

Variabili, tipi primitivi e costrutti condizionali

Laboratorio di Programmazione I

Corso di Laurea in Informatica
A.A. 2017/2018



Calendario delle lezioni

Ogni lezione consta di una spiegazione assistita da slide, e seguita da esercizi in classe

Lezione 1 (19-20/09/2017) - Introduzione all'ambiente Linux

Lezione 2 (26-27/09/2017) - Introduzione al C

Lezione 3 (3-4/10/2017) - Tipi primitivi e costrutti condizionali

Lezione 4 (10-11/10/2017) - Costrutti iterativi ed array

Lezione 5 (17-18/10/2017) - Funzioni, stack e visibilità variabili

Lezione 6 (24-25/10/2017) - Puntatori e memoria

Lezione 7 (7-8/11/2017) - Debugging

Lezione 8 (14-15/11/2017) - Tipi di dati utente

Lezione 9 (21-22/11/2017) - Liste concatenate e librerie.

Lezione 10 (28-29/11/2017) - Ricorsione

Lezione 11 (5-6/12/2017) - CAML

Lezione 12 (12-13/12/2017) - Simulazione prova pratica

Sommario

- 1 Variabili
 - Tipi di dato primitivi
 - Operatori aritmetici
 - Conversione tra tipi di dato
 - Booleani
- 2 Printf e scanf
- 3 Comando condizionale
- 4 Piattaforma di autovalutazione

Outline

1 Variabili

- Tipi di dato primitivi
- Operatori aritmetici
- Conversione tra tipi di dato
- Booleani

2 Printf e scanf

3 Comando condizionale

4 Piattaforma di autovalutazione

Dichiarazioni di variabili

La sintassi da utilizzare per dichiarare una variabile la seguente:

```
tipo nome_variabile [=valore iniziale];
```

Dichiarazioni di variabili

La sintassi da utilizzare per dichiarare una variabile la seguente:

```
tipo nome_variabile [=valore iniziale];
```

- tipo della variabile;

Dichiarazioni di variabili

La sintassi da utilizzare per dichiarare una variabile la seguente:

```
tipo nome_variabile [=valore iniziale];
```

- tipo della variabile;
- nome della variabile: i nomi possono essere composti da lettere e cifre, ma devono sempre cominciare con una lettera. Ricordatevi le regole stilistiche viste la scorsa volta;

Dichiarazioni di variabili

La sintassi da utilizzare per dichiarare una variabile la seguente:

```
tipo nome_variabile [=valore iniziale];
```

- tipo della variabile;
- nome della variabile: i nomi possono essere composti da lettere e cifre, ma devono sempre cominciare con una lettera. Ricordatevi le regole stilistiche viste la scorsa volta;
- si può definire un valore iniziale della variabile. È fortemente consigliato inizializzare le variabili (in alcuni casi di default hanno un valore casuale).

Sommario

- 1 Variabili
 - Tipi di dato primitivi
 - Operatori aritmetici
 - Conversione tra tipi di dato
 - Booleani
- 2 Printf e scanf
- 3 Comando condizionale
- 4 Piattaforma di autovalutazione

Tipi di dato primitivi: int (1)

int: un intero rappresentabile sulla macchina (segnaposto %d)

- l'occupazione in memoria e di conseguenza l'intervallo di rappresentazione del tipo **int** dipende dalla macchina su cui viene compilato il programma;

Tipi di dato primitivi: int (1)

int: un intero rappresentabile sulla macchina (segnaposto %d)

- l'occupazione in memoria e di conseguenza l'intervallo di rappresentazione del tipo **int** dipende dalla macchina su cui viene compilato il programma;
- la funzione predefinita `sizeof()` fornisce l'occupazione in byte di un qualsiasi tipo o variabile C. Ad esempio se lanciamo sulle macchine del laboratorio:

```
#include <stdio.h>
int main()
{
    printf("%d\n", sizeof(int));
}
```

Tipi di dato primitivi: int (1)

int: un intero rappresentabile sulla macchina (segnaposto %d)

- l'occupazione in memoria e di conseguenza l'intervallo di rappresentazione del tipo **int** dipende dalla macchina su cui viene compilato il programma;
- la funzione predefinita `sizeof()` fornisce l'occupazione in byte di un qualsiasi tipo o variabile C. Ad esempio se lanciamo sulle macchine del laboratorio:

```
#include <stdio.h>
int main()
{
    printf("%d\n", sizeof(int));
}
```

Otteniamo in output:

4

Tipi di dato primitivi: int (2)

Maggiore è il numero di byte per un tipo, piu' grande è il massimo valore assoluto rappresentabile.

Gli attributi `short` e `long` indicano interi che occupano un diverso numero di byte.

Tipi di dato primitivi: int (2)

Maggiore è il numero di byte per un tipo, piu' grande è il massimo valore assoluto rappresentabile.

