Fondamenti di Informatica

Ingegneria Clinica

Lezione 23/11/2009

Raffaele Nicolussi

FUB - Fondazione Ugo Bordoni

Via B. Castiglione 59 - 00142 Roma

Docente	**Raffaele Nicolussi**	*rnicolussi@fub.it* *0654803323*
Lezioni Aula 54 (ex aula 4) Via del Castro Laurenziano, 7	**Lunedì, Giovedì, Venerdì**	**12:00 – 13:30**
Esercitazioni Aula 15 Via Tiburtina, 205	**Giovedì**	**14:00 – 16.30**
Ricevimento:	**Per appuntamento**	**in FUB, per email, per telefono**
Sito web:	**http://w3.uniroma1.it/IngClinFondinf**	

Esercizio 5

Scrivere un programma che legge da standard input un carattere **c** e due numeri interi **b** ed **h**, e stampa un rettangolo di base **b** ed altezza **h** di caratteri **c**

```
Inserire il carattere da stampare: &

Inserire base e altezza: 7 3

&&&&&&&
&&&&&&&
&&&&&&&

Premere un tasto per continuare . . .
```

Soluzione 5

```
int b, h ,i, j;
    char car;

    printf("\nInserire il carattere da stampare: ");
    scanf("%c", &car);
    printf("\nInserire base e altezza: ");
    scanf("%d%d", &b, &h);

    printf("\n");
    i = 1;
    while ( i <= h) {
        j = 1;
        while ( j <= b ) {
            printf ("%c", car);
            j = j + 1;
        }
        printf("\n");
        i = i + 1;
    }
```

```
while ( i <= h) {
  j = 1;
  while ( j <= b ) {
     printf ("%c", car);
     j = j + 1;
  }
   printf("\n");
   i = i + 1;
}
```

Qualificatori di tipo

- tipi di dato primitivi
- qualificatori di tipo
- short int
- int
- long int
- unsigned short int
- short
- unsigned int
- unsigned long int
- byte
- float
- double
- specifiche di conversione per gli interi
- operandi interi
- tipo di una costante
- operatori aritmetici per interi
- operatori di assegnamento
- conversione di tipo
- Espressioni che coinvolgono tipi di dato primitivi numerici diversi
- Assegnazioni fra tipi di dato primitivi numerici diversi
- Esercitazione 4 – 22 Maggio 2007

Tipi di dato primitivi

- Per definire un tipo di dato si deve specificare:
  - **Dominio:** insieme dei possibili valori rappresentabili nella memoria del calcolatore mediante il tipo di dato primitivo (sempre un insieme finito di valori)
  - **Operazioni:** operatori del linguaggio che permettono di effettuare operazioni elementari su valori del tipo primitivo (Es. `+`, `-`, `/`, `*`, ecc.)
  - **Letterali:** simboli del linguaggio che definiscono i valori del tipo primitivo (Es. `10`, `3.14`, `'A'`, `true`, ecc.)

C: qualificatori di tipo

- Tipi base: char (caratteri), int (interi), float e double (reali)
- E' possibile modificare alcune caratteristiche dei tipi base mediante i *qualificatori* short, long, signed e unsigned
- short e long possono modificare il numero di bit utilizzato per la rappresentazione
- signed e unsigned possono modificare l'insieme dei valori rappresentati
- Esempio:

  long int (interi, intervallo rappresentato include quello rappresentato con int)

C: Tipo short int

- Intervallo *minimo*: [-32767, 32767]
- Intervallo *vero*: dipende dall'implementazione; è specificato da SHRT_MIN e SHRT_MAX in "limits.h"
- Occupazione: dipende dall'implementazione; per l'intervallo minimo sono sufficienti 16 bit
- Rappresentazione: dipende dall'implementazione; tipicamente, è quella in complemento a 2
- Nelle dichiarazioni, short int o short
- Specifica di conversione: %hd

C: Tipo int

- Intervallo *minimo*: [-32767, 32767]
- Intervallo *vero*: dipende dall'implementazione; è specificato da INT_MIN e INT_MAX in "limits.h"; deve includere l'intervallo usato per il tipo short int
- Occupazione: dipende dall'implementazione; per l'intervallo minimo sono sufficienti 16 bit
- Rappresentazione: dipende dall'implementazione; tipicamente, è quella in complemento a 2
- Nelle dichiarazioni int
- Specifica di conversione: %d

