LABORATORIO DI PROGRAMMAZIONE 1

Docenti: Vincenzo Bonnici (A - L)
Maurizio Boscaini (M - Z)

Lezione 3 – a.a. 2016/2017

Costrutti di Iterazione

- I costrutti di iterazione consentono di eseguire ripetutamente una serie di istruzioni.

- Il linguaggio C offre tre forme di iterazione:

  - L'istruzione `for` .
  - L'istruzione `while`.
  - L'istruzione `do-while`.

Istruzione For

```
for( espr_iniziale; cond_ciclo; espr ciclo ){
  istruzione_1;
  ...
  istruzione_n;
}
```

- `espr_iniziale`: utilizzata per impostare i valori iniziali *prima* che cominci il ciclo (es. variabile indice).
- `cond_ciclo`: condizione (o condizioni) necessarie affinché il ciclo possa continuare.
  - Il ciclo continua finché la condizione è soddisfatta, quando la condizione non è più soddisfatta l'esecuzione continua con l'istruzione posta subito dopo il ciclo `for`.
- `espr_ciclo`: valutata *dopo* che il corpo del ciclo è stato eseguito.
  - Generalmente utilizzata per modificare il valore della variabile indice.
- `istruzione_1; ... istruzione_n`: *corpo del ciclo.*
- non servono le parentesi {} se c'è solo una *istruzione* nel corpo del ciclo (ma è buona prassi metterle comunque)

Istruzione For

- Esecuzione ciclo `for`:
  1. Viene valutata l'espressione iniziale (`espr_iniziale`).
  2. Viene valutata la condizione del ciclo (`cond_ciclo`).
     - Se la condizione è falsa il ciclo termina e si procede con l'istruzione immediatamente successiva al ciclo.
     - Se la condizione è vera viene eseguito una volta il corpo del ciclo.
  3. Viene valutata l'espressione del ciclo (`espr_ciclo`).
     - Generalmente modifica la variabile indice.
  4. Si ritorna al punto 2.

Istruzione For

- Programma che calcola la somma dei primi *n* numeri interi:

```
int i;
int n;
int somma = 0;
...
for( i = 1; i <= n; i += 1 ){
  somma = somma + i;
}
...
```

Espressioni booleane

- La condizione del ciclo può essere qualsiasi espressione booleana (valore vero o falso).

- Operatori relazionali:
  - `a < b`: `a` minore di `b`,
  - `a <= b`: `a` minore o uguale a `b`,
  - `a > b`: `a` maggiore di `b`,
  - `a >= b`: `a` maggiore o uguale a `b`,
  - `a == b`: `a` uguale a `b`.
  - `a != b`: `a` diverso da `b`.

- Le espressioni booleane possono essere combinate con gli operatori logici binari di and (`&&`) e or (`||`), oppure negate con l'operatore logico (unario) di negazione (`!`).
  - A `&&` B vera sse A vera e B vera
  - A `||` B vera sse almeno una tra A e B è vera
  - `!` A vera sse A è falsa

- Gli operatori relazionali hanno priorità più bassa rispetto agli operatori aritmetici:
  - `a < b + c` ⟶ `a < (b+c)`

Esercizio 1

- Scrivere un programma C che stampa la tabellina di un numero *n* inserito dall'utente: la tabellina deve essere calcolata da 1 fino ad un valore *x* sempre inserito dall'utente.

- Il programma deve:
  - Richiedere il numero *n*.
  - Richiedere il numero *x*.
  - Il numero *n* deve essere moltiplicato per un valore *i* che va da 1 fino a *x*.

Operatori di Incremento/Decremento

- Abbiamo visto gli *operatori di assegnamento*:
  - Operatori aritmetici (+,-,*,/,%) associati all'operatore di assegnamento.
  - `i += 10` → `i = i+10`

- Quando il valore di una variabile deve essere solo incrementato o decrementato di 1 si possono usare le forme compatte degli operatori di incremento/decremento:
  - `i++` → `i = i+1`
  - `i--` → `i = i-1`

Operatori di Incremento/Decremento

I simboli ++ e — possono essere posti prima o dopo il nome della variabile producendo lo stesso effetto di incremento sulla variabile:

- `++i`, `--i` (forma *prefissa*)
- `i++`, `i--` (forma *postfissa*)

- La differenza tra la forma prefissa è postfissa riguarda solo quando questi costrutti sono considerati come espressioni:
  - *Forma prefissa*: il valore dell'espressione è quello della variabile dopo l'incremento.
  - *Forma postfissa*: il valore dell'espressione è quello della variabile prima dell'incremento.