Gli attributi `short` e `long` indicano interi che occupano un diverso numero di byte.

	sizeof	min	max
<code>short int</code>	2	-32768	32767
<code>int</code>	4	-2147483648	2147483647
<code>long int</code>	8	-9223372036854775808	9223372036854775807

Tipi di dato primitivi: int (2)

Maggiore è il numero di byte per un tipo, piu' grande è il massimo valore assoluto rappresentabile.

Gli attributi `short` e `long` indicano interi che occupano un diverso numero di byte.

	sizeof	min	max
<code>short int</code>	2	-32768	32767
<code>int</code>	4	-2147483648	2147483647
<code>long int</code>	8	-9223372036854775808	9223372036854775807

Se i valori da rappresentare sono positivi, l'attributo `unsigned` permette di usare tutti i byte per il numero positivo

Tipi di dato primitivi: int (2)

Maggiore è il numero di byte per un tipo, piu' grande è il massimo valore assoluto rappresentabile.

Gli attributi `short` e `long` indicano interi che occupano un diverso numero di byte.

	sizeof	min	max
<code>short int</code>	2	-32768	32767
<code>int</code>	4	-2147483648	2147483647
<code>long int</code>	8	-9223372036854775808	9223372036854775807

Se i valori da rappresentare sono positivi, l'attributo `unsigned` permette di usare tutti i byte per il numero positivo

	sizeof	min	max
<code>unsigned short</code>	2	0	65535
<code>unsigned int</code>	4	0	4294967295
<code>unsigned long</code>	8	0	18446744073709551615

Tipi di dato primitivi: char

char: un singolo byte, in grado di contenere un carattere (codifica ASCII). Segnaposto %c.

```
char a='a'; // i caratteri si indicano tra apici
```

Tipi di dato primitivi: char

char: un singolo byte, in grado di contenere un carattere (codifica ASCII). Segnaposto `%c`.

```
char a='a'; // i caratteri si indicano tra apici
```

- i caratteri sono visti in C come interi. Infatti:

```
int a='a';  
printf ("%c\n",a);  
printf ("%d\n",a);
```

Tipi di dato primitivi: char

char: un singolo byte, in grado di contenere un carattere (codifica ASCII). Segnaposto `%c`.

```
char a='a'; // i caratteri si indicano tra apici
```

- i caratteri sono visti in C come interi. Infatti:

```
int a='a';  
printf ("%c\n",a);  
printf ("%d\n",a);
```

Stampa in output:

a

97

97 corrisponde alla codifica ASCII di a

Tipi di dato primitivi: char (2)

- Alcune costanti carattere e i relativi valori interi:

Costante carattere	'a'	'b'	...	'z'
Valore intero	97	98	...	122
Costante carattere	'A'	'B'	...	'Z'
Valore intero	65	66	...	90
Costante carattere	'0'	'1'	...	'9'
Valore intero	48	49	...	57

Non c'è alcuna relazione tra una costante carattere cifra e la cifra stessa: **'2'** non ha codice 2.

Tutti i codici delle cifre, delle minuscole e delle maiuscole sono **consecutivi**.

Usatelo! **'5'-'0'=5**, se la variabile `ch` contiene una costante carattere cifra `ch - '0'` ci dice a che cifra corrisponde.

Tipi di dato primitivi: numeri reali

float, **double**: sono i tipi utilizzati per rappresentare numeri reali (precisione singola o doppia)

```
float x=123.34;
```

```
double y=100.1e5; //anche notazione scientifica
```

Tipi di dato primitivi: numeri reali

float, **double**: sono i tipi utilizzati per rappresentare numeri reali (precisione singola o doppia)

```
float x=123.34;  
double y=100.1e5; //anche notazione scientifica
```

- segnaposto `%f` e `%lf`;
- dimensioni differenti: `double` viene tipicamente rappresentato con 8 byte, `float` con 4;
- è possibile utilizzare l'attributo `long` per variabili di tipo `double`.

Costanti

L'attributo `const` puo' essere applicato alla dichiarazione di qualsiasi variabile, con l'effetto di affermare che il suo valore non cambiera':

```
const double pi=3.141592;  
const int cinque=5;
```

Costanti

L'attributo `const` puo' essere applicato alla dichiarazione di qualsiasi variabile, con l'effetto di affermare che il suo valore non cambiera':

```
const double pi=3.141592;  
const int cinque=5;
```

I tentativi di modifica a costanti sono tipicamente segnalati dal compilatore con un errore.

Costanti

L'attributo `const` puo' essere applicato alla dichiarazione di qualsiasi variabile, con l'effetto di affermare che il suo valore non cambiera':

```
const double pi=3.141592;  
const int cinque=5;
```

I tentativi di modifica a costanti sono tipicamente segnalati dal compilatore con un errore.

