Introduzione al C

Lezione 1

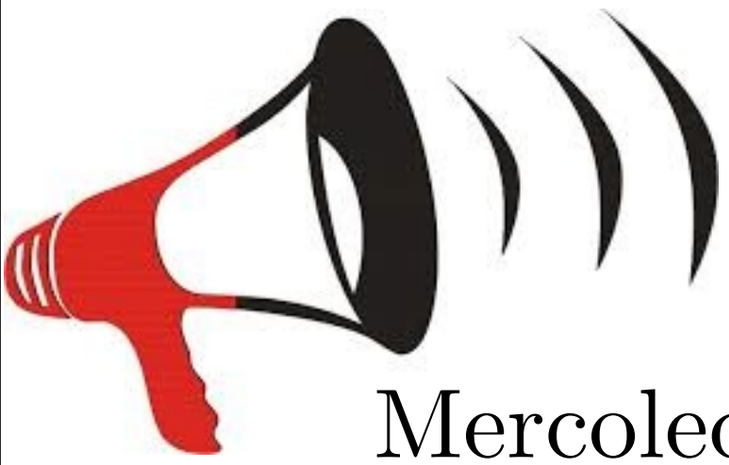
Elementi

Rossano Venturini

rossano@di.unipi.it

Pagina web del corso

http://didawiki.cli.di.unipi.it/doku.php/informatica/all-b/start

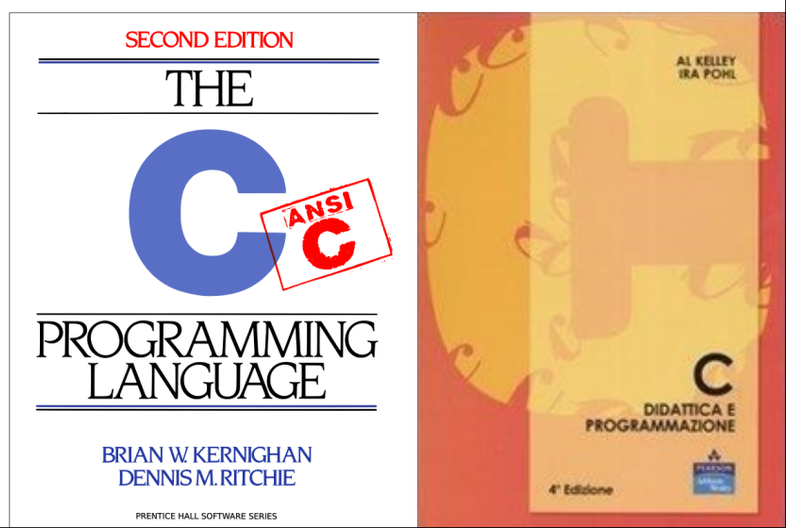


Lezioni di ripasso C

Mercoledì 19	11-13	Aula A-B
Giovedì 20	16-18	Aula A-B
Mercoledì 26	11-13	Aula A-B
Giovedì 27	16-18	Aula A-B

