Fondamenti di Informatica

per la Sicurezza

a.a. 2007/08

Elementi di programmazione

Stefano Ferrari

UNIVERSITÀ DEGLI STUDI DI MILANO
DIPARTIMENTO DI TECNOLOGIE DELL'INFORMAZIONE

Definizione

virtuale [vir-tu-à-le] *agg.*

1. Che esiste in potenza, ma non si è ancora realizzato;

2. Fittizio, non reale.

Macchina virtuale

Si usa il termine macchina virtuale per indicare l'astrazione della macchina fisica realizzata dal software ai vari livelli.

Esempio:

- per le periferiche mappate in memoria si usano delle operazioni che non hanno nulla a che vedere con la realtà fisica, ma facilitano l'uso dei dispositivi.

Virtualizzazione

A livello di utente:

- sistema operativo;
- interfaccia:
  - tira l'angolino;
  - schiaccia il pulsante.

A livello di programmatore:

- paradigmi di programmazione;
- linguaggi.

Risolvere un problema

Dato un problema, l'individuazione delle procedure necessarie alla sua soluzione richiede le seguenti fasi:

- descrizione dei requisiti che la soluzione dovrà soddisfare per essere considerata corretta (specifiche);
- procedimento con cui si individua o si inventa una soluzione (progetto);
- tecnica che consente di risolvere il problema (soluzione).

Nell'informatica, la soluzione è espressa tramite un algoritmo.

Algoritmo

Dato un problema e un esecutore, l'algoritmo:

- è una successione finita di passi elementari (direttive);
- eseguibile senza ambiguità dall'esecutore;
- risolve il problema dato.

Caratteristiche di un algoritmo

Ogni algoritmo possiede le seguenti caratteristiche:

- sequenzialità:
  - viene eseguito un passo dopo l'altro secondo un ordine specificato (flusso di esecuzione);
- univocità:
  - i passi elementari devono poter essere eseguiti in modo univoco dall'esecutore, e quindi devono essere descritti in una forma eseguibile per l'esecutore.

Algoritmo e programmazione

Un buon algoritmo deve prevedere tutti i casi significativi che derivano dall'esecuzione di ogni passo elementare.

La descrizione di un algoritmo per un esecutore automatico deve avere una formulazione generale: la soluzione individuata non deve dipendere solo da valori predefiniti dei dati.

Esempi

Esempio negativo:

- programma che calcola l'area del cerchio di raggio 3 cm.

Esempio positivo:

- programma che calcola l'area del cerchio di raggio dato dall'utente.

Elementi di un algoritmo

"Oggetti": le entità su cui opera l'algoritmo vengono generalmente chiamate dati, e comprendono sia i dati iniziali del problema che i risultati intermedi.

Operazioni: gli interventi che si possono effettuare sugli oggetti, cioè sui dati sono calcoli, confronti, assegnamenti ed operazioni di I/O.

Flusso di controllo: indica le possibili evoluzioni dell'esecuzione delle operazioni, cioè le possibili successioni dei passi dell'algoritmo.

Esempi di algoritmi

Algoritmi adatti ad un esecutore umano:

- ricette di cucina;
- spartiti musicali;
- istruzioni di montaggio di un mobile.

Generalmente contengono riferimenti a:

- quantità soggettive;
- descrizioni approssimative;
- informazioni mancanti perché implicite.

Descrizione di un algoritmo

Un algoritmo adatto per un esecutore automatico deve essere descritto tramite un formalismo (linguaggio) rigoroso e non ambiguo costituito da:

- **vocabolario**: insieme di elementi per la descrizione di oggetti, operazioni e flusso di controllo;
- **sintassi**: insieme di regole per la composizione degli elementi del linguaggio in frasi eseguibili e costrutti di controllo;
- **semantica**: insieme di regole per l'interpretazione degli elementi e delle istruzioni sintatticamente corrette.

Linguaggio di programmazione

In un linguaggio di programmazione:

- gli oggetti vengono descritti tramite nomi simbolici (detti anche identificatori);
- le operazioni vengono descritte tramite operatori, funzioni e procedure;
- il flusso di controllo viene descritto tramite opportuni costrutti di controllo.

Flusso di controllo vs. flusso di esecuzione

Il flusso di controllo è differente dal flusso di esecuzione.

Il flusso di controllo descrive tutte le possibili successioni di operazioni che possono essere realizzate dal programma nella sua esecuzione.

Il flusso di esecuzione è la sequenza di operazioni percorsa durante una particolare esecuzione del programma.