Differenza tra `#define` e `const`:

- `#define` é una direttiva del preprocessore e sostituita all'interno del codice prima della compilazione;
- una variabile definita con l'attributo `const` viene maneggiata dal compilatore: ha un tipo e un indirizzo.

Sommario

- 1 Variabili
 - Tipi di dato primitivi
 - Operatori aritmetici
 - Conversione tra tipi di dato
 - Booleani
- 2 Printf e scanf
- 3 Comando condizionale
- 4 Piattaforma di autovalutazione

Operatori aritmetici

Nel C abbiamo essenzialmente i seguenti operatori aritmetici:

+ - * / %

che rappresentano rispettivamente le usuali operazioni di addizione, sottrazione, moltiplicazione, divisione e modulo. Vengono utilizzati per definire, trasformare il valore delle vostre variabili.

Operatori aritmetici

Nel C abbiamo essenzialmente i seguenti operatori aritmetici:

+ - * / %

che rappresentano rispettivamente le usuali operazioni di addizione, sottrazione, moltiplicazione, divisione e modulo. Vengono utilizzati per definire, trasformare il valore delle vostre variabili.

Ricordiamo che in matematica il valore di ***a* modulo *b*** e' dato dal **resto della divisione di *a* per *b***: ad esempio $5\%3 = 2$. L'operatore modulo non puo' essere applicato a variabili `float` e `double`.

Operatori aritmetici(2)

Gli operatori possiedono regole di **precedenze** e **associativita'** che determinano come avviene la valutazione delle espressioni.

Operatori aritmetici(2)

Gli operatori possiedono regole di **precedenze** e **associativita'** che determinano come avviene la valutazione delle espressioni.

Così come avviene nell'aritmetica tradizionale, $+$ e $-$ hanno lo stesso grado di precedenza, inferiore a quello di $*$, $/$ e $\%$.

Operatori aritmetici(3)

Altri operatori:

- operatori contratti: permettono di effettuare un'operazione su una variabile e assegnarne il risultato alla stessa:
 - `var op = expr` equivale a `var = var op expr`

Operatori aritmetici(3)

Altri operatori:

- operatori contratti: permettono di effettuare un'operazione su una variabile e assegnarne il risultato alla stessa:
 - `var op = expr` equivale a `var = var op expr`
 - ad esempio: `j*=i+2` \Leftrightarrow `j=j*(i+2)`

Operatori aritmetici(3)

Altri operatori:

- operatori contratti: permettono di effettuare un'operazione su una variabile e assegnarne il risultato alla stessa:
 - `var op = expr` equivale a `var = var op expr`
 - ad esempio: `j*=i+2` \Leftrightarrow `j=j*(i+2)`
- incremento/decremento unitario: operatori `++` e `--`. Sono utilizzabili sia come prefisso (ossia prima della variabile: `++n`) o suffisso (dopo la variabile: `n++`). L'effetto è quello di incrementare `n` ma:

Operatori aritmetici(3)

Altri operatori:

- operatori contratti: permettono di effettuare un'operazione su una variabile e assegnarne il risultato alla stessa:
 - `var op = expr` equivale a `var = var op expr`
 - ad esempio: `j*=i+2` \Leftrightarrow `j=j*(i+2)`
- incremento/decremento unitario: operatori `++` e `--`. Sono utilizzabili sia come prefisso (ossia prima della variabile: `++n`) o suffisso (dopo la variabile: `n++`). L'effetto è quello di incrementare `n` ma:
 - `++n` esegue l'incremento **prima** di usare il valore `n`:
`x=++n` equivale a `n=n+1; x=n;`

Operatori aritmetici(3)

Altri operatori:

- operatori contratti: permettono di effettuare un'operazione su una variabile e assegnarne il risultato alla stessa:
 - `var op = expr` equivale a `var = var op expr`
 - ad esempio: `j*=i+2` \Leftrightarrow `j=j*(i+2)`
- incremento/decremento unitario: operatori `++` e `--`. Sono utilizzabili sia come prefisso (ossia prima della variabile: `++n`) o suffisso (dopo la variabile: `n++`). L'effetto è quello di incrementare `n` ma:
 - `++n` esegue l'incremento **prima** di usare il valore `n`:
`x=++n` equivale a `n=n+1; x=n;`
 - `n++` lo fa **dopo** l'impiego del valore:
`x=n++` equivale a `x=n; n=n+1`

Sommario

- 1 Variabili
 - Tipi di dato primitivi
 - Operatori aritmetici
 - Conversione tra tipi di dato
 - Booleani
- 2 Printf e scanf
- 3 Comando condizionale
- 4 Piattaforma di autovalutazione

Conversioni di tipo (1)

Le espressioni aritmetiche hanno un valore ed un tipo dettato da quello delle variabili in gioco.

Conversioni di tipo (1)

Le espressioni aritmetiche hanno un valore ed un tipo dettato da quello delle variabili in gioco.

