Introduzione ai Linguaggi di Programmazione

Armando Tacchella

Linguaggi di Programmazione

- Un calcolatore, per eseguire compiti specifici, necessita di un programma
- Un programma è una sequenza di istruzioni (paradigma imperativo)
- Un linguaggio di programmazione definisce in maniera formale
  - la sintassi delle istruzioni, e
  - la semantica delle istruzioni

Il linguaggio Simple

- Non corrisponde a nessun linguaggio tra quelli esistenti, ma assomiglia a molti fra i più noti
- Il termine tecnico è pseudo-codice
- La costruzione di Simple avviene in modo incrementale, a partire dai costrutti fondamentali

Tipi di dato

- Quali sono le entità fondamentali in Simple?
- Numeri e testi
- Formalmente:
  - Un numero è una sequenza di cifre 0..9, eventualmente seguita dal punto decimale e da un'ulteriore sequenza di cifre.
  - Un testo è una sequenza di caratteri alfanumerici, ossia cifre, lettere e segni di interpunzione, delimitata da singoli apici.

Esempi

- Possibili dati numerici in Simple

```
96
0.345
.32     (equivale a 0.32)
10.     (equivale a 10.0)
```

- Possibili testi in Simple

```
‘Ciao!‘
‘RTtt89()%èmmF!‘
```

Istruzione di Visualizzazione

- L’istruzione SCRIVI consente di visualizzare su schermo un’espressione
- Sintassi

```
SCRIVI <espressione>
```

- Esempi

```
SCRIVI 96
SCRIVI ‘Ciao!‘
```

Espressioni in Simple

- Espressioni numeriche
  - un numero è un'espressione
  - un operatore unitario (+,-) davanti ad una espressione forma una nuova espressione
  - un'operatore binario (+,-,*,/) nel mezzo di due espressioni forma una nuova espressione
- Espressioni testuali
  - un testo è un'espressione
  - un operatore binario (+) nel mezzo di due testi forma una nuova espressione

Primi programmi in Simple - 1

- Il programma

```
SCRIVI 10+5
```

  visualizza `15` sullo schermo
- Il programma

```
SCRIVI '(10.2/2)+1='
SCRIVI 10.2/2+1
```

  visualizza `(10.2/2)+1=6.1` sullo schermo

Primi programmi in Simple - 2

- Il programma

  ```
  SCRIVI ‘Ciao a tutti! ‘
  ```

  e il programma

  ```
  SCRIVI ‘Ciao ‘+ ‘a ‘ + ‘tutti! ‘
  ```

  visualizzano `Ciao a tutti!` sullo schermo
- Cosa visualizza il programma seguente?

  ```
  SCRIVI ‘7‘+‘8‘
  ```

Potere Espressivo - 1

- Supponiamo di voler scrivere un programma per calcolare l’area di un generico rettangolo con le istruzioni sinora introdotte.
- Siamo in grado di calcolare un’area per uno specifico rettangolo, per esempio:

  ```
  SCRIVI 6*5
  ```

  calcola l’area di un rettangolo con base pari a 6 e altezza pari a 5.
- Non possiamo, a meno di modificare il programma, calcolare l’area di un rettangolo diverso!

Potere Espressivo - 2

- Il potere espressivo di un linguaggio misura la sua capacità di risolvere problematiche applicative
- Al crescere del potere espressivo cresce la complessità del linguaggio
- Dobbiamo introdurre nuovi costrutti in Simple per poter risolvere il "problema dell'area di un rettangolo generico"

Variabili

- Obiettivo: scrivere programmi che funzionino per numeri o testi diversi da quelli previsti in fase di scrittura.
- Sono necessari "contenitori" per numeri e testi: le variabili.
- Una variabile
  - ha un tipo associato (numerico o testo)
  - ha un nome simbolico (x, y, base, ecc.)
  - rappresenta un dato nella memoria del calcolatore
- Le variabili possono essere utilizzate nelle espressioni al posto di numeri e testi.

