LABORATORIO DI MATEMATICA

IL LINGUAGGIO DI PROGRAMMAZIONE DI DERIVE

ESERCITAZIONE GUIDATA

Dalla versione 5 troviamo in Derive la possibilità di usare un linguaggio di programmazione.
La struttura di un programma scritto nel linguaggio di Derive è la seguente:

- **il nome, stabilito dal programmatore;**
- **un elenco di variabili poste fra parentesi tonde, che servono per assumere i valori dei dati d'ingresso;**
- **il simbolo di assegnazione : = (due punti uguale);**
- **la parola chiave PROG seguita da una sequenza di istruzioni, separate da virgole, posta fra parentesi tonde;**
- **la parola chiave RETURN seguita da un'espressione (l'uscita del programma), che chiude l'elaborazione del programma.**

Le istruzioni possono essere di assegnazione, di selezione o di iterazione.

Un programma con assegnazioni

L'istruzione di **assegnazione** consiste in una variabile seguita dal simbolo : = e da un'espressione che può contenere qualunque operatore noto a Derive.

Scriviamo un programma che calcoli e mostri il perimetro e l'area di un rettangolo, dopo aver ricevuto le misure della base e dell'altezza:
Rett_1(b, h) : = PROG (duep : = 2 · (b + h), S : = b · h, RETURN [duep, S]).

Per **comunicare il programma** a Derive, lo scriviamo nella riga di editazione delle espressioni e lo immettiamo con INVIO nella zona algebrica. Derive lo scrive in un'etichetta in modo «indentato», usa una riga per ogni istruzione e non mostra le virgole, che separano le singole istruzioni, e le parentesi tonde, che contengono il corpo del programma, come vediamo in figura 1.

```
      Rett_1(b, h) :=
        Prog
#1:       duep := 2·(b + h)
          S := b·h
          RETURN [duep, S]
```

◀ **Figura 1** Il programma.

```
#2:   Rett_1(4, 3)

#3:   [14, 12]
```

◀ **Figura 2**

Per **eseguire il programma**, scriviamo nella riga di editazione il suo nome seguito fra parentesi dai dati (qui diamo 4 per la base e 3 per l'altezza), corrispondenti in numero e ordine a quelli stabiliti per i parametri d'ingresso, e con INVIO lo immettiamo nella #2. Su di essa applichiamo uno dei comandi di *Semplifica* (o *Base*, o *Sviluppa*, o *Approssima*). Derive elabora i dati e mostra i risultati nella #3 (figura 2).

Un programma con selezioni

Scriviamo un programma che, lette le coordinate di due punti, determini l'equazione dell'eventuale retta passante per essi. Lo proviamo poi con le seguenti coppie di punti: (3; 2) e (− 1; 4); (3; 2) e (3; 4); (3; 2) e (3; 2).

L'analisi

Per determinare l'equazione della retta passante per i punti $(x_1; y_1)$ e $(x_2; y_2)$ usiamo la formula:

$$y = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + y_1.$$

Cerchiamo le eccezioni alla formula, delle quali dobbiamo tener conto nella costruzione dell'algoritmo risolutivo del problema. Notiamo che, se $x_1 = x_2$, la formula non è valida e distinguiamo due casi: se $x_1 = x_2$ e $y_1 = y_2$, i due punti coincidono, quindi la retta non è determinata univocamente; se $x_1 = x_2$ e $y_1 \neq y_2$, la retta risulta parallela all'asse y e ha equazione $x = x_1$.

L'algoritmo risolvente

Per **algoritmo** intendiamo una sequenza di istruzioni che risolve un problema. Esprimiamo l'algoritmo in un linguaggio libero rivolto a un generico esecutore, che chiamiamo **linguaggio di progetto**.

Inizio,
Leggi x_1, y_1, x_2, y_2,
Se $x_1 = x_2$,
  allora Se $y_1 = y_2$,
    allora Scrivi I due punti coincidono, Fine.
    altrimenti Scrivi x = x1, Fine.
  altrimenti Scrivi y = (y2 − y1)/(x2 − x1) (x − x1) + y1, Fine.

La traduzione dell'algoritmo nel linguaggio di Derive

Per costruire un'istruzione di **selezione** usiamo la funzione IF. Essa ha la sintassi IF(*cond*, *istr*1, *istr*2): se *cond* è vera, allora il programma svolge *istr*1, altrimenti *istr*2.
Prima di battere il programma scegliamo *Parola* nel campo *Nome delle variabili* posto all'interno di *Opzioni_Modalità Input*, in modo che le variabili possano essere indicate con più di un carattere.
• Nella riga di editazione scriviamo: Retta_2p(x1, y1, x2, y2) := PROG(IF(x1 = x2, IF(y1 = y2, RETURN "I due punti coincidono", RETURN x = x1), RETURN y = (y2 − y1)/(x2 − x1) · (x − x1) + y1)).
• Con INVIO immettiamo il listato del programma nell'etichetta #1 (figura 3).

```
      Retta_2p(x1, y1, x2, y2) :=
        Prog
          If x1 = x2
#1:          If y1 = y2
                RETURN "I due punti coincidono"
                RETURN x = x1
             RETURN y = (y2 - y1)/(x2 - x1)·(x - x1) + y1
```

◀ **Figura 3** Il programma che determina l'equazione di una retta passante per due punti.

