DOMANDE POSSIBILI IN LINGUAGGI DI PROGRAMMAZIONE PER CAPITOLO

Capitolo 1

1. Quali sono le qualità di un linguaggio?

   La sua leggibilità (chiaro, naturale, semplice), la sua scrivibilità (facilità per passare da un algoritmo al rispettivo codice), la sua affidabilità (facile verifica e non presenta errori nascosti) e il suo costo in termini di efficienza

2. Quali sono gli aspetti di un linguaggio di programmazione?

   La sua sintassi, la semantica, la pragmatica, l'implementazione, le librerie e i tools

3. Cos'è una macchina astratta (virtuale)? //OCCIO

   Una macchina astratta è una macchina capace di processare del codice che proviene da una macchina differente di più alto livello che consente di arrivare fino allo sfruttamento del hardware della macchina fisica, quindi passando da un linguaggio di alto livello, si arriva all'assembly o al codice macchina tramite le macchine astratte che convertono passo passo il codice (JVM).

4. Cosa si intende per compilazione pura?

   Si intende la sola compilazione del programma da livello alto alla traduzione ad un equivalente programma di livello più basso (verso l'hardware), fino ad arrivare al codice macchina o al codice assembly + il codice per l'utilizzo nel sistema operativo.

5. Cosa si intende per interpretazione pura?

   Si intende una traduzione istantanea con dati e programma sorgente e le singole istruzioni sono eseguite immediatamente

6. Quali sono i vantaggi di compilazione ed interpretazione?

   - Compilazione: migliori prestazioni generali e gli errori vengono presentati al momento dei controlli prima della sua esecuzione
   - Interpretazione: molto più flessibile, più semplice di implementazione, esecuzione diretta del codice e debugging più semplice

7. Cosa si intende per bootstrap?

   Scrittura di un nuovo compilatore che da come output un codice più efficiente e ricompilo il codice prodotto con la nuova versione che risulterà più efficiente di quella precedente

8. Cos'è l'analisi lessicale?

   Si intende il processo di scansione (scanner), divisione in lessemi, con la produzione di un token per lessema e la semplificazione dell'analisi di un programma. Le singole classi di lessemi sono descritti da linguaggi regolari e lo scanner implementa un DFA.

9. Cos'è il parsing?

   si analizza l'intero programma, definendo la sua struttura. Viene descritto tramite linguaggi liberi dal contesto e viene costruito l'albero della sintassi astratta una rappresentazione ad albero della struttura del programma.

10. Cos'è l'analisi semantica?

    Esegue controlli sul codice statici non implementabili dal parser ed altri controlli come le matrici fuori limite, quindi dinamici.

11. Che cos'è la tabella dei simboli?

    La tabella che contiene tutti i simboli dell fasi, quindi contiene tutti gli identificatori del programma e di ciò che il compilatore sa di loro.

Capitolo 2

1. Cos'è la BNF?

   La Backus-Naur Form è una metasintassi, ovvero un formalismo attraverso cui è possibile descrivere la sintassi di linguaggi formali, descrivendoli in modo preciso e non ambiguo. Può essere descritto come un formalismo per descrivere grammatiche libere dal contesto.

2. Che cosa sono le grammatiche ambigue?

   Sono delle grammatiche che possono riconoscere delle stringhe uguali partendo da due punti differenti degli alberi di derivazione, quindi stesso risultato ma da due punti di inizio differenti.

3. Come si disambigua?

   Rendendo la grammatica non ambigua inserendo nuovi non terminali oppure ottenere una nuova grammatica che genera lo stesso linguaggio ma che è più complessa; si può anche convivere con la grammatica ambigua, inserendo informazioni aggiuntive su come risolcere le ambiguità.

