    ore = numero/60;
    minuti= numero%60;
    printf("%d h, %d m", ore, minuti);
}
```

34

Cygwin



```
Santone@PC ~
$ gcc ore.c -o ore.exe
Santone@PC ~
$ ./ore.exe
Dammi il tempo in minuti 134
2 h, 14 m
Santone@PC ~
$
```

35

Tipo reale

I numeri reali vengono usati per rappresentare prezzi, pesi, misure, per calcoli matematici, ecc.

36

Campo di variabilità

Intervallo finito

Tipo	Dimensione (byte)	Valore minimo	Valore massimo
float precisione singola	4	$-3.2 \cdot 10^{\pm 38}$	$+3.2 \cdot 10^{\pm 38}$
double precisione doppia	8	$-1.7 \cdot 10^{\pm 308}$	$+1.7 \cdot 10^{\pm 308}$

37

Notazione per i valori costanti

Esistono due modi di scrivere numeri reali

- parte intera punto parte decimale
4.34
- parte intera e o E esponente con segno
-3E3 rappresenta $-3 \cdot 10^3$ cioè -3000
5e-2 rappresenta $5 \cdot 10^{-2}$ cioè 0.05

38

Operazioni

somma, differenza unaria e binaria, prodotto, divisione reale, esponenziali, logaritmi, funzioni trigonometriche, ...

39

Funzioni aritmetiche

```
#include<math.h>
```

x e y di tipo double e restituiscono un double

- ❖ `pow(x, y)` x^y
- ❖ `sin(x)` seno di x, con x espresso in radianti
- ❖ `cos(x)` coseno di x, con x espresso in radianti
- ❖ `exp(x)` e^x
- ❖ `log(x)` logaritmo naturale di x
- ❖ `sqrt(x)` radice quadrata x, $x \geq 0$
- ❖ `log10(x)` logaritmo in base 10 di x
- ❖

40

Metodi di input/output

```
float x;
```

Output

```
printf("%f", x);
```

Input

```
scanf ("%f", &x);
```

41

Metodi di input/output

```
double x;
```

Output

```
printf("%lf", x);
```

Input

```
scanf ("%lf", &x);
```

42

... approfondimento

43

Limiti di precisione

Operazioni aritmetiche sui reali non sono necessariamente esatte

Per brevi computazioni questo di solito non rappresenta un problema, se invece la computazione contiene molte operazioni, come la risoluzione numerica di equazioni differenziali, è importante conoscere gli effetti degli errori dovuti alla rappresentazione

Di questi aspetti si occupa l'**analisi numerica**

44

Esercizi

... sul tipo reale

45

Esercizio 1

Scrivere un programma che calcola l'area di un cerchio di raggio r immesso dall'utente


46

Soluzione: esercizio 1

```
#include<stdio.h>
main()
{
    float r, area;
    printf("Dammi il raggio: ");
    scanf("%f", &r);
    area = r*r*3.14;
    printf("L'area del cerchio di raggio %f = %f\n", r, area);
}
```

47

Cygwin



```
Santone@PC ~
$ gcc esercizio.c -o esercizio.exe
Santone@PC ~
$ ./esercizio.exe
Dammi il raggio: 2
L'area del cerchio di raggio 2.000000 = 12.560000
Santone@PC ~
$ =
```

48

Esercizio 2

Scrivere un programma che effettua la conversione da LIRE ITALIANE a EURO

Esempio

Se immetto 1000 LIRE

1000 LIRE = 0.516457 EURO

49

Soluzione: esercizio 2

```
#include<stdio.h>
main()
{
    float euro, lira;
    printf("Dammi il numero in LIRE: ");
    scanf("%f", &lira);
    euro = lira / 1936.27;
    printf("%f LIRE = %f EURO", lira,euro);
}
```

50

Cygwin



```
Santone@PC ~
$ gcc esercizio.c -o esercizio.exe
Santone@PC ~
$ ./esercizio.exe
Dammi il numero in LIRE: 1000
1000.000000 LIRE = 0.516457 EURO
Santone@PC ~
$
```

51

Migliorare stampa

