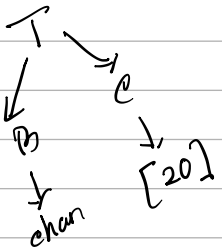
17/12/24 TUESDAY

```
int x;
float y;
Student k;
```
↳ instance of student type

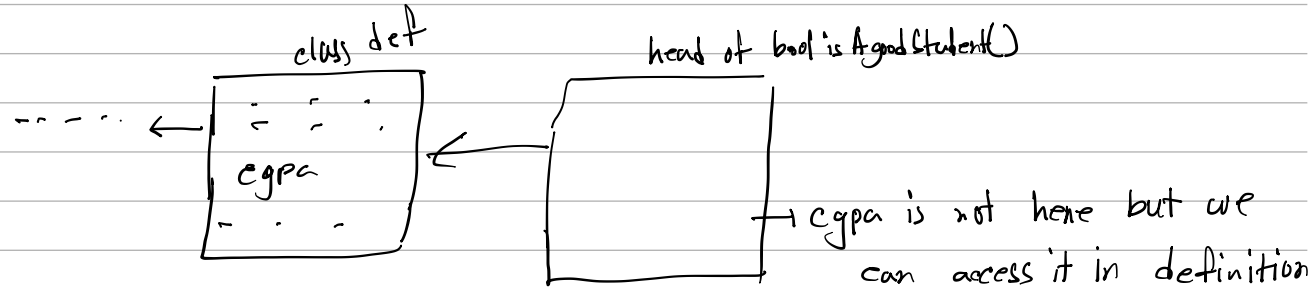
```
class Student{
int id;
float cgpa;
int batch;
string name;
boolean isAgoodStudent(){
return cgpa>2;
}
}
```

```
record{
int id;
float cgpa;
int batch;
char [20] name;
}
```



student1, student2, student3 ....;
↳ instances.

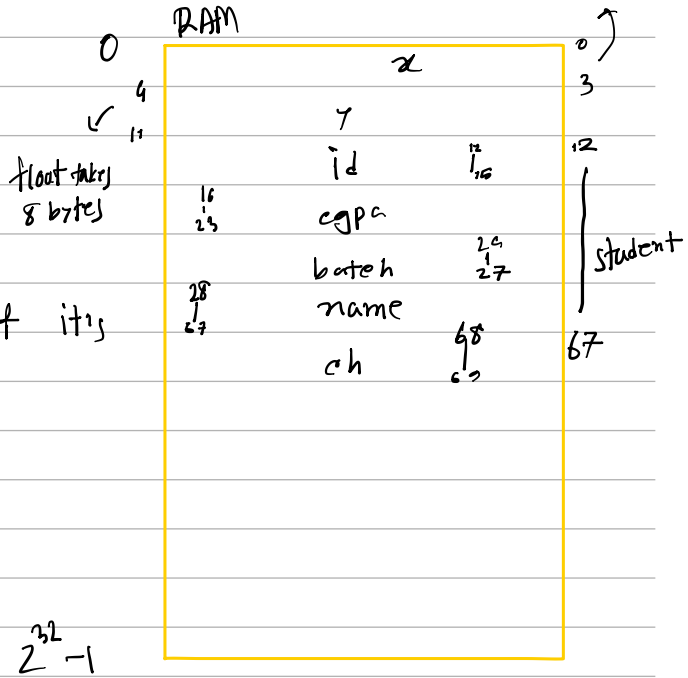
∴ Each record will have a scope ∴ symbol table for a record.



That is why inside class we don't need obj to get symbol.
but outside class we need.

int takes 4 bytes: 4X8=32 bits

TLB does the mapping from logical to physical address.