4. Cosa sono gli abstract syntax tree?

   Sono gli alberi sintattici a cui sono stati eliminati i nodi rindondanti ai fini dell'analisi, dando solo informazioni utili, consentendo una computazione più pratica ed evidenziano meglio il significato di una stringa di caratteri

5. Cos'è la semantica?

   La semantica di un programma definisce il suo significato, descrive il suo comportamento a run-time.

6. Quali sono le operazioni eseguite sui linguaggi?

   L'unione, la concatenazione e la chiusura di Kleene dalle quali posso costruire l'insieme delle espressioni regolari, ossia l'algebra di Kleene (metodo per rappresentare i linguaggi in modo sintetico).

7. Quali sono le estensioni?

Olte alle operazioni base si possono definire altre operazioni:

- la chiusura positiva
- la zero o una istanza
- la n concatenazioni di parole
- uno tra (un or tra i vari valori di una certa lista di elementi)
- opposto (rappresentato da ^ davanti ad i caratteri successivi come [^a-z] quindi prende solo le minuscole)

8. Enuncia il teorema di equivalenza.

   I linguaggi regolari possono essere descritti in modi diversi:

   - espressioni regolari
   - grammatiche regolari
   - automi finiti non deterministici
   - automi finiti deterministici

   Esiste anche un automa deterministico minimo (quindi con il minor numero di stati), chiamato il DFA minimo.

9. Cos'è uno scanner?

   Uno scannere (lexer) deve risolvere un problema più complesso del semplice riconoscimento di una singola espressione regolare, ovvero deve dividere la stringa d'ingresso in lessemi, ciascuno riconosciuto da un'espressione regolare.

10. Come viene costruito uno scanner?

    Costruisco un automa per ogni espressione regolare e sulla stringa di ingresso, simulo il funzionamento parallelo degli automi e riconosco un lessema quano nessun automa può continuare.

11. Che cosa sono gli automi a pila? (automi LL)

    Sono degli automi con un uso limitato di memoria (quindi stati finiti) che accetteranno la stringa se la pila è vuota o se si raggiunge uno stato finale durante l'analisi della stringa.

12. Che cosa sono gli automi LL?

    Sono degli automi che costruiscono l'albero di derivazione in modo top-down a partire dal simbolo iniziale, esaminando al più n simboli della stringa non consumata (lookahead) e determina la prossima regola (espansione) da applicare

13. Che cos'è un analizzatore LR?

    Un analizzatore LR è un analizzatore creato come quello LL solo con l'approccio bottom-up, quindi a partire dalla stringa di input applico una serie di contrazioni (regole al contrario) fino a contrarre tutto l'input nel simbolo iniziale della grammatica.

Capitolo 3

1. Che cosa sono il legame e l'ambiente?

   Il legame (binding) è l'associazione esistente tra un nome e un oggetto mentre l'ambiente (environment) sono gli insiemi dei legami esistenti dipendenti da uno specifico punto del programma, quindi può essere condizionato dal codice eseguito in precedenza quindi la storia del programma.

2. Come si creano i binding?

   I binding si creano con il linguaggio (quindi tipi primitivi, operazioni primitive, costanti predefinite) e quelli che crea il programmatore (sottoscrizione di variabili, parametri formali, procedure, ...).

3. Quando si possono creare i binding?

   - durante la definizione del linguaggio
   - la scrittura del codice
   - il caricamento del programma in memoria
   - l'esecuzione
   - ...

4. Quali tipi di binding esistono?

   Esistono 2 tipi di binding, ovvero il binding statico (viene eseguito prima dell'esecuzione della prima istruzione) e binding dinamico (che avviene durante l'esecuzione)

5. Che cosa sono le dichiarazioni?

   Le dichiarazioni sono un meccanismo (implicito o esplicito) col quale si crea un legame e si modifica l'ambiente

6. Che cosa sono i blocchi?

   Sono regioni del programma nelle quali sono contenute dichiarazioni

7. Quali operazioni si possono effettuare su un ambiente?

   - Si possono creare associazioni nome-oggetto denotato (naming)
   - Si possono distruggere le associazioni nome-oggetto denotato (unnaming)
   - Si possono fare riferimenti ad oggetto denotato meiante il suo nome (referencing)
   - Si possono diasttivare associazione nome-oggetto denotato
   - Si possono riattivare ddelle associazioni nome-oggetto denotato

8. Quali sono le operazioni possibili sugli oggetti denotabili?

   La creazione, l'accesso, la modifica (se l'oggetto è modificabile) e la distruzione di tali oggetti. La creazione e la distruzione fanno riferimento a dati in memoria (variabili) e il codice (procedure).

