TODO

- Guardare CRT, CRT a pila nascosta e A-list
- Andare a prendere le soluzioni degli esercizi
- Re-binding?
- Nelle implementazioni di catena statica con display, i puntatori di catena statica normale rimangono?
- La CRT normale ha il bit di visibilità?

IMPORTANTE: x++ incrementa immediatamente finita la valutazione di x, quindi se x=2, print(x++ + x)
stampa 2+3 = 5

IMPORTANTE #2: A prescindere dallo scoping, la visibilità è “statica”! Oppure no?

Storia dei linguaggi

[Ha senso?]

Macchine astratte

Quali sono gli step del ciclo Fetch-Decode-Execute?

- Fetch: Lettura dell’istruzione
- Decode: Decodifica dell’istruzione
- Execute
    - Lettura valore degli operandi
    - Esecuzione dell’istruzione
    - Memorizzazione operandi

(Se si parla del ciclo Fetch-Execute, il Decode viene incorporato nel Fetch)

Quali sono i componenti di una macchina astratta?

- Memoria
- Interprete
- Implementazione delle operazioni

Cos’è la semantica di un programma?

Dato un programma P_r^L scritto in un linguaggio L, la semantica di P_r^L è una funzione parziale P^L tale che $P^L(input) = output$ dove $input$ e $output$ appartengono all’insieme dei dati D.

Qual è la definizione di interprete?

Un interprete per un linguaggio L scritto in L_0 è un programma la cui semantica è la funzione $I_L^{L_0} : (Prog^L \times D) \to D$, tale che
$I_L^{L_0}(P_r^L, input) = P^L(input)$

Qual è la definizione di compilatore?
Un compilatore da L a L_0 è un programma la cui semantica è la funzione $C_{L,L_0} : Prog^L \to Prog^{L_0}$ tale che, dato un programma P_r^L, se $C_{L,L_0}(P_r^L) = P_r c^{L_0}$, allora per ogni $input \in D$,
$P^L(input) = P_c^{L_0}(input)$

Come viene solitamente realizzata un'implementazione di tipo interpretativo?
- Compila l'interprete di un linguaggio intermedio nel linguaggio ospite
- Compila il programma nel linguaggio intermedio
- Esegui il programma sull'interprete

Come viene solitamente realizzata un'implementazione di tipo compilativo?
- Compila il supporto a runtime nel linguaggio ospite
- Compila il programma nel linguaggio ospite
- Il programma viene eseguito interagendo con il supporto a runtime

Nomi

Cos'è un nome?
È una sequenza di caratteri che denota qualcos'altro.

Cos'è l'ambiente?
È l'insieme delle associazioni tra i nomi e gli oggetti denotabili esistenti a runtime in un punto specifico del programma in un momento specifico dell'implementazione.

Cos'è una dichiarazione?
È un meccanismo (implicito o esplicito) con cui si crea un'associazione nell'ambiente.

Cos'è la portata (aka scope)?
È l'estensione della visibilità di un nome.

Cos'è il tempo di vita?
La durata di un nome.

Quando avviene il binding time?
- Statico
    - Progettazione del linguaggio: Tipi primitivi, nomi delle operazioni, costanti predefinite

    - Scrittura del programma: Definizione dei nomi (ma non vengono ancora legati)
    - Compilazione, collegamento e caricamento: Legame di alcuni nomi (es. funzioni e variabili globali)
- Dinamico: Legame di tutti i nomi ancora non legati (es. variabili locali)

Nota: Binding statico/dinamico =/= Scoping statico/dinamico

Cos'è un blocco?
È una regione testuale (identificata da un segnale di inizio e uno di fine) all'interno del quale possono essere definiti dei nomi locali a quella regione.

Quali operazioni possono essere eseguite su un ambiente?
- Creazione di associazione tra un nome e un oggetto denotato
- Riferimento a un oggetto mediante il suo nome
- Disattivazione di un'associazione
- Riattivazione di un'associazione
- Distruzione di un'associazione

Qual è la differenza tra scoping statico e dinamico?
- Scoping statico: Si parte dallo scope attuale e si cerca la prima dichiarazione, eventualmente passando a uno scope più esterno
- Scoping dinamico: Si cerca il primo record a partire dal RdA attuale e risalendo lo stack

Come si può aggiungere il supporto per definizioni ricorsive a un linguaggio?
Due opzioni:
- Permettere definizioni incomplete
- Permettere di usare una funzione/un tipo prima che sia definito

Gestione memoria

Qual è la differenza tra allocazione statica e dinamica della memoria?
- Statica: La memoria viene allocata a compile time
- Dinamica: La memoria viene allocata a runtime

Nota: Quindi non dipende solo da se sai in anticipo le dimensioni o no!