Il tipo di dato int

Tipo	int	
Dimensione	32 bit (4 byte)	
Dominio	l'insieme dei numeri interi nell'intervallo $[-2^{31},+2^{31}-1]$ (oltre 4 miliardi di valori)	
Operazioni	+	somma
	-	sottrazione
	*	prodotto
	/	divisione intera
	%	resto della divisione intera
Letterali	sequenze di cifre che denotano valori del dominio (es. 275930)	

```
int a,b,c;        // Dichiarazione di variabile di tipo int
a = 1;            // Uso di letterali
b = 2;
c = a + b;        // Espressione aritmetica
```

C: Tipo long int

- Intervallo *minimo*: [-2147483647, 2147483647]
- Intervallo *vero*: dipende dall'implementazione; è specificato da LONG_MIN e LONG_MAX in "limits.h"; deve includere l'intervallo usato per il tipo int
- Occupazione: dipende dall'implementazione; per l'intervallo minimo sono sufficienti 32 bit
- Rappresentazione: dipende dall'implementazione; tipicamente, è quella in complemento a 2
- Nelle dichiarazioni: long int o long
- Specifica di conversione: %ld

Il tipo di dato long

Tipo	long	
Dimensione	64 bit (8 byte)	
Dominio	l'insieme dei numeri interi nell'intervallo $[-2^{63},+2^{63}-1]$	
Operazioni	+	somma
	-	sottrazione
	*	prodotto
	/	divisione intera
	%	resto della divisione intera
Letterali	sequenze di cifre terminate con una l o L che denotano valori del dominio (es.9000000000L)	

```
long a,b;              // Dichiarazione di variabile di tipo long
a = 9000000000L;       // Uso di letterali
b = a+1;               // Espressione aritmetica
```

C: Tipo unsigned short int

- Intervallo *minimo*: [0, 65535]
- Intervallo *vero*: dipende dall'implementazione, è specificato da USHRT_MAX in "limits.h" (valore minimo è sempre 0)
- Occupazione: dipende dall'implementazione; per l'intervallo minimo sono sufficienti 16 bit
- Rappresentazione: dipende dall'implementazione; tipicamente, è quella posizionale
- Nelle dichiarazioni: unsigned short int o unsigned short
- Specifica di conversione: %hu, %ho (ottale), %hx o %hX (esadecimale, lettere minuscole o maiuscole)

Il tipo di dato short

Tipo	short	
Dimensione	16 bit (2 byte)	
Dominio	l'insieme dei numeri interi nell'intervallo $[-2^{15},+2^{15}-1] = [-32768, 32767]$	
Operazioni	+	somma
	-	sottrazione
	*	prodotto
	/	divisione intera
	%	resto della divisione intera
Letterali	sequenze di cifre che denotano valori del dominio (es. 22700)	

```
short a,b;        // Dichiarazione di variabile di tipo short
a = 22700;     // Uso di letterali
b = a+1;                    // Espressione aritmetica
```

C: Tipo unsigned int

- Intervallo *minimo*: [0, 65535]
- Intervallo *vero*: dipende dall'implementazione, è specificato da UINT_MAX in "limits.h" (valore minimo è sempre 0)
- Occupazione: dipende dall'implementazione; per l'intervallo minimo sono sufficienti 16 bit
- Rappresentazione: dipende dall'implementazione; tipicamente, è quella posizionale
- Nelle dichiarazioni: unsigned int o unsigned
- Specifica di conversione: %u, %o (ottale), %x o %X (esadecimale, lettere minuscole o maiuscole)

C: Tipo unsigned long int

- Intervallo *minimo*: [0, 4294967295]
- Intervallo *vero*: dipende dall'implementazione, è specificato da ULONG_MAX in "limits.h" (valore minimo è sempre 0)
- Occupazione: dipende dall'implementazione; per l'intervallo minimo sono sufficienti 32 bit
- Rappresentazione: dipende dall'implementazione; tipicamente, è quella posizionale
- Nelle dichiarazioni: unsigned long int o unsigned long
- Specifica di conversione: %lu, %lo (ottale), %lx o %lX (esadecimale, lettere minuscole o maiuscole)