9. Differenze tra scope statico e dinamico

   Lo scope statico ha le informazioni complete dal testo del programma, le associazioni sono note a tempo di compilazione è complesso da implementare ma è più efficiente Lo scope dinamico ha le informazioni derivanti dall'esecuzione e spesso a causa di programmi meno leggibili, però è più flessibile e quindi le modifiche sono possibili all'istante ed è più semplice da implementare

10. Cosa si intende per aliasing?

    Si intende una uguale denomina dello stesso oggetto (puntatori, passaggio di parametro per riferimento alle procedure), possono portare all'overloading e quindi serve il contesto per poter capire il vero funzionamento della funzione

11. Da che cosa è determinato l'ambiente?

    L'ambiente è determianto da regole di scope (statico o dinamico), dalle regole specifiche (come ad esempio quando sia possibile la visibilità di una dichiarazione nel blocco), regole per il passaggio dei parametri e le regole di binding (shallow o deep)

12. Cosa sono i moduli e a che cosa servono?

    Per evitare grossi conglomerati di codice e problemi nell'information hiding, si sviluppano i moduli che sono grosse parti di codice che portano altre funzionalità e che permettono anche l'information hiding, però con nomi simili possono possono portare a un overloading provocando comportamenti imprevedibili

---

Capitolo 4

1. Come viene riorganizzata la Ram solitamente?

    - 0 - 0xFFF è la parte riservata al sistema operativo
    - 0x1000 - ww è la parte riservata al .text del codice del programma
    - ww - xx è la parte riservata alle costanti, variabili del programma principale nel .data
    - xx - yy è la parte della heap per i dati dinamici come liste e alberi
    - yy -zz è la parte dedicata allo stack per le chiamate di procedura e allo stack di attivazione
    - il registro r13 (SP, stack pointer) è il registro dedito a puntare alla cima dello stack
    - il registro r11 (FP, frame pointer) è il redistro dedito a puntare al "frame" della procedura in esecuzione

2. Quali sono i tipi di allocazione alla memoria?

    - Vi sono 3 meccanismi di allocazione della memoria:
        - statica, la memorai è allocata a tempo di compilazione
        - dinamica, la memoria è allocata a tempo di esecuzione di cui si devono ricordare
            - la pila (stack) dove gli oggetti sono allocati con politica LIFO (last in first out)
            - la heap dove gli oggetti sono allocati e de-alloccati in qualsiasi momento

3. Cosa si intende per allocazione statica?

    Si intende che gli oggetti hanno un indirizzo assoluto, mantenito per tutta l'esecuzione. Le variabili globali, le costanti e le tabelle usate dal supporto a run time sono solitamente alloccate saticamente. Non permettono la ricosione e viene forzato più spazio rispetto a quello necessario; daltronde sono ad accesso diretto e veloce

4. Cosa si intende per allocazione dinamica?

Si intende che ogni istanza di procedura in esecuzione possiede un record di attivazione (RdA o freame) e può contenere qualsiasi tipo di oggetto

5. Che cos'è il link statico?

   Il link statico è il metodo per poter accedere ai RdA degli antenati, poiché ogni RdA contiene un puntatore all'ultimo RdA del genitore, definendo una catena statica, ovvero la lista dei RdA degli antenati

6. Riassunto dell'ultima parte

   Il link dinamico (procedura chiamante) dipende dalla sequenza di esecuzione del progeamma, definendo la catena dinamica Il link statico (procedura genitore) dipende dalla struttura delle dichiarazioni di procedure, definendo la catena statica

7. Come si crea il link statico?

   La procedura chiamante determina il link statico della procedura chiamata, quindi passa alla procedura chiamata che lo inserisce nel suo RdA.