Quando un'espressione del tipo $x \text{ op } y$ coinvolge operandi di tipo diverso, avviene una conversione implicita secondo le seguenti regole:

Conversioni di tipo (1)

Le espressioni aritmetiche hanno un valore ed un tipo dettato da quello delle variabili in gioco.

Quando un'espressione del tipo $x \text{ op } y$ coinvolge operandi di tipo diverso, avviene una conversione implicita secondo le seguenti regole:

- 1 ogni valore di tipo `char` o `short` viene convertito in `int`

Conversioni di tipo (1)

Le espressioni aritmetiche hanno un valore ed un tipo dettato da quello delle variabili in gioco.

Quando un'espressione del tipo $x \text{ op } y$ coinvolge operandi di tipo diverso, avviene una conversione implicita secondo le seguenti regole:

- 1 ogni valore di tipo `char` o `short` viene convertito in `int`
- 2 se l'espressione è ancora eterogenea si converte l'operando di tipo inferiore facendolo divenire di tipo superiore secondo la seguente gerarchia:

`int` \rightarrow `long` \rightarrow `float` \rightarrow `double` \rightarrow `long double`

Conversioni di tipo (1)

Le espressioni aritmetiche hanno un valore ed un tipo dettato da quello delle variabili in gioco.

Quando un'espressione del tipo $x \text{ op } y$ coinvolge operandi di tipo diverso, avviene una conversione implicita secondo le seguenti regole:

- 1 ogni valore di tipo `char` o `short` viene convertito in `int`
- 2 se l'espressione è ancora eterogenea si converte l'operando di tipo inferiore facendolo divenire di tipo superiore secondo la seguente gerarchia:

`int` \rightarrow `long` \rightarrow `float` \rightarrow `double` \rightarrow `long double`

Esempio

Se x ha tipo `int` e y ha tipo `float`, il risultato dell'espressione $x+y$ viene automaticamente convertito a `float`

Conversioni di tipo (2)

Conversione nell' assegnamento: si ha in $x = \text{exp}$ quando i tipi di x e exp non coincidono.

Conversioni di tipo (2)

Conversione nell' assegnamento: si ha in $x = \text{exp}$ quando i tipi di x e exp non coincidono.

- La conversione avviene **sempre** a favore del tipo della variabile a sinistra:

Conversioni di tipo (2)

Conversione nell' assegnamento: si ha in $x = \text{exp}$ quando i tipi di x e exp non coincidono.

- La conversione avviene **sempre** a favore del tipo della variabile a sinistra:
 - se si tratta di una **promozione** non si ha perdita di informazione

Conversioni di tipo (2)

Conversione nell' assegnamento: si ha in $x = \text{exp}$ quando i tipi di x e exp non coincidono.

- La conversione avviene **sempre** a favore del tipo della variabile a sinistra:
 - se si tratta di una **promozione** non si ha perdita di informazione
 - se si ha una **retrocessione** si può avere perdita di informazione

Conversioni di tipo (2)

Conversione nell' assegnamento: si ha in $x = \text{exp}$ quando i tipi di x e exp non coincidono.

- La conversione avviene **sempre** a favore del tipo della variabile a sinistra:
 - se si tratta di una **promozione** non si ha perdita di informazione
 - se si ha una **retrocessione** si può avere perdita di informazione

Esempio

```
int i;  
float x = 2.3, y = 4.5;  
i = x + y;  
printf("%d, i); /* stampa 6 */
```

- Se la conversione non è possibile si ha errore.

Casting

Oltre alle conversioni implicite, sono possibili anche conversioni esplicite dette **cast** secondo la sintassi:

```
(tipo) espressione;
```

Casting

Oltre alle conversioni implicite, sono possibili anche conversioni esplicite dette **cast** secondo la sintassi:

```
(tipo) espressione;
```

Esempio:

```
int somma, n;
```

```
float media;
```

```
...
```

```
media = somma/n; /* divisione tra interi */
```

```
media = (float)somma/n; /* divisione tra reali */
```


Casting

Oltre alle conversioni implicite, sono possibili anche conversioni esplicite dette **cast** secondo la sintassi:

```
(tipo) espressione;
```

Esempio:

```
int somma, n;  
float media;  
...  
media = somma/n; /* divisione tra interi */  
media = (float)somma/n; /* divisione tra reali */
```

L'operatore di cast ha precedenza piu' alta degli operatori binari e associa da destra a sinistra. Dunque `(float)somma/n` equivale a `((float)somma)/n`

Sommario

- 1 Variabili
 - Tipi di dato primitivi
 - Operatori aritmetici
 - Conversione tra tipi di dato
 - Booleani
- 2 Printf e scanf
- 3 Comando condizionale
- 4 Piattaforma di autovalutazione

Booleani ed operatori

In C non esiste un tipo Booleano. Si usa il tipo `int`:

- 0 rappresenta **FALSO**;
- 1 (o qualsiasi valore diverso da zero) rappresenta **VERO**.