Il tipo di dato byte

Tipo	byte	
Dimensione	8 bit (1 byte)	
Dominio	l'insieme dei numeri interi nell'intervallo $[-2^7,+2^7-1]$ = [-128, 127]	
Operazioni	+	somma
	-	sottrazione
	*	prodotto
	/	divisione intera
	%	resto della divisione intera
Letterali	sequenze di cifre che denotano valori del dominio (es. 47)	

```
int a,b,c;      // Dichiarazione di variabile di tipo byte
a = 1;          // Uso di letterali
b = a+1;                    // Espressione aritmetica
```

Il tipo di dato float

<table>
<tr><td>Tipo</td><td colspan="4">float</td></tr>
<tr><td>Dimensione</td><td colspan="4">32 bit (4 byte)</td></tr>
<tr><td rowspan="3">Dominio</td><td rowspan="3">insieme di 2^{32} numeri reali + e -</td><td>|min|</td><td colspan="2">$1.4012985 * 10^{-38}$</td></tr>
<tr><td>|max|</td><td colspan="2">$3.4028235 * 10^{+38}$</td></tr>
<tr><td>Precisione</td><td colspan="2">~7 cifre decimali</td></tr>
<tr><td rowspan="4">Operazioni</td><td>+</td><td colspan="3">somma</td></tr>
<tr><td>-</td><td colspan="3">sottrazione</td></tr>
<tr><td>*</td><td colspan="3">prodotto</td></tr>
<tr><td>/</td><td colspan="3">divisione</td></tr>
<tr><td rowspan="2">Letterali</td><td colspan="4">sequenze di cifre con punto decimale terminate con una f (o F) che denotano valori del dominio (es. 3.14f)</td></tr>
<tr><td colspan="4">rappresentazione in notazione scientifica (es. 314E-2f)</td></tr>
</table>

Il tipo di dato float

Esempio:

```
float a;    // Dichiarazione di variabile di tipo float
a = 3.14f;  // Uso di letterali
a = a*2f;   // Espressione aritmetica
```

Il tipo di dato double

Tipo	double		
Dimensione	64 bit (8 byte)		
Dominio	insieme di 2^{64} numeri reali + e -	|min|	$1.7976931348623157 * 10^{-308}$
		|max|	$2.250738585072014 * 10^{+308}$
		Precisione	~ 15 cifre decimali
Operazioni	+	somma	
	-	sottrazione	
	*	prodotto	
	/	divisione	
Letterali	sequenze di cifre con punto decimale opzionalmente terminate con una d o (D), che denotano valori del dominio (Es. 3.14 oppure 3.14d)		
	Rap. in not. scientifica (Es. 314E-2 oppure 314E-2d)		

Specifiche di conversione per tipi interi

short int	int	long int
%hd	%d	%ld

unsigned short int	unsigned int	unsigned long int
%hu	%u	%lu
%ho	%o	%lo
%hx	%x	%lx
%hX	%X	%lX

C: Operandi interi (1)

- Variabili di uno dei tipi int
- Costanti usate direttamente nelle espressioni
- Esempio: b = 2 * a + 33 * b - c / 19 ;
- Una costante viene specificata con

  segno: + o -, opzionale

  sequenza di cifre: in base 10, 8 (prefisso 0, [0-7]), 16 (prefisso 0x o 0X, [0-9], [a-f] o [A-F])

  suffisso: u o U per unsigned, l o L per long
- Esempi: -165438L, 0xFFFFFFFFl, -0765, 0XaAaA1, +2147483647L

C: Operandi interi (2)

- Costanti introdotte con #define
- Forma: #define nome valore
- Effetto: *ogni* occorrenza successiva di nome sarà rimpiazzata con valore (*qualunque* esso sia!)
- Nome: stesse regole date per il nome di variabili
- #define è una direttiva per il compilatore (elaborata dal preprocessore a *tempo di compilazione*)
- Uso tipico: per modificare valori di costanti *senza interventi pesanti* sul testo del programma (si ricompila il programma dopo aver aggiornato solo il valore che compare nella #define)