Le successive lezioni di laboratorio saranno

Corso B Giovedì	14-16	Aula H-M
Corso A Giovedì	16-18	Aula H-M

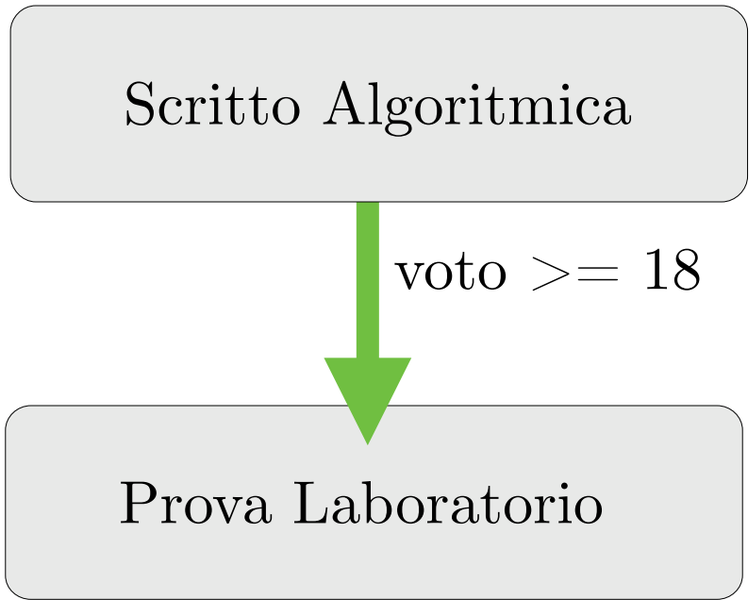


Esame

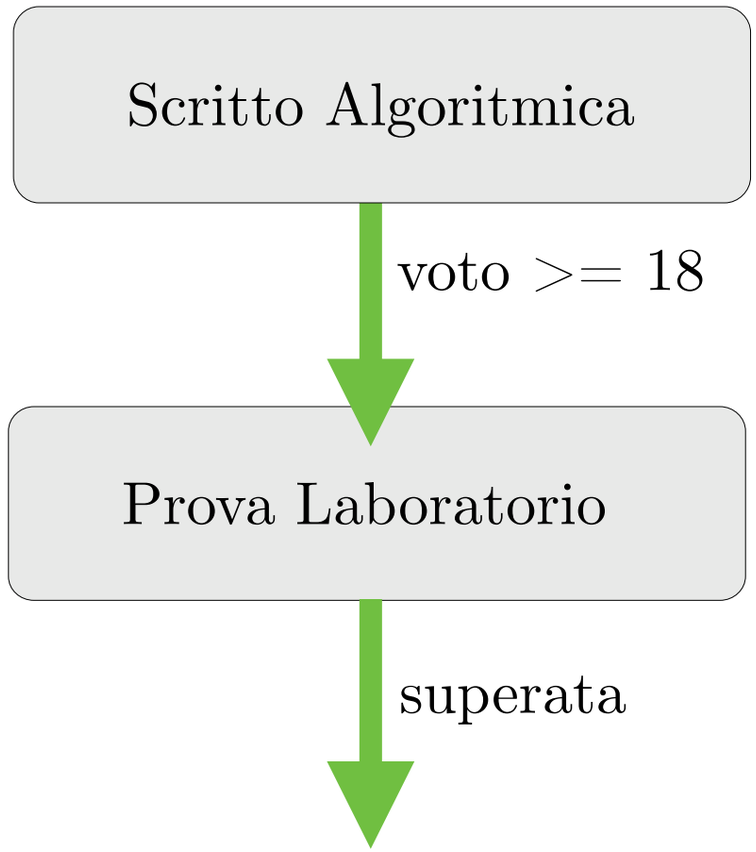
Esame

Scritto Algoritmica

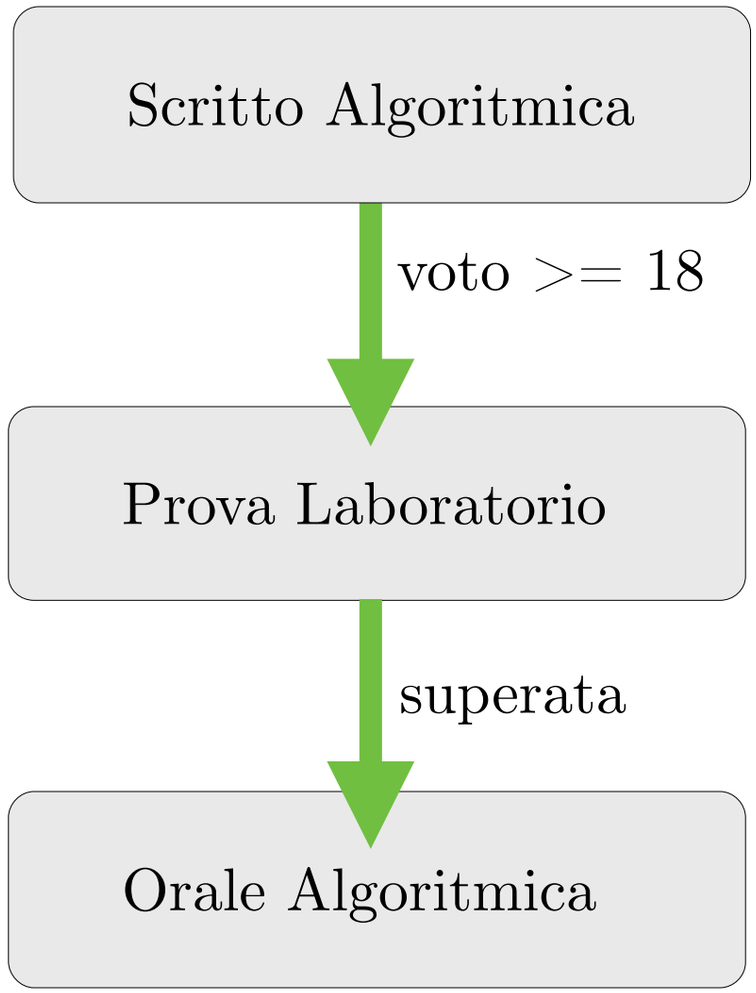
Esame



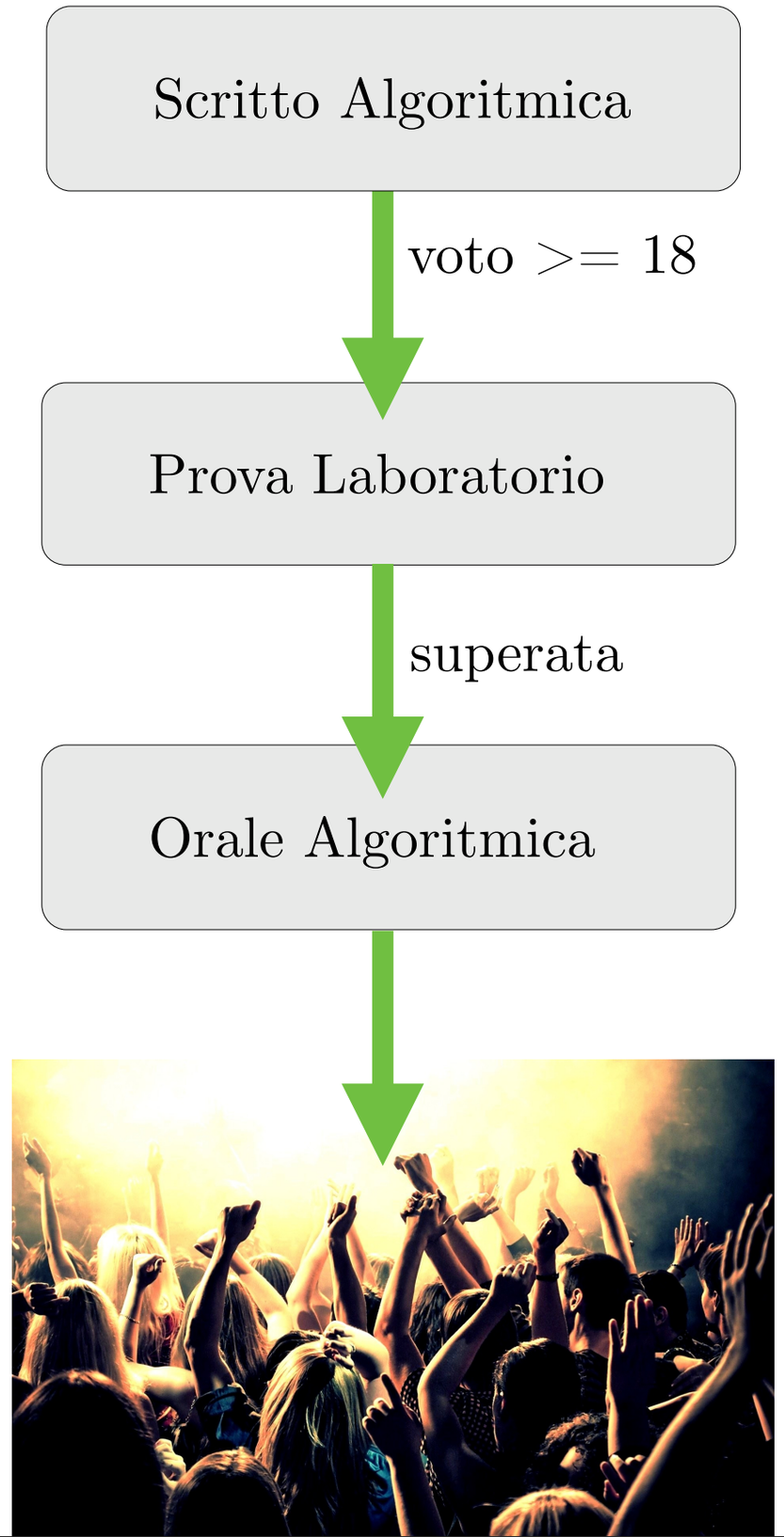
Esame



Esame


Scritto Algoritmica
voto >= 18
Prova Laboratorio
superata
Orale Algoritmica

Esame

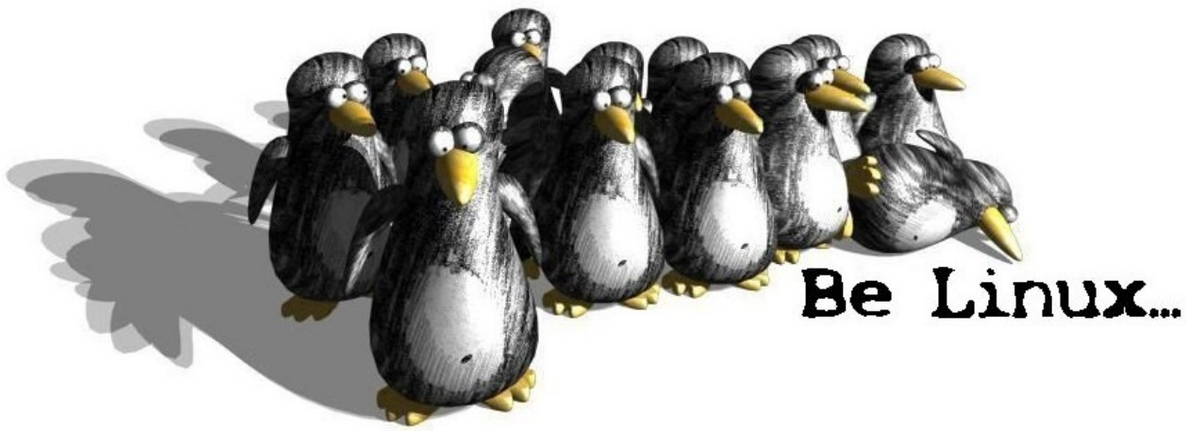

Scritto Algoritmica
voto >= 18
Prova Laboratorio
superata
Orale Algoritmica

Introduzione al C

- Strumenti che utilizzeremo nelle esercitazioni
  - Editare sorgenti: editor di testo generico (ad esempio, gedit per Linux)
  - Compilare: gcc per Linux, o tool di pubblico dominio per Windows (vedere la pagina Web del corso)

Introduzione al C

- Strumenti che utilizzeremo nelle esercitazioni
  - Editare sorgenti: editor di testo generico (ad esempio, gedit per Linux)
  - Compilare: gcc per Linux, o tool di pubblico dominio per Windows (vedere la pagina Web del corso)



Struttura di un programma C

Direttive al preprocessore

```
#include <*.h>
#define MAX (100)
#include <stdio.h> // generalmente necessaria
```

Dichiarazioni di variabili globali

```
char pippo;
...
int pluto;
```

Definizioni/dichiarazioni di funzioni

```
void foo();

int main () {
    /* esecuzione inizia da qui */
    ...
    return 0;  // necessario! Valore diverso da 0 indica un errore.
}
```

Scheletro di un programma C

```c
#include <stdio.h>

int main () {
    /* esecuzione inizia da qui */

    // corpo del funzione
    // scrivi qui il tuo codice
    // e spera per il meglio!

    return 0;
}
```

Esempio: hw.c

Aggiungiamo una chiamata alla funzione di libreria printf() (definita in stdio.h) per stampare una stringa.

Esempio: hw.c

Aggiungiamo una chiamata alla funzione di libreria printf() (definita in stdio.h) per stampare una stringa.

```
$ gedit hw.c &
```

Esempio: hw.c

Aggiungiamo una chiamata alla funzione di libreria printf() (definita in stdio.h) per stampare una stringa.

```
$ gedit hw.c &
```

```
#include <stdio.h>

int main () {
    printf("Hello World!\n");
    return 0;
}
```

Esempio: hw.c

Aggiungiamo una chiamata alla funzione di libreria printf() (definita in stdio.h) per stampare una stringa.

```
$ gedit hw.c &
```

```
#include <stdio.h>

int main () {
    printf("Hello World!\n");
    return 0;
}
```

Le costanti stringa in C sono sempre specificate tra una coppia di apici

Esempio: hw.c

Aggiungiamo una chiamata alla funzione di libreria printf() (definita in stdio.h) per stampare una stringa.

```
$ gedit hw.c &
```

```
#include <stdio.h>

int main () {
    printf("Hello World!\n");
    return 0;
}
```

Le costanti stringa in C sono sempre specificate tra una coppia di apici

\n è la *sequenza di escape* che codifica il carattere di fine linea

Esempio: hw.c

Aggiungiamo una chiamata alla funzione di libreria printf() (definita in stdio.h) per stampare una stringa.

```
$ gedit hw.c &
```

```
#include <stdio.h>

int main () {
    printf("Hello World!\n");
    return 0;
}
```

Compilazione ed esecuzione

```
```

Esempio: hw.c

Aggiungiamo una chiamata alla funzione di libreria printf() (definita in stdio.h) per stampare una stringa.