Size of record = sum of size/width of its variables/properties.

record of

4 ← int id ;

8 ← float cgpa ;

4 ← int batch ;

40 ← char [20] name ;

2 each

} student

2 ← char ch ;

x = student.batch + student.id ;

add $R1, $zero, 12 → student

lw $s1, $R1 [12] ⇒ student.batch

lw $s2, $R1 [0] → student.id

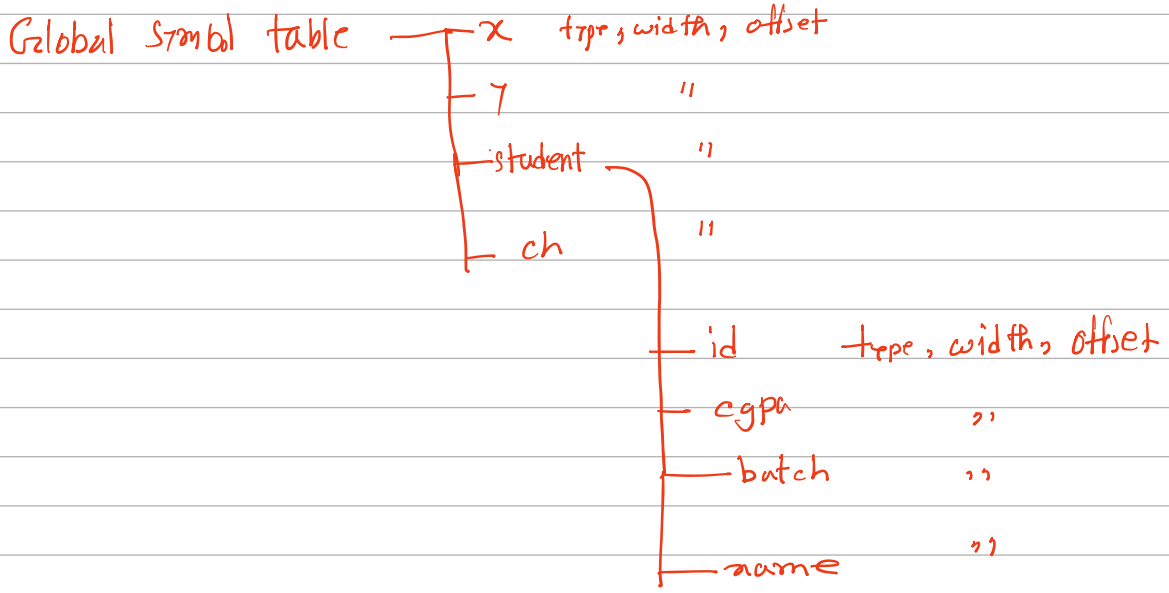
destination → $s2

add $s3, $s1, $s2

add $R1, $zero, 0 → x

sw $s3, $R1 [0] → x ← $s3

source → $s3



$P \rightarrow D$

$D \rightarrow T\ id\ ;\ D$

$D \rightarrow \varepsilon$

$T \rightarrow BC$ → Basic → supported directly by hardware
→ Complex → built upon the

$B \rightarrow int$

$B \rightarrow float$

$B \rightarrow char$

$T \rightarrow record\ \{\ D\ \}$

$C \rightarrow [num]\ C$

$C \rightarrow \varepsilon$

* Bias is introduced to avoid floating point operations.