C: Tipo di una costante

- Il *tipo* di una costante intera *dipende* da come viene specificata
- Base 10, senza suffisso: primo tipo possibile tra int, long int e unsigned long int
- Base 8 o 16, senza suffisso: primo tipo possibile tra int, unsigned int, long int e unsigned long int
- Con suffisso u o U: primo tipo possibile tra unsigned int e unsigned long int
- Con suffisso l o L: primo tipo possibile tra long int e unsigned long int

C: Operatori aritmetici per interi (1)

- Operatori binari (due operandi): somma (+), sottrazione (-), prodotto (*), quoziente (/), resto (%)
- Operatori unari (un operando): segno (+), inversione segno (-)
- Attenzione: se il risultato di un'operazione eccede i limiti della rappresentazione, il comportamento *dipende* dall'implementazione
- Attenzione: se almeno uno degli operandi è negativo, il comportamento di / e % *dipende* dall'implementazione

C: Operatori di assegnamento

- Operatore = usato nella forma

  variabile = espressione
- Operatore +=, -=, *=, /=, %= (indicato qui con op=) usato nella forma

  variabile op= espressione
- Significato

  variabile = variabile op (espressione)

  Esempi:

  a += 3*b + c → a = a + (3*b + c)

  x *= 3 + b - 2*c → x = x * (3 + b - 2*c)

C: Conversione di tipo (1)

- Conversione *implicita* nel caso di operatori binari utilizzati con due operandi di tipo diverso: uno dei due tipi (inferiore) viene promosso all'altro (superiore) ed il risultato è del tipo superiore
- ESEMPIO di regola (informale)

  oper1 * oper2: se oper1 è double e oper2 è float, oper2 *promosso* a double e risultato è double
- Attenzione: a , b float e c double

  a = b + c ; assegnerà il valore double risultato della somma ad una variabile float con potenziale *perdita di informazione* (in un assegnamento il tipo della variabile destinazione definisce il tipo finale)

C: Conversione di tipo (2)

- Conversione *implicita* nella chiamata di funzione
- Gli argomenti passati ad una funzione *dichiarata con prototipo* vengono convertiti secondo quanto specificato nella dichiarazione con potenziale *perdita di informazione*
- ESEMPIO: se gli argomenti formali sono float e int, una chiamata con argomenti attuali int e float provoca una potenziale *perdita di informazione*

C: Conversione di tipo (3)

- Conversione *implicita* nella return all'interno di una funzione
- In return (espressione), il tipo assegnato al valore di espressione è quello specificato nella dichiarazione della funzione (potenziale *perdita di informazione*)
- ESEMPIO: b float e c int, tipo di ritorno int return b * c ; convertirà il risultato float del prodotto in un int con potenziale *perdita di informazione*

C: Conversione di tipo (4)

- Conversione *esplicita* di tipo può essere forzata con operatore cast
- (tipo) espressione provoca la valutazione di espressione come se il risultato dovesse essere assegnato ad una variabile del tipo forzato
- ESEMPIO di utilizzazione: conversione tipo argomento nella chiamata funzioni di libreria (potrebbero non fare uso di dichiarazione tramite prototipo)
- sqrt((double) n) chiamata di sqrt su n int convertito a double (valore di n immutato!)

Espressioni che coinvolgono tipi di dato primitivi numerici diversi (1)

- Vediamo una tabella che descrive il tipo risultante di una espressione della forma a+b per ciascuna coppia di tipi possibili di a e di b

Espressioni che coinvolgono tipi di dato primitivi numerici diversi (2)

a+b	byte b	short b	int b	long b	float b	double b
byte a	byte	short	int	long	float	double
short a	short	short	int	long	float	double
int b	int	int	int	long	float	double
long b	long	long	long	long	float	double
float b	float	float	float	float	float	double
double b	double	double	double	double	double	double

Assegnazioni fra tipi di dato primitivi numerici diversi (1)

- Un valore di un tipo di dato *non* può essere assegnato ad una variabile di un tipo di dato con minor dimensione, altrimenti si rischia perdita di precisione dell'informazione.
- Un valore reale non può essere assegnato ad una variabile intera.

  int a; long b; a = b;

  Errore: un valore long non può essere assegnato ad una variabile int

  int a; float b; a = b;