```
$ gedit hw.c &
```

```
#include <stdio.h>

int main () {
    printf("Hello World!\n");
    return 0;
}
```

Compilazione ed esecuzione

```
$ gcc -o hw hw.c
```

Esempio: hw.c

Aggiungiamo una chiamata alla funzione di libreria printf() (definita in stdio.h) per stampare una stringa.

```
$ gedit hw.c &
```

```
#include <stdio.h>

int main () {
    printf("Hello World!\n");
    return 0;
}
```

Compilazione ed esecuzione

```
$ gcc -o hw hw.c
$ ./hw
```

Esempio: hw.c

Aggiungiamo una chiamata alla funzione di libreria printf() (definita in stdio.h) per stampare una stringa.

```
$ gedit hw.c &
```

```
#include <stdio.h>

int main () {
    printf("Hello World!\n");
    return 0;
}
```

Compilazione ed esecuzione

```
$ gcc -o hw hw.c
$ ./hw
Hello World!
$
```

Dichiarazione e assegnamento

Una dichiarazione introduce il nome di una variabile e specifica il tipo di dato che essa conterrà.

Dichiarazione e assegnamento

Una dichiarazione introduce il nome di una variabile e specifica il tipo di dato che essa conterrà.

```
int x;
int y = 0;
char pippo, pluto;
```

Dichiarazione e assegnamento

Una dichiarazione introduce il nome di una variabile e specifica il tipo di dato che essa conterrà.

```
int x;
int y = 0;
char pippo, pluto;
```

Il contenuto di una variabile si può inizializzare o modificare mediante *assegnamenti* e *pre/post-incrementi*.

Dichiarazione e assegnamento

Una dichiarazione introduce il nome di una variabile e specifica il tipo di dato che essa conterrà.

```
int x;
int y = 0;
char pippo, pluto;
```

Il contenuto di una variabile si può inizializzare o modificare mediante *assegnamenti* e *pre/post-incrementi*.

```
x = 0;                    (forme equivalenti)
```

Dichiarazione e assegnamento

Una dichiarazione introduce il nome di una variabile e specifica il tipo di dato che essa conterrà.

```
int x;
int y = 0;
char pippo, pluto;
```

Il contenuto di una variabile si può inizializzare o modificare mediante *assegnamenti* e *pre/post-incrementi*.

```
x = 0;                    (forme equivalenti)
x = x + 1;                ++x; oppure x++;
```

Dichiarazione e assegnamento

Una dichiarazione introduce il nome di una variabile e specifica il tipo di dato che essa conterrà.

```
int x;
int y = 0;
char pippo, pluto;
```

Il contenuto di una variabile si può inizializzare o modificare mediante *assegnamenti* e *pre/post-incrementi*.

```
x = 0;                    (forme equivalenti)
x = x + 1;                ++x; oppure x++;
x = y + x;                x = x + y; oppure x += y;
```

Dichiarazione e assegnamento

Una dichiarazione introduce il nome di una variabile e specifica il tipo di dato che essa conterrà.

```
int x;
int y = 0;
char pippo, pluto;
```

Il contenuto di una variabile si può inizializzare o modificare mediante *assegnamenti* e *pre/post-incrementi*.

```
x = 0;                  (forme equivalenti)
x = x + 1;              ++x; oppure x++;
x = y + x;              x = x + y; oppure x += y;
x = x * 3;              x *= 3;
```

Tipi di dato primitivi

Tipi di dato primitivi

Tutti i tipi primitivi in C sono numerici
e gli operatori standard (+,-,*,/,%) sono definiti su di essi.