Booleani ed operatori

In C non esiste un tipo Booleano. Si usa il tipo `int`:

- 0 rappresenta **FALSO**;
- 1 (o qualsiasi valore diverso da zero) rappresenta **VERO**.

Operatori logici:

- `!`: NOT (operatore unario). Esempio: `!a`;
- `&&`: AND (operatore binario). Esempio: `a && b`;
- `||`: OR (operatore binario). Esempio: `a || b`;

Restituiscono un valore intero pari a 0 o 1 a seconda del valore (**falso/vero**) dell'espressione.

Altri lavorano sui singoli bit: ad esempio operatori di shift (`<<`, `>>`), AND (`&`), OR (`|`), XOR (`^`) ...

Operatori relazionali e di uguaglianza

Gli operatori relazionali sono:

< > <= >=

sono tutti binari: hanno come operandi due espressioni e restituiscono un risultato di tipo `int` che puo' essere 0 o 1.

Operatori relazionali e di uguaglianza

Gli operatori relazionali sono:

`<` `>` `<=` `>=`

sono tutti binari: hanno come operandi due espressioni e restituiscono un risultato di tipo `int` che puo' essere 0 o 1.

Ad esempio l'espressione relazione `a<b`:

- se `a` e' minore di `b` assume valore 1 (vero);
- altrimenti assume valore 0 (falso).

Operatori relazionali e di uguaglianza

Gli operatori relazionali sono:

`<` `>` `<=` `>=`

sono tutti binari: hanno come operandi due espressioni e restituiscono un risultato di tipo `int` che puo' essere 0 o 1.

Ad esempio l'espressione relazione `a<b`:

- se `a` e' minore di `b` assume valore 1 (vero);
- altrimenti assume valore 0 (falso).

Abbiamo poi gli operatori di uguaglianza `==` e `!=`, sempre binari, che assumono valore 1 (vero) se i due operandi sono rispettivamente uguali o diversi. 0 in caso contrario.

Outline

- 1 Variabili
 - Tipi di dato primitivi
 - Operatori aritmetici
 - Conversione tra tipi di dato
 - Booleani
- 2 **Printf e scanf**
- 3 Comando condizionale
- 4 Piattaforma di autovalutazione

La libreria `stdio.h`

Nel C le funzionalita' di input/output sono demandate a librerie esterne. Ad esempio la libreria `stdio.h` implementa un semplice modello di input/output di dati testuali.

La libreria `stdio.h`

Nel C le funzionalita' di input/output sono demandate a librerie esterne. Ad esempio la libreria `stdio.h` implementa un semplice modello di input/output di dati testuali.

Le funzioni `printf` e `scanf` sono in essa definite: la prima viene utilizzata per l'output, la seconda per l'input. La lettera `f` alla fine dei nomi delle due funzioni sta per formatted.

La libreria `stdio.h`

Nel C le funzionalità di input/output sono demandate a librerie esterne. Ad esempio la libreria `stdio.h` implementa un semplice modello di input/output di dati testuali.

Le funzioni `printf` e `scanf` sono in essa definite: la prima viene utilizzata per l'output, la seconda per l'input. La lettera `f` alla fine dei nomi delle due funzioni sta per formatted.

Sia `printf()` che `scanf()` ricevono una **stringa di controllo**, che può contenere le specifiche di conversione indicate con il simbolo `%` (segnaposto), e una serie di **parametri**, che possono essere ad esempio le variabili da stampare o leggere.

printf

Per stampare delle variabili di un determinato tipo dobbiamo utilizzare i segnaposto relativi:

- **interi**: %d, %u (unsigned). Si antepone h per short e l per long;
- **reali**: %f, %e (notazione scientifica), %g la piu' breve tra notazione standard e scientifica. Per i double non si antepone nulla, per i long double si antepone L;
- **caratteri**: %c;
- **stringhe**: %s (le vedremo piu' avanti).

printf(2)

Altri flag, vanno messi subito dopo il %:

- -: allinea a sinistra

printf(2)

Altri flag, vanno messi subito dopo il %:

- -: allinea a sinistra
- un numero intero n : specifica l'ampiezza minima del campo. Esempio:

```
printf ("%c%3c%5c\n", 'A', 'B', 'C');
```

Stampa: A B C

printf(2)

Altri flag, vanno messi subito dopo il %:

- -: allinea a sinistra
- un numero intero n : specifica l'ampiezza minima del campo. Esempio:

```
printf ("%c%3c%5c\n", 'A', 'B', 'C');
```

Stampa: A B C

- un parametro $.d$: per i numeri reali specifica il numero di cifre decimali d (con arrotondamento). Esempio:

```
printf (".3f\n", 123.4557454);
```

Stampa: 123.456

scanf

La funzione `scanf` e' analoga alla `printf` ma viene utilizzata per la lettura. Anche in questo caso abbiamo una stringa di controllo, mentre gli altri parametri sono gli indirizzi delle variabili in cui saranno memorizzati i valori letti.

scanf

La funzione `scanf` e' analoga alla `printf` ma viene utilizzata per la lettura. Anche in questo caso abbiamo una stringa di controllo, mentre gli altri parametri sono gli **indirizzi** delle variabili in cui saranno memorizzati i valori letti.