  Errore: un valore float non può essere assegnato ad una variabile int

Assegnazioni fra tipi di dato primitivi numerici diversi (2)

a=b	byte b	short b	int b	long b	float b	double b
byte a	OK	ERRORE	ERRORE	ERRORE	ERRORE	ERRORE
short a	OK	OK	ERRORE	ERRORE	ERRORE	ERRORE
int b	OK	OK	OK	ERRORE	ERRORE	ERRORE
long b	OK	OK	OK	OK	ERRORE	ERRORE
float b	OK	OK	OK	OK	OK	ERRORE
double b	OK	OK	OK	OK	OK	OK

Qualche esercizio …

Esercizio 1

Scrivere un programma che, letta da stdin una sequenza di caratteri terminata da invio, per ciascun carattere della sequenza esegue la seguente azione:

- se il carattere è una lettera minuscola stampa su stdout la corrispondente lettera maiuscola
- se il carattere è una lettera maiuscola stampa su stdout la corrispondente lettera minuscola
- in tutti gli altri casi stampa uno spazio

Nella rappresentazione ASCII

-le lettere maiuscole hanno codici compresi tra 65 ('A') e 90 ('Z')

-le lettere minuscole hanno codici compresi tra 97 ('a') e 122 ('z')

```
Inserire una sequenza di caratteri terminata da
                    invio:
                abc12.Ief2g
                ABC   iEF G
        Premere un tasto per continuare . . .
```

Soluzione 1


```
char carattere;

printf("\nInserire una sequenza di caratteri terminata da invio: \n");
scanf("%c", &carattere);

while ( carattere != '\n' ) {
      if ( carattere >= 65 && carattere <= 90 )  /* lettera maiuscola */
                        printf ("%c", carattere + 32);
      else if  ( carattere >= 97 && carattere <= 122 )  /* lettera minuscola */
                        printf ("%c", carattere -32);
      else
                  printf ("%c", ' ');
      scanf("%c", &carattere);
 }
```

Esercizio 2

Scrivere un programma che legge da stdin un carattere c e due interi b ed h e stampa—utilizzando un "ciclo for"—un rettangolo di base b ed altezza h di caratteri c, come specificato in figura
Se c = 'A', b = 5, h = 3, il risultato sarà

AAAAA
AAAAA
AAAAA

Usare una prima scanf per il carattere c ed una seconda per i due interi b ed h

Soluzione 2


```
int base , altezza , i , j ;
char car ;

printf( "Inserire un carattere:\n" ) ;
scanf( "%c" , &car ) ;

printf( "Inserire base rettangolo (intero >= 1):\n" ) ;
scanf( "%d" , &base ) ;

printf( "Inserire altezza rettangolo (intero >= 1):\n" ) ;
scanf( "%d" , &altezza ) ;

for ( i = 1 ; i <= altezza ; i = i+1 )
{
   for ( j = 1 ; j <= base ; j = j+1 )
      printf( "%c" , car ) ;
   printf( "\n" ) ;
}
```

Esercizio 3

Scrivere un programma C che legga da stdin una sequenza di numeri positivi la cui lunghezza non è nota a priori, terminata da un numero negativo. Per ogni numero letto il programma deve stampare su stdout la media di tutti i numeri letti fino a quel momento.

```
Inserisci un numero positivo (o negativo per terminare): 2.2
          Media attuale (1 numero/i): 2.200000
Inserisci un numero positivo (o negativo per terminare): 3.3
          Media attuale (2 numero/i): 2.750000
Inserisci un numero positivo (o negativo per terminare): 5.5
          Media attuale (3 numero/i): 3.666667
Inserisci un numero positivo (o negativo per terminare): 0
          Media attuale (4 numero/i): 2.750000
Inserisci un numero positivo (o negativo per terminare): -1
          Premere un tasto per continuare . . .
```

Soluzione 3

```
int i = 1;
float media, num, somma = 0;

printf ("Inserisci un numero positivo (o negativo per terminare): ");
scanf("%f",&num);
while (num >= 0)
{
    somma = somma + num;
    media = somma/i;
    printf ("Media attuale (%d numero/i): %f\n", i, media);
    printf ("Inserisci un numero positivo (o negativo per terminare): ");
    scanf("%f",&num);
    i++;
}
```

Esercizio 4

Leggere da stdin una sequenza di 0 e 1 terminata da 2 (acquisire i valori uno per volta) e stampare la lunghezza della più lunga sottosequenza di soli 0 presente nella sequenza letta