Variabili in Simple - 1

- Serve un'istruzione per stabilire il nome e il tipo delle variabili utilizzate
- Istruzione di dichiarazione.
- Sintassi in Simple:

```
<tipo della variabile> <nome della variabile>
```

- Esempi di dichiarazione:

```
NUMERO x
TESTO s
```

Variabili in Simple - 2

- Serve un modo per assegnare un valore ad una variabile
- Istruzione di assegnamento
- Sintassi in Simple:

```
<nome della variabile> = <espressione>
```

- Esempi:

```
x = 5 * 6
z = 7 + x
```

Calcolo di aree - 1

- Consideriamo il seguente programma

```
NUMERO base
NUMERO altezza
altezza = 5
base = 6
SCRIVI base * altezza
```

- Abbiamo utilizzato le variabili, ma non abbiamo ancora risolto il problema di partenza! Perché? Cosa manca?

Istruzione di Richiesta Dati

- Serve un'istruzione per introdurre i valori di base e altezza durante l'esecuzione del programma!
- Istruzione di richiesta dati
- Sintassi in Simple:

```
CHIEDI <variabile>
```

- Esempio:

```
CHIEDI base
```

Calcolo di aree - 2

- Adesso abbiamo risolto il problema:

```
NUMERO base, altezza
CHIEDI base
CHIEDI altezza
SCRIVI base * altezza
```

- Abbiamo scritto il nostro primo programma "interattivo" in Simple!
- Notare la forma abbreviata per la dichiarazione delle variabili

Potere Espressivo - 3

- Supponiamo di voler scrivere un programma per calcolare le soluzioni dell'equazione di secondo grado

$$ax^2 + bx + c = 0$$

- Dobbiamo calcolare il discriminante e decidere il tipo di soluzioni (coincidenti, reali, complesse coniugate).

Equazioni di Secondo Grado - 1

- Il programma in Simple inizia così

```
NUMERO a, b, c
CHIEDI a
CHIEDI b
CHIEDI c
NUMERO delta
delta = b*b – 4 * a * c
```

- Come facciamo a scegliere in base ai valori di `delta` cosa fare nel seguito del programma?

Scelta Condizionata

- Serve un'istruzione per scegliere sulla base del risultato di un test vero/falso
- Istruzione di scelta condizionata
- Sintassi in Simple

```
SE <test> ALLORA
   <istruzioni nel caso test sia vero>
ALTRIMENTI
   <istruzioni nel caso test sia falso>
```

- La parte `ALTRIMENTI` può essere omessa

Equazioni di Secondo Grado - 2

- Ora possiamo continuare il programma in questo modo:

```
SE delta >= 0 ALLORA
   SCRIVI 'Soluzioni reali'
ALTRIMENTI
   SCRIVI 'Soluzioni immaginarie'
```

- Si noti la forma del test: si tratta di un'espressione logica costruita con l'operatore relazionale ">=" (maggiore o uguale)

Espressioni logiche - 1

- Abbiamo di fatto introdotto un nuovo tipo di dato (oltre a `NUMERO` e `TESTO`)
- Il tipo `LOGICO`: vero o falso (1 o 0)
- Un'espressione logica è costruita utilizzando
  - operatori relazionali
    - uguale (==), diverso (`!=`), minore (<), minore o uguale (<=), maggiore (>), maggiore o uguale (>=)
  - operatori logici
    - congiunzione (`E`), disgiunzione (`O`)

  a partire da espressioni numeriche o testuali

Espressioni logiche - 2

- Formalmente
  - `<espressione> <op.rel.> <espressione>` è un'espressione logica
  - `<espr.logica> E <espr.logica>` è un'espressione logica
  - `<espr.logica> O <espr.logica>` è un'espressione logica
- Esempi

```
delta > 0
delta > 0 E a != 0
```

Equazioni di secondo grado - 3

- La formula risolutiva basata sul calcolo di `delta` prevede un'equazione non degenere, ossia i valori dei coefficienti a, b e c sono tutti diversi da 0
- Il programma per risolvere le equazioni deve gestire anche i casi "degeneri"
- Questa tipologia di errore è una delle più frequenti cause di malfunzionamento

Equazioni di secondo grado - 4

- Supponiamo di aver già chiesto a, b e c

```
SE a==0 ALLORA
    ...
ALTRIMENTI
    SE b==0 ALLORA
          ...
    ALTRIMENTI
          SE c==0 ALLORA
                ...
          ALTRIMENTI
                delta = b*b – 4*a*c
                ...
```

- Cosa è necessario al posto dei puntini?

