Corso “Programmazione 1”
Capitolo 03: Istruzioni

Docente: Marco Roveri - marco.roveri@unitn.it
Esercitatori: Giovanni De Toni - giovanni.detoni@unitn.it
Stefano Berlato - stefano.berlato-1@unitn.it

C.D.L.: Informatica (INF)
A.A.: 2021-2022
Luogo: DISI, Università di Trento
URL: https://bit.ly/2VgfYwJ

Ultimo aggiornamento: 27 settembre 2021

Terms of Use and Copyright

The material (text, figures) in these slides is authored mostly by Roberto Sebastiani, with contributions by Marco Roveri, Alessandro Armando, Enrico Giunchiglia e Sabrina Recla.

Struttura di un programma

- Un programma consiste in un insieme di funzioni, eventualmente suddivise in più file
  - La funzione che costituisce il programma principale si deve necessariamente chiamare `main`
- Ogni programma contiene una lista di istruzioni di ogni tipo:
  istruzioni semplici o istruzioni strutturate

Esempio:

```
(...)
int main() {
  int x=2, y=6, z;
  z=x*y;
  return 0;
}
```

Istruzioni semplici

- Le istruzioni semplici sono la base delle istruzioni più complesse (istruzioni strutturate)
- Sono sempre terminate da un punto-e-virgola "`;`"
- Si distinguono in:
  - definizioni/dichiarazioni (declaration-statement) di
    - variabili, es. `int x,y,z;`
    - costanti, es. `const int kilo=1024;`
  - espressioni (expression-statement)
    - di input, es: `cin >> x`
    - di output, es: `cout << 3*x`
    - di assegnamento, es. `x=2*(3-y)`
    - matematiche, es. `(x-3)*sin(x)`
    - logiche, es. `x==y && x!=z`
    - costanti, es. `3*12.7`
    - condizionali (a seguire)

  ogni espressione seguita da un "`;`" è anche un'istruzione

L'espressione condizionale

- Sintassi: `exp1 ? exp2 : exp3` :
  Se exp1 è vera equivale a exp2, altrimenti equivale a exp3

Esempio

```
prezzo = valore * peso * (peso>10) ? 0.9 : 1;
```

- se il peso è maggiore di 10, equivale a:
  `prezzo = valore * peso * 0.9;`
- altrimenti, equivale a
  `prezzo = valore * peso * 1;`

Esempio di uso di espressione condizionale:
{ `ESEMPI_ITE/espressione_condizionale.cc` }

Istruzioni strutturate

- Le istruzioni strutturate consentono di specificare azioni complesse
- Si distinguono in
  - istruzione composta (compound-statement)
  - istruzioni condizionali (conditional-statement)
  - istruzioni iterative (iteration-statement)
  - istruzioni di salto (jump-statement)

Istruzione Composta

- Trasforma una sequenza di istruzioni in una singola istruzione
  - sequenza delimitata per mezzo della coppia di delimitatori '{' e '}'
  - la sequenza così delimitata è detta blocco
- Le definizioni possono comparire in qualunque punto del blocco
  - sono visibili solo all'interno del blocco
  - possono accedere ad oggetti definiti esternamente
  - in caso di identificatori identici, prevale quello più interno

```
{
  int a=4;
  a*=6;
  char b='c';
  b+=3;
}
```

Esempio di blocchi e visibilità:

{ ESEMPI_ITE/visibilita.cc }

L'Istruzione Condizionale `if-then`

- Istruzione "**`if`**" semplice (if-then):
  - Sintassi:
    ```
    if (exp)
        istruzione1
    ```
  - Significato: se `exp` è vera, viene eseguita `istruzione1`, altrimenti non viene eseguito nulla
  - `istruzione1` può a sua volta essere un'istruzione complessa

Esempio

```
if (x!=0)
   y=1/x;
```

Esempio di if-then:

{ `ESEMPI_ITE/divisibilita.cc` }

L'Istruzione Condizionale `if-then-else`

- Istruzione "**`if`**" composta (if-then-else):
  - Sintassi: **`if`** `(exp) istruzione1` **`else`** `istruzione2`
  - Significato: se `exp` è vera, viene eseguita `istruzione1`, altrimenti viene eseguita `istruzione2`
- `istruzione1` e `istruzione2` possono essere a loro volta istruzioni complesse (un blocco, un'altro if-then-else, ...)

Esempio

```
if (x<0)
   y=-x;
else
   y=x;
```

Esempio di if-then-else:

{ `ESEMPI_ITE/divisibilita2.cc` }

If annidati

- Nei costrutti if-then e if-then-else, `istruzione1` e `istruzione2` possono essere a loro volta istruzioni complesse
  (un blocco, un'altro if-then-else, ...)
- L'annidamento di if-then-else e l'uso di operatori logici permettono di costruire strutture decisionali complesse
  - Uso di if-then-else annidati,...:
    { `ESEMPI_ITE/eq_1grado.cc` }
  - ..., con diverso ordine,...:
    { `ESEMPI_ITE/eq_1grado2.cc` }
  - ... e con operatori logici:
    { `ESEMPI_ITE/eq_1grado3.cc` }

L'indentazione del codice è importantissima!!!

- individua a colpo d'occhio l'inizio e la fine del codice/blocco
- contribuisce grandemente alla leggibilità del codice

If-then-else annidati: esempi

- Esempi di alternative all'utente:
  { `ESEMPI_ITE/conversione2.cc` }
- Esempio di "Dangling else":
  { `ESEMPI_ITE/dangling_else.cc` }
- Esempio di "Dangling else" (2):
  { `ESEMPI_ITE/dangling_else2.cc` }
- Errore tipico con "if":
  { `ESEMPI_ITE/ifeq_err.cc` }
- Versione corretta:
  { `ESEMPI_ITE/ifeq_corr.cc` }
- Minimo tra due numeri:
  { `ESEMPI_ITE/minimo.cc` }
- Minimo tra tre numeri:
  { `ESEMPI_ITE/minimo2.cc` }
- Scelte tra valori multipli:
  { `ESEMPI_ITE/simple_calc.cc` }

L'Istruzione Condizionale `switch`

- Sintassi

```
switch (exp) {
  case const-exp1: istruzione1 break;
  case const-exp2: istruzione2 break;
  ...
  default:  istruzione-default
}
```

- L'esecuzione dell'istruzione **`switch`** consiste
  - nel calcolo dell'espressione `exp`
  - nell'esecuzione dell'istruzione corrispondente all'alternativa specificata dal valore calcolato
  - se nessuna alternativa corrisponde, se esiste, viene eseguita `istruzione-`**`default`**

Scelte tra valori multipli con switch:

{ `ESEMPI_ITE/simple_calc2.cc` }

Scelte multiple con `switch`

- Se dopo l'ultima istruzione di un'alternativa non c'è un **`break`**, viene eseguita anche l'alternativa successiva
- Questo comportamento è sconsigliato ma può essere giustificato in alcuni casi

Esempio

```
switch (giorno)
{ case lun: case mar:
  case mer: case gio:
  case ven: orelavorate+=8; break;
  case sab: case dom: break;
}
```

Esercizi Proposti

Esercizio su istruzioni condizionali:
{ `ESEMPI_ITE/ESERCIZI_PROPOSTI.txt` }