```
#include<stdio.h>
main()
{
    float euro, lira;
    printf("Dammi il numero in LIRE: ");
    scanf("%f", &lira);
    euro = lira / 1936.27;
    printf("%f LIRE = %f EURO",
    lira,euro);
}
```

`%.nf`

stampa un numero frazionario con n cifre dopo il punto decimale

52

Cygwin



```
Santone@PC ~
$ gcc esercizio.c -o esercizio.exe
Santone@PC ~
$ ./esercizio.exe
Dammi il numero in LIRE: 1000
1000 LIRE = 0.52 EURO
Santone@PC ~
$
```

53

Altri esercizi ...

54

Esercizio

Scrivere un programma che calcola l'area di un triangolo di base **b** ed altezza **h**, immessi dall'utente

55

Soluzione

```
#include<stdio.h>
main()
{
    float base, altezza, area;
    printf("Dammi la base: ");
    scanf("%f", &base);
    printf("Dammi l'altezza: ");
    scanf("%f", &altezza);
    area = (base * altezza)/2;
    printf("L'area: %f\n", area);
}
```

56

Cygwin



```
Santone@PC ~
$ gcc triangolo.c -o triangolo.exe
Santone@PC ~
$ ./triangolo.exe
Dammi la base: 3
Dammi l'altezza: 5
L'area: 7.500000
Santone@PC ~
$ ==
```

57

Esercizio

Scrivere un programma che effettua la conversione da EURO a LIRE ITALIANE

Esempio

Se immetto 100 EURO

100 Euro = 193627 LIRE

58

Soluzione

```
#include<stdio.h>
main()
{
    float euro, lira;
    printf("Dammi il numero in EURO: ");
    scanf("%f", &euro);
    lira = euro * 1936.27;
    printf("%f EURO = %f LIRE", euro,lira);
}
```

59

Cygwin



```
Santone@PC ~
$ gcc euro.c -o euro.exe
Santone@PC ~
$ ./euro.exe
Dammi il numero in EURO: 100
100.000000 EURO = 193627.000000 LIRE
Santone@PC ~
$ ==
```

60

Tipo carattere

Finora abbiamo lavorato con valori numerici. I numeri costituiscono molta parte del lavoro dei computer, ma non tutta. I computer sono macchine per il trattamento dell'*informazione* e l'informazione è costituita per la maggior parte da testi, che a loro volta sono composti da *caratteri*

61

Campo di variabilità

Intervallo finito

Tipo	Dimensione (byte)
char	1

62

Codifiche binarie

Ogni carattere è rappresentato da uno specifico codice binario:

ad ogni carattere corrisponde una rappresentazione numerica univoca

Le codifiche binarie più diffuse nel mondo informatico sono:

- ❖ Codifica **ASCII**
(American Standard Code for Information Interchange)
- ❖ Codifica **EBCDIC**
(Extended Binary Coded Decimal Interchange Code)

63

Codice ASCII

Rappresenta 128 simboli diversi (codice di 7 bit):

lettere dell'alfabeto, cifre, segni di punteggiatura e altri simboli

64

Codice ASCII (cont.)

Carattere	Decimale	Binario
{	123	1111011
a	97	1100001
A	65	1000001
B	66	1000010
;	59	0111011
3	51	0110011
&	38	0100110

'A' è maggiore di ';' che è maggiore di '&'
Si ha che: a<b<c<.....<z
A<B<C<.....<Z
0<1<2<.....<9

65

Codice ASCII (cont.)

0 ... 9 valori nell'intervallo 48 ...57
A ... Z valori nell'intervallo 65 ...90
a ... z valori nell'intervallo 97 ... 122

66

Notazioni per valori costanti

- ❖ per assegnare un valore costante ad una variabile `char` lo si deve racchiudere tra apici, es.

```
x = 'A';  
y = ';' ;  
z = '&';
```

- ❖ per quelli non stampabili si usa `\` (*sequenza di escape*)

<code>\n</code>	a capo	<code>\\</code>	backslash
<code>\t</code>	tabulazione	<code>\'</code>	carattere apice
<code>\"</code>	doppio apice	<code>\r</code>	ritorno carrello

Le sequenze di escape vanno racchiuse tra apici come i simboli dei caratteri

67

Cosa stampa?