Linguaggi di programmazione (1)

I linguaggi di programmazione sono classificabili in almeno quattro grandi categorie, parzialmente sovrapposte:

Imperativi: le istruzioni descrivono in modo esplicito le modifiche del contenuto della memoria;
- esempi: Basic, Pascal, C, Fortran, Cobol;

Funzionali: il programma è costituito da una funzione che ha per argomenti altre sottofunzioni;
- esempio: Lisp;

Linguaggi di programmazione (2)

Dichiarativi (o logici): il programma è costituito da una serie di asserzioni e di regole, e la sua esecuzione consiste in una dimostrazione di veridicità di un'asserzione, senza indicare il flusso di esecuzione;
- esempio: Prolog;

Orientati agli oggetti: il programma è costituito da entità (oggetti) che comunicano tra loro scambiandosi messaggi e che godono delle proprietà di incapsulamento, ereditarietà e polimorfismo;
- esempi: Ada, Smalltalk, Java.

Linguaggi di descrizione

I linguaggi semiformali hanno regole ben definite, ma permettono descrizioni in linguaggio naturale, non rigorose.

Sono usati nella fase di sviluppo di algoritmi:

schema a blocchi (diagramma di flusso): usa blocchi di varie forme geometriche connessi da archi orientati la cui direzione indica il flusso di controllo, mentre la forma dei blocchi indica la natura del blocco stesso (operazione, confronto, operazione di I/O);

pseudocodice: usa strutture di controllo di un linguaggio di programmazione e operazioni descritte in linguaggio naturale.

Sottoprogrammi

Programmi complessi vengono progettati seguendo uno schema gerarchico in cui il problema iniziale viene suddiviso in sottoproblemi e così via (metodologia top-down).

Questo approccio ha i seguenti vantaggi:

- chiarezza del programma principale;
- sintesi;
- efficienza.

Vantaggi

Chiarezza: i dettagli di basso livello sono descritti a parte nei sottoprogrammi, evidenziando la struttura di controllo generale;

Sintesi: per i sottoproblemi presenti in più parti del problema principale, richiamando il sottoprogramma, si evitano di ripetere le stesse sequenze di operazioni;

Efficienza: sottoproblemi di uso comune sono disponibili, già risolti da esperti programmatori, raccolti nelle cosiddette librerie.

Programma eseguibile

Perché un algoritmo sia eseguibile da un calcolatore, esso deve essere tradotto in una sequenza di istruzioni codificate nel codice operativo caratteristico della CPU del calcolatore che lo eseguirà (binario).

L'operazione di traduzione da un linguaggio di programmazione ad un altro è una procedura ripetitiva, che può essere automatizzata.

Traduzione in binario

La traduzione in codice eseguibile è effettuata da un programma apposito, che può essere di due tipi:

- **interprete**: traduce solo le istruzioni del flusso di esecuzione;
  - la traduzione viene effettuata ad ogni esecuzione;
- **compilatore**: traduce l'intero programma;
  - la traduzione viene effettuata una sola volta.

Programmare

La generazione di un programma eseguibile consiste nella traduzione automatica

- di un algoritmo espresso in un linguaggio simbolico (programma sorgente)
- nello stesso algoritmo espresso in codice macchina (programma eseguibile).

Catena di programmazione

Le fasi che portano un algoritmo ad essere eseguito sono:

editing: composizione del programma sorgente;

compiling: traduzione in codice binario;

linking: collegamento con i sottoprogrammi di libreria;

loading: caricamento in memoria;

esecuzione: esecuzione del programma.

Editing

Generalmente un algoritmo viene codificato in un programma sorgente.

Lo strumento utilizzato per la scrittura è un apposito programma chiamato editor.

Questa fase ha termine con la produzione di un file di testo detto file programma sorgente.

Compiling

La fase di traduzione, o compilazione, utilizza un programma (detto compilatore).

Esso elabora il file sorgente, riconoscendo i simboli, le parole e i costrutti del linguaggio e producendo:

- la forma binaria del codice macchina corrispondente (file programma oggetto);
- oppure, una segnalazione degli errori sintattici.

Errore sintattico

Gli errori sintattici sono violazioni delle regole sintattiche del linguaggio di programmazione.

Essi rendono il programma non associabile ad un significato e quindi ne impediscono la traduzione.

Esempio:

```
3 * (5 +/ 2)
```

Linking

Nella fase di linking (collegamento), un programma (detto linker) collega il file oggetto con le funzioni di libreria.

Esso produce:

- un programma eseguibile;
- oppure, una segnalazione di errore nella citazione delle routine di libreria (linker error).

Loading

Per poter essere eseguito, un file eseguibile deve essere caricato in memoria centrale.

Questa funzione viene svolta da un programma chiamato loader (caricatore), che individua una regione di memoria adeguata all'esecuzione del programma.

Se tale spazio in memoria non esiste, segnala un errore di caricamento per memoria insufficiente.

Esecuzione

Il programma in esecuzione elabora i dati in ingresso e produce i risultati in uscita.

Possibili errori (non di tipo sintattico!):

- calcoli matematicamente impossibili (run-time error);
  - esempio: divisione per zero;
- operazioni fisicamente impossibili (run-time error);
  - esempio: memoria insufficiente;
- calcoli scorretti;
  - esempio: overflow;
- errori semantici;
  - esempio: errore nella scrittura di un'operazione.