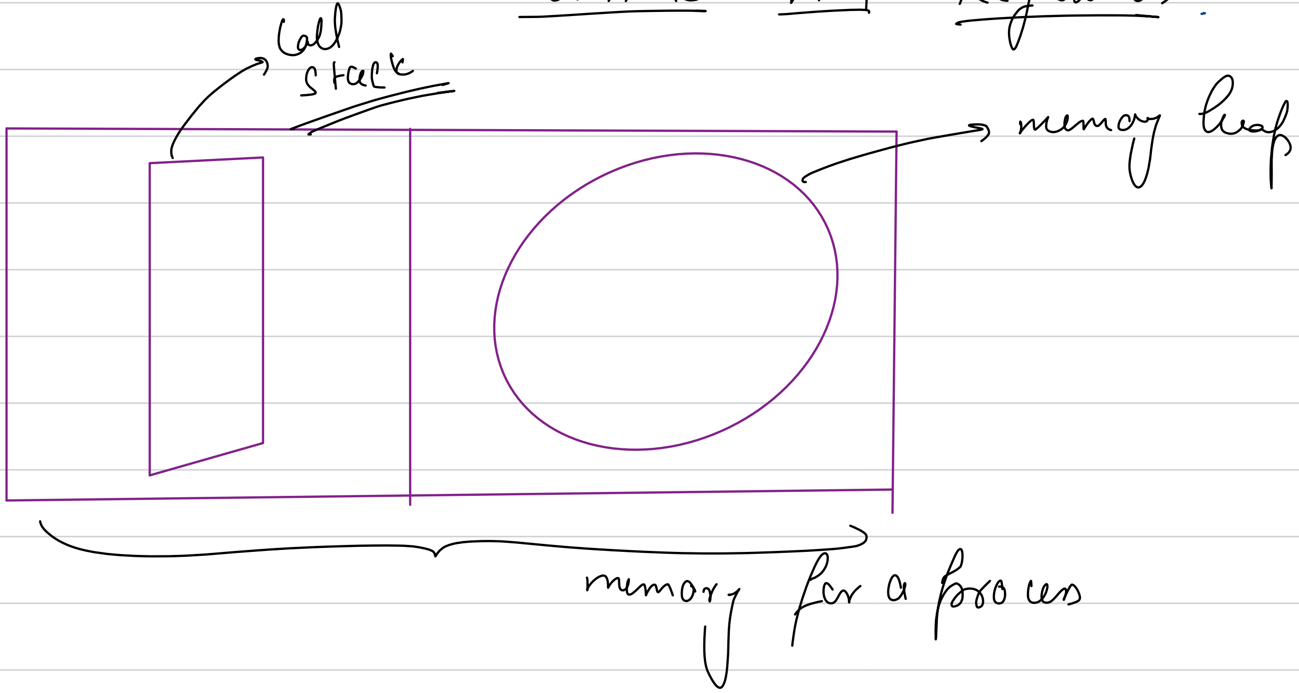
Pointers And References

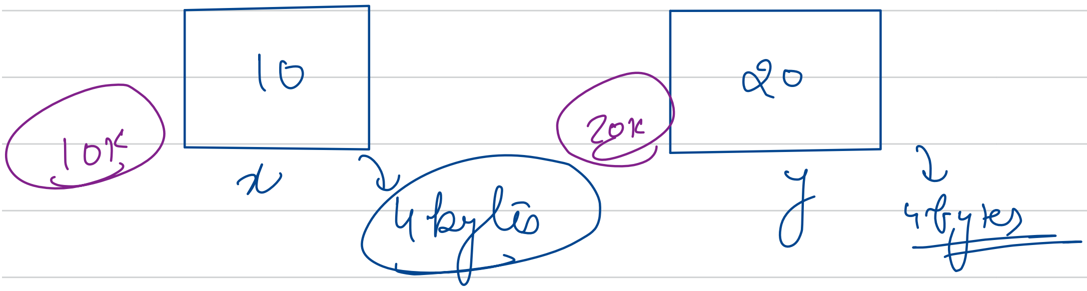


label value

int x = 10 ; int y = 20 ; float z = 3.14 ;

OS
microprocessor &
microcontroller

STLD



unique location → address → hexadecimal value

pointers → these are variables that stored memory address.

boolean variable → bool <var-name> = <value>;

pointer → we can specifically store address of a certain known type.