Ad esempio:

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

legge un intero e lo memorizza il valore all'indirizzo di `x`, indicato con la notazione `&x`.

scanf

La funzione `scanf` e' analoga alla `printf` ma viene utilizzata per la lettura. Anche in questo caso abbiamo una stringa di controllo, mentre gli altri parametri sono gli **indirizzi** delle variabili in cui saranno memorizzati i valori letti.

Ad esempio:

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

legge un intero e lo memorizza il valore all'indirizzo di `x`, indicato con la notazione `&x`.

Gli specificatori di formato sono quelli visti con la `printf`, tranne che per i **reali**:

- `double`: si antepone `l`;
- `long double`: si antepone `L`.

Outline

- 1 Variabili
 - Tipi di dato primitivi
 - Operatori aritmetici
 - Conversione tra tipi di dato
 - Booleani
- 2 Printf e scanf
- 3 **Comando condizionale**
- 4 Piattaforma di autovalutazione

Comando condizionale

Le istruzioni di un programma vengono normalmente eseguite in sequenza, ma la maggior parte dei programmi richiede una modifica al normale flusso sequenziale del controllo.

Comando condizionale

Le istruzioni di un programma vengono normalmente eseguite in sequenza, ma la maggior parte dei programmi richiede una modifica al normale flusso sequenziale del controllo.

Le istruzioni `if` e `if-else` ci permettono ad esempio di scegliere tra azioni alternative. Sintassi:

```
if (espressione) {  
    blocco1  
}  
else {  
    blocco2  
}
```

`espressione` e' un' espressione booleana. Se vera vengono eseguite le istruzioni in `blocco1` altrimenti vengono eseguite le istruzioni in `blocco2`. Il ramo `else` e' opzionale.

Comando condizionale(2)

Esempio: si acquisisca l'eta' da tastiera e si stabilisca se l'utente sia maggiorenne o meno.

```
#include <stdio.h>
int main(void)
{
    int eta;
    printf("Inserisci la tua eta' in anni: ");
    scanf("%d",&eta);
    if (eta >= 18)
        printf("Sei maggiorenne\n");
    else
        printf("Sei minorenne\n");
    return 0;
}
```

Comando condizionale(2)

Esempio: si acquisisca l'eta' da tastiera e si stabilisca se l'utente sia maggiorenne o meno.

```
#include <stdio.h>
int main(void)
{
    int eta;
    printf("Inserisci la tua eta' in anni: ");
    scanf("%d",&eta);
    if (eta >= 18)
        printf("Sei maggiorenne\n");
    else
        printf("Sei minorenne\n");
    return 0;
}
```

NOTE:

- ricordate sempre l'indentazione;
- se i rami dell'`if` sono composti da una sola istruzione, potete non inserire le parentesi graffe.

Switch

L'istruzione `switch` puo' essere usata per realizzare una selezione a piu' vie.

Switch

L'istruzione `switch` puo' essere usata per realizzare una selezione a piu' vie.

Sintassi:

Switch

L'istruzione `switch` puo' essere usata per realizzare una selezione a piu' vie.

Sintassi:

```
switch (espressione) {  
    case valore_1: istruzioni_1  
        break;  
    ...  
    case valore_n: istruzioni_n  
        break;  
    default : istruzioni_default  
}
```

Switch(2)

Semantica:

- 1 viene valutata *espressione* (deve restituire un intero) e confrontata con i valori contenuti nelle varie etichette (che devono essere costanti);
- 2 quando viene raggiunta un'etichetta uguale all'espressione esegue tutti i comandi che seguono (per interrompere tale esecuzione si usa il comando `break`);
- 3 se nessuna etichetta e' uguale al valore dell'espressione si esegue il ramo `default` se definito (e' opzionale).