8. Come si possono ridurre i costi?

   Per ridurre i costi si è implementato il display, che è un unico array contenente il link ai RdA visibili al momento attuale, quindi evitando di dover ripercorrere tutta la catena statica

9. Come si aggiorna il display?

   Per aggiornare il display è necessario aggiornare il display all'ingresso di ogni procedura, all'uscita della procedura viene ripristinato il valore originale

10. Che cos'è la CRT?

    La CRT è la tabella centrale dei riferimenti, quindi evita le lunghe scansioni della A-list, mantenendo tutti i nomi distinti del programma, associati alla lista delle associazioni di quel nome; tempo costante per l'accesso alle variabili

---

Capitolo 5

1. In che cosa differiscono i paradigmi di programmazione funzionale ed imperativo?

   L'imperativo e il funzionale differiscono nei meccanismi di controllo adottati, quindi l'assegnazione, sequenzializzazione e iterazione fanno parte di un paradigma imperativo, mentre valutazione di espressioni e ricorsione sono per il dichiarativo

2. Che cosa sono le espressioni?

   Sono un insieme di identificatori, letterali e operatori che possono produrre un risultato, un possibile effetto collaterale e possono divergere

3. Che cos'è lo zucchero sintattico?

Intendiamo per zucchero sintattico, delle scritture alternative di una espressione (comando) per migliorare la sua leggibilità

4. Cos'è la notazione polacca?

   Vi sono due notazioni polacche, diretta ed inversa. Sono due metodi di notazioni che non necessitano parentesi, cambiando però l'ordine di identificatori e operatori. C'è la possibilità di omettere le parentesi

5. Quali sono le reogle di precedenza?

   Ogni linguaggio di programmazione fissa le sue regole di precedenaza tra operatori; solitamente gli operatori aritmetici hanno la precedenza su quelli di confronto che hanno precedenza su quelli logici solitamente.

6. Quali sono gli effetti collaterali?

   Gli effetti collaterali sono tipicamente la restituzione di un valore ma vi è la modifica dello stato del programma (esempio, valutazione espressione -> chiamate di funzione -> funzione modifica la memoria)

7. Cosa si intende per linguaggi funzionali puri?

   Si intende che la computazione con tali linguaggi si riduce alla sola valutazione di espressioni senza effetti collaterali

8. Che cosa si intende la valutazione lazy?

   Si intende che vengono valutati gli operandi quando sono strettamente necessari

9. Cosa si intende per valutazione corto-circuito?

   Si intende non è necessario valutare ogni singolo booleano poiché è sufficiente la prima valuazione in se, poichè non modificherebbe il risultato

10. Che cosa sono i comandi?

    I comandi sono la parte del codice la cui valutazione non restituisce un valore ma ha un effetto collaterale (modifica dello stato). Solitamente sono tipici del paradigma imperativo e non sono molto presenti nei linguaggi funzionali e logici

11. Che cos'è l'assegnamento?

    L'assegnamento è l'inserimento in una locazione, cella di un valore ottenuto valutando una espressione.

12. Che cosa sono rval e lval?

    Sono due metodi di assegnamento di una variabile, lval solitamente è la denotazione della locazione di memoria, mentre rval è il conetenuto della locazione

13. Differenza tra espressione e comando

    Sintatticamente il coamndo è importante per l'effetto collaterale ottenuto da tale comando, mentre l'espressione è importante per il valore restituito.

