Compilers

Report for the Lecture of 11th November 2013

Submitted by- Shubham Gupta 11CS30035

Group No 42

In this scribe, we discuss the **declaration and usage of array elements** ie we formulate the SDD and also give the semantic actions to find out the various attributes.

We use the following **naming convention**:

int a[20] is written as array(20,int)

So int a[20][25] is written as array(20,array(25,int))

We first give the grammar rules and the semantic actions associated with each production.

Now we write the Grammar

P->D

D->T id ; D

T -> BC

D-> epsilon

B -> int

B-> float

C -> [num]C

C -> epsilon

Now we write the SDT ie the syntax directed translation for this language

T -> B { t= B.type , w=B.width } C { T1.type= C.type , T.width=C.width }

B-> int {B.type=int, B.width=2}

B->float {B.type=float, B.width=4}

C-> epsilon { C.type=t, C.width=w}

C-> [num]C1 { C.type=array(num.val, C1.type) , C.width=C1.width*num.val }

In symbol table we store the type as array(20,array(25,int)).

Another point to keep in mind is the 3-Address codes which are valid.

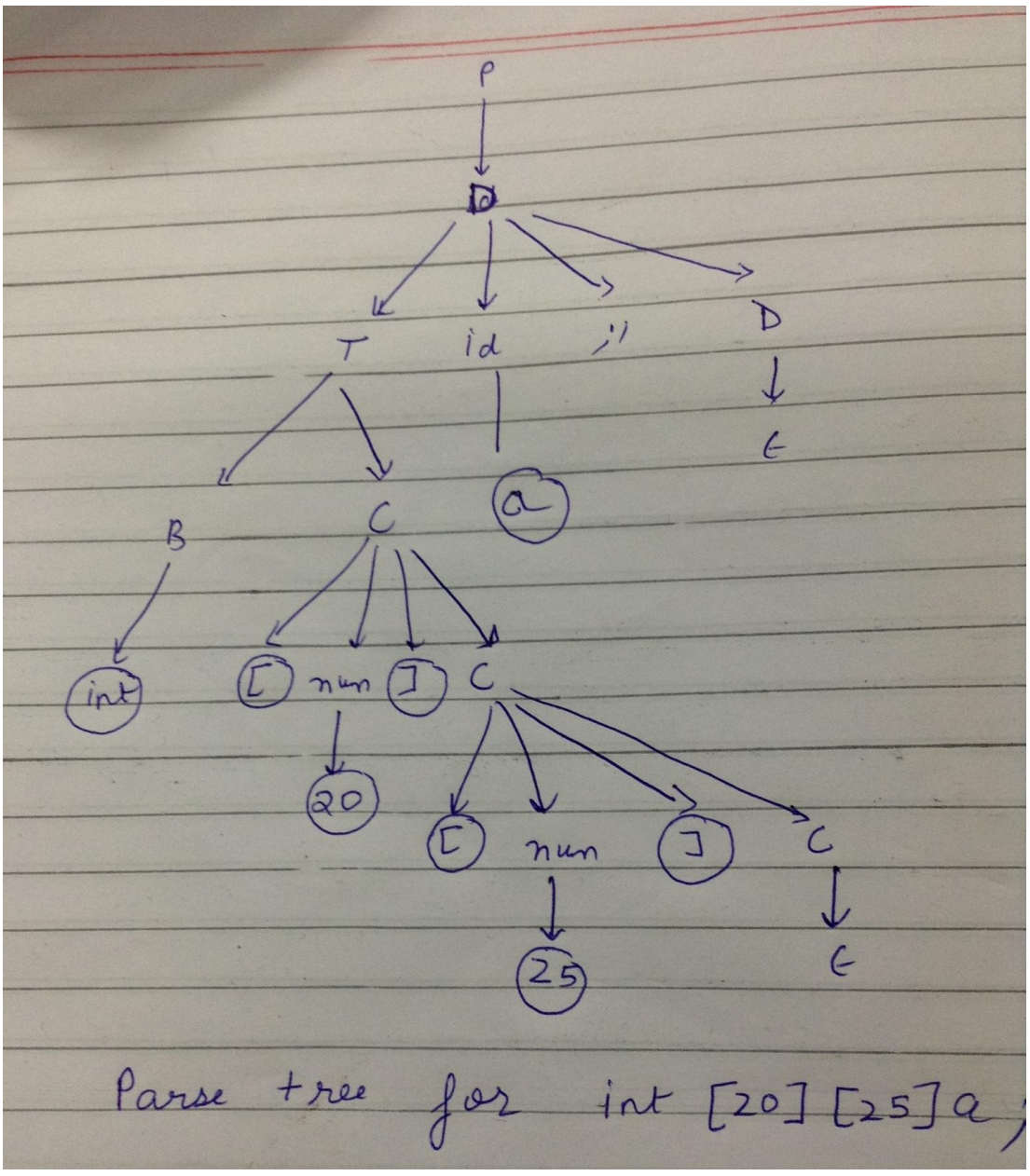
So x=y[i] is a valid 3-address code and so is y[i]=x .

Determining the address of an array entry a[i][j]

The address is t= base+ i*width (array(4 ,int)) + j*width(int)

Where base corresponds to the address a[0][0]

Now we **illustrate** how the parse tree for the expression int [20][25] a; is generated.



Parse tree for int [20] [25] a;

Let us consider another grammar with the following productions:

S-> id=E

E-> id

E -> L

L -> id[E]

L->L1[E]

Now we **add semantic actions to the grammar** and obtain the SDT:

S-> id=E{ gen(id= E.addr) }

E-> id {E.addr=id.val}

E -> L { E.addr=new temp gen{ E.addr=L.array_base[L.addr] } }

L -> id[E] { L.array=id.val, L.type=L.array.type.elem
L.addr= new Temp gen(L.addr= E.addr * L.type.width) }

L- L1[E]{ L.array=L1.array
L.type=L1.tye.elem
t= new temp gen(t=E.addr *L.type.width)
L.addr= new Temp (gen(L.addr=L1.addr))

Now we give the parse tree for it.