Cosa contiene un Record di Attivazione?
- Puntatore di catena dinamica
- Puntatore di catena statica (se si usa scoping statico)
- Indirizzo di ritorno
- Indirizzo del risultato
- Parametri
- Variabili locali
- Risultati intermedi

Cosa avviene durante l'ingresso in un blocco?

- Modifica il program counter
- Alloca il RdA
- Modifica il puntatore al RdA
- Passa i parametri
- Salva i registri
- Esegui eventuali inizializzazioni
- Trasferisci il controllo

Cosa avviene durante l'uscita da un blocco?

- Restituisci i valori
- Ripristina i registri allo stato precedente (incluso il puntatore al RdA)
- Esegui eventuali finalizzazioni
- Dealloca il RdA
- Ripristina il valore del program counter

Perché serve l'heap?

Perché se è possibile eseguire allocazioni dinamiche di memoria perché non si hanno garanzie sull'ordine di deallocazione.
Nota: in teoria non dovrebbero esserci problemi con il fatto che non si sa in anticipo di quanto allocare il RdA.

Come viene gestito l'heap?

Si usa una lista libera che contiene uno o più blocchi contigui. Quando un blocco viene allocato, lo si rimuove dalla lista, mentre quando viene deallocato lo si riaggiunge.

Quand'è che l'allocazione statica senza stack è sufficiente?

Quando non ci sono allocazioni dinamiche di memoria e non ci sono funzioni ricorsive.

Cosa indica k?

È un numero stabilito a compile time di quanti step della catena statica si devono risalire per calcolare il puntatore al RdA che verrà assegnato al puntatore di catena statica del nuovo RdA.

Cosa indica h?

È un numero stabilito a compile time di quanti step della catena statica si devono risalire per accedere a un nome. 0 significa che il nome è locale.

Come viene gestito l'heap in modo da supportare blocchi di dimensioni diverse?

- Buddy system
    - n liste
    - La k-esima lista contiene blocchi di dimensione 2^k
    - Se non ci sono blocchi di dimensione 2^k, si divide un blocco di dimensione 2^{k+1}
    - Se due blocchi di dimensione 2^k sono liberi, vengono uniti in un blocco di dimensione 2^k
- Fibonacci system: Come il Buddy system, ma usa i numeri di Fibonacci

A cosa serve il display?
È un array di dimensione pari al massimo livello di indentazione dove il k-esimo elemento punta all'ultimo RdA di un blocco di livello k aggiunto allo stack.

Quali sono le principali implementazioni dello scoping dinamico?
- Naive
- A-List
- CRT
- CRT con pila nascosta

Cos'è una A-List?
È una lista di tutte le variabili attive dove la variabile aggiunta per ultima è prima nella lista.

Cos'è una CRT?
È una struttura che associa ogni nome a una lista di tutte le variabili con quel nome, dove la variabile aggiunta per ultima è prima nella lista.

Cos'è una CRT con pila nascosta?
È una struttura che associa ogni nome alla variabile aggiunta per ultima con quel nome e a un bit che indica se quella variabile è al momento visibile. Vi è anche una "pila nascosta" che contiene tutte le variabili nascoste da altre variabili. La pila nascosta è tale che l'ultima variabile nascosta è la prima nella pila.

Strutturare il controllo

Quali sono i tipi di notazione?
- Infissa: L'operatore è tra gli operandi, es. a + b
- Prefissa: L'operatore è prima degli operandi, es. + a b
- Postfissa (aka Polacca inversa): L'operatore è dopo gli operandi, es. a b +

Quale notazione è preferibile?
Quella polacca inversa perché non è ambigua e può essere implementata direttamente con una macchina a stack.

Qual è la corrispondenza tra visita e notazione?
- Visita infissa (aka simmetrica): Notazione infissa
- Visita prefissa (aka anticipata): Notazione prefissa
- Visita postfissa (aka differita): Notazione postfissa

Perché l'ordine di esecuzione può influenzare il comportamento del programma?
- Side effects
- Problemi legati all'aritmetica finita
- Operatori non definiti

Qual è la differenza tra lazy evaluation e eager evaluation?