Esempio: per la sequenza

0 0 1 0 0 0 1 1 1 1 0 0 2

la risposta cercata è 3

Algoritmo Esercizio 4

Variabili utilizzate (tipo intero):
bit: valore letto, contatore: numero di 0 accumulati lmax: massima lunghezza sottosequenza di 0

- contatore=0; lmax=0 (inizializzazione)
- leggi un numero (valore registrato in bit)
- finché bit è diverso da 2
  - se bit è pari a 0:
    - incrementa contatore
    - se contatore > lmax: lmax=contatore
  - altrimenti
    - contatore=0
  - leggi un altro numero
- stampa lmax

Soluzione 4

```
int bit, cont = 0, maxlung = 0;
printf("Inserisci una sequenza di 0 e 1 terminata da 2\n");
scanf("%d", &bit);

while ( bit!=2)
{
  if ( bit == 0)
  {
    cont = cont + 1;
    if (cont > maxlung)
      maxlung = cont;
  }
  else
    cont = 0;
  scanf("%d", &bit);
}
printf("La lunghezza della piu\' lunga sottosequenza di 0 e\' %d\n", maxlung);
```

Esercizio 5

Un intero N > 1 è detto primo se i suoi unici divisori sono 1 e N

Scrivere un programma che legge da stdin un intero e determina se è primo

Algoritmo (*inefficiente*): provare se tra i numeri compresi tra 2 e N-1 c'è un divisore di N

Algoritmo(*inefficiente*) Esercizio 5

Variabili utilizzate (tipo intero):
numero: valore letto, provadiv: possibile divisore di numero,
trovatodiv: diventa vero (1) se si trova un divisore di numero

- provadiv=2; trovatodiv=0 (inizializzazione)
- leggi valore (registrato in numero)
- finché provadiv < numero
  - se provadiv divide numero: trovato=1
  - provadiv=provadiv+1
- se trovato=1: numero non è primo
  altrimenti: numero è primo

Soluzione 5


```
int numero, provadiv = 2, trovatodiv = 0;
printf ("Inserire un numero intero maggiore di uno: \n");
scanf ("%d",&numero);

while (provadiv < numero)
{
   if ((numero % provadiv) == 0)
       trovatodiv = 1;
   provadiv = provadiv + 1;
}
if (trovatodiv==0)
   printf("Il numero %d e\' un numero primo\n", numero);
else
   printf("\nIl numero %d non e\' un numero primo\n", numero);
```

Esercizio 6

Scrivere un programma che, letti da stdin un intero positivo *n* e un numeri reale x, calcola lo sviluppo di Taylor di ordine *n* della funzione e^x, dato dalla seguente formula:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!}$$

```
Calcolo dello sviluppo di Taylor di ordine n di e(x)
Inserire x: 1.5
Inserire n: 6
4.477539
Premere un tasto per continuare . . .
```

Soluzione 6

```
float x,num=1,risultato=1,den=1; int i=1,n;
printf("Calcolo dello sviluppo di Taylor di ordine n di e(x)\n");
printf("Inserire x: ");
scanf("%f",&x);
printf("Inserire n: ");
scanf("%d",&n);

for (i = 1; i <= n; i++)
{
    num = num*x;
    den = den * i;
    risultato = risultato + (num/den);
 }
 printf("%f\n",risultato);
```

Esercizio 7

Scrivere un programma che letti da stdin un carattere c ed un intero h, stampi un triangolo di altezza h di caratteri c, come specificato in figura

Se c = 'T', h = 4, il risultato sarà

T
TT
TTT
TTTT

Usare una prima scanf per il carattere c ed una seconda per l'intero h

Soluzione 7

```
char car;
int h , i , j ;

printf( "Inserire un carattere:\n" ) ;
scanf( "%c" , &car ) ;

printf( "Inserire altezza triangolo (intero >=1):\n" ) ;
scanf( "%d" , &h ) ;

/* attenzione alla condizione di continuazione del ciclo inerno */
for ( i = 1 ; i <= h ; i = i+1 )
{
 for ( j = 1 ; j <= i ; j = j+1 )
    printf( "%c" , car ) ;
 printf( "\n" ) ;
}
```

Riga 1 : 1 carattere
Riga 2: 2 caratteri
...
Riga n: n caratteri