Tipi di dato primitivi

Tutti i tipi primitivi in C sono numerici
e gli operatori standard (+,-,*,/,%) sono definiti su di essi.

Interi: char, short, int, long

Si differenziano in base alla loro occupazione e, quindi, al range di valori rappresentabili.

Tipi di dato primitivi

Tutti i tipi primitivi in C sono numerici
e gli operatori standard (+,-,*,/,%) sono definiti su di essi.

Interi: char, short, int, long

Si differenziano in base alla loro occupazione e, quindi, al range di valori rappresentabili.

char	1 byte	usato per rappresentare caratteri ASCII

Tipi di dato primitivi

Tutti i tipi primitivi in C sono numerici
e gli operatori standard (+,-,*,/,%) sono definiti su di essi.

Interi: char, short, int, long

Si differenziano in base alla loro occupazione e, quindi, al range di valori rappresentabili.

char	1 byte	usato per rappresentare caratteri ASCII
short	2 byte	

Tipi di dato primitivi

Tutti i tipi primitivi in C sono numerici
e gli operatori standard (+,-,*,/,%) sono definiti su di essi.

Interi: char, short, int, long

Si differenziano in base alla loro occupazione e, quindi, al range di valori rappresentabili.

char	1 byte	usato per rappresentare caratteri ASCII
short	2 byte	
int	4 byte	
long	8 byte	

Tipi di dato primitivi

Tutti i tipi primitivi in C sono numerici
e gli operatori standard (+,-,*,/,%) sono definiti su di essi.

Interi: char, short, int, long

Si differenziano in base alla loro occupazione e, quindi, al range di valori rappresentabili.

char	1 byte	usato per rappresentare caratteri ASCII
short	2 byte	
int	4 byte	
long	8 byte	

Il modificatore unsigned restringe i valori rappresentabili ai soli positivi.

Tipi di dato primitivi

Tutti i tipi primitivi in C sono numerici
e gli operatori standard (+,-,*,/,%) sono definiti su di essi.

Interi: char, short, int, long

Si differenziano in base alla loro occupazione e, quindi, al range di valori rappresentabili.

char	1 byte	usato per rappresentare caratteri ASCII
short	2 byte	
int	4 byte	
long	8 byte	

Il modificatore unsigned restringe i valori rappresentabili ai soli positivi.

Floating-point: float, double

Si differenziano in base alla loro occupazione e, quindi, alla precisione della rappresentazione.

Tipi di dato primitivi

Tutti i tipi primitivi in C sono numerici
e gli operatori standard (+,-,*,/,%) sono definiti su di essi.

Interi: char, short, int, long

Si differenziano in base alla loro occupazione e, quindi, al range di valori rappresentabili.

char	1 byte	usato per rappresentare caratteri ASCII
short	2 byte	
int	4 byte	
long	8 byte	

Il modificatore unsigned restringe i valori rappresentabili ai soli positivi.

Floating-point: float, double

Si differenziano in base alla loro occupazione e, quindi, alla precisione della rappresentazione.

float	4 byte
double	8 byte

Costrutto condizionale: if-else

Costrutto condizionale: if-else

```
if ( guardia ) {
    // blocco 1
}  else  { // optional
    // blocco 2
}
```

Costrutto condizionale: if-else

```
if ( guardia ) {
    // blocco 1
}  else  { // optional
    // blocco 2
}
```

Indentate correttamente il vostro codice!

Costrutto condizionale: if-else

```
if ( guardia ) {
    // blocco 1
} else { // optional
    // blocco 2
}
```

Esempio

```
if ( voto >= 18 ) {
    printf("Promosso!");
} else {
    printf("Chi è il prossimo?");
}
```

Operatori logici e di confronto

Le guardie sono ottenute tipicamente combinando espressioni attraverso operatori logici e di confronto.

Operatori logici e di confronto

Le guardie sono ottenute tipicamente combinando espressioni attraverso operatori logici e di confronto.

Operatori logici

&&	And	!	Not
||	Or	^	Xor

Operatori logici e di confronto

Le guardie sono ottenute tipicamente combinando espressioni attraverso operatori logici e di confronto.

Operatori logici

&&	And	!	Not
||	Or	^	Xor

Operatori di confronto

<	Minore	<=	Minore o uguale
>	Maggiore	>=	Maggiore o uguale
==	Uguaglianza	!=	Diverso da

Operatori logici e di confronto

Le guardie sono ottenute tipicamente combinando espressioni attraverso operatori logici e di confronto.

Operatori logici

&&	And	!	Not
||	Or	^	Xor

Operatori di confronto

<	Minore	<=	Minore o uguale
>	Maggiore	>=	Maggiore o uguale
==	Uguaglianza	!=	Diverso da

Attenzione a non confondere
x==y (confronto) e
x=y (assegnamento)

Operatori logici e di confronto

Le guardie sono ottenute tipicamente combinando espressioni attraverso operatori logici e di confronto.

Operatori logici

&&	And	!	Not
||	Or	^	Xor

Operatori di confronto

<	Minore	<=	Minore o uguale
>	Maggiore	>=	Maggiore o uguale
==	Uguaglianza	!=	Diverso da

Esempio di guardia complessa

```
if ( anno%400 == 0 ||
     ( anno%100 != 0 && anno%4 == 0) ) {
     printf("Bisestile");
}
```

Booleani in C

In C non esiste un tipo primitivo per i booleani.
Questi vengono codificati con interi.

Booleani in C

In C non esiste un tipo primitivo per i booleani.
Questi vengono codificati con interi.

Un qualsiasi valore è interpretato come

Falso	se == 0
Vero	se != 0

Booleani in C

In C non esiste un tipo primitivo per i booleani.
Questi vengono codificati con interi.

Un qualsiasi valore è interpretato come

Falso	se == 0
Vero	se != 0

Gli operatori logici e di confronto sono a tutti gli effetti operatori aritmetici che restituiscono

0	se Falsi
1	se Veri

Booleani in C

In C non esiste un tipo primitivo per i booleani.
Questi vengono codificati con interi.