La lazy evaluation valuta solo le espressioni necessarie, mentre la eager evaluation valuta tutte le espressioni.
Nota: Per le operazioni booleane la lazy evaluation viene anche detta short circuit.

Qual è la definizione di comando?
Entità sintattica che non restituisce necessariamente un valore ma può avere dei side effects.

Quali distinzioni si possono fare tra espressioni e comandi?
- Expression oriented: tutto è un'espressione
- Distinzione tra espressioni e comandi
- Distinzione tra espressioni e comandi, ma con alcuni casi speciali (es. in C gli assegnamenti restituiscono un valore e si può usare un'espressione al posto di un comando)

Quali sono i comandi di controllo sequenza?
- Comandi di controllo sequenza esplicito
  - ;
  - Blocchi: Creano un comando composto
  - goto: Esegue un salto
- Comandi condizionali
  - if
  - case
- Comandi iterativi
  - for: In teoria esegue il corpo un numero determinato di volte
  - while: Esegue il corpo un numero indeterminato di volte

Quand'è che un comando è strutturato?
Quando ha un solo punto di ingresso e un solo punto di uscita.

Come viene implementato il case?
- Si calcola il valore v dell'espressione
- Se v è fuori dal bound, salta a L(n+1)
- Altrimenti salta a L0+v
- Dopo L0, ci sono i jump a L1, L2, … Ln
- Ogni Li (incluso L(n+1)) termina con un jump a dopo il case

Come viene implementato il for?
- Implementazione standard
  - Calcola inizio e fine
  - I = inizio
  - Se il passo è positivo: Se I > fine esci dal ciclo, altrimenti esegui il corpo e torna qui
  - Se il passo è negativo: Se I < fine esci dal ciclo, altrimenti esegui il corpo e torna qui
- Implementazione con iteration counter: calcoli $ic = \left\lfloor \frac{fine - inizio + passo}{passo} \right\rfloor$

- Altre implementazioni che permettono di eseguire iterazione non determinata

La ricorsione è equivalente all'iterazione?
Sì, il potere espressivo è lo stesso.

Quand'è che una chiamata è una tail call?
Se f chiama esegue una tail call di g, restituisce il valore di g senza ulteriore computazione

Quand'è che una funzione è tail recursive?
Quando contiene solo tail calls.

Come si può convertire una funzione ricorsiva normale in una funzione tail recursive?
Con il continuation passing style: [...]

Astrazione sul controllo

Quali sono i principali metodi di passaggio?
- Valore: Chiamante ⇒ Chiamato
- Riferimento: Chiamante ⇔ Chiamato
- Risultato: Chiamante ⇐ Chiamato
- Valore-Risultato: Chiamante ⇔ Chiamato
- Nome: Chiamante ⇔ Chiamato

Come viene implementato il passaggio per nome?
Viene passato sia l'espressione da valutare che il puntatore all'ambiente in cui valutarla.

Cos'è il downward funarg problem?
Consiste nel determinare in che ambiente va valutata una funzione passata come argomento.

Quali sono gli approcci principali al downward funarg problem?
- Shallow binding: La funzione viene eseguita nell'ambiente in cui viene chiamata
- Deep binding: La funzione viene eseguita nell'ambiente in cui è definita
    - Si passa una closure (chiusura) <code, env> dove env è l'ambiente in cui è definita la funzione
    - L'ambiente viene determinato a runtime nel momento del passaggio risalendo la catena statica di un tot di step (calcolati a compile time)

Cos'è l'upward funarg problem?
Consiste nel determinare in che ambiente va valutata una funzione restituita da un'altra funzione.

Il deep binding passa il puntatore all'RdA del chiamante o all'RdA dove è definita la funzione legata?
Passa il puntatore all'RdA dove è definita la funzione legata.

Le eccezioni vengono propagate lungo la catena statica o dinamica?
Lungo la catena dinamica.

Qual è l'implementazione naive delle eccezioni?
- All'inizio di un blocco protetto si mette il gestore sulla pila
- Quando un'eccezione viene sollevata, si rimuove il primo gestore e si guarda se è adeguato
- Se non lo è, si ripete

Tipi di dati

Cos'è un tipo?
Una collezione di valori (omogenei ed effettivamente rappresentati) dotato di un insieme di operazione per manipolarli.

Quali sono le regole per definire i tipi?
- Equivalenza: [...]
- Compatibilità: [...]
- Inferenza: [...]

Qual è la differenza tra valori esprimibili, denotabili e memorizzabili?
[...]