14. Cosa si intende per pre e post incremento?

    Per pre incremento si intende la restituzione immediata del valore incrementato, mentre il post è restituito il valore originale ma dopo viene incrementato successivamente al termine della procedura

15. Che cos'è il comando goto?

    Il comando goto è il comando che permette di saltare parti del codice per passare da una sezione ad un'altra senza processare il resto del codice. Molto flessibile ma rende illeggibile il codice, complicandolo.

16. Che cos'è l'iterazione?

    L'iterazion e la ricorsione son i due meccanismi che permettono di ottenere tutte le funzioni computabili. Può essere indeterminata con cicli controllati logicamente o determinata con cicli controllati numericamente.

17. Che cos'è la tail recursion?

    La tail recursion si dice come di coda, è la chiamata di una funzione, che attende il suo risultato senza fare nulla, ad un'altra funzione che deve riportare il risultato. Simulo un ciclo while

---

Capitolo 6

1. Quali sono i due modi principali per il passaggio dei parametri?

   Uno è per valore, quindi il parametro formale è una variabile locale, preso quindi l'rvale L'altro è per riferimento che è passato per riferimento (indirizzo) all'attuale, l-value

2. Quali sono i metodi per il passaggio delle funzioni?

   Alcuni linguaggi possono passare funzioni come argomenti di altre funzioni (procedure), dove le funzioni sono passate come oggetti di secondo livello. Altri possono restituire funzioni come risultato di funzioni, ovvero sono trattate come funzioni di primo livello

3. Quali sono le politiche di binding e su quale ambiente vengono applicate?

   Vengono applicate sull'ambiente in uso se si parla di scope statico mente per lo scope dinamico, il deep binding viene creato l'ambiente al momento della creazione del legame, mentre lo shallow binding viene creato al momento della chiamata di varie funzioni

4. Come funziona la gestione delle eccezioni?

   Viene definito un insime di eccezioni, il programma la rileva l'eccezione e viene fatto un raise an exception, un gestore si occupa dell'eccezione sollevata e per arrivare al gestore, se necessario, va cercato tramite la catena dinamica del RdA per arrivare al primo gestore disponibile.

5. Cosa succede se un'eccezione si propaga?

   La routine termina. l'eccezione viene risollevata al punto di chiamata e se l'eccezione non è gestita dal chimante l'eccezione si propaga lungo la catena dinamica si toglie un RdA e si passa al relativo

chiamante, finché non si arriva al top-level che fornisce un gestore deaful e ferma il programma con un messaggio di errore.

Capitolo 7

1. Quando avviene il controllo di tipo?

   Dipende se statico o dinamico, poiché lo statico avviene a tempo di compilazione, mentre quello dinamico viene fatto durante l'esecuzione del codice.

2. Quali sono i vantaggi e gli svantaggi del controllo di tipo statico?

   Il vantaggio maggiore è la prevenzione e anticipazione degli errori di tipo; poi si garantisce un carico di lavoro minore, poiché a tempo di esecuzione molti controlli possono essere omessi ma come contro ha la possibilità di essere prolisso e meno flessibile.

3. Quali sono i vantaggi e svantaggi del controllo di tipo dinamico?

   I vantaggi sono la flessibilità e possibilità di avere un codice più conciso e test ottimizati il più possibile (test non pesanti) ma il grosso svantaggio è che va eseguito il codice per trovare l'errore.

4. Cosa è il strong type system?

   Il strong type system è il sistema di tipi forte che rende difficile che errori di tipo vengano rilevati (type safe), quindi il problema di errore di tipo non avverrà mai durante l'esecuzione

5. Cosa è il weak type system?

   Il weak type system è il sistema di tipi debole che permette una maggiore flessibilità ma a costo della sicurezza poiché è possibile avere problemi relazionati ad errori di tipo sia durante l'esecuzione, sia durante la compilazione