```
#include<stdio.h>  
  
main()  
{  
    printf("ab\n");  
    printf("cd\r");  
    printf("e\n");  
}
```



68

Soluzione

```
#include<stdio.h>
```

```
main()  
{  
    printf("ab\n");  
    printf("cd\r");  
    printf("e\n");  
}
```

```
ab  
ed
```

69

Operazioni

Ad ogni carattere corrisponde una rappresentazione numerica univoca

I caratteri sono totalmente ordinati

Possibili operazioni:

- ❖ restituire il carattere che segue/precede;
- ❖ operazioni di uguaglianza/disuguaglianza;
- ❖ chiedersi se un carattere è maggiore/minore di un altro
- ❖ ...

70

Metodi di input/output

```
char x;
```

Output

```
printf("%c", x);
```

Input

```
scanf ("%c", &x);
```

71

... approfondimento

In C il tipo `char` è un tipo intero su 1 byte utilizzato principalmente per rappresentare caratteri

```
#include<stdio.h>  
  
main()  
{  
    char x=65;  
    printf("%c", x); // stampa A  
    printf("%d", x); // stampa 65  
    x='A';  
    x=x+1;  
    printf("%c", x); // stampa B  
}
```

72

Esercizi

... sul tipo carattere

73

Cosa stampa?

```
#include<stdio.h>

main()
{
    char a=10;
    printf("%d", 'a'%10);

    printf("%d", a%10);           // stampa 7

    printf("%d", a+'a');        // stampa 0

    printf("%d", a+'a');        // stampa 107
}
```

74

Esercizio

Scrivere un programma che legge prima due caratteri e poi li stampa per due volte in ordine inverso.

Esempio:

ab → baba

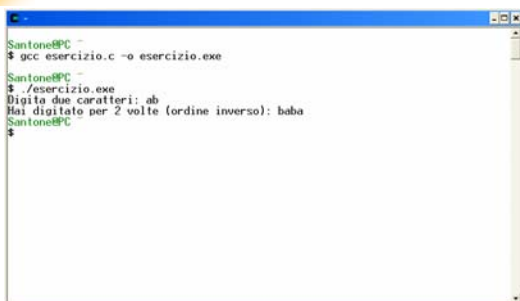
75

Soluzione

```
#include<stdio.h>
main()
{
    char x, y;
    printf("Digita due caratteri: ");
    scanf("%c%c", &x, &y);
    printf("Hai digitato per 2 volte (ordine inverso): ");
    printf("%c%c%c%c", y, x, y, x);
}
```

76

Cygwin



```
Santone@PC ~
$ gcc esercizio.c -o esercizio.exe
Santone@PC ~
$ ./esercizio.exe
Digita due caratteri: ab
Hai digitato per 2 volte (ordine inverso): baba
Santone@PC ~
$
```

77

Esercizio

Scrivere un programma che legge prima un carattere e poi stampa i due caratteri che lo precedono

Esempio:

Se leggo d stampa: cb


78

Soluzione

```
#include<stdio.h>
main()
{
    char x;
    printf("Digita un carattere: ");
    scanf("%c", &x);
    x=x-1;
    printf("%c", x);
    x=x-1;
    printf("%c", x);
}
```

79

Cygwin



```
Santone@PC ~
$ gcc esercizio.c -o esercizio.exe
Santone@PC ~
$ ./esercizio.exe
Digita un carattere: d
cb
Santone@PC ~
$
```

80

Altri esercizi ...

81

Esercizio

Scrivere un programma che legge prima tre caratteri e poi li stampa in ordine inverso.


82

Soluzione

```
#include<stdio.h>
main()
{
    char x, y, z;
    printf("Digita tre caratteri: ");
    scanf("%c%c%c", &x, &y, &z);
    printf("Hai digitato (ordine inverso): ");
    printf("%c%c%c", z, y, x);
}
```

83

Cygwin