Potere espressivo - 4

- Supponiamo di voler scrivere un programma che chieda all'utente
  - Un numero arbitrario N maggiore di 0
  - N valori di cui calcolare la media aritmetica
- Serve un'istruzione che ci consenta di ripetere N volte la richiesta di dati
- Il problema è che N non è noto a priori!

Iterazione Condizionale

- Introduciamo un costrutto che consente di ripetere un certo insieme di istruzioni fintantoché una certo test è soddisfatto
- Sintassi in Simple

```
SE <test> RIPETI
   <istruzione>
   ...
   <istruzione>
```

Calcolo di Media Aritmetica - 1

- Il programma in Simple per il calcolo della media di N valori

```
NUMERO N, i, x, somma
CHIEDI N
somma = 0
i = N
SE i>0 RIPETI
   CHIEDI x
   somma = somma + x
   i = i - 1
SCRIVI somma / N
```

Iterazione semplice

- Si consideri l'iterazione condizionale

```
i = N
SE i>0 RIPETI
    ...
    i = i – 1
```

- Il significato è ripeti le istruzioni "..." N volte
- Introduciamo un'abbreviazione

```
PER <var.> DA <espr.num.> A <espr.num.> RIPETI
    <istruzione>
    ...
    <istruzione>
```

Calcolo di Media Aritmetica - 2

- Sfruttando l'iterazione semplice si ottiene un programma più compatto

```
NUMERO N, x, somma
CHIEDI N
somma = 0
PER i DA 0 A N RIPETI
   CHIEDI x
   somma = somma + x
SCRIVI somma / N
```

- L'iterazione semplice può essere ricondotta all'iterazione condizionale ma non viceversa!

Potere Espressivo - 5

- Supponiamo di voler scrivere un programma che chieda una sequenza di N numeri e la restituisca ordinata in senso ascendente
- Bisogna prima memorizzare la sequenza e poi ordinarla
- Simple non consente di memorizzare sequenze!

Vettori - 1

- Un vettore si definisce come una sequenza di dati di tipo omogeneo
- Dichiarazione di vettore in Simple

```
VETTORE <tipo>[<dim>] <nome>
```

- Esempi

```
VETTORE INTERO[10] vettoreDiInteri
VETTORE TESTO[15] vettoreDiTesti
```

Vettori - 2

- Come si assegnano valori ai vettori?
- Utilizzando l'operatore di indicizzazione
- Sintassi in Simple

```
<vettore>[<posizione>]
```

- Per esempio

```
vettoreDiInteri[2] = 10
```

assegna il valore `10` alla posizione `2` del vettore `vettoreDiInteri`

Memorizzare una Sequenza

- Questo programma legge N valori e li memorizza in un vettore

```
CHIEDI N
VETTORE INTERO[N] v
PER i DA 1 A N RIPETI
   CHIEDI v[i]
```

- Ogni posizione (cella) del vettore è accessibile indipendentemente dalle altre!

Ordinare una Sequenza

- Supponiamo che v contenga una sequenza di N numeri

```
PER i DA 1 A N-1 RIPETI
   PER j DA i+1 A N RIPETI
        SE v[i] > v[j] ALLORA
             Scambia( v[i], v[j] )
```

- `Scambia` sostituisce al valore di `v[i]` il valore di `v[j]` e viceversa.

Potere Espressivo - 6

- Supponiamo di voler scrivere un programma per memorizzare i dati di un certo gruppo di persone
- Ogni persona è contraddistinta da un insieme eterogeneo di dati (nome, età, data di nascita)
- Possiamo risolvere il problema in Simple?