Operatori di Incremento/Decremento

```
int a = 2;
printf( "Il valore di a e' %i", a++ );
```

```
  Il valore di a e' 2;
```

```
printf( "Il valore di a e' %i", a );
```

```
   Il valore di a e' 3;
```

```
int a = 2;
printf( "Il valore di a e' %i, ++a );
```

```
   Il valore di a e' 3;
```

```
printf( "Il valore di a e' %i, a );
```

```
   Il valore di a e' 3;
```

Operatori di Incremento/Decremento

- Quando gli operatori di incremento/decremento sono usati in un ciclo `for` come espressione di ciclo, l'effetto della forma prefissa e postfissa è lo stesso perché prima viene valutata l'espressione ed effettuato l'assegnamento e poi viene valutata la condizione di ciclo.

```
for( i = 0; i < n; i++ ){ ... }
          ==
for( i = 0; i < n; ++i ){ ... }
```

Esercizio 2

- Scrivere un programma C che calcola il fattoriale di un numero intero *n* richiesto all'utente.

- Il fattoriale di un numero intero *n* (scritto *n*!) è il prodotto dei numeri interi da 1 a *n*.
  - *n*! = 1 * 2 *... * *n*;
  - Caso particolare: il fattoriale di 0 è 1.

- Quando avete scritto il programma... provate a calcolare i fattoriali di 30, 31, 32, 33, 34.

Varianti del Ciclo For

- I cicli for possono essere annidati tra loro.

```
for( ... ) {
  for( ... ) {
  }
}
```

- Espressioni multiple: è possibile indicare più espressioni separate da virgola come `espr_iniziale` e/o `espr_ciclo`:

```
for( i = 0, j = 0; i < 10; ++i, j-=5 ) { ... }
```

- Campi omessi: è possibile omettere uno o più campi (ma non il corrispondente punto e virgola).
  - Tipicamente si omette l'espressione iniziale se questa è complessa e calcolata prima dell'esecuzione del ciclo.

```
j = ...;
for ( ; j < 100; ++j ){ ... }
```

Formattazione: Larghezza Campo

- Nelle stringhe di formato usate nella funzione `printf` è possibile specificare una larghezza di campo:
  - `%2i`: specifica che l'intero da stampare deve occupare due colonne (allineamento a destra dei valori con una sola cifra).
  - `%.2f`: specifica che la parte decimale del valore in virgola mobile deve occupare due colonne (troncamento solo in output).

Esercizio 3

- Scrivere un programma C che stampa una matrice *n*x*n* dove *n* è un valore richiesto all'utente. In ogni cella della matrice va stampata la coppia (*i*,*j*) dove *i* è il numero di riga e *j* il numero di colonna.
  - Utilizzare opportunamente le stringhe di formato per impostare la larghezza di campo. Assumere che *n* sia composto al massimo da 2 cifre.

Esercizio 3 [esempio di output]
Matrici 4x4 e 6x6

```
Inserire un numero intero: 4
( 1, 1) ( 1, 2) ( 1, 3) ( 1, 4)
( 2, 1) ( 2, 2) ( 2, 3) ( 2, 4)
( 3, 1) ( 3, 2) ( 3, 3) ( 3, 4)
( 4, 1) ( 4, 2) ( 4, 3) ( 4, 4)
```

```
Inserire un numero intero: 6
( 1, 1) ( 1, 2) ( 1, 3) ( 1, 4) ( 1, 5) ( 1, 6)
( 2, 1) ( 2, 2) ( 2, 3) ( 2, 4) ( 2, 5) ( 2, 6)
( 3, 1) ( 3, 2) ( 3, 3) ( 3, 4) ( 3, 5) ( 3, 6)
( 4, 1) ( 4, 2) ( 4, 3) ( 4, 4) ( 4, 5) ( 4, 6)
( 5, 1) ( 5, 2) ( 5, 3) ( 5, 4) ( 5, 5) ( 5, 6)
( 6, 1) ( 6, 2) ( 6, 3) ( 6, 4) ( 6, 5) ( 6, 6)
```