```
Santone@PC ~
$ gcc car.c -o car.exe
Santone@PC ~
$ ./car.exe
Digita tre caratteri: abc
Hai digitato (ordine inverso): cba
Santone@PC ~
$
```

84

Esercizio

Scrivere un programma che legge prima un carattere e poi stampa il carattere che lo segue e quello che lo precede

Esempio:

Se leggo D stampa: E C

85

Soluzione

```
#include<stdio.h>
main()
{
    char x;
    printf("Digita un carattere: ");
    scanf("%c", &x);
    x=x+1;
    printf("Il succ = %c\n", x);
    x=x-2;
    printf("Il prec = %c\n", x);
}
```

86

Cygwin



```
Santone@PC ~
$ gcc car2.c -o car2.exe
Santone@PC ~
$ ./car2.exe
Digita un carattere: D
Il succ = E
Il prec = C
Santone@PC ~
$ =
```

87

Operatore sizeof()

Operatore unario

Prende in input una variabile od un identificatore di tipo e restituisce la dimensione in byte

Esempio

```
int i;
printf("dimensione di i: %d", sizeof(i));
//stampa 4
```

88

Esercizio

Utilizzando l'operatore sizeof(...) determinare e stampare la dimensione occupata dai seguenti tipi: int, char, float, double, short int, long int

89

```
...
printf("dimensione di int: %d\n", sizeof(int));
//stampa 4
printf("dimensione di char: %d\n", sizeof(char));
//stampa 1
printf("dimensione di float: %d\n", sizeof(float));
//stampa 4
printf("dimensione di double: %d\n", sizeof(double));
//stampa 8
printf("dimensione di short int: %d\n", sizeof(short int));
//stampa 2
printf("dimensione di long int: %d\n", sizeof(long int));
//stampa 4
```

90

Riepilogo: stringa di formato

```
%d "int"
%hd "short int"
%ld "long int"

%u "unsigned int"
%hu "unsigned short int"
%lu "unsigned long int"

%f "float"
%lf "double"

%c tipo char
```

91

Conversioni

92

Conversioni

Il risultato di un'operazione dipende dal tipo di dato che è coinvolto nell'operazione

```
int x=5;          float x=5;
int y=2;          float y=2;
... x/y ... vale 2   ... x/y ... vale 2.5
```

Questo perché, a seconda del tipo di dato coinvolto nelle operazioni, le operazioni sono svolte in maniera diversa

93

Le conversioni di tipo

I valori di una variabile possono essere convertiti da un tipo all'altro

- ❖ implicitamente (dal compilatore)
- ❖ esplicitamente (dal programmatore)

94

Conversioni implicite

Le conversioni più significative sono:

- ❖ se due operandi di un'operazione binaria sono tra loro diversi, l'operando di tipo più piccolo viene convertito nel tipo più grande, senza perdita di informazioni

```
int i; float f;
...
f=f*i;          // i convertito in float
```

- ❖ nell'assegnamento, il C converte il valore del lato destro nel tipo del lato sinistro, eventualmente perdendo informazioni quando si passa da un tipo ad un tipo più piccolo

```
int i;
float f=3.6;
i=f;           //i vale 3
```

95

Conversioni esplicite

Il programmatore può richiedere esplicitamente la conversione di un valore da un tipo ad un altro:

(nome_nuovo_tipo) espressione

```
int x=5;
int y=2;
float f;
f=(float)x/(float)y;   f vale 2.5
```

96

Nota: cosa stampa??