Quand'è che un linguaggio è type-safe?
Quando a runtime non possono avvenire errori non segnalati derivanti da un errore di tipo.

Può esistere un sistema di controllo statico perfetto?
No, perché sapere in anticipo il tipo del valore che verrà assegnato a una variabile è indecidibile (variante del problema dell'arresto).

Cos'è un tipo ordinale?
Un tipo dove ogni elemento ha un elemento precedente e uno successivo (tranne rispettivamente il primo e l'ultimo).

Cos'è un record?
Un tipo di dato che permette di manipolare in modo unitario dati eterogenei.

Cos'è un record variante?
Un tipo di record in cui alcuni campi sono alternativi. Solitamente contiene un tag (solitamente di tipo ordinale).

Cos'è un array?
Una collezione di dati omogenei. Più formalmente, è una funzione da un tipo indice (solitamente discreto) a un tipo degli elementi.

Quali sono le operazioni principali su un array?

- Selezione di un elemento
- Slicing (selezione di elementi contigui)

Come si accede a un array tridimensionale?
Dato un array $[l_1 \dots u_1, l_2 \dots u_2, l_3 \dots u_3]$ di posizione a, dove la dimensione di un array 2d è s_1, di un array 1d è s_2 e di un elemento è s_3, l'indirizzo dell'elemento (i, j, k) è $a + (i - l_1) \cdot s_1 + (j - l_2) \cdot s_2 + (k - l_3) \cdot s_3$.

Perché l'ordinamento row-first è favorevole se si itera riga per riga?
Perché quando si ha un cache miss viene caricato in cache l'intero blocco contiguo, quindi iterando riga per riga si hanno operazioni più efficienti.

Come può essere la forma di un array?
- Statica: Nota a compile time
- Fissata al momento di inizializzazione dell'array
    - Il RdA viene diviso in parte fissa e parte dinamica
    - La parte fissa contiene il dope vector, che contiene
        - Puntatore all'inizio dell'array
        - Numero di dimensioni
        - Limite inferiore (+ eventualmente limite superiore)
        - Occupazione per ogni dimensione (s_1, s_2...)
- Dinamica
    - Può cambiare forma dopo la creazione
    - Salvato sull'heap

Quali sono le operazioni di un puntatore?
- Dereferenziazione
- Allocazione di memoria
- Deallocazione di memoria

Cos'è una dangling reference?
Un riferimento a un oggetto non più valido.

Come funziona una tombstone?
Invece di puntare direttamente a un oggetto, i puntatori puntano a una tombstone, che contiene il puntatore all'oggetto vero solo se l'oggetto è ancora valido.

Come funziona il sistema locks and keys?
Ogni puntatore ha una chiave e ogni oggetto puntato ha un lucchetto. È possibile dereferenziare un oggetto solo se la chiave coincide con il lucchetto. Quando un oggetto viene deallocato, il suo lucchetto viene cancellato.

Come funziona la garbage collection con reference count?
Ogni oggetto ha un contatore di riferimenti a quell'oggetto. Ogni volta che una variabile inizia a fare riferimento a quell'oggetto, si incrementa il contatore. Se una variabile smette di fare riferimento all'oggetto, si decrementa il contatore. Quando il contatore è a 0, si dealloca l'oggetto. Problema: non identifica i riferimenti circolari.

Come funziona la garbage collection con mark and sweep?
Si marcano tutti gli oggetti dell'heap come unused. Poi a partire dai puntatori fuori dall'heap (nello stack e nei registri) si seguono tutti i puntatori e si marcano gli oggetti incontrati come used. Gli oggetti unused vengono deallocati. Il pointer reversal permette di evitare un utilizzo ulteriore di memoria per eseguire la ricerca in profondità. Problema: stop the world effect.

Da cosa viene descritto un sistema di tipi?
- Tipi predefiniti
- Tipi che si possono definire
- Quali valori sono
    - Esprimibili
    - Denotabili
    - Memorizzabili
- Regole sui tipi
    - Forme di controllo dei tipi
        - Statico
        - Dinamico
    - Meccanismi di controllo
        - Regole di equivalenza
        - Regole di compatibilità
        - Regole e tecniche di inferenza

Cos'è l'equivalenza?
[...]

Cos'è l'equivalenza per nome?
Due tipi sono equivalenti per nome solo se hanno lo stesso nome (quindi un tipo è equivalente solo a se stesso).

