Bar is function so it’s type is T1 -> T2

Zar T3 -> T4

Cell is : real -> interger => T3 = real

+ ap dung integer => T4 = interger

* Zar: real -> interger

Vi x la integer => T1 = integer => T2 = T4 = integer

* Bar: integer -> integer

T1 -> T2

T1 = ( T3 \* T4 \* T5) - > T2

Map: T6 -> T7

T6 = (T8-> T9) \* List[T8]

T7 = List[T9]

T8 = T3

T8-> T9 = T5

List[T8] = T4

(T8\*List[T8]\*(T8->t9)) -> List[T9]

(T\*List[T]\*(T->T1))-> List[T1]

Logo

Description automatically generated with low confidence

T1 -> T2

T1 = (t3 \* t4 \*t5)

T4 : T6 -> T7

X == f(x) => T3 = T6 = T7

H(x) = f(x) => T3

* (T\* (T -> T) \* (T -> T) ) -> T

Main: a // 1, b // 1, c//1, sub1

Sub1: a // 2, b //1, c//1, sub1, sub2, sub3

Sub2: a // 3, b //1 , c //3, f // 3, sub1, sub2, sub3

Sub3: a//2, b//4, c// 1, sub1, sub2, sub3

Text, letter

Description automatically generated

Goi dia chi lenh goi sub1, sub2, f(a), f(c) lan luot la I1, I2 I3 I4

R1: Main{ip = , ep = , scp = , a = 0, b = 1, c = 5}

R2: sub1 {ip= I1 , ep= R1 , scp = R1 , a = 5 }

R3: sub2 {ip= I2 , ep= R2 , scp = R1 , a = 2, c = 3, f - > code sub3} -> ( -5 –10) + 2 = 7

~~R4: sub3 {ip= I3, ep= R3 , scp = R2, b = 2 } -> 5 – 2 \* 5 = -5~~

~~R5: sub3 {ip= I4, ep= R3 , scp = R2, b = 3 } -> 5 – 3\* 5 = -10~~

= > b = 7

Graphical user interface, text, application, email

Description automatically generated

1. Tri ket qua:

A[0], j =0 sum ( 5, 0)

S = 5 + 5 + 5 = 15

A[0] = 5 – 1 -1 -1 = 2

* 15 2 9 41

1. Tham Khao:

S = 5 + 4 + 3 = 12

A[0] = 5 – 1 -1 -1 =2

12 2 9 41

1. Name:

S = 5 + 9 + 41 = 55

A[0] = 5 -1 = 4

A[1] = 8

A[2] = 40

* 55 4 8 40

5)

Dangling pointer will happen if a pointer pointing to data that not exist anymore. To prevent this error, we have two ways to resolve:

* Lock and key:

Diagram

Description automatically generated

When memory is allocated, space for one more cell called lock cell is also allocated.

The pointer, which points to this memory have both address and key of this memory. A pointer is allowed to access the memory if the value of the lock and keys match. Otherwise, it will be throwed errors out immediately.

* Tombstome:

Diagram

Description automatically generated

As soon as memory is allocated, it is assigned to a pointer called the tombstome. The pointer in our program, which are supposed to point to this memory, point to the tomstome instead.

When this memory is deallocated, the value of tombstone will be assigned to Null, and when we try to access this memory \*(\*Null), so it will be throwed errors.

Diagram

Description automatically generated

7)

8)

* Simple call return: khong cho phép gọi đệ quy, lệnh gọi tường minh rõ ràng, điểm vào chương trình chỉ có một điểm, thực hiện lệnh gọi một cách tức thời, thực thi đơn ( chương trình a gọi chương trình b-> chuyển qua chương trình B chạy).
* De quy:

Đệ quy trực tiếp và gián tiêps

Những đặc tính khác is the same with simple call return

* Bien co(exception)

Không có lện gọi tường minh, ví dụ( xử lí event khi bấm chuột

Error handler( try catch)

Requires:

Những biến cố nào đã được định nghĩa và làmthế nào để họ định nghia một biến cố

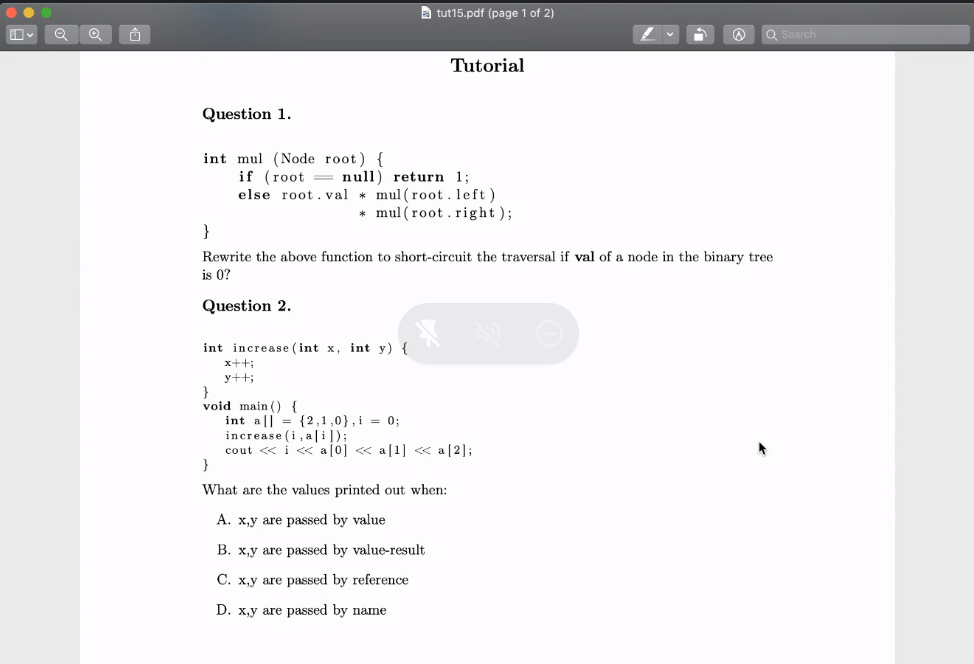
Cách làm sao để raise 1 biến cố

Làm cách nào để một biến cố có thể được handle

* Coroutine: Khác với 1 entry, trình cộng hành có nhiều điểm vào trong một chương trình.
* Task: có thể thực hiện dồng thời vs nhiều task khác, run trên multiple processor, chạy song song chuyển qua chuyển lại lẫn nhau theo cơ chế time sharing( khác vs couroutine – được chuyển dựa theo ý của người lập trình). Tuy nhiên xảy ra các vấn đề về deadlock, race condition.
* Scheduled subroutine:

Khi gọi có thể không thực hiên trực tiếp, thực hiện theo định thời( runtime) hoặc định thời theo độ ưu tiên

Được điều khiển theo một cái schedule, được ứng dụng trong lập trình real time



1)

int mul(Node root){

try{

return mul\_exception(root)

}

catch(int x){

return 0;

}

}

int mul\_exception(Node root){

if(root == null) return 1;

else if(root.val == 0) throw 0;

else return root.val\* mul(root.left)\*mul(root.right);

}