Sorgente : http://holsi.hasanaj.com/
Autore : Holsi Hasanaj
Professore: Alfonzo Miola/Limongelli
Corso di Ingegneria Informatica Roma TRe
Anno di produzione: 2010-2011

Fondamenti di Informatica

Sorgente : http://holsi.hasanaj.com/
Autore : Holsi Hasanaj
Professore: Alfonzo Miola/Limongelli
Corso di Ingegneria Informatica Roma Tre
Anno di produzione: 2010-2011

Fondamenti di Informatica — domande di Teoria — 1/3

PREPARAZIONE ESAME

1) Descrivere la macchina di Von Neumann

- La macchina di Von Neumann è un modello semplificato dei calcolatori moderni e che come tale non è una macchina reale
  - Von Neumann è stato il progettista del primo calcolatore in cui i programmi potevano essere memorizzati anziché codificati mediante cavi ed interruttori.

La macchina di Von Neumann è composta da 4 tipologie di componenti funzionali fondamentali:

1) CPU: unità centrale di elaborazione → componente in grado di eseguire istruzioni per l'elaborazione dei dati → svolge inoltre funzioni di controllo
2) MEMORIA CENTRALE: memorizza e fornisce l'accesso a dati e programmi. Una memoria si compone di celle ed ogni cella è caratterizzata da: a) un indirizzo che è un numero che identifica la cella e ne consente l'accesso b) da un valore che è la sequenza di bit memorizzata nella cella.
3) INTERFACCE DI INGRESSO E USCITA: componenti di collegamento con le periferiche del calcolatore → scambio di dati tra calcolatore e utente
4) BUS: svolge la funzionalità di trasferimento di dati e di informazioni di controllo tra le varie componenti funzionali

Lo scopo fondamentale del calcolatore è di permettere l'elaborazione di informazioni → rappresentate sottoforma di dati

2) Descrivere i tipi PRIMITIVI in JAVA:

- Sono un numero limitato di tipi predefiniti del linguaggio JAVA che permettono di rappresentare numeri interi e reali, caratteri e valori logici → sono chiamati PREDEFINITI proprio perché predefiniti
  1) INT: Rappresentazione: in complemento a 2 con 32 bit
     Dominio: interi relativi compresi -2^{31} e $+2^{31}-1$
     Operatori: binari +, −, *, /, %
     Viene inoltre utilizzato l'aritmetica modulare.
  2) BYTE: rappresenta un intero relativo con rappresentazione a 8 bit
  3) SHORT: interi relativi con rappresentazione a 16 bit
  4) LONG: interi relativi con rappresentazione a 64 bit
  5) CHAR: rappresenta un carattere dell'alfabeto UNICODE 2.0 (16 bit)
  6) DOUBLE: rappresenta un numero reale approssimato nella notazione in virgola mobile con una rappresentazione a 64 bit.
  7) FLOAT: rappresentazione dei numeri reali con rappresentazione a 32 bit
  8) BOOLEAN: rappresenta l'algebra di Boole con i valori di verità TRUE/FALSE

3) Descrivere le modalità di gestione della memoria della JAVA VIRTUAL MACHINE

- della memoria
- La gestione dei programmi a tempo di esecuzione prende il nome di MODELLO RUNTIME. La JVM a tempo di esecuzione deve gestire diverse zone di memoria:
  - Zona codice: contiene il java bytecode ovvero eseguibile solo dalla JVM. Inoltre la dimensione della memoria viene determinata a tempo di compilazione
  - Zona Heap: questa zona è dedicata agli oggetti, essa viene gestita dinamicamente ovvero → è area di memoria per un oggetto viene allocata solo nel momento della creazione dell'oggetto e deallocata quando l'oggetto non ha più riferimenti. Quando un oggetto non è più utilizzato è viene cancellato → dalla GARBAGE COLLECTOR
  - Pila dei Record di Attivazione: è una zona di memoria per i dati locali ai metodi, variabili e parametri → cresce e decresce dinamicamente durante l'esecuzione.
    Viene gestita da un meccanismo a pila, che ha una struttura dati con accesso LIFO (Last in First out) che non è altro che un registro.