```
#include<stdio.h>
main()
{
  int x=5;
  int y=2;
  float f;
  f=(float) (x/y);
  printf("%f", f);
}
```

2.0

97

```
#include<stdio.h>
main()
{
  int x=5;
  int y=2;
  float f;
  ISTRUZIONE
  printf("%f", f);
}
```

I) ISTRUZIONE

f=(x/y);

2.0

III) ISTRUZIONE

f=(float) (x/y);

2.0

II) ISTRUZIONE

f=(float) x / (float) y;

2.5

IV) ISTRUZIONE

f=(float)x / y;

2.5

98

Esercizi

```
int a;
float b=6, c=18.6;
...
a=c/b;
```

3

```
int a=27, b=6;
float c;
...
c=a/(float)b;
```

4.5

99

Esercizio

```
int b=6;
float a, c=18.6;
...
a=(int) c/b;
```

3.0



100

Assegnamento multiplo

```
int i, j, k;
j=9;
k=j=i=10;
```

assegna prima il valore 10 a i
assegna il valore 10 a j
assegna il valore 10 a k

- ❖ assegnamento è un'espressione
- ❖ assegnamenti sono associativi da destra a sinistra

101

102

Problema: conversioni di tipo

```
int i;  
double g,x;  
x = i = g = 10.3;  
printf("x=%f i=%d g=%f", x, i,g);
```

```
// stampa  
// x=10.0 i=10 g=10.3
```

```
int i;  
double g,x;  
i = x = g = 10.3;  
printf("x=%f i=%d g=%f", x, i,g);
```

```
// stampa  
// x=10.3 i=10 g=10.3
```

103

Operatori di assegnamento

```
i=i+2;
```

variabile sul lato sinistro ripetuta immediatamente sul lato destro

può essere scritta, usando una forma compatta, come

```
i+=2;
```

Operatori di assegnamento sono:

+= -= *= ...

104

Operatori di incremento e decremento

Operatori di incremento e decremento

Operatori unari

++ aggiunge uno

-- sottrae uno

x++ equivale a x=x+1

x-- equivale a x=x-1

105

106

Operatori di incremento e decremento

Operatori postfissi

x++; prima usa x, poi incrementala

Operatori prefissi

++x; prima incrementa x, poi usala

107

Operatori di incremento e decremento

La variabile viene comunque incrementata

Attenzione: quando compaiono in istruzioni meno semplici

```
x++;      //equivale a x=x+1;
```

```
++x;      //equivale a x=x+1;
```

```
y=x++;    //equivale a y=x; x=x+1;
```

```
y=++x;    //equivale a x=x+1; y=x;
```

108

Operatori di incremento e decremento

```
int n, m=0;      int n, m=0;
n=m++;          n=++m;
```

```
n vale 0        n vale 1
m vale 1        m vale 1
```

```
int n, m=0;
n=m=m+1;        //(n=(m=m+1));
n vale 1
m vale 1
```

109

Costanti

110

Costanti

```
/* Calcolo area cerchio */
#include <stdio.h>
#define PI_GRECO 3.14

main()
{
    float raggio, area;

    printf("Dammi raggio: ");
    scanf("%f", &raggio);

    area = raggio*raggio*PI_GRECO;

    printf("Area: %f", area);
}
```

La definizione di costante
implica che il suo valore non può
essere modificato

NOTA

Le direttive non
terminano con il ;

111

Perché usare costanti?

- ❖ Evitare di scrivere più volte in un programma un'espressione che rappresenta un numero, per esempio quando è molto complicata, o per garantire che non ci siano difformità tra le varie occorrenze
- ❖ Migliorare la leggibilità dei programmi, per esempio usare sempre una costante per *pi greco*
- ❖ Riutilizzare i programmi per esempio un programma che manipola matrici quadrate di dimensione 100 può essere facilmente riutilizzato per le matrici di dimensione 200, se tale dimensione è rappresentata da una costante

112

Costanti in C

1. Costanti testuali (#define)

2. Variabili read only (const)

113

#define

Sintassi

```
#define NomeVariabile costante
```

Semantica

Tutte le occorrenze di *NomeVariabile* (purché non siano racchiuse tra apici e non facciano parte di un'altra stringa) vengono rimpiazzate con *costante*

114

Nota

Nomi delle costanti scritti con caratteri maiuscoli (per convenzione)

Dopo alla fine `#define` non serve il ;

115

Esempio