Un qualsiasi valore è interpretato come

Falso	se == 0
Vero	se != 0

Gli operatori logici e di confronto sono a tutti gli effetti operatori aritmetici che restituiscono

0	se Falsi
1	se Veri

Esempio

```
int x =67947;
if ( x ) {
    printf("x diverso da 0");
}
```

Costrutto iterativo: while

Costrutto iterativo: while

```
while ( guardia ) {
    // corpo del while
}
```

Costrutto iterativo: while

```
while ( guardia ) {
    // corpo del while
}
```

Esempio

```
int counter = 0;
while ( counter < 1000 ) {
    printf("Una corretta indentazione favorisce la leggibilità del codice");
}
```

Costrutto iterativo: while

```
while ( guardia ) {
    // corpo del while
}
```

Esempio

```
int counter = 0;
while ( counter < 1000 ) {
    printf("Una corretta indentazione favorisce la leggibilità del codice");
    counter++;
}
```

Non c'è bisogno di ripeterlo all'infinito, vero? ;-)

Costrutto iterativo: for

Costrutto iterativo: for

```
for( inizializzazione; guardia; incremento ) {
    // corpo del for
}
```

Costrutto iterativo: for

```
for( inizializzazione; guardia; incremento ) {
    // corpo del for
}
```

Esempio equivalente al precedente

```
int counter;
for ( counter = 0; counter < 1000; counter++ ) {
    printf("Una corretta indentazione favorisce la leggibilità del codice");
}
```

Costrutto iterativo: for

```
for( inizializzazione; guardia; incremento ) {
    // corpo del for
}
```

Esempio equivalente al precedente

```
int counter;
for ( counter = 0; counter < 1000; counter++ ) {
    printf("Una corretta indentazione favorisce la leggibilità del codice");
}
```

Il for può essere sempre riscritto come

```
inizializzazione;
while( guardia ) {
    // corpo del for
    incremento;
}
```

Array (1)

Ad alto livello, un array è una *collezione di oggetti delle stesso tipo*, raccolti sotto un unico nome e identificati da un indice intero compreso tra `0` e `n-1`, con `n` dimensione dell'array.

Array (1)

Ad alto livello, un array è una *collezione di oggetti delle stesso tipo*, raccolti sotto un unico nome e identificati da un indice intero compreso tra 0 e n-1, con n dimensione dell'array.

La sintassi per dichiarare array di *dimensione costante* in C è

tipo nome-array [dimensione]

Array (1)

Ad alto livello, un array è una *collezione di oggetti delle stesso tipo*, raccolti sotto un unico nome e identificati da un indice intero compreso tra 0 e n-1, con n dimensione dell'array.

La sintassi per dichiarare array di *dimensione costante* in C è

tipo nome-array [dimensione]

Dimensione è costante (ad es. 10, 100, 34526 ecc.) non una variabile (ad es. n, m, k, pippo ecc.).

Array (1)

Ad alto livello, un array è una *collezione di oggetti delle stesso tipo*, raccolti sotto un unico nome e identificati da un indice intero compreso tra 0 e n-1, con n dimensione dell'array.

La sintassi per dichiarare array di *dimensione costante* in C è

tipo nome-array [dimensione]

Alcuni esempi

```
int a[10];
char s[24];
int b[5] = {55,3,77,14,22}; // dichiara array e lo inizializza
```

Array (1)

Ad alto livello, un array è una *collezione di oggetti delle stesso tipo*, raccolti sotto un unico nome e identificati da un indice intero compreso tra 0 e n-1, con n dimensione dell'array.

La sintassi per dichiarare array di *dimensione costante* in C è

tipo nome-array [dimensione]

Alcuni esempi

```
int a[10];
char s[24];
int b[5] = {55,3,77,14,22}; // dichiara array e lo inizializza
```

Cosa non fare

```
int n = 20;
char s[n];
```

Array (2)

Ovviamente è possibile accedere/modificare il valore di un qualunque elemento di un array specificando il suo indice.

Array (2)

Ovviamente è possibile accedere/modificare il valore di un qualunque elemento di un array specificando il suo indice.

Alcuni esempi

```
int b[5] = {55,3,77,14,22};
b[0] = 6;
int x = b[4] + 1;
```

Array (2)

Ovviamente è possibile accedere/modificare il valore di un qualunque elemento di un array specificando il suo indice.

Alcuni esempi

```
int b[5] = {55,3,77,14,22};
b[0] = 6;
int x = b[4] + 1;
```

Cosa non fare

```
int b[5] = {55,3,77,14,22};
b[5] = 4; // accesso out-of-bound
```

Array (2)

Ovviamente è possibile accedere/modificare il valore di un qualunque elemento di un array specificando il suo indice.

Alcuni esempi

```
int b[5] = {55,3,77,14,22};
b[0] = 6;
int x = b[4] + 1;
```

Cosa non fare

```
int b[5] = {55,3,77,14,22};
b[5] = 4; // accesso out-of-bound
```

Nessun errore a tempo di compilazione, segmentation fault o comportamenti indesiderati a tempo di esecuzione.

Array (3)

Tipicamente si usa un ciclo for per scandire gli elementi di un array.

Array (3)

Tipicamente si usa un ciclo for per scandire gli elementi di un array.

Esempio: inizializzare gli elementi di un array con interi (pseudo-)casuali

Array (3)

Tipicamente si usa un ciclo for per scandire gli elementi di un array.

Esempio: inizializzare gli elementi di un array con interi (pseudo-)casuali

```
#include <stdlib.h>
#include <time.h>

int main() {
    ...
    /* inizializza il generatore di numeri pseudocasuali con seme time(NULL).
       Usate un numero qualunque su Windows. */
    srand(time(NULL));

    int a[10];
    for ( int i = 0; i < 10; i++) {
        a[i] = rand() % 100; // intero tra 0 e 99
        // fai qualcosa con a[i]
    }
    return 0;
}
```

Array (3)

Tipicamente si usa un ciclo for per scandire gli elementi di un array.

Esempio: inizializzare gli elementi di un array con interi (pseudo-)casuali

```
#include <stdlib.h>
#include <time.h>

int main() {
    ...
    /* inizializza il generatore di numeri pseudocasuali con seme time(NULL).
       Usate un numero qualunque su Windows. */
    srand(time(NULL));

    int a[10];
    for ( int i = 0; i < 10; i++) {
        a[i] = rand() % 100; // intero tra 0 e 99
        // fai qualcosa con a[i]
    }
    return 0;
}
```

Array (3)

Tipicamente si usa un ciclo for per scandire gli elementi di un array.

Esempio: inizializzare gli elementi di un array con interi (pseudo-)casuali

```
#include <stdlib.h>
#include <time.h>

int main() {
    ...
    /* inizializza il generatore di numeri pseudocasuali con seme time(NULL).
       Usate un numero qualunque su Windows. */
    srand(time(NULL));

    int a[10];
    for ( int i = 0; i < 10; i++) {
        a[i] = rand() % 100; // intero tra 0 e 99
        // fai qualcosa con a[i]
    }
    return 0;
}
```

Array (3)

Tipicamente si usa un ciclo for per scandire gli elementi di un array.

Esempio: inizializzare gli elementi di un array con interi (pseudo-)casuali

```
#include <stdlib.h>
#include <time.h>

int main() {
    ...
    /* inizializza il generatore di numeri pseudocasuali con seme time(NULL).
       Usate un numero qualunque su Windows. */
    srand(time(NULL));

    int a[10];
    for ( int i = 0; i < 10; i++) {
        a[i] = rand() % 100; // intero tra 0 e 99
        // fai qualcosa con a[i]
    }
    return 0;
}
```

Array (3)

Tipicamente si usa un ciclo for per scandire gli elementi di un array.

