delete karna all at compile time.

Recap of Lecture

Who ~~divides~~ cleans up the Heap ?

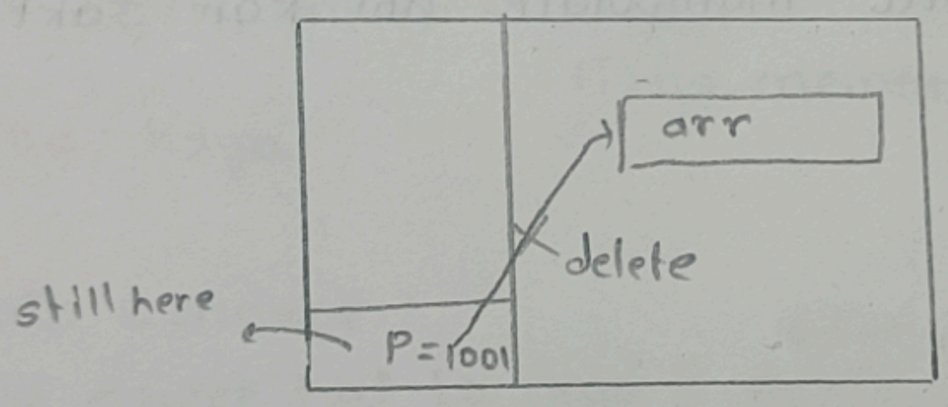
Approach -1] Manual Memory Management (C/C++)

- You are responsible for assigning memory in heap (new) and you are responsible for explicitly freeing it when you're done (delete).

- Increases Performance , fastest code run

- Problem :
  1. Forgetting to delete
  2. Dangling pointers : Freeing memory (delete) but still having a pointer to it. Agar pointer use kiya toh invalid memory access error

⇒ Hum jab delete kar dete hai memory from heap , still uska reference stack mein store hota hai and stack se sirf program/ func complete hone ke baad hi nikalta hai agar humne recall/reuse kiya woh ptr toh error aata hai



3. Double Free : Trying to delete same memory twice

Approach-2] Garbage Collector (Java , JS , Python)

- Create as many object you want and you need not to worry about deleting them. A separate process , Garbage collector runs in background.

- GC periodically pauses your program and scans heap to check for no longer being referenced objects

- Problem
  1. Performance hinder
  2. include lags
  3. slow

Need of Rust : The Third Way

⊙ Rust was born from one principle : Can we get both of these options

=> Is it possible to achieve the guaranteed memory safety of a garbage-collector language with performance of C and C++.

Ans: Yes by Rust's concept of Ownership and Borrowing

Do Rust has Garbage Collector?

=> No, coz it doesn't need one. Instead it has a very strict "compiler" that analyzes your code and proves at compile time that our program is memory-safe

=> That means, yeh compile time par hi sab analyse karta hai ki konse variable ki ownership transfer hui, konse variable ko delete karna all at compile time.

=> Suppose humne memory allocate ki heap mein - a = "sakshi" and then b = a kiya toh ab a print kiya toh error ayenga coz ownership transfer to b.

=> This only works for data stored in heap and not stack i.e. let b = 10 ✓
a = b ✓ } a & b both will have individual value as 10.

=> Another Doubt : How will it understand ki a ki ownership goyi so that is stored in separate table created by Rust at runtime.

⊙ Rust gives us -
↳ Speed of C++
↳ Safety of Python

And this combination makes it choice for Solana.

-Ru. #Recap of Lecture

Variables and Mutability

↳detail Notes in Notion

-

R

Sa