Sorgente : http://holsi.hasanaj.com/
Autore : Holsi Hasanaj
Professore: Alfonzo Miola/Limongelli
Corso di Ingegneria Informatica Roma Tre
Anno di produzione: 2010-2011

(4) Descrivere le modalità di gestione della memoria nell'esecuzione di metodi JAVA

- l'esecuzione di un metodo può essere descritta in 3 passi:
  1. Il metodo viene invocato e ad esso viene allocato un record di attivazione per gestire le informazioni relative a questa attivazione del metodo. Nel momento dell'allocazione la JVM stabilisce il riferimento all'esecutore e il p.to di ritorno ed effettua il legame tra parametri formali ed attuali
  2. Successivamente viene eseguito il corpo del metodo istruzione dopo istruzione
  3. In fine, quando il metodo termina il controllo torna dove specificato nel punto di ritorno. Alla terminazione del metodo, il record di attivazione del metodo viene deallocato così da permettere agli altri metodi di utilizzare lo spazio di memoria da esso occupato

(5) Descrivere le modalità di gestione della memoria nell'esecuzione di metodi ricorsivi. Fornire un esempio di esecuzione di un metodo ricorsivo a scelta.

- l'esecuzione dei metodi ricorsivi: viene gestita attraverso il modello basato su record di attivazione e pila di attivazione, con l'osservazione che ogni invocazione ricorsiva in un metodo richiede l'allocazione di un nuovo record di attivazione. Il modello di esecuzione dei metodi ricorsivi è basato sul meccanismo della pila di attivazione che alloca un nuovo record di attivazione ogni volta che un metodo viene invocato e lo dealloca ogni volta che il metodo termina. Ciascun metodo di attivazione può accedere solo alle proprie variabili locali allocate nel proprio record di attivazione. In questo caso è possibile che la pila di attivazione contenga molteplici attivazioni dello stesso metodo. In altre parole, la pila inizierà a ridursi o svuotarsi solo dopo aver trovato il passo base come nel caso del metodo fattoriale definito ricorsivamente

→ Ex METODO RICORSIVO: fattoriale(3) viene eseguito come segue:
- il fattoriale di 3 dato che è diverso da zero si invoca il metodo fattoriale con parametro formale 2
- dato che 2 è diverso da zero si invoca il metodo fattoriale con parametro formale 1
- dato che 1 è diverso da 0 si invoca di nuovo lo stesso metodo con parametro formale 0, il fattoriale di zero è il caso base e fa 1
- Il metodo fattoriale 0 restituisce 1 al metodo fattoriale 1.
- Il metodo fattoriale 1 restituisce 1 al metodo fattoriale 2,
- Il metodo fattoriale 2 restituisce 2 al fattoriale di 3 così da ottenere il risultato corretto

(6) Compilazione & interpretazione & strategia JAVA

- Ogni calcolatore ha un suo linguaggio macchina (linguaggio a basso livello) che che non rispecchia la logica del programmatore.
- I linguaggi di programmazione sono dei linguaggi simbolici (ad alto livello) più vicini alla logica del programmatore.
- Il calcolatore può eseguire solo programmi scritti in linguaggi macchina
- Il calcolatore per TRADURRE i programmi può seguire 2 metodi diversi:
  1. COMPILAZIONE: tramite un elaboratore (compilatore), il calcolatore ricevendo un programma sorgente, scritto in linguaggio simbolico, esegue una traduzione producendo un programma eseguibile che, viene eseguito direttamente dal calcolatore.
  2. INTERPRETAZIONE: tramite un elaboratore (interprete) il programma scritto in linguaggio simbolico, viene tradotto ed eseguito, simultaneamente, istruzione per istruzione

LA COMPILAZIONE È ANALOGA alla traduzione di un libro → l'interpretazione è analoga alla traduzione simultanea

- I file compilati hanno un file eseguibile specifico per ogni piattaforma ma sono più veloci nell'esecuzione
- I file interpretati hanno un file interrogabile in qualunque piattaforma ma sono più lenti nell'esecuzione.