Esempio: inizializzare gli elementi di un array con interi (pseudo-)casuali

```
#include <stdlib.h>
#include <time.h>

int main() {
    ...
    /* inizializza il generatore di numeri pseudocasuali con seme time(NULL).
       Usate un numero qualunque su Windows. */
    srand(time(NULL));

    int a[10];
    for ( int i = 0; i < 10; i++) {
        a[i] = rand() % 100; // intero tra 0 e 99
        // fai qualcosa con a[i]
    }
    return 0;
}
```

Array (3)

Tipicamente si usa un ciclo for per scandire gli elementi di un array.

Esempio: inizializzare gli elementi di un array con interi (pseudo-)casuali

```
#include <stdlib.h>
#include <time.h>

int main() {
    ...
    /* inizializza il generatore di numeri pseudocasuali con seme time(NULL).
       Usate un numero qualunque su Windows. */
    srand(time(NULL));

    int a[10];
    for ( int i = 0; i < 10; i++) {
        a[i] = rand() % 100; // intero tra 0 e 99
        // fai qualcosa con a[i]
    }
    return 0;
}
```

Stampare: printf

printf è una funzione di libreria (in stdio.h) per stampare testo formattato sullo standard output.

Stampare: printf

printf è una funzione di libreria (in stdio.h) per stampare testo formattato sullo standard output.

printf(“formato-output”, lista-argomenti)

Stampare: printf

printf è una funzione di libreria (in stdio.h) per stampare testo formattato sullo standard output.

printf(“formato-output”, lista-argomenti)

printf rimpiazza in formato-output ogni place-holder con il corrispondente argomento e stampa il risultato sullo standard output.

Stampare: printf

printf è una funzione di libreria (in stdio.h) per stampare testo formattato sullo standard output.

printf(“formato-output”, lista-argomenti)

printf rimpiazza in formato-output ogni place-holder con il corrispondente argomento e stampa il risultato sullo standard output.

Esempio

```
int x = 5, y = 10;
printf(“%d + %d = %d\n”, x, y, x+y);
```

Stampare: printf

printf è una funzione di libreria (in stdio.h) per stampare testo formattato sullo standard output.

printf(“formato-output”, lista-argomenti)

printf rimpiazza in formato-output ogni place-holder con il corrispondente argomento e stampa il risultato sullo standard output.

Esempio

```
int x = 5, y = 10;
printf(“%d + %d = %d\n”, x, y, x+y);
```

```
5 + 10 = 15
```

Stampare: printf

printf è una funzione di libreria (in stdio.h) per stampare testo formattato sullo standard output.

printf(“formato-output”, lista-argomenti)

printf rimpiazza in formato-output ogni place-holder con il corrispondente argomento e stampa il risultato sullo standard output.

Esempio

```
int x = 5, y = 10;
printf(“%d + %d = %d\n”, x, y, x+y);
```

Ogni tipo ha il suo place-holder.
%d è quello degli interi.

Stampare: printf

printf è una funzione di libreria (in stdio.h) per stampare testo formattato sullo standard output.

printf(“formato-output”, lista-argomenti)

printf rimpiazza in formato-output ogni place-holder con il corrispondente argomento e stampa il risultato sullo standard output.

Esempio

\n è il ritorno a capo

```
int x = 5, y = 10;
printf(“%d + %d = %d\n”, x, y, x+y);
```

```
5 + 10 = 15
```

Stampare: printf

printf è una funzione di libreria (in stdio.h) per stampare testo formattato sullo standard output.

printf("formato-output", lista-argomenti)

printf rimpiazza in formato-output ogni place-holder con il corrispondente argomento e stampa il risultato sullo standard output.

Alcuni place-holder

%d	int	%c	char
%s	stringa	%p	indirizzo
%f	float	%lf	double

Leggere: scanf

scanf è una funzione di libreria (in stdio.h) per leggere valori o testo dallo standard input.

Leggere: scanf

scanf è una funzione di libreria (in stdio.h) per leggere valori o testo dallo standard input.

scanf(“formato-input”, lista-argomenti)

Leggere: scanf

scanf è una funzione di libreria (in stdio.h) per leggere valori o testo dallo standard input.

scanf(“formato-input”, lista-argomenti)

Sintassi simile a printf ma comportamento simmetrico.

Leggere: scanf

scanf è una funzione di libreria (in stdio.h) per leggere valori o testo dallo standard input.

scanf(“formato-input”, lista-argomenti)

Sintassi simile a printf ma comportamento simmetrico.

Esempio

```
int x = 5;
printf(“%d\n”, x);
scanf(“%d”, &x);
printf(“%d\n”, x);
```

Leggere: scanf

scanf è una funzione di libreria (in stdio.h) per leggere valori o testo dallo standard input.

scanf(“formato-input”, lista-argomenti)

Sintassi simile a printf ma comportamento simmetrico.

Esempio

```
int x = 5;
printf(“%d\n”, x);
scanf(“%d”, &x);
printf(“%d\n”, x);
```

Richiede l’indirizzo della variabile che conterrà il valore letto (per questo &x). Più dettagli nella prossima lezione!

Leggere: scanf

scanf è una funzione di libreria (in stdio.h) per leggere valori o testo dallo standard input.

scanf(“formato-input”, lista-argomenti)

Sintassi simile a printf ma comportamento simmetrico.

Esempio

```
int x = 5;
printf(“%d\n”, x);
scanf(“%d”, &x);
printf(“%d\n”, x);
```

```
5
```

Leggere: scanf

scanf è una funzione di libreria (in stdio.h) per leggere valori o testo dallo standard input.

scanf(“formato-input”, lista-argomenti)

Sintassi simile a printf ma comportamento simmetrico.

Esempio

```
int x = 5;
printf(“%d\n”, x);
scanf(“%d”, &x);
printf(“%d\n”, x);
```

```
5
10
```

Leggere: scanf

scanf è una funzione di libreria (in stdio.h) per leggere valori o testo dallo standard input.

scanf(“formato-input”, lista-argomenti)

Sintassi simile a printf ma comportamento simmetrico.

Esempio

```
int x = 5;
printf(“%d\n”, x);
scanf(“%d”, &x);
printf(“%d\n”, x);
```

```
5
10
10
```

Esempio printf e scanf

Esempio printf e scanf

Area di un cerchio di raggio dato

```
#include <stdio.h>
#define PI (3.1415f)

int main() {
    float r; // raggio del cerchio
    float a; // area del cerchio

    scanf("%f", &r);
    a = PI * r * r;
    printf("%f\n", a);

    return 0;
}
```

