Fondamenti di Informatica

per la Sicurezza

a.a. 2003/04

◇ *Elementi di programmazione* ◇

Stefano Ferrari

Università degli Studi di Milano

Dipartimento di Tecnologie dell'Informazione

Macchina virtuale

virtuale [vir-tu-à-le] *agg.* che esiste in potenza ma non si è ancora realizzato; fittizio, non reale.

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

- Esempio:
  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*, *schiaccia*)
- A livello di programmatore
  - linguaggi
  - paradigmi di programmazione

Risoluzione automatica di problemi

Dato un problema:

specifiche descrizione dei requisiti a cui dovrà soddisfare la soluzione per essere considerata corretta

progetto procedimento con cui si individua o si inventa una soluzione

soluzione tecnica che consente di risolvere il problema

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)
- eseguibili senza ambiguità dall'*esecutore*,
- che risolve il *problema* dato.

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.

Esempio:

- programma che calcola l'area del cerchio di raggio *3 cm* (NO)
- programma che calcola l'area del cerchio di raggio dato dall'utente (SÌ)

Elementi di un algoritmo

“oggetti” Entità su cui opera l’algoritmo: vengono generalmente chiamate *dati*, ma, oltre ai dati iniziali del problema, possono anche essere risultati intermedi.

operazioni Interventi che si possono effettuare sugli oggetti, cioè sui dati: calcoli, confronti, assegnamenti, 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

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

- 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*

Nota: il *flusso di controllo* è differente dal *flusso di esecuzione*

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

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

Linguaggi di descrizione

Oltre ai linguaggi di programmazione sono stati sviluppati dei linguaggi, detti *semiformali*, adatti alla fase di sviluppo di algoritmi: essi presentano alcune caratteristiche di formalità tipiche dei linguaggi di programmazione, ma permettono l'utilizzo di descrizioni in linguaggio naturale, non rigorose.

schemi a blocchi (diagrammi di flusso) blocchi di varie forme geometriche connessi da archi orientati. La direzione degli archi indica il flusso di controllo, mentre la forma dei blocchi indica la natura del blocco stesso (operazione, confronto, operazione di I/O)

pseudocodice 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.
- Risolvere ogni sottoproblema con un sottoprogramma comporta:

  Chiarezza del programma principale 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.
- L'operazione di traduzione da un linguaggio di programmazione ad un altro è una procedura ripetitiva, che può essere automatizzata.
- Tale operazione 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

Catena di programmazione

La generazione di un programma eseguibile consiste quindi nella traduzione automatica

- di un algoritmo espresso il un linguaggio simbolico (*programma sorgente*)
- in un algoritmo espresso in codice macchina (*programma eseguibile*)

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

Editing

Generalmente un algoritmo viene codificato in un programma sorgente tramite 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*

Gli errori sintattici, cioè le violazioni delle regole della sintassi del linguaggio, rendono il programma non correttamente associabile ad un significato e quindi ne impediscono la traduzione.

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

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 scorretti esempio: overflow

calcoli impossibili esempio: divisione per zero

errori semantici esempio: errore nella scrittura di un'operazione

Linguaggi di programmazione

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

Imperativi le istruzioni descrivono esplicitamente le modifiche del contenuto della memoria (Basic, Pascal, C, Fortran, Cobol)

Funzionali il programma è costituito da una funzione che ha per argomenti altre sottofunzioni (Lisp)

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

Orientati agli oggetti il programma è costituito da entità (oggetti) che comunicano tra loro scambiandosi messaggi; incapsulamento, ereditarietà e polimorfismo (Ada, Smalltalk, Java)