→ Java adotta una strategia mista:
- il file sorgente viene prima compilato generando un file in Bytecode (file scritto in linguaggio macchina per una macchina non esistente da JAVA VIRTUAL MACHINE
- Per eseguire il BYTECODE, bisogna interpretarlo su ogni piattaforma in questo modo non viene distribuito il file sorgente ma il Bytecode che è lo stesso per ogni piattaforma.
  In questo modo il diritto d'autore del programma rimane al proprietario in possesso del programma sorgente.

Sorgente : http://holsi.hasanaj.com/
Autore : Holsi Hasanaj
Professore: Alfonzo Miola/Limongelli
Corso di Ingegneria Informatica Roma Tre
Anno di produzione: 2010-2011

Fondamenti di Informatica

PREPARAZIONE ESAME

(7) Descrivere il processo di compilazione ed esecuzione di programmi in JAVA

- Il calcolatore con un programma, COMPILATORE, ricevendo un programma sorgente in linguaggio simbolico esegue la traduzione, producendo in uscita il corrispondente programma eseguibile in linguaggio macchina. In particolare il suo hardware esegue solo programmi in linguaggio macchina → se un programma è in linguaggio macchina può essere quindi direttamente eseguibile
Se un programma è in linguaggio simbolico non può essere eseguito direttamente, c'è bisogno di un processo di TRADUZIONE che viene realizzato tramite opportune applicazioni già disponibili che sono quindi evidentemente in linguaggio macchina
→ ogni istruzione in memoria centrale a sua volta viene eseguita in 3 FASI
  (1) FETCH (lettura): legge dalla memoria la prossima istruzione da eseguire
  (2) DECODE (decodifica): determina il tipo di istruzione che deve essere eseguito
  (3) EXECUTE: (esecuzione): richiede lo svolgimento di tutte le azioni necessarie per l'esecuzione dell'istruzione - ciascuna azione viene richiesta al componente opportuno

(8) Descrivere le modalità di conversione tra tipi primitivi numerici

- Talvolta è possibile/necessario effettuare conversioni da un tipo ad un altro per trasformare la rappresentazione di un valore in un altra rappresentazione
Esistono due tipi di conversione:
  (1) Implicita: (ex) double reale = int intero;
  Il valore intero (come sequenza di bit) non può essere semplicemente trascritto nella variabile reale, ma bisogna prima convertire il valore di intero a un valore equivalente di tipo double → conversione implicita o promozione
  Una assegnazione variabile = espressione è valida se variabile ed espressione hanno lo stesso tipo oppure se il tipo di variabile è più ampio di quello di espressione → la semantica dell'assegnazione consiste nella valutazione di espressione, nella conversione implicita del valore calcolato al tipo di variabile e in un assegnazione a variabile
  Ex: → Math.sqrt(144) → 7.0/2
  (2) Esplicita - si deve effettuare in tutti quei casi in cui è possibile una perdita di precisione. Se T è un tipo, allora la forma sintattica (T) è un operatore unario - operatore di CAST - che indica una conversione di un valore al tipo T → intero = (int) reale
  La conversione esplicita può essere solo realizzata dall'utente che prevede un eventuale troncamento nel valore del dato che diventando un elemento di un dominio più piccolo, potrebbe perdere delle informazioni

(9) Formalismo EBNF: (Extended-Backus-Naur-form)

- È un meta-linguaggio utilizzato per descrivere in modo corretto la grammatica di un qualsiasi linguaggio di programmazione. ~~Inizialmente ma~~ in seguito vennero apportati alcuni cambiamenti per poter avere una maggiore "potenza" nella descrizione della sintassi. L'EBNF come le BNF mette a disposizione dei simboli terminali, cioè "l'alfabeto" del linguaggio che poi può essere esposto a disinsura con i simboli non terminali, cioè tutto quell'insieme di parole e frasi che si possono creare a partire dall'alfabeto. Solitamente i simboli non terminali sono racchiusi tra questi simboli <>
(ad esempio <espressione> qui espressione è un simbolo non terminale) chiamati tag

(10) Sintassi (con notazione EBNF) e semantica delle istruzioni ripetitive disponibili in JAVA → descrivere i criteri di scelta nell'uso di tali istruzioni