Esempio printf e scanf

Area di un cerchio di raggio dato

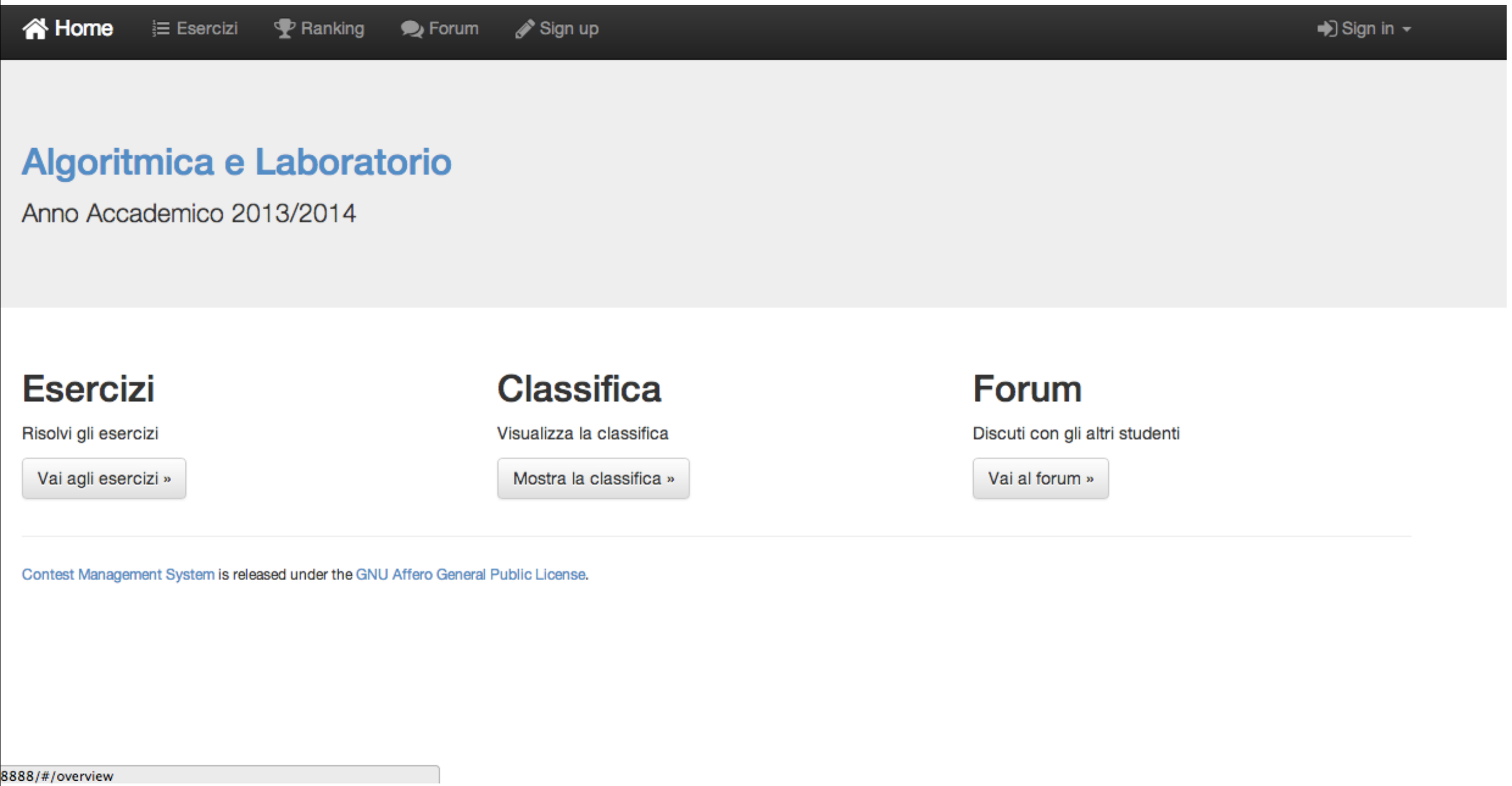
```
#include <stdio.h>
#define PI (3.1415f)

int main() {
    float r; // raggio del cerchio
    float a; // area del cerchio

    scanf("%f", &r);
    a = PI * r * r;
    printf("%f\n", a);

    return 0;
}
```

Esempio printf e scanf

Area di un cerchio di raggio dato

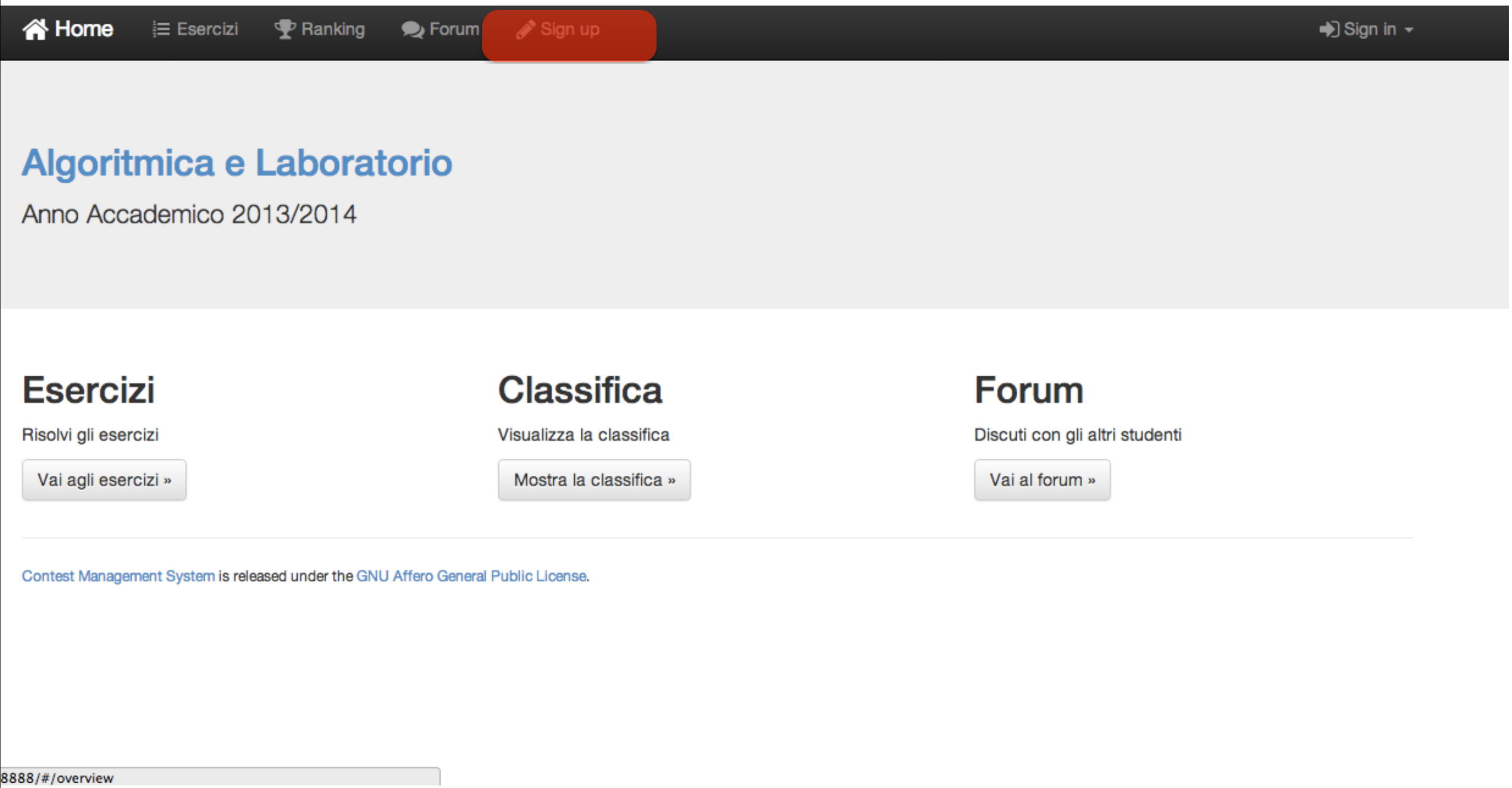
```
#include <stdio.h>
#define PI (3.1415f)

int main() {
    float r; // raggio del cerchio
    float a; // area del cerchio

    scanf("%f", &r);
    a = PI * r * r;
    printf("%f\n", a);

    return 0;
}
```

