

Assegnamenti:

- Sintassi:
 - nomevariabile = espressione
- Espressione può essere un semplice valore, oppure una espressione complessa.

```
a=3;  
b = (a+2)*3;  
  
a = a+1;
```

Cosa significa il terzo
assegnamento?

Input formattato

- Per l'input da tastiera esistono molte possibilità. La più semplice è usare la funzione `scanf()` che, anche se ha qualche problema, per adesso va più che bene.

- La `scanf()` si usa in un modo molto simile alla `printf()`.

```
int a;  
printf("Inserire il valore di a:");  
scanf("%d", &a);
```

```
printf("Ora a vale %d", a);
```

Prestare attenzione al
segno di &

Operatori aritmetici

- Sono quelli tradizionali:
 - +, -, *, /
 - % è il resto della divisione intera
 - Esistono inoltre delle forme abbreviate:
 - `a += b;` equivale a scrivere `a=a+b;`
 - esistono gli analoghi: `-=`, `*=`, `/=`
 - `a++;` equivale a scrivere `a=a+1;`
 - `++a;` equivale a scrivere `a=a+1;` a differenza di prima, prima si incrementa la variabile, e poi si valuta l'espressione
 - `a--;` equivale a scrivere `a=a-1;`
 - `--a;` equivale a scrivere `a=a-1;` a differenza di prima, prima si decrementa la variabile, e poi si valuta l'espressione

Operatori aritmetici

- Confrontare:

```
a=10;  
b=a++;
```

```
a=10;  
b=++a;
```

Sperimentiamolo in
diretta!

Operatori relazionali

- Useremo questi operatori per scrivere condizioni logiche che controllano il flusso di esecuzione.
- Niente di sorprendente nella sintassi:
 - `<`, `<=`, `>`, `>=`
- L'uguaglianza è l'operatore `==`
 - attenzione a non confonderlo con l'operatore di assegnamento! Sono due concetti molto diversi.
- La disuguaglianza è `!=`
- La negazione logica è `!`
- AND logico è `&&`
- OR logico è `||`

Strutture di controllo

- Sequenziale
- Condizionale
- Iterativa
 - Con controllo in testa
 - Con controllo in coda

Struttura sequenziale

- Le istruzioni sono eseguite una dopo l'altra, nel medesimo ordine in cui sono scritte.

```
printf("Anzitutto stampa questo...\n");  
printf("...poi quest'altro...\n");  
printf("Ed infine questo");
```

- Non è detto che le istruzioni debbano essere tutte dello stesso tipo.

Struttura condizionale: if

- Viene valutata una condizione:
 - se vera (cioè non-zero) si esegue il **primo blocco** di istruzioni
 - se falsa (cioè zero) si esegue il **secondo blocco** di istruzioni

```
if ( (a%2) == 0 ) {  
    printf("La variabile a ");  
    printf("contiene un numero pari.");  
} else {  
    printf("In a è contenuto ");  
    printf("un numero dispari.");  
}
```

Le parentesi intorno
alla condizione sono
OBBLIGATORIE!

Struttura condizionale: if

- La sezione else è facoltativa.
 - In sua assenza, se la condizione è falsa, non viene eseguita alcuna istruzione.
- Viceversa, devono essere specificate delle istruzioni da eseguire se la condizione è vera!

```
if ( (a%2) == 0 ){  
    printf("La variabile a ");  
    printf("contiene un numero pari.");  
}  
printf("Questa linea è stampata in ogni caso");
```

Struttura condizionale: if

- Se uno (o entrambi) dei blocchi è composto da una istruzione solamente, allora è possibile omettere le graffe di quel blocco:

```
if ( (a%2) == 0 )  
    printf("La variabile a contiene un numero pari ");  
else  
    printf("In a è contenuto un numero dispari.");
```

Struttura condizionale: if

- È possibile innestare più if uno dentro l'altro:

```
if ( (a%2) == 0 ) {  
    printf("La variabile a contiene un numero pari ");  
    if (a>5) {  
        printf("Il valore di a è maggiore di 5");  
    }  
}  
  
} else {  
    printf("In a è contenuto un numero dispari.");  
}
```

Esercizio:
Costruire il diagramma di
flusso

Struttura iterativa: while

- Viene valutata la condizione. Se è vera, si continua ad eseguire il corpo del ciclo fintantoché la condizione resta vera.

```
int i = 10;
while (i >= 0) {
    printf("%d...\n", i);
    i = i-1;
}
printf("Fuoco!\n");
```

Attenzione a non dimenticarsi di decrementare la variabile!

- Nota: il corpo del ciclo potrebbe non essere mai eseguito... (quando?)

Struttura iterativa: do-while

- Viene eseguito il corpo del ciclo, quindi viene valutata la condizione. Se è vera, si continua ad eseguire il corpo del ciclo fintantoché la condizione resta vera.

```
int i = 10;
do {
    printf("%d...\n", i);
    i--;
} while (i >= 0);
printf("Fuoco!\n");
```

- Nota: il corpo del ciclo è sempre eseguito almeno una volta...

Strutture di controllo

- Queste sono le strutture di controllo fondamentali del linguaggio C.
- In realtà ne esistono alcune altre, ma non aumentano la potenza espressiva.
- Tra queste la più importante è il for, che altro non è che un modo più comodo di riscrivere un certo tipo di while

Esercizio:

- Calcolare il MCD tra due numeri inseriti da tastiera. Vedere i lucidi di teoria per l'algoritmo.

Commenti

- Spesso è utile inserire del testo che spiega alcune parti di codice. Questo testo è detto *commento*, ed è marcato in modo particolare: infatti il compilatore deve riconoscerlo per poterlo ignorare.
- I commenti servono non al compilatore bensì agli esseri umani che devono leggere il programma (magari per modificarlo).

Commenti

- Tutto ciò che è compreso tra la sequenza `/*` e la sequenza `*/` è considerato un commento.
- Il commento può anche estendersi su più righe

```
/* Questo è  
un commento */
```

Commenti

- Una estensione del linguaggio, riconosciuta da molti (ma non tutti!) i compilatori, permette di iniziare i commenti con la sequenza //
- Questo tipo di commenti non deve essere chiuso: termina automaticamente alla fine della riga

```
// Questo è un altro commento
```