- La sintassi di un linguaggio di programmazione (ma anche di un qualunque linguaggio) serve a definire in modo preciso e chiaro quando una frase è corretta sotto il punto di vista grammaticale. Nel caso di un linguaggio di programmazione la frase sarà un istruzione e ogni linguaggio, ha una sua sintassi precisa per definire una certa istruzione
- La SEMANTICA: di un linguaggio invece ci dice se quella frase ha senso in quel linguaggio. Nel caso di un linguaggio di programmazione ad esempio JAVA, System.out(), è un istruzione sintatticamente corretta in quanto viene richiamato il metodo println della classe System → l'unico problema è che →

nella classe System di Java non esiste il metodo ciao, quindi è semanticamente errata. Esistono 3 istruzioni ripetitive:

1. FOR
   - La sintassi → for(<inizializzazione>; <espressione>; <aggiornamento>){ <istruzione>; }
   - La semantica: Inizialmente eseguo una volta l'inizializzazione poi valuto l'espressione booleana; → se è vera eseguo in modo ordinato le istruzioni contenute nel corpo della for, eseguo l'aggiornamento e poi ritorno alla valutazione dell'espressione booleana.
     Se B falsa il ciclo for termina e il programma continua ad eseguire in modo ordinato tutte le istruzioni a partire dalla prima dopo il corpo del ciclo for
   - Il ciclo for lo utilizzo solamente se già so a priori il numero di iterazioni che la mia istruzione ripetitiva dovrà effettuare.
2. WHILE:
   - La sintassi → while(<espressione>){ <istruzione> }
   - La semantica → Valuto l'espressione booleana; se è vera eseguo in modo ordinato tutte le istruzioni all'interno del corpo della while e poi torno al punto di valutazione della condizione booleana se è falsa esco dal ciclo while e continuo con tutte le altre istruzioni del programma

   Il ciclo while lo utilizzo quando non conosco a priori il numero di ripetizioni che dovrò effettuare e questo numero potrebbe essere maggiore o uguale a 0. ~~L'ultimo, ma non per~~
3. DO-WHILE:
   - La sintassi → do{ <istruzione> } while(<condizione>);
   - La semantica → Eseguo il corpo del do while, valuto l'espressione booleana, se è vera eseguo nuovamente il corpo del do-while se è falsa esco dal ciclo.

   In questa istruzione ripetitiva il corpo del ciclo è eseguito sicuramente almeno una volta quindi utilizzerò questa istruzione quando sono sicuro che dovrò eseguire le istruzioni ripetitive almeno una volta

(11) Descrivere in cosa consiste il test sperimentale di correttezza di un programma.

- Il test sperimentale di correttezza di un programma è uno degli strumenti fondamentali dei prodotti software → il test è composto da tre fasi principali:
  - si determina un insieme di dati input I
  - si esegue il programma con i dati I
  - si verificano i risultati ottenuti

Generalmente il test di un programma non permette di stabilirne la correttezza a meno che non vengano provate tutte le possibili scelte di dati in ingresso → non permette quindi di dimostrarne la correttezza assoluta

Due dei test maggiormente utilizzati:

1. TEST A SCATOLA NERA: l'insieme dei dati di ingresso viene scelto solo in riferimento alla specifica del metodo, senza far riferimento ai dettagli realizzativi del metodo
   L'insieme dei possibili dati di ingresso per il metodo viene partizionato in sottoinsiemi (detti insiemi di equivalenza)
   il metodo viene eseguito utilizzando una sola combinazione di valori per ciascun insieme di equivalenza
2. TEST A SCATOLA TRASPARENTE: l'insieme dei dati di ingresso viene scelto in riferimento ai dettagli realizzativi del metodo
   L'idea alla base di questo test è che ciascun test può considerarsi concluso solo quando ciascuna istruzione del metodo sia stata eseguita almeno una volta → questo richiede infatti di scegliere i dati di ingresso in modo tale che ciascuna istruzione sia eseguita almeno una volta.

