Il SIMULA

Il linguaggio per la simulazione di sistemi discreti.

Generalità dei linguaggi per la simulazione

- Un linguaggio di simulazione deve possedere delle proprietà sia algoritmiche che sistematiche.
- Le proprietà computazionali dei tradizionali linguaggi algebrici
- Gli strumenti necessari ad esprimere in modo sistematico i componenti essenziali di un sistema discreto (utenti, canali di servizio, code, discipline e relative interazioni)

Generalità dei linguaggi per la simulazione

- I linguaggi generali come il Fortran o il C sono dotati di proprietà algoritmiche, ma mancano di quelle sistemistiche. Per esprimere un modello in questi linguaggi bisogna costruire sottoprogrammi che simulano i componenti del sistema, le loro interazioni ed i tempi in cui queste interazioni avvengono.
- Altri linguaggi come il linguaggio GPSS (IBM) pur essendo dotati di notevoli proprietà sistemistiche, mancano di alcuni strumenti computazionali e quindi non posseggono quella flessibilità necessaria per costruire un modello di simulazione.

Generalità dei linguaggi per la simulazione

L'uso di linguaggi completamente orientati alla simulazione presenta però un basso tasso di diffusione aziendale:

- necessità di rivolgersi ad esperti
- limitazione delle possibilità di controllo diretto del lavoro
- riduzione della possibilità di sviluppo del modello attraverso modifiche successive

Caratteristiche principale di un linguaggio di simulazione

- diversi tipi di variabili records, matrici, liste ecc;
- *flessibilità* nel creare e manipolare le strutture dati;
- *efficiente routine* di temporizzazione
- *elasticità* nella definizione di procedure e sottoprogrammi che possono essere definiti e richiamati in modo semplice consentendo di arricchire il linguaggio con nuove
- *semplice creazione* di variabili statistiche di forma standard
- *funzionalità per la facilitazione della sperimentazione* del modello di simulazione: per iterazioni multiple, per cambiare i valori dei parametri e dai dati, per analizzare i risultati, ecc.

Il SIMULA 67

Il linguaggio Simula, è stato progettato da *Nygaard* e *Dahl* presso il *Norwegian Computing Center*.

Fu costruito ed implementato originariamente come linguaggio per la *simulazione discreta*, ma poi successivamente re-implementato come un linguaggio di *programmazione generale*.

Sebbene il Simula non abbia avuto una larga diffusione, la sua influenza sulla moderna metodologia di programmazione (OOP) è stata molto elevata.

La sua importanza principale è stata l'introduzione del concetto di classe.

Il SIMULA

- Il Simula I (**Simu**lation **La**nguage) è basato sull'ALGOL di cui riprende le strutture logiche ed a cui aggiunge strumenti che permettono di esprimere in modo immediato gli aspetti sistemistici del modello discreti.
- Il Simula 67 (**Sim**ple **u**niversal **la**nguage) è un linguaggio generale orientato agli oggetti con sistemi di classi che supportano le "linked list" (il sistema Simset) e i processi discreti orientati alla simulazione (il sistema Simulation)
- Il Simula utilizzato è il Simula 67 in quanto il Simula I non è stato più utilizzato.

La programmazione in Simula

- La programmazione in Simula è leggermente diversa da quella in altri linguaggi quali il Fortran o il Pascal.
- Il Simula infatti prevede che il programma sia organizzato o strutturato in modo preciso e permette la descrizione di azioni che accadono simultaneamente o in parallelo.
- Un programma in Simula è costituito da una sequenza di blocchi (un insieme o più di istruzioni) che agiscono indipendentemente, ma che vengono poi combinate insieme per ottenere l'effetto finale desiderato.
- L'approccio per la costruzione di un programma è quello top-down

La programmazione in Simula

- Un programma in Simula è costituito da una serie di procedure che rappresentano i processi che compongono il sistema.
- Ad ogni processo è associato un insieme di dati locali che ne stabiliscono le caratteristiche ed un pattern di comportamento che ne descrive la storia nel tempo; in questa storia sono via via elencate le trasformazioni che subiscono i parametri caratterizzanti il processo, le relazioni con gli altri processi. La storia termina, quando necessario, con l'eliminazione del processo stesso.

La programmazione in Simula

- I processi sono raggruppati in classi dette attività , caratterizzate dall'uguaglianza delle regole operative. La manipolazione dei processi è effettuata attraverso delle liste ausiliarie dette elementi , che contengono i nomi dei processi stessi.
- Anche gli elementi possono essere raggruppati durante la simulazione fissandone l'appartenenza a sets.
- Il controllo della sequenza degli oggetti viene effettuato mediante una routine automatica, (sequencing set) che però può essere modificata attraverso una serie di istruzioni.

La programmazione in Simula

Esistono procedure particolari per:

- attivare e terminare i processi,
- creare variabili casuali e/o aventi determinate distribuzioni di probabilità,
- creare oggetti di tipo lista, coda, ecc. con o senza relazioni di priorità

Tipi di variabili in Simula

- **Integer** (numero intero Es: 0,1,2,) (Valore di Default 0)
- **Real** (numero reale Es: 0.1,0.2) (Valore di Default 0.0)
- **Boolean** (valore booleano Es: true, false) (Valore di Default false)
- **Character** (variabile simbolica Es: a,A,b,B) (Valore di Default iso null (invisibile))
- **text (**array di simboli Es: ciccio, pippo) (Valore di Default no text (i.e. ""))

Dichiarazione delle variabili

- La sintassi per la dichiarazione di una variabile è il seguente:

TipoVariabile Identificatore1, Identificatore2,.....IdentificatoreN, ...

Esempi

- ***integer*** NumeroStudenti; !dichiariamo la variabile NumeroStudenti di tipo intero;
- ***text*** NomeStudente; !dichiariamo la variabile NomeStudente di tipo array di simboli;

Il ; e i commenti

- Ogni istruzione deve terminare con il simbolo ; e che eventuali commenti possono essere inseriti dopo il simbolo ! terminandoli come al solito con ;.
- Occorre infine ricordare di moderare la lunghezza del nome di un identificatore e che questa non deve mai iniziare con un carattere numerico.

Istruzioni di assegnazione

- Un discorso a parte deve essere invece fatto per le variabili di tipo puntatore

Esempio

- **ref(**TipoVariabile**)** P,Q; !P e Q sono variabili puntatore al tipo complesso TipoVariabile;
- **P :- Q;** !P e Q condividono la stessa locazione di memoria;

Istruzioni di I/O

	Input	Output
variabile intera	**InInt**	**OutInt**(NomeVariabile,m)
variabilereale	**InReal**	**OutReal**(NomeVariabile,n,m)
variabile simbolica	**InChar**	**OutChar**(NomeVariabile)
variabile array di simboli	**InText***(s)*	**OutText**(puntatore)

m dimensione del campo da usare quando il valore della variabile verrà visualizzato, **n** è il numero di cifre dopo la virgola che devono essere visualizzate, ed **s** è la lunghezza massima di caratteri che può contenere l'array di simboli.

Operatori aritmetici

I principali operatori aritmetici:

- + addizione
- - sottrazione
- * moltiplicazione
- ** elevamento a potenza
- / divisione tra reali
- // divisione tra interi

Connettivi

- per tipi semplici = > < >= <=
- per puntatori = =/=
- per tipi booleani **eqv, not** and **or**
- concatenazione di array di simboli **&**