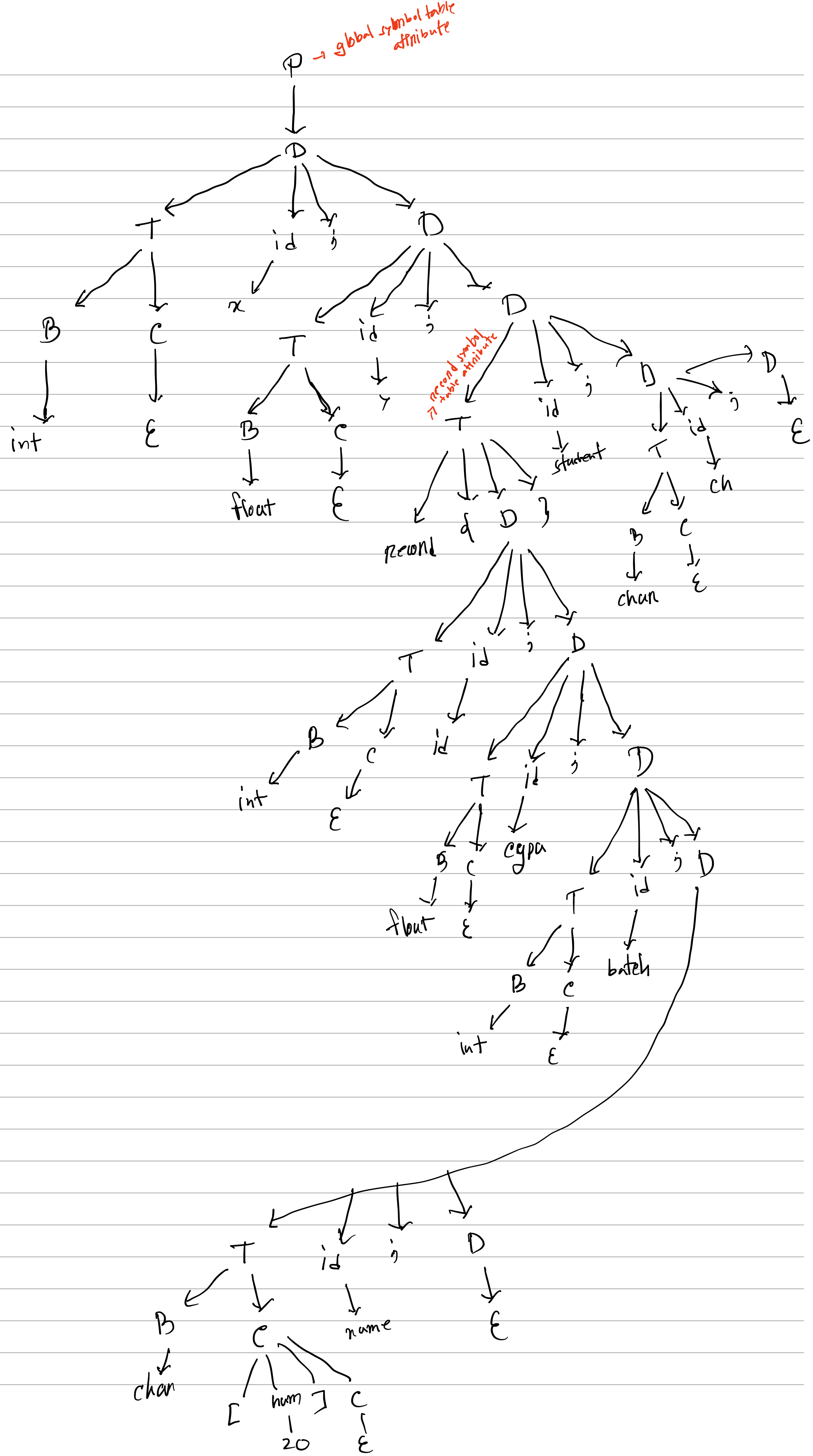
if (x > 3.7)

{ we have to do substraction to compare.

To avoid complexity bias can treat floats as integers.

since int operations are less costly.

```
int x ;
float y ;
record {
    int id ;
    float cgpa ;
    int batch ;
    char [20] name ;
} student
char ch ;
```


P → global symbol table attribute
record symbol table attribute
D
T
id
;
B
C
int
ε
x
float
y
record
{
}
student
char
ch
cgpa
batch
name
[
num
]
20

$P \rightarrow$ {of = 0 ; tab = newTable() ; } D {P.table = tab}

$D \rightarrow \varepsilon$

$D \rightarrow T$ id ; {tab.insert(id.lexeme, {T.type, T.width, of}) ; of = of + T.width;} D_1

(key: id.lexeme; values: {T.type, T.width, of})