Cos'è l'equivalenza strutturale?
È la minima relazione di equivalenza che soddisfa le seguenti proprietà:
- Un nome di tipo è equivalente a se stesso
- Se un tipo T è introdotto come type T = espressione, allora T è equivalente a espressione
- Se due tipi sono costruiti applicando lo stesso costruttore di tipo a tipi equivalenti, allora essi sono equivalenti

Quand'è che due tipi sono compatibili?
T è compatibile con S se un valore di tipo T è ammesso in qualsiasi contesto in cui sarebbe ammesso un valore di tipo S.

Quali tipi di conversione di tipo ci sono?
- Conversione implicita (coercizione)
- Conversione esplicita (cast)

Come viene implementata la conversione di tipo?
- Coercizione

    - Se T e S hanno la stessa rappresentazione, la coercizione è solo sintattica
    - Se T e S hanno valori diversi ma la stessa rappresentazione nell'intersezione (es. int e intervalli), si esegue solamente un controllo dinamico di appartenenza all'intersezione
    - Se esiste un metodo di conversione da T a S, si applica quello
- Cast: Come per la coercizione

Cos'è un non converting type cast?
Un cast dove si cambia il tipo di una variabile ma non la sua rappresentazione in memoria.
Nota: molto pericoloso.

Cos'è l'inferenza di tipo?
Il processo di attribuzione di un tipo a un'espressione nella quale non appaiono dichiarazioni esplicite di tipo dei suoi componenti.

Quand'è che un sistema di tipi è polimorfo?
Quando uno stesso oggetto può avere più di un tipo.

Quand'è che un oggetto è polimorfo?
Quando ha più di un tipo.

Quand'è che un nome è overloaded?
Quando corrispondono più oggetti a quel nome (in numero finito). Es.: + per indicare sia somma tra interi che somma tra float.

Quand'è che un valore ha polimorfismo parametrico?
Quando ha un'infinità di tipi diversi che si ottengono per istanziazione da un unico schema di tipo generale.
Nel polimorfismo di parametrico non si usano le informazioni di tipo.

Quand'è che un valore ha polimorfismo di sottotipo?
Quando ha un'infinità di tipi diversi che si ottengono per istanziazione da un unico schema di tipo generale, sostituendo a un opportuno parametro i sottotipi di un tipo assegnato.
I generics "generali" hanno polimorfismo parametrico, i generics che ereditano da un tipo hanno polimorfismo di sottotipo.

Paradigma Orientato agli Oggetti

Cos'è un tipo di dato astratto?
Un tipo di dato che può essere descritto e manipolato considerando esclusivamente la sua interfaccia. [Definizione non precisa]

Quali sono i componenti principali di Java?
- Compilatore: Converte il codice Java in bytecode
- Loader: Carica i file .class (eventualmente da internet)

- Linker: Aggiunge le classi e le interfacce compilate al runtime system
- Verifier: Controlla la correttezza del bytecode
- Interprete
    - Esegue il bytecode
    - Esegue controlli runtime
    - Può compilare in codice nativo
    - Può chiamare metodi nativi

Quali sono gli step della metodologia OO?
- Identificare quali sono gli oggetti a un certo livello di astrazione
- Identificare la semantica degli oggetti
- Identificare le relazioni tra gli oggetti
- Implementare gli oggetti

Quali sono i modificatori di visibilità in Java?
- Public: Visibile ovunque
- Private: Visibile solo nella classe
- Protected: Visibile nella classe e nelle sue sottoclassi (e secondo Oracle anche nel package)
- Default (Package): Visibile nel codice contenuto nel package)

Quali sono i tre tipi "fondamentali" di Java?
- Object, da cui ereditano
    - Object[]
    - Throwable

Cos'è l'array covariance?

[Capire bene]

Che cos'è l'ereditarietà?
[...]

Qual è la differenza tra shadowing e overriding?
- Shadowing
    - Usato per le variabili
    - Vengono nascoste, ma sono accessibili nel loro contesto o specificando che si sta facendo riferimento alla variabile della superclasse (ricorda: super.super non è ammesso)
- Overriding
    - Usato per i metodi
    - I metodi della superclasse overridati non vengono nascosti, vengono proprio rimpiazzati dalla loro versione nuova. Ciò significa che il codice della superclasse anche se crede di chiamare metodi della superclasse chiamerà i metodi della sottoclasse

Che particolarità ha una classe astratta?
- Contiene metodi astratti (senza corpo)

- Non può essere istanziata
- Se una sottoclasse implementa tutti i metodi astratti può essere istanziata

[Qualcosa su lookup dinamico]

Cos'è il problema della classe fragile?
Se si cambia la vtable di una superclasse, bisogna modificare la vtable di tutte le sottoclassi

Come si risolve il problema della classe fragile?
Bytecode rewriting: si calcolano gli offset a runtime la prima volta che sono necessari e si riscrivono le istruzioni inserendo l'offset.