Sistema di autovalutazione



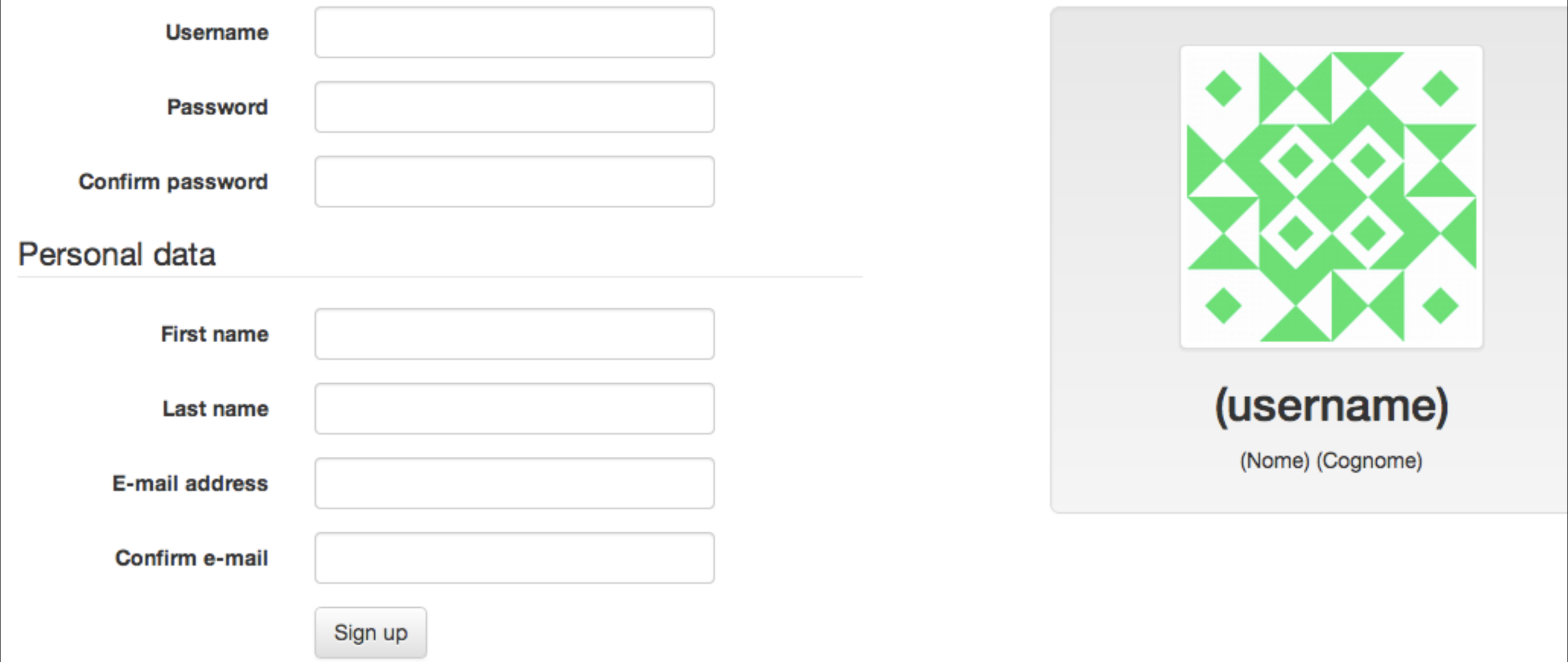
vinello.isti.cnr.it:8888/

Sistema di autovalutazione



vinello.isti.cnr.it:8888/

Sistema di autovalutazione



vinello.isti.cnr.it:8888/

Sistema di autovalutazione

vinello.isti.cnr.it:8888/

Sistema di autovalutazione

vinello.isti.cnr.it:8888/

Sistema di autovalutazione

L01E01 Ciao Mondo

Statement | Attachments 1 | Stats | </> Submissions | 1 sec | 512 MiB

Page: 1 of 1 | Automatic Zoom

Ciao Mondo!

Esercizio

Scrivere un programma che stampi `Ciao Mondo!` sul terminale.

Esempio

Input	Output
	`Ciao Mondo!`

vinello.isti.cnr.it:8888/

Sistema di autovalutazione

L01E01 Ciao Mondo

Statement | Attachments 1 | Stats | </> Submissions | 1 sec | 512 MiB

Page: 1 of 1 | Automatic Zoom

Ciao Mondo!

Esercizio

Scrivere un programma che stampi `Ciao Mondo!` sul terminale.

Esempio

Input	Output
	`Ciao Mondo!`

vinello.isti.cnr.it:8888/

Sistema di autovalutazione

L01E01 Ciao Mondo

Statement | Attachments 1 | Stats | </> Submissions | 1 sec | 512 MiB

Page: 1 of 1 | Automatic Zoom

Ciao Mondo!

Esercizio

Scrivere un programma che stampi `Ciao Mondo!` sul terminale.

Esempio

Input	Output
	`Ciao Mondo!`

vinello.isti.cnr.it:8888/

Esercizio 1

Ciao Mondo!

Esercizio

Scrivere un programma che stampi `Ciao Mondo!` sul terminale.

Esempio

Input	Output
	`Ciao Mondo!`

Esercizio 2

Primo

Esercizio

Scrivere un programma che legga da tastiera un intero e stabilisca se il numero è primo.
L'input consiste di una sola riga conente l'intero x.
Il programma stampa in output 1 se x è primo, 0 altrimenti.

Esempi

Input	Output
226	0

Input	Output
13	1

Esercizio 3

Somma

Esercizio

Scrivere un programma che legga da tastiera una sequenza di numeri interi terminata da 0 e ne stampi la somma.
L'input una sequenza di numeri interi terminata da 0, un intero per riga.
Il programma stampa in output la somma degli interi.

Esempi

Input	Output
3 9 1 2 3 0	18

Esercizio 4

Invertire un array

Esercizio

Scrivere un programma che legga da input gli N interi di un array A. Il programma deve invertire A in loco (cioè senza utilizzare un array di appoggio), ossia scambiare il contenuto della prima e dell'ultima cella, della seconda e della penultima, ecc.
Si assuma che $N \leq 10000$.
La prima riga dell'input è il valore N. Seguono N interi, uno per riga.
Il programma stampa in output gli elementi dell'array invertito, uno per riga.

Esempi

Input	Output
5 3 1 4 0 4	4 0 4 1 3