$T \rightarrow B$ C {T.type = C.type, T.width = C.width}

$B \rightarrow$ int {t = int ; w = 4}

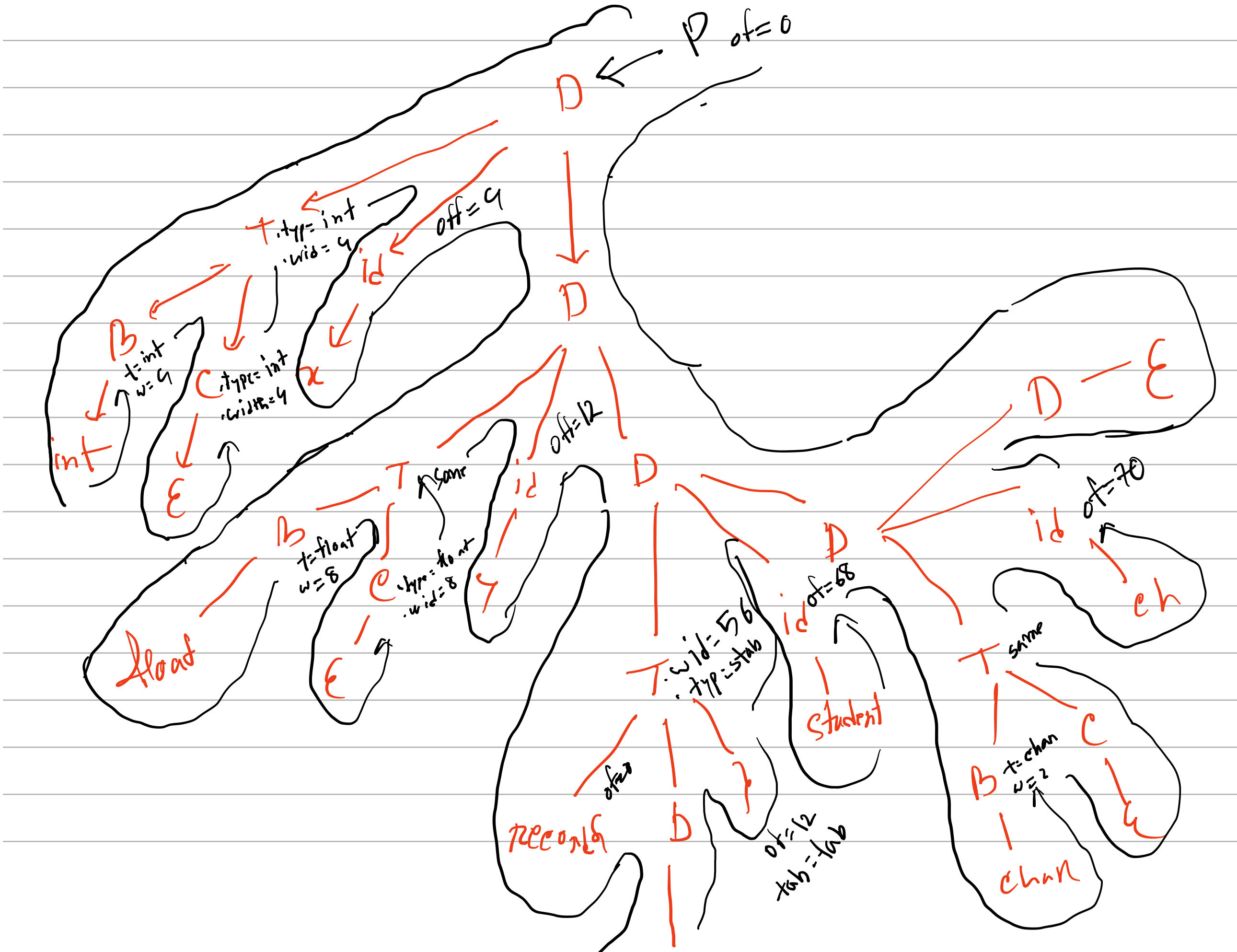
$B \rightarrow$ float {t = float, w = 8}