Switch(3)

Esempio:

```
int giorno;  
...  
switch (giorno) {  
    case 1: printf("Lunedì\n");  
            break;  
    case 2: printf("Martedì\n");  
            break;  
    case 3: printf("Mercoledì\n");  
            break;  
    case 4: printf("Giovedì\n");  
            break;  
    case 5: printf("Venerdì\n");  
            break;  
    default: printf("Week end\n");  
}
```

Outline

- 1 Variabili
 - Tipi di dato primitivi
 - Operatori aritmetici
 - Conversione tra tipi di dato
 - Booleani
- 2 Printf e scanf
- 3 Comando condizionale
- 4** Piattaforma di autovalutazione

Piattaforma autovalutazione

The screenshot shows a web application interface. At the top, there is a dark navigation bar with icons and labels for 'Home', 'Esercizi', 'Ranking', 'Forum', and 'Sign up', along with a 'Sign in' button on the right. Below this is a light gray header area containing the text 'Programmazione 1 e laboratorio' and 'Anno Accademico 2014/2015'. The main content area features three columns: 'Esercizi' with the subtext 'Risolvi gli esercizi' and a button 'Vai agli esercizi >'; 'Ranking' with the subtext 'Visualizza la classifica' and a button 'Mostra la classifica >'; and 'Forum' with the subtext 'Discuti con gli altri studenti' and a button 'Vai al forum >'. At the bottom of the page, there is a small line of text: 'Content Management System is released under the G1AJ Affero General Public License.'

`http://pr11718.dijkstra.di.unipi.it/`

Registrazione

The screenshot shows a web application interface. At the top, there is a dark navigation bar with icons and labels for 'Home', 'Esercizi', 'Ranking', 'Forum', and 'Sign up'. Below this is a light gray header area containing the text 'Programmazione 1 e laboratorio' and 'Anno Accademico 2014/2015'. The main content area features three columns: 'Esercizi' with a 'Vai agli esercizi >' button, 'Ranking' with a 'Mostra la classifica >' button, and 'Forum' with a 'Vai al forum >' button. At the bottom, a small line of text reads 'Content Management System is released under the GNU Affero General Public License.'

Home Esercizi Ranking Forum Sign up

Sign in

Programmazione 1 e laboratorio
Anno Accademico 2014/2015

Esercizi
Risolvi gli esercizi
Vai agli esercizi >

Ranking
Visualizza la classifica
Mostra la classifica >

Forum
Discuti con gli altri studenti
Vai al forum >

Content Management System is released under the GNU Affero General Public License.

Registrazione

The screenshot shows a registration form with a dark navigation bar at the top containing icons for Home, Esercizi, Ranking, Forum, and Sign up, along with a Sign in link. The form is divided into two main sections: 'Login data' and 'Personal data'. The 'Login data' section includes fields for Username, Password, and Confirm password. The 'Personal data' section includes fields for First name, Last name, E-mail address, and Confirm e-mail, followed by a 'Sign up' button. To the right of the form is a 'User profile preview' area showing a square profile picture with a green geometric pattern, the text '(username)', and '(Nome) (Cognome)' below it.

Contest Management System is released under the [GNU Affero General Public License](#).

- inserite i vostri dati;
- **usate un Username nella forma cognome.nome.corso**
ad esempio: `pisanti.nadia.AB`

Esercitazioni

 Home  Esercizi  Ranking  Forum  Sign up

▼ Lezione 3

> Esercizio 1: Size of

> Esercizio 2: Capitalize

> Esercizio 3: Average

[Contest Management System](#) is released under the [GNU Affero General Public License](#).

Le varie esercitazioni sono organizzate per lezione (a partire dalla 3, quella odierna)

Statement

The screenshot shows a web interface for an online programming exercise. At the top, there is a navigation bar with links for Home, Esercizi, Ranking, and Forum, and a user profile for 'tester'. Below this, the exercise title 'Esercizio 3-1: Size of' is displayed. The main content area is titled 'Statement' and contains the following text:

Esercizio

Scrivere un programma che legga da tastiera un numero intero x e stabilisca il numero di byte necessario a memorizzare x variabili di tipo intero.

The interface also includes a toolbar with options for Attachments (1), Stats, and Submissions, along with performance metrics (1 sec, 128 MiB) and a page indicator (Page: 1 of 1).

Per ogni esercizio trovate sotto **statement** il testo (con alcuni esempi di input/output)

Submission

The screenshot shows a submission page for 'Esercizio 3-1: Size of'. At the top, there is a navigation bar with 'Home', 'Esercizio', 'Ranking', and 'Forum'. Below this, the exercise title is displayed. There are tabs for 'Statement', 'Attachments', 'Stats', and 'Submissions'. A timer shows '1 sec' and a file size limit of '128 MB'. The main area is titled 'Submit a solution' and contains a file upload section for 'Lez3E1.c' with a 'Choose File' button and 'No file chosen' text, and a green 'Submit' button. Below this is a 'Previous submissions' table.

ID	Time and date	Status	File(s)
5	10/14/2014, 4:00:28 PM	100 / 100	Scarica ▾
4	10/14/2014, 3:59:58 PM	40 / 100	Scarica ▾

Potete sottomettere la vostra soluzione (il file .c) dalla scheda **Submission**.