it denotes init of a new pointer var ← $\star$ → this will act differently in LHS & RHS

<var-type> $\star$ <ptr-name> = <address>;

$\underbrace{\text{<ptr-name>}}$ ↓ all rules for variable naming

int $\star$ → stores address of only int variable
bool $\star$ → " " " " bool "

'&' operator → &x → returns address of bucket x.
→ RHS

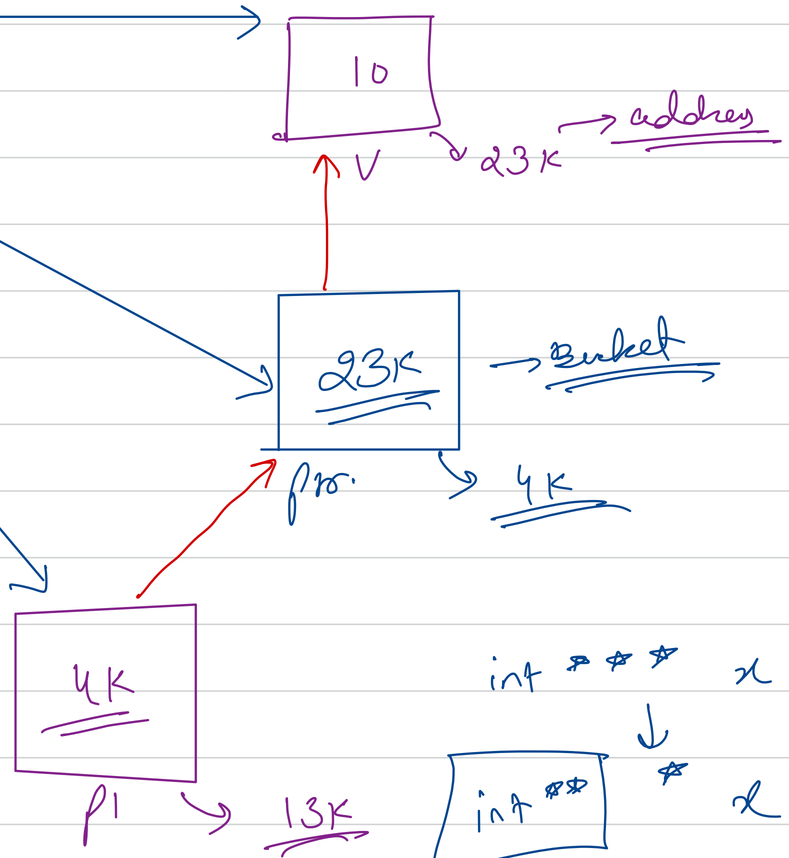
int v = 10

→ int * ptr = &v;

int ** p1 = &ptr;

↓

This is a pointer that points to integer pointer. (pointer to a pointer)



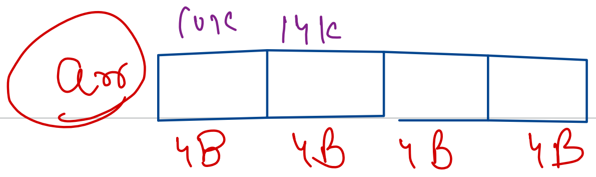
int *** x

↓

$\boxed{\text{int **}}$ * x

int ** p $\xrightarrow{\text{read as}}$ $\boxed{\text{int *}}$ * p

p points to what kind of bucket



→ array → int

int x = 10;
x += 4
x → 14

Pointer Arithmetic → (++) (--) (+x) (+= x)

(-x) (-= x) (difference b/w ptrs)

Name of the array is a pointer to the 0th index element.

Now we will introduce dereferencing operators.

$$\downarrow$$

$(*)$

```
int x = 10;

int* p = &x;

cout << (*p) << endl;
```

References

It is an alternative name for existing variable.

born has label of name bullet

int x = 10;

Reference → ref

x = 11

p = &x

p' = &x

fun(int* x)

int & r = x;

$$\begin{tikzcd} x \arrow[r] \arrow[d] & x' \arrow[d] \\ 10 & 10 \end{tikzcd}$$

$$\underbrace{\text{Pass By value / copy}}_{\downarrow}$$

primitive

$$\underbrace{\text{Pass by address / reference}}$$

generally non primitive

int x = 10 ;

address → &x RHS

int * p = &x ;

*p RHS