(12) Descrivere cosa si intende per specifica di un programma e quando esso è soddisfatto

- L'espressione <pre, P, post> rappresenta la specifica del programma P. La specifica di un programma P si dice soddisfatta se vale la seguente proprietà: Per ogni insieme di dati che soddisfano la pre-condizione (Pre), se il programma P termina quando viene eseguito su tali dati allora i dati in uscita prodotti dal programma soddisfano la post-condizione
  Si può inoltre dire che un programma P è corretto rispetto alla specifica se l'insieme di dati di ingresso che soddisfa la pre-condizione pre, il programma P termina e fornisce come risultato che soddisfa la post condizione post

Sorgente : http://holsi.hasanaj.com/
Autore : Holsi Hasanaj
Professore: Alfonzo Miola/Limongelli
Corso di Ingegneria Informatica Roma Tre
Anno di produzione: 2010-2011

Fondamenti di Informatica

PREPARAZIONE ESAME

(13) Descrivere su che base si dice che un programma è corretto.

- Un programma si dice parzialmente (P) è corretto rispetto alla specifica <pre, P, post> se per ogni insieme di dati di ingresso che soddisfa la pre-condizione pre, ie programma P termina e fornisca come risultato un valore che soddisfa la post-condizione post

(14) Descrivere ie principio di induzione matematica e fornisc un esempio dee suo uso

- Un esempio è la LA RICORSIONE → una tecnica di programmazione basata sulla definizione induttiva (o ricorsiva) di funzioni su domini che sono definiti in modo induttivo (o ricorsivo)
  - Una funzione è definita in modo induttivo, se è definita, direttamente o indirettamente, in termini di se stessa
  - Un dominio è definito in modo induttivo se è definito direttamente o indirettamente, in termini di se stesso

La ricorsione è connessa alla nozione di induzione matematica
La fondatezza della ricorsione può essere dimostrata sulla base del seguente principio di induzione MATEMATICA:
- Sia P(n) una proposizione definita per n numero naturale
- La proposizione P(n) è vera se per ogni n se:
  - P(0) è vera - (per definizione) passo base dell'induzione
  - Per ogni naturale K, se P(K) è vera (ipotesi induttiva) allora anche P(K+1) è vera - passo induttivo

Ex: Sia P(n) la proposizione definita per n, numero naturale per cui $n! = n*(n-1)!$
la proposizione P(n) è vera per ogni numero naturale n se
- P(0) è vera, passo base ($0!=1$) - è per definizione
- Per ogni naturale K, se P(K) è vera (ipotesi induttiva) allora anche P(K+1) è vera - passo induttivo
  - Se P(K) è vera: $K! = K*(K-1)!$
  - allora anche P(K+1) è vera $(K+1)! = (K+1)*K! = \ldots$

(15) Fornire almeno un insieme di esempio di insieme definito induttivamente di una funzione definita induttivamente → con relativo metodo ricorsivo JAVA

- L'insieme N dei numeri naturali può essere definito in modo induttivo come
  - $0 \in N$ → lo zero è un numero naturale
  - se $n \in N$, allora anche $n+1 \in N$ → ie successore di un numero naturale è naturale
- L'insieme S_Σ delle stringhe su un alfabeto Σ di caratteri può essere definito in modo induttivo come segue
  - $\varepsilon \in S_\Sigma$ → la stringa vuota è una stringa su Σ
  - se $c \in \Sigma$ e $S \in S_\Sigma$, allora anche $c+S \in S_\Sigma$, un carattere in Σ concatenato a una stringa su Σ è una stringa su Σ

In generale quindi nella definizione induttiva di un insieme è possibile identificare
- uno o più casi base, un caso base descrive l'appartenenza di alcuni elementi all'insieme in modo diretto
- uno o più casi induttivi, un caso induttivo descrive l'appartenenza di alcuni elementi all'insieme in modo indiretto, ovvero in termini dell'appartenenza di altri elementi all'insieme stesso

- DEFINIZIONE RICORSIVA DELLA FUNZIONE FATTORIALE:

$$fatt(n) = \begin{cases} 1 & \text{se } n=0 \text{ - caso base} \\ n * fatt(n-1) & \text{se } n>0 \text{ - caso induttivo} \end{cases}$$