Gestione Elenco Anagrafico - 1

- Una soluzione basata sui vettori per memorizzare i dati di `N` persone

```
VETTORE TESTO[N] nomi
VETTORE TESTO[N] cognomi
VETTORE NUMERO[N] età
...
```

- Per accedere ai dati di una persona è necessario consultare tanti vettori quanti sono i dati anagrafici

Dati Strutturati

- Un costrutto per raggruppare dati eterogenei
- Sintassi in Simple

```
STRUTTURA <nome> DI
   <tipo> <nome del campo>
   ...
   <tipo> <nome del campo>
```

- Definisce un nuovo tipo chiamato `<nome>`

Dati Strutturati - 2

- Definizione di un tipo di dato strutturato

```
STRUTTURA Persona DI
    TESTO nome
    TESTO cognome
    NUMERO anni
    ...
```

- Dichiarazione e utilizzo di una variabile di tipo Persona

```
Persona p
p.nome = ‘Ernesto‘
```

Gestione Elenco Anagrafico - 2

- Programma per inserire i dati anagrafici relativi ad N persone

```
CHIEDI N
VETTORE Persona[N] elenco
PER i DA 1 A N RIPETI
   CHIEDI elenco[i].nome
   CHIEDI elenco[i].cognome
   CHIEDI elenco[i].anni
   ...
```

Procedure e Funzioni

- Nel programma per ordinare un vettore di numeri abbiamo utilizzato il concetto di procedura
- La procedura `Scambia(x, y)` scambiava il valore di `x` con quello di `y` e viceversa
- `Scambia` può essere scritta in Simple

```
NUMERO t
t = x
x = y
y = t
```

Definizione di Procedure - 1

- Sintassi in Simple

```
PROCEDURA <nome> ( <parametri> )
   <istruzione>
   ...
   <istruzione>
```

- Consente di riutilizzare più volte lo stesso codice senza doverlo riscrivere

Definizione di Procedure - 2

- La procedura Scambia è definita come

```
PROCEDURA Scambia( NUMERO x,y )
   NUMERO t
   t = x
   x = y
   y = t
```

- `x` e `y` sono i parametri della procedura, entrambe di tipo `NUMERO`

Definizione di Funzioni - 1

- Una funzione è una procedura che restituisce un risultato
- Sintassi in Simple

```
FUNZIONE <nome> ( <parametri> )
   <istruzione>
   ...
   <istruzione>
   RESTITUISCI <espressione>
```

Definizione di Funzioni - 2

- Esempio: funzione per il calcolo del massimo tra due valori

```
FUNZIONE Max( NUMERO x, y )
   NUMERO t
   SE x > y ALLORA
        t = x
   ALTRIMENTI
        t = y
   RESTITUISCI t
```

Utilizzo delle Funzioni

- Programma in Simple per calcolare il massimo fra tre valori che utilizza la funzione `Max`

```
NUMERO x, y, z
CHIEDI x, y, z
SE Max(x, y)==x ALLORA
    SE Max(x, z)==x ALLORA
          SCRIVI 'Il massimo è x'
    ALTRIMENTI
          SCRIVI 'Il massimo è z'
ALTRIMENTI
    SE Max(y, z)==y ALLORA
          SCRIVI 'Il massimo è y'
    ALTRIMENTI
          SCRIVI 'Il massimo è z'
```

Una nuova prospettiva - 1

- Il linguaggio Simple ci consente di
  - Inserire/visualizzare dati
  - Eseguire calcoli
  - Manipolare testi
  - Scegliere fra istruzioni alternative
  - Ripetere istruzioni
  - Gestire sequenze di dati omogenei
  - Gestire dati strutturati
  - Suddividere il programma in procedure e funzioni

Una nuova prospettiva - 2

- Combinando la possibilità di gestire dati strutturati con la possibilità di definire procedure e funzioni otteniamo un concetto nuovo: gli oggetti
- Un oggetto è una struttura che incorpora dati (attributi) e procedure (metodi) che agiscono sui dati
- Solo i metodi dell'oggetto possono modificare il valore degli attributi (incapsulazione)

Programmazione ad oggetti

- Nella programmazione ad oggetti il programma consta di due parti:
  - definizione delle classi di oggetti
  - utilizzo degli oggetti (istanze delle classi)
- Ad esempio, la struttura Persona definita in precedenza diventa un oggetto se associata con le procedure che consentono di intervenire sui dati in essa contenuta