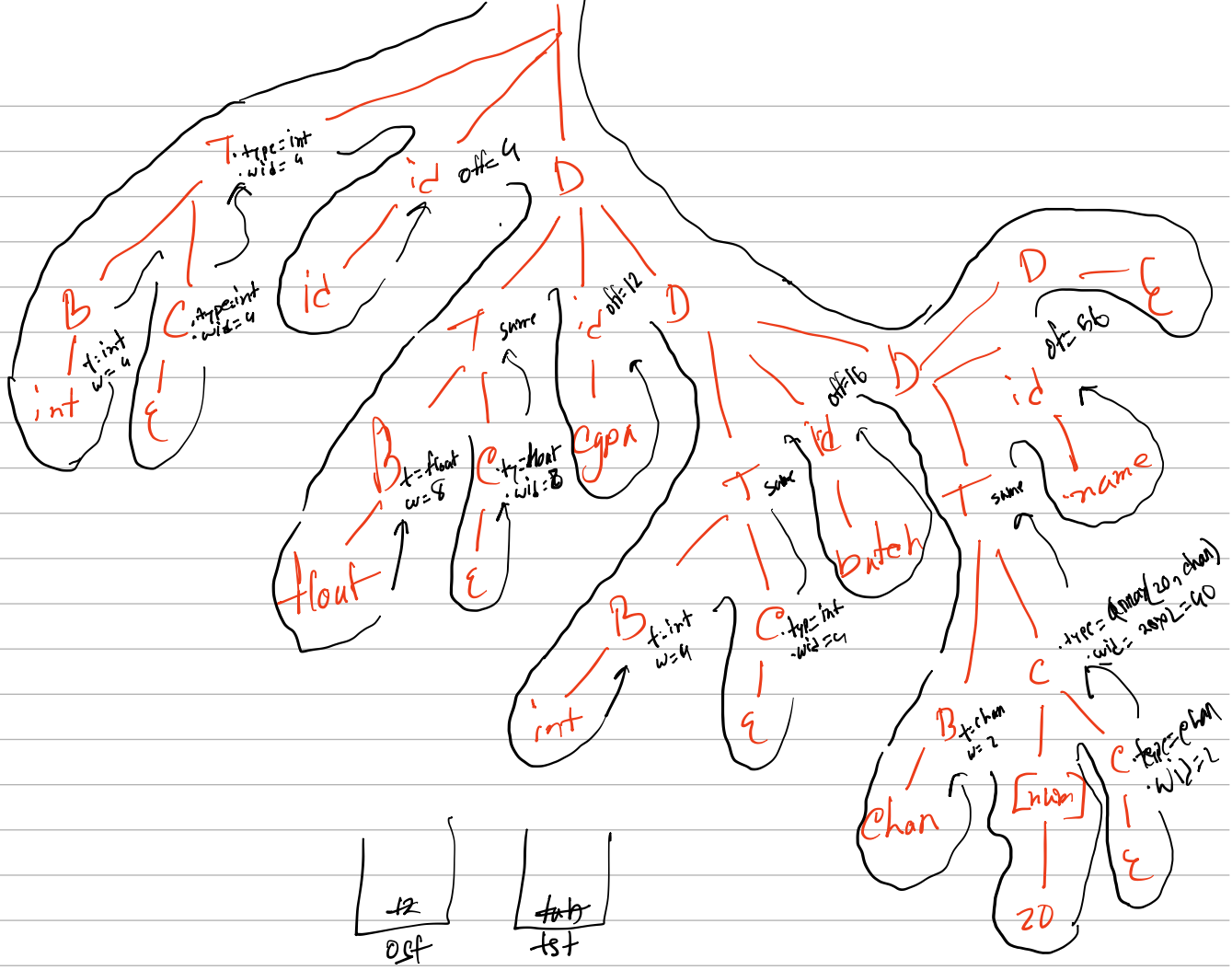
$C \rightarrow$ [num] C_1 {C.type = array(num.value, C_1.type); C.width = num.value × C_1.width;}

$T \rightarrow$ record { {ost.push(of); tst.push(tab); of = 0 ; tab = new Table();} D } {T.width = of ; T.type = tab ; of = ost.pop(); tab = tst.pop()}

$C \rightarrow \varepsilon$ {C.type = t, C.width = w}

```
int x ;
float y ;
record {
    int id ;
    float cgpa ;
    int batch ;
    char [20] name ;
} student
char ch ;
```





tab

ptab

a	int	4	0
f	float	8	4
student	stab	56	12
ch	chan	2	68

stab

id	int	4	0
cgpa	float	8	4
batch	int	4	12
name	array(20,chan)	40	16