6. Cosa si intende per Dope Vector (descrittore del vettore)?

   Si intende un vettore dove sono mantenute tutte le informazioni necessarie, come il puntatore all'inizio dell'array, il numero dimensioni, il limite inferiore o superiore e l'occupazione per ogni dimensione. Il dope vector è memorizzato nella parte fissa del RdA, e per fare l'accesso al vettore dati solitamente si calcola l'indirizzo a run-time, usando esattamente il dope vector

7. Cosa sono i puntatori e come si possono usare?

   I puntatori identificano una locazione di memoria, l-value e fanno riferimento a dei valori o a delle costanti (solitamente null); i puntatori sono nel tipo del puntatore specifico del tipo dell'oggetto puntato e le operazioni fattibili sono la creazione, la dereferenziazione e il test di uguaglianza. In c si può anche eseguire l'aritmetica sui puntatori.

8. Cosa si intende per Dangling refernce?

   Questo è uno dei proble con l'uso dei puntatori, ovvero si fa riferimento ad unazona della memoria che contiene dati di un tipo non compatibile, quindi arbitrari. Le possibile cause sono l'aritmetica sui

puntatori (uscita dalla memoria alloccata), la deallocazione dello spazio sulla heap (free(p)) o la deallocazione dei RdA. Per risolvere il problema va ristretto l'uso dei puntatori e non deve essereci nessuna esplicita deallo cazione e implementato una garbage collection.

9. Quando si dice che due relazioni di tipo sono equivalenti?

   Quando due espressioni denotano "lo stesso tipo".

10. Quando invece di dicono di compatibilità?

    Quando la prima funzione è compatibile con la seconda, ovvero i valori della prima possono essere usati in contesti dove ci si attende valori della seconda ma non è necessariamente detto che sia possibile fare anche viceversa.

11. Quali tipi di equivalenza ci sono?

    - Per nome, quindi due tipi sono equivalenti se sono lo stesso nome, identificatore di tipo enon è sufficiente che leespressioni siano di tipo uguali
    - Per nome lasca, non sono per nome ma sono degli alias per determinare la stessa variabile ma non genera un nuovo tipo, è solo una rinomina della variabile.
    - Per equivalenza strutturale, ovvero è solo necessario controllare che se l'espansione delle definizioni di una struttura genera la stessa espressione di tipo della seconda, allora le strutture sono equivalenti strutturalmente.

12. Cos'è il polimorfismo'

    Si intende per polimorfismo, una stessa espressione che può assumere diversi tipi, distinguendole tra polimorfismo ad hoc o overloading (dove uno stesso simbolo denota diversi significati) e due tipi di polimorfismi universali, parametrico esplicito o implicito (infinità di diversi tipi, ottenuti per istanziazione da un unico schema di tipo generale) e il polimorfismo di sottotipo (usato nei linguaggi ad oggetti e può assumere diverse forme a seconda del linguaggio di programmazione dove i tipi sono compatibili)

Capitolo 7

1. Cosa si intende peer principio di incapsulamento?

   Si intende l'implementazione di una parte di codice o di un programma che è indipendente dalla rappresentazione (ovvero due implementazioni corrette di un tipo non sono distinguibili dal contesto), sono modificabili senza dover interferire col contesto ed è anche usato per il nascondimento di informazioni (information hiding)

2. Cosa sono i moduli?

   I moduli sono un costrutto generale per l'information hiding, evitanod i conflitti di nome, facendo si che vi sia una compilazione separata, consesnte la creazione di interfaccie più sofisticate e i moduli sono visibili all'esterno, ma in maniera ristretta. Vanno eseguite delle importazioni ed esportazioni per utilizzare i costrutti di tali moduli.

Capitolo 8

1. Cos'è una classe?

   Una classe definisce un insieme di oggetti specificando, un insieme di campi con relativo tipo, un insieme di metodi con relativo codice, visibilità dei campi e metodi e costruttori degli oggetti.