Le esecuzioni del programma

• Scriviamo l'espressione Retta_2p(3, 2, −1, 4) e la immettiamo nella #2 (figura 4).
• Su di essa applichiamo il comando *Semplifica_Sviluppa*, che causa l'esecuzione del programma con lo sviluppo del risultato. Osserviamo infatti nella #3 l'equazione sviluppata della retta.
• Operiamo in modo simile per gli altri due casi, ottenendo la #5 e la #7.

```
#2:  Retta_2p(3, 2, -1, 4)

#3:  y = 7/2 - x/2

#4:  Retta_2p(3, 2, 3, 4)

#5:  x = 3

#6:  Retta_2p(3, 2, 3, 2)

#7:  I due punti coincidono
```

► **Figura 4** Le esecuzioni del programma corrispondenti ai tre casi proposti.

Un programma con un'iterazione

Scriviamo un programma che stabilisca se un numero n, intero e positivo, è o non è primo. Proviamo il programma con 8, 17, 8,2.

L'analisi

Per stabilire se un numero n sia primo o meno, calcoliamo con un ciclo iterativo i resti della divisione fra n e i numeri 2, 3 e così via. Se raggiungiamo la radice quadrata r di n o il numero intero immediatamente inferiore senza trovare alcun divisore di n, segnaliamo che n è primo. Infatti, se n ammettesse un divisore d_1 maggiore di r, allora esisterebbe un altro divisore d_2 di n, tale che $d_1 \cdot d_2 = n = r \cdot r$, con d_2 minore di r.

L'algoritmo risolvente

Inizio,
Leggi n,
Controlla che n sia intero e maggiore di 2,
Assegna la parte intera della radice quadrata di n a r,
Assegna 1 a i,
  Ripeti
  (Incrementa i di 1,
  Se (Resto fra n e i) $= 0$,
    allora Scrivi n non è primo, Fine.
  Se $i = r$ allora Esci),
  Scrivi n è primo, Fine.

◀ La parola Esci causa l'uscita dell'algoritmo del ciclo iterativo iniziato con la parola Ripeti e il proseguimento del percorso dell'algoritmo. Le parole Ripeti ed Esci corrispondono alle parole LOOP ed EXIT di Derive.

La traduzione dell'algoritmo nel linguaggio di Derive

Per effettuare un'**iterazione** scriviamo la parola LOOP e il ciclo che il programma deve ripetere, finché non trova la parola EXIT, posto fra parentesi tonde.

• Scriviamo l'espressione Primo_?(n) := PROG(IF(n ≠ FLOOR(n) OR n ≤ 2, RETURN "Il numero deve essere intero e maggiore di 2"), r := FLOOR(SQRT(n)), i := 1, LOOP (i := i + 1, IF(MOD(n, i) = 0, RETURN [n; "non è primo"]), IF (i = r, EXIT)), RETURN [n; "è primo"]) e la immettiamo nella #1 (figura 5).

Le esecuzioni del programma

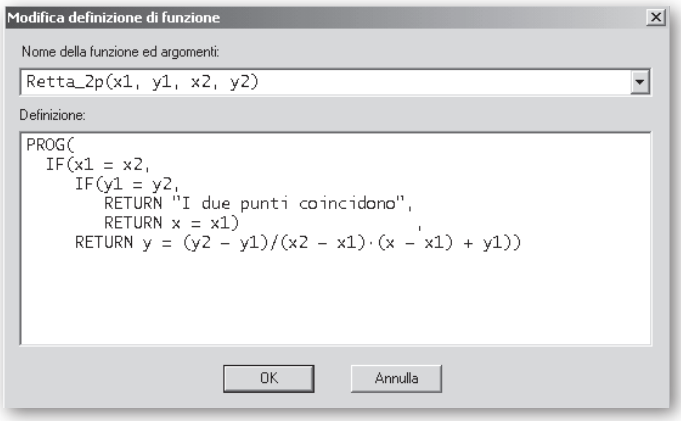
• Immettiamo Primo_?(8) nella #2.
• Su di essa applichiamo *Semplifica_Base*, ottenendo la risposta nella #3.
• Operiamo in modo simile per gli altri due numeri, ricavando la #5 e la #7.

```
      Primo_?(n) :=
        Prog
          If n ≠ FLOOR(n) ∨ n ≤ 2
            RETURN " n deve essere intero e maggiore di 2"
          r := FLOOR(√n)
          i := 1
#1:       Loop
            i := i + 1
            If MOD(n, i) = 0
              RETURN [n; "non è primo"]
            If i = r exit
          RETURN [n; "è primo"]

#2:   Primo_?(8)

#3:   [ 8 ; non è primo ]

#4:   Primo_?(17)

#5:   [ 17 ; è primo ]

#6:   Primo_?(8.2)

#7:    n deve essere intero e maggiore di 2
```

▲ **Figura 5** Il programma che indica se un numero è primo e alcune sue esecuzioni.

L'editor delle espressioni

Derive, nella versione 6, mette a disposizione un editing multilinea particolarmente utile per variare la struttura di un programma. Per applicarlo evidenziamo l'etichetta che contiene il listato del programma e usiamo il comando *Modifica_Espressione* o facciamo clic con il tasto destro del mouse. Vediamo comparire una finestra come quella di figura 6, nella quale possiamo operare le modifiche nel modo usuale (portando il cursore nella posizione desiderata con un clic del mouse o con i tasti freccia e battendo da tastiera le variazioni desiderate). Per andare a capo battiamo contemporaneamente i tasti CTRL e INVIO. Con il tasto INVIO, invece, concludiamo le variazioni rendendo noto a Derive il programma nella nuova versione.



▲ **Figura 6** La finestra per l'editor dei programmi.