- 1 il sistema valuta la vostra soluzione;
- 2 se corretta vi assegna il massimo del punteggio;

Submission

The screenshot shows a 'Submission details' window. It contains a table with the following data:

Testcase	Result	Details	Time	Memory
0	Correct	Output is correct	0.000s	128 KiB
1	Not correct	Output isn't correct	0.000s	128 KiB
2	Not correct	Output isn't correct	0.000s	128 KiB
3	Not correct	Output isn't correct	0.000s	128 KiB

Below the table, there is a 'Compilation output' section with the following information:

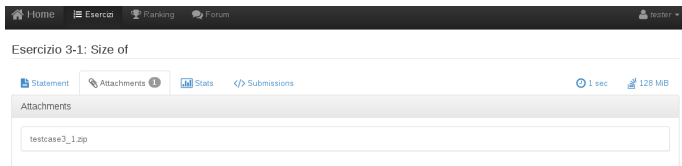
Compilation outcome:	ok
Compilation time:	0.136s
Used memory:	8.7 MiB

At the bottom, there is a 'Standard output' section.

Potete sottomettere la vostra soluzione (il file `.c`) dalla scheda **Submission**.

- 1 il sistema valuta la vostra soluzione;
- 2 se corretta vi assegna il massimo del punteggio;
- 3 altrimenti vi segnala errore. Cliccando sull 'ID della sottomissione potete conoscere su quali test case il vostro programma ha fallito).

Test cases



The screenshot shows a web interface for an online judge. At the top, there is a navigation bar with links for Home, Esercizi, Ranking, and Forum, and a user profile for 'tester'. Below this, the title 'Esercizio 3-1: Size of' is displayed. A secondary navigation bar contains tabs for Statement, Attachments (which is active), Stats, and Submissions. To the right of these tabs, there are indicators for '1 sec' and '128 MB'. The 'Attachments' section shows a list with one entry: 'testcase3_1.zip'.

Trovate i vari test case sotto la scheda **Attachments**. Sono inseriti in un file zip e per ognuno avete una coppia `input` e `output` (atteso). Per controllare la correttezza del vostro programma potete usare i file di input:

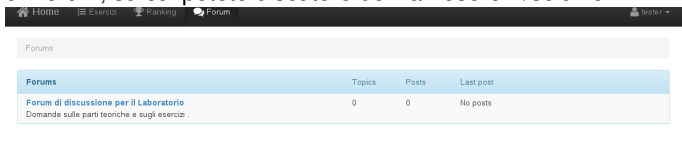
```
cat input.txt | ./<nome_eseguibile>  
cat input.txt | ./<nome_eseguibile> | diff -  
output.txt
```

Ricordatevi di terminare con il `return 0`.

Ricordatevi di terminare con il `return 0`.

Cosa altro c'è:

- un forum, su cui potete discutere dei vari esercizi/soluzioni



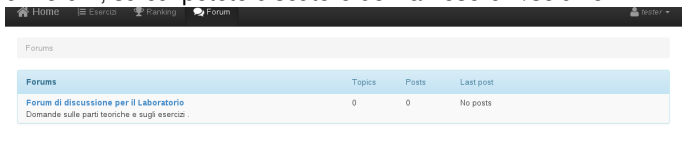
The screenshot shows a forum interface with a navigation bar at the top containing 'Home', 'Esercizi', 'Ranking', and 'Forum'. Below the navigation bar is a table with the following data:

Forums	Topics	Posts	Last post
Forum di discussione per il Laboratorio Domande sulle parti teoriche e sugli esercizi .	0	0	No posts

Ricordatevi di terminare con il `return 0`.

Cosa altro c'è:

- un forum, su cui potete discutere dei vari esercizi/soluzioni



The screenshot shows a forum interface with a navigation bar at the top containing 'Home', 'Esercizi', 'Ranking', and 'Forum'. Below the navigation bar, there is a table with the following data:

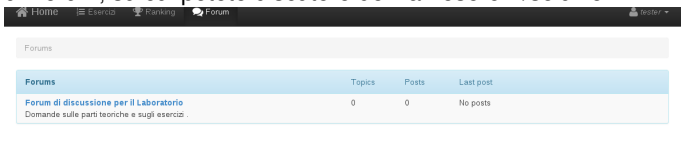
Forums	Topics	Posts	Last post
Forum di discussione per il Laboratorio Domande sulle parti teoriche e sugli esercizi .	0	0	No posts

- una classifica ottenuta considerando i punteggi che accumulate esercizio dopo esercizio.

Ricordatevi di terminare con il `return 0`.

Cosa altro c'è:

- un forum, su cui potete discutere dei vari esercizi/soluzioni



The screenshot shows a forum interface with a navigation bar at the top containing 'Home', 'Esercizi', 'Ranking', and 'Forum'. Below the navigation bar is a table with the following data:

Forums	Topics	Posts	Last post
Forum di discussione per il Laboratorio Domande sulle parti teoriche e sugli esercizi .	0	0	No posts

- una classifica ottenuta considerando i punteggi che accumulate esercizio dopo esercizio.

Buon Lavoro!