2. Quali sono gli aspetti nella definizione di una classe?

   l'information hiding e incapsulamento e definisco cosa deve essere visibile all'esterno o meno (quindi uso private, public e protected), l'astrazione sui dati e sul controllo quindi assegno un nome alla classe, ai campi e ai metodi ed infine la possibilità di estendere e riusare il codice, quindi poter definire delle sottoclassi

3. Cosa sono i linguaggi prototype based?

   Sono dei linguaggi OO che usano gli oggetti come record con metodi, non hanno la necessità di pattern predefiniti e sono preseti nei linguaggi con sistema di tipo dinamico (esempio JS).

4. Cosa sono i linguaggi a modello riferimento-variabile?

   Normalmente gli oggetti usano il modello a riferimento (quindi variabile contiene un puntatore all'oggetto immagazzinato nella memoria). I modelli a valore invece sono variabili contenute nei campi dell'oggetto che sono memorizzati nello stack, quindi si ha una creazione implicita degli oggetti (maggiore efficienza ma spreco di memoria).

5. Cosa sono le sottoclassi?

   Le sottoclassi sono classi che creano nuovi metodi o campi ed estendono le classi principali, non toccando la classe principale (avendo anche tipo compatibili tra classi)

6. Cosa si intende per ereditarietà?

   Si intende la definzione di sottoclassi, con un meccanismo di riuso e condivisione del codice, permettendo l'estensione del codice e la modificabilità del codice.

7. Cosa si intende per ereditarietà singola?

   Si intende che ogni classe eredita al più da una sola super classe immediata.

8. Cosa si intende per ereditarietà multipla?

   una classe può ereditare da più di una super-classe immediata.

9. Cosa si intende per polimorfismo di sottotipo?

   Si può dire che è un meccanismo che permette di usare un oggetto in un altro contesto ed è la relazione tra le interfacce di due classi (interfaccia = insime dei campi e metodi pubblici di una classe)

10. Cos'è il duck typing?

Sostanzialmente è se l'uso di oggetti in un contesto è lecito e se possiede tutti i metodi necessari, indipendentemente dal suo tipo (sistema a tipo dinamico).

Capitolo 9

1. Differenze tra Haskell e Scheme

   Sono entrambi fortemente tipati ma Scheme ha come controllo di tipo dinamico e Haskell usa un controllo di tipo statico. Poi Scheme è un linguaggio eager, quindi una valutazione viene fatta quando tutto appena possibile, mentre Haskell è lazy, quindi la valutazione di un argomento viene fatta solo se necessaria.

2. Vantaggi e svantaggi della valutazione lazy.

   Il vantaggio maggiore è quello di evitare lavoro inutile quindi se non necessario, non valuto nemmeno il codice. Più programmi terminano è più programmi non generano errori. Per gli svantaggi, valuto la possibilità di avere potenziali ripetizioni delle esecuzioni, evitabili con il call-by-need e le espressioni con side-effect sono complicate da gestire.

3. Che cosa sono i thunk?

   Sono dei sottoprogrammi che vengono creati, automaticamente, per assistere una chiamata ad un'altra subroutine, quindi sono pricipalmente rappresentati da un calcolo aggiuntivo di un sottoprogramma che deve calcolare espressioni complesse.

Capitolo 10 e 11 sono tutti esercizi per poter fare Haskell, Lex e Yacc

Capitolo 12

1. Cosa si intende per programmazione concorrente?

   Si intende una programmazione che mira a sfruttare l'hardware, quindi aumentando il parallelismo e la possibilità di divide un compito in sottocompiti da svolgere in parallelo e in maniera indipendente, gestendo l'hardware distribuito.

2. Come si realizza la concorrenza fisica?

   Viene fatta a livello di macchina fisica e si prevede l'esecuzione simultanea di istruzioni, con diversi gradi di granularità (pipeline, superscalari, istruzioni vettoriali, ...)