JAVA

```
public static int fattoriale (int n) {
    int f;
    if (n==0) f = 1;
    else f = n* fattoriale (n-1);
    return f;
}
```

Sorgente : http://holsi.hasanaj.com/
Autore : Holsi Hasanaj
Professore: Alfonzo Miola/Limongelli
Corso di Ingegneria Informatica Roma TRe
Anno di produzione: 2010-2011

(16) Descrivere in cosa consiste l'analisi asintotica della complessità:

- Nello studio del costo dei programmi è importante valutare la funzione di costo al crescere della dimensione dell'input → in questo caso si possono trascurare costanti moltiplicative e termini di ordine inferiore
  Lo studio ASINTOTICO del costo
  - fornisce un'idea dell'andamento del costo all'aumentare della dimensione dell'input
  - generalizza (è approssimato) ulteriormente il modello di costo adottato e consente quindi di semplificare i calcoli.

(17) Descrivere il relativo modello di costo per l'analisi asintotica della complessità e il significato della notazione O grande.

- analisi della complessità ha lo scopo di determinare una funzione chiamata funzione costo → questa funzione non viene espressa in termini di unità di tempo ma in base al numero di operazioni elementari. La complessità computazionale viene calcolata considerando uniforme il costo speso nell'esecuzione di ciascuna operazione elementare
  - Nell'analisi di complessità, la funzione di costo viene in genere determinata rispetto al caso peggiore
  In pratica la funzione di costo asintotica di un metodo approssima la funzione di costo del metodo quando la dimensione dei dati in ingresso cresce in modo illimitato → studia il comportamento a limite.
  Esso descrive l'ordine di esecuzione del costo O [grande].
  - COMPLESSITÀ COSTANTE - COMPLESSITÀ LINEARE
  - COMPLESSITÀ LOGARITMICA - COMPLESSITÀ QUADRATICA
  - COMPLESSITÀ POLINOMIALE - COMPLESSITÀ ESPONENZIALE

(18) ORdinamento

- Il problema dell'ordinamento consiste nel trasformare una sequenza di elementi rendendola ordinata rispetto ad un certo criterio
  Ci sono diversi algoritmi di ordinamento che si differenziano su come vengono selezionate e scambiate le coppie di elementi da confrontare. L'algoritmo di ordinamento procede dunque per fasi che sono chiamate passate

COMPLEX. ASINTOT. n^2 (1) SELECTION SORT: è basato sulla strategia iterativa → finché la sequenza non è ordinata → seleziona l'elemento di valore minimo della sequenza tra quelli che non sono stati ancora ordinati e dispone nella posizione definitiva

COMPLEX ASINTOT: n^2 (2) BUBBLE SORT: nell'ordinamento a bolle gli elementi da ordinare sono partizionati in due insiemi contigui, uno di elementi ordinati che hanno sicuramente raggiunto la loro posizione definitiva, e uno di elementi non ordinati che devono ancora raggiungere la loro posizione definitiva.
Una passata dell'ordinamento a bolle ha lo scopo di ordinare l'elemento di valore massimo tra quelli non ancora ordinati.

COMPLEX ASINTOT: n^2 (3) INSERTION SORT: gli elementi da ordinare sono partizionati in due insiemi uno di elementi relativamente ordinati (a sinistra) e uno non relativamente ordinati (dex) → inizialmente il primo elemento della sequenza viene considerato relativamente ordinato e tutti gli altri no. In ciascuna passata il primo tra gli elementi non relativamente ordinati viene collocato tra gli elementi relativamente ordinati → inserendolo nella sua posizione corretta

COMPLEX ASINTOT: $n \log n$ (4) MERGE: (fusione) se la sequenza da ordinare contiene più di due elementi allora viene divisa → gli elementi della sequenza vengono partizionati in due sottosequenze, le due sottosequenze vengono ordinate separatamente, le due sottosequenze ordinate vengono fuse in un unico sequenza ordinata

(5) QUICK SORT: è l'algoritmo di fusione viene utilizzato maggiormente poiché pur avendo un costo asintotico nel caso peggiore uguale a n^2, mediamente possiede un costo $n \log n$ dove questo n è molto minore di quello del merge.