Programmazione Concorrente

Cosa significa concurrent?
"Che avviene o viene eseguito allo stesso tempo".

Cos'è la programmazione concorrente?
Un tipo di programmazione in cui ci sono più thread/processi/programmi attivi contemporaneamente.

Qual è la differenza tra concorrenza fisica e logica?
Nella concorrenza fisica, vengono eseguiti simultaneamente più thread. Nella concorrenza logica, a livello di linguaggio di programmazione viene percepita l'esecuzione simultanea di più thread (ma non necessariamente a livello fisico).
A livello del linguaggio non c'è una differenza.

Qual è la differenza tra programmazione multithreaded e programmazione distribuita?
Nella programmazione multithreaded più thread vengono eseguiti sulla stessa macchina fisica (con memoria condivisa). Nella programmazione distribuita vengono eseguiti su macchine realizzate da strutture fisiche distribuite (no memoria condivisa).

Cos'è il service oriented computing?
È un paradigma di programmazione basato sulla composizione di servizi che interagiscono tra di loro.

Cos'è il cloud computing?
Un insieme di tecnologie che permettono di sviluppare applicazioni distribuite su scala planetaria, acquistando servizi elementari (CPU, Spazio disco…) e accedendo via internet.

Quali sono i meccanismi di comunicazione?
- Memoria condivisa: Lettura/Scrittura in una zona comune di memoria
- Scambio di messaggi: Primitive di invio e di ricezione
- Blackboard: Modello intermedio con send/receive a una zona comune di memoria

Quali sono le primitive di comunicazione?

- Send
    - Aggiunge un messaggio alla coda del canale
    - Bloccante solo in comunicazione sincrona
- Receive
    - Preleva un messaggio dalla coda del canale
    - Bloccante in entrambi i tipi di comunicazione

Quali sono i tipi di comunicazione?

- Sincrona: Avviene logicamente allo stesso tempo
    - Send bloccante
    - Receive bloccante
- Asincrona: Avviene in momenti diversi
    - Send non bloccante
    - Receive bloccante

Quali sono i meccanismi di sincronizzazione?

- Mutua esclusione: Solo un processo alla volta può entrare nelle regioni critiche del codice
- Sincronizzazione su condizione: Si sospende l'esecuzione di un processo fino al verificarsi di una condizione

Come vengono implementati i meccanismi di sincronizzazione?

- Attesa attiva
    - Tipi
        - Busy waiting
        - Spinning
    - Richiede l'utilizzo di operazioni atomiche come il Test-and-Set
    - Problemi di fairness
    - Problemi legati all'accesso in memoria dovuto ai bus
- Sincronizzazione con scheduler
    - Meccanismi strutturati
        - Semafori
        - Monitor
    - Gestiti dallo scheduler di processi (del kernel)
    - Un processo sospeso rilascia la CPU

Da cosa è costituito un semaforo?

- L'insieme dei numeri interi >= 0
- Due operazioni atomiche
    - P(s): Accesso al semaforo
        - Se s > 0, s--
        - Se s == 0, sospendi
    - V(s): Rilascio del semaforo
        - s++

Cos'è un monitor?

Un tipo di dato che incapsula un oggetto con stato e fornisce operazioni per agire sull'oggetto. Contiene:

- Variabili permanenti
- Procedure
- Init
- Dall'esterno sono visibili solo nomi e signature delle procedure

Permette di realizzare la sincronizzazione condizionale grazie a:

- Wait(variabile condizionale)
- Signal(variabile condizionale)

Quali sono i meccanismi di gestione delle risposte?

- Remote Procedure Call (RPC): La ricezione del messaggio causa la creazione di un processo per gestire la risposta
- Rendez-vous: Il destinatario ha già in esecuzione un processo attivo per gestire la risposta

In cosa consiste il nondeterminismo?

Quando un programma deve scegliere quale azione compiere (es. sono soddisfatte diverse condizioni che portano all'esecuzione di comandi diversi), lo fa senza uno schema prevedibile.

Perché è importante avere il nondeterminismo?

Per evitare deadlock e starvation.