3. Come si realizza la concorrenza logica?

   Si lavora a livello di linguagio di programmazione, quindi si cerca di parallelizzare l'esecuzione di un singolo programma e tramite anche all'esecuzione di algoritmi in parallelo. Un altro tipo di programmazione è lo sfruttamento del multithread, ovvero più thread o processi attivi

contemporaneamente che usano la stessa macchina fisica. Come altro modo vi è la programmazione distribuita, ovvero tramite i programmi concorrenti, eseguito su macchine separate (come i multicomputer) e non si assume la memoria condivisa.

4. Cos'è la SOC?

   La SOC è un paradigma di programmzazione basata sulla composizione di componenti (servizi) che interagiscono (Service Oriented Computing)

5. Cos'è il cloud computing?

   Ha lo stesso scopo della SOC ma con meno struttura, usando una granularità più fine, usate per lo sviluppo di applicazioni distribuite su scala planetaria, con accesso on demand.

6. Alcuni aspetti della programmazione concorrente

   Molti sono i modi per strutturare la programmazione concorrente ma in particolare, sono necessarie due parti fondamentali:

   - la comunicazione tra i thread per avere le informazioni necessarie
   - la sincronizzazione per regolare la velocità dei thread

7. Cosa sono i thread?

   I thread del controllo è l'esecuzione di comandi, una specifica computazione con spazio di indirizzamento comune

8. Cosa sono i processi?

   Sono un generico insieme di istruzioni in esecuzione con il proprio spazio di indirizzamento e un processo può contenere più thread divisi

9. Quali sono i meccanisimo di comunicazione?

   - Memoria condivisa, accesso in lettura e scrittura ad una zona comune di memoria
   - Scambio di messaggi, usando primitive esplicite di invio e ricezione di messaggi, tramite un canale usando un percorso logico fra mittente e destinatario
   - Blackboard, si intende un modello intermedio fra i precedenti ovvero che condividono la stessa zona di memoria e comunicano tramite send e recive.

10. Quali sono i meccanismi di sincronizzazione'

    Sono i meccanismi che permettono di controllare l'ordine relativo delle varie attività dei processi. Le race condition sono uno degli strumenti per questo scopo, che valuta i programmi concorrenti intrinsecamente, dando un diverso ordine di esecuzione, escludendo quelle non accettabili. Poi parliamo di mutua esclusione dove una regione di codice dove ci sono dati che non possono essere accessibili contemporaneamente a più processi. Poi abbiamo la sincronizzazione su condizione e semplicemente sospende l'esecuzione di un processo fino al verificarsi di una opportuna condizione.

11. Come si implementa la sincronizzazione?

    - Attesa attiva (busy waiting o spinning), ha senso solo su multiprocessori

    - sincronizzazione basata sugli scheduler, con sospensione del thread

12. Cosa è la sincronizzazione su condizione con barriere?

    Si tratta di un meccanismo di sincronizzazione e si basa sull'attesa che una condizione globale non venga soddisfatta.

13. Cosa sono i semafori?

    Si tratta di un costrutto per la sincronizzazione di thread per bloccare o far partire i singoli thread.

14. Cosa sono i monitor?

    Sono dei semplici contatori condivisi per la gestione corretta delle risorse codnivise, venendo affidata al programmatore.

15. Come può essere eseguita la comunicazione mediante scambio di messaggi?

    Può essere o sincrona (quindi l'invio e la ricezione di un messaggio avviene allo stesso tempo) ed asincrona (l'invio e la ricezione di un messaggio possono avvenire in momenti differenti)

16. Cosa sono i canali?

    Sono strutture simi alle porte ma più in generale sono astratte, usate nella comunicazione tra processi e non si fanno ipotesi sul numero di processi coinvolti o sulla direzione dei dati

17. Cos'è l'RPC?

    Si tratta di un ulteriore meccanismo di comunicazione che viene usato su una macchina remota solitamente (quindi sistemi distribuiti e client-server). Si invola la procedura su una macchina remota e la comunicazione avviene passando dei parametri o il risultato e viene attivato il processo all'arrivo della richiesta.