```
#define SIZE 10
```

```
int i=SIZE;
```

Viene tradotto dal *preprocessore* in

```
int i=10;
```

116

Esempio

```
#include<stdio.h>
#define SIZE 3
```

```
main()
{
  int x=SIZE;
  int y= SIZE+2;
  printf("%d %d", x, y);
}
```

3

3

Stampa:

3 5

117

Esempio

```
#include<stdio.h>
#define SIZE 3
```

```
main()
{
  printf("SIZE");
}
```

Semantica

Tutte le occorrenze di `NomeVariabile` (purché non siano racchiuse tra apici e non facciano parte di un'altra stringa) vengono rimpiazzate con costante

Stampa:

SIZE

118

Esempio

```
#include<stdio.h>
#define SIZE 3
```

```
main()
{
  int MYSIZE = 2;
  int y = MYSIZE;
  printf("%d", y);
}
```

Stampa:

2

SIZE parte di MYSIZE, quindi non si rimpiazza con 3

119

Preprocessore

Codice sorgente C

preprocessore

compilatore

120

Cosa fa il preprocessore?

1. Processa le cosiddette direttive al preprocessore:
 - ❖ inclusione di file
`#include<nomefile>`
prende il file `nomefile` e lo inserisce al posto della direttiva
 - ❖ definizioni di costanti
`#define NomeVariabile costante`
ogni volta che incontra `NomeVariabile` lo sostituisce con `costante`
2. Elimina i commenti contenuti nel sorgente

121

Un esempio di come lavora il processore

prima

```
/* file header.h */  
  
extern int i;  
extern double f;  
  
/* fine header.h */
```

```
/* file prova.c */  
  
#include "header.h"  
/* size vettore */  
#define SIZE 10  
int vettore[SIZE];
```

dopo il passaggio col processore

```
extern int i;  
extern double f;  
int vettore[10];
```

122

Costanti in C

1. Costanti testuali (`#define`)
2. Variabili read only (`const`)

123

const

Sintassi

```
const tipo NomeVariabile = costante;
```

Semantica

Variabile che non può più essere modificata dopo aver fissato un suo valore iniziale (e che quindi possa essere solo letta)

- ❖ Ogni successiva assegnazione a tale variabile verrà considerato un errore
- ❖ È possibile dichiarare queste costanti in ogni punto dove è possibile dichiarare una variabile

124

Esempio

```
#include<stdio.h>  
main()  
{  
    const int A=100;  
    printf("%d", A);    //stampa 100  
    A=A+1;             // errore  
}
```

125

... abbiamo visto

- ❖ Tipi semplici
 - ❖ intero
 - ❖ reale
 - ❖ carattere
- ❖ Costanti

126

Esercizi

127

Esercizio

Scrivere quattro differenti istruzioni in C che aggiungano 1 alla variabile intera x

128

Soluzione

1. `x = x+1;`
2. `x += 1;`
3. `++x;`
4. `x++;`

129

Esercizio

```
int a, b=0, c=0;
```

```
a = ++b + ++c;
```

```
a: 2 b: 1 c: 1
```

```
a = b++ + c++;
```

```
a: 0 b: 1 c: 1
```

```
a = ++b + c++;
```

```
a: 1 b: 1 c: 1
```

```
a = b-- + --c;
```

```
a: -1 b: -1 c: -1
```

130

Cosa stampa?

```
#include<stdio.h>

main()
{
    int i=1, k=3;
    k = (i + 0.7) + (float) (k/2);

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

2

131

Cosa stampa?

```
#include<stdio.h>
main()
{
    int product=5, x=5;
    product *= ++x;
    printf("product = %d, x = %d", product, x);
}
product = 30, x = 6

#include<stdio.h>
main()
{
    int product=5, x=5;
    product *= x++;
    printf("product = %d, x = %d", product, x);
}
product = 25, x = 6
```

132

Cosa stampa?

```
#include<stdio.h>
main()
{
    int result=5, x=5;
    result = ++x + x ;
    printf("result = %d, x = %d", result, x);
}
```

result = 12, x = 6

```
#include<stdio.h>
main()
{
    int result=5, x=5;
    result = x++ + x ;
    printf("result = %d, x = %d", result, x);
}
```

result = 10, x = 6