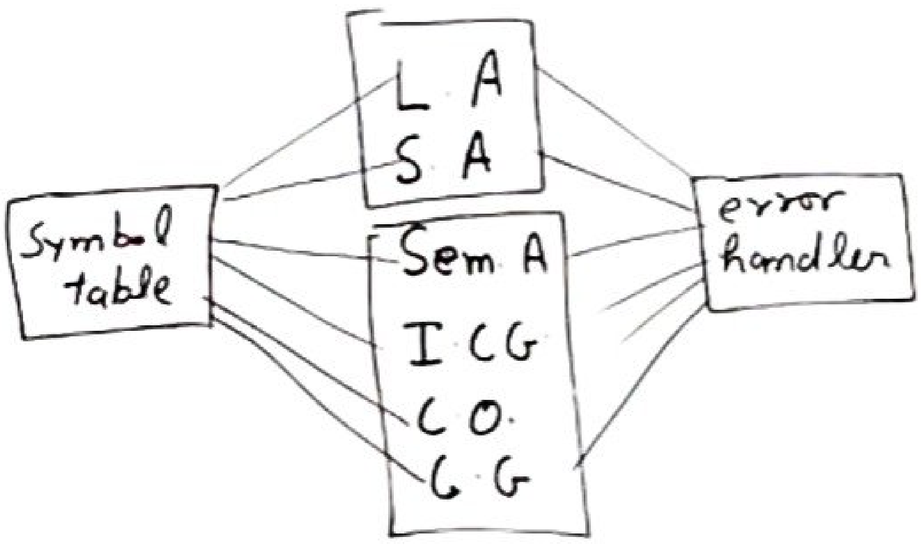
Symbol Table

Symbol table is a data structure created and ~~and~~ maintained by compiler in order to store Information about variables, function, class, object etc



Formate of Symbol table. Compiler uses following type Information in symbol table. 1. Data type, 2 Name, 3 Scope 4 Address 5. other attribute

e.g static int a;
float b;

S N	Name	Type	Attribute
1.	a	int	static
2.	b	float	–

Symbol table Representation

1. Fixed length
2. Variable length

Example int calculate;
int sum;
int a, b;

Name									Type
c	a	l	c	u	l	a	t	e	int
s	u	m							int
a									int
b									int

Fixed

Starting index	length	type
0	10	int
10	4	int
14	2	int
16	2	int

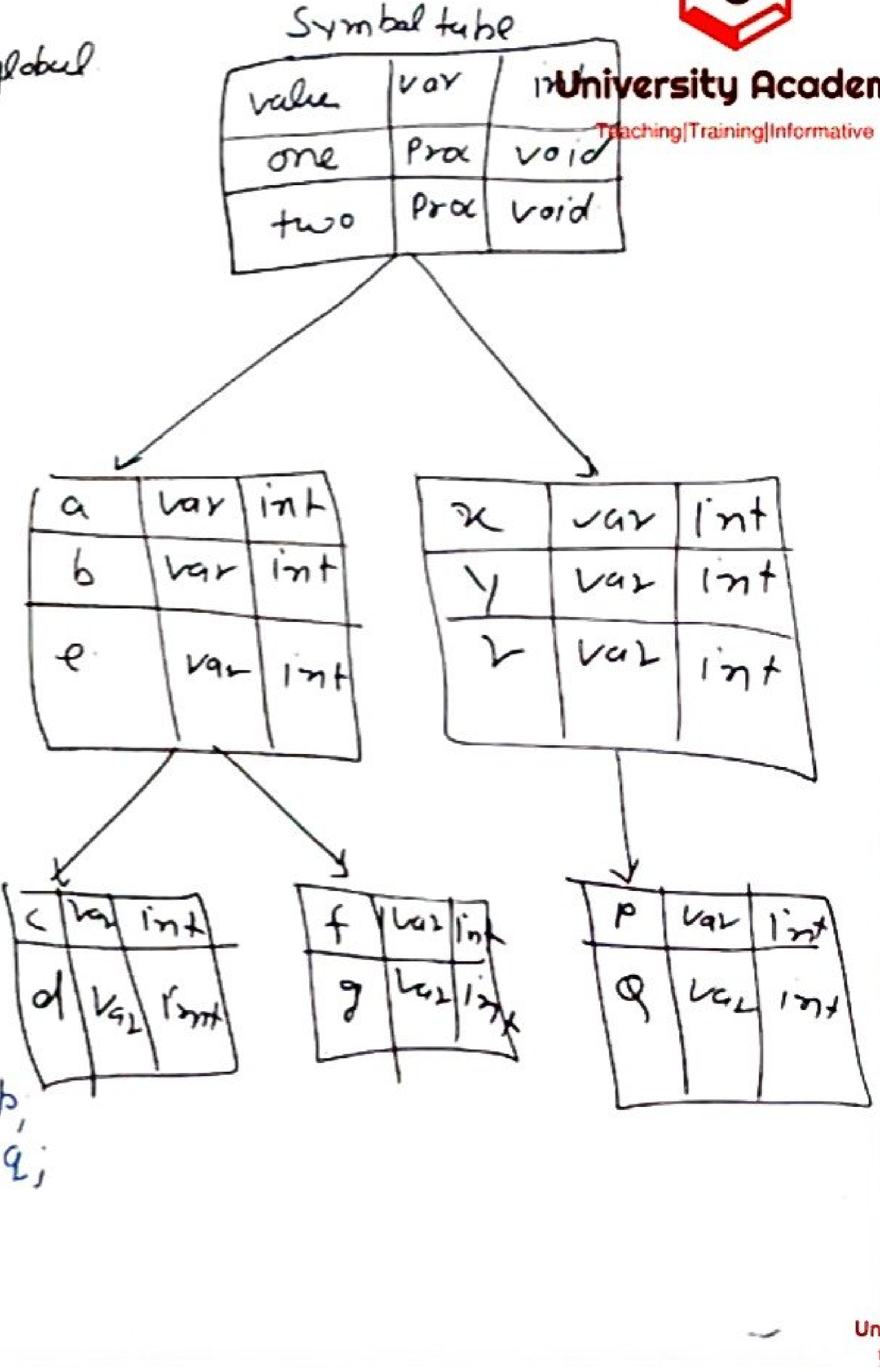
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
c	a	l	c	u	l	a	t	e	$	s	u	m	$	a	$	b	$

Symbol Table

Operation on Symbol table.

1. Insert(): insert ($\overset{\text{Name}}{a}$, $\overset{\text{type}}{\text{int}}$)
2. Lookub(): Lookub(Symbol) :- Lookub(a)
3. Delete( ): delele(Simbul) : delete(a)
4. Scope mgmt: local and Global Varia.

```
eg  int value;   global
⇒ void one()
  {
   → int a;
   → int b;
     {
     → int c;
     → int d;
     }
   → int e;
     {
     → int f;
     → int g.
     }
  }
⇒ void two()
  {
   → int x;
   → int y;
     {
     → int p;
     → int q;
     }
   → int r;
  }
```



value	var	[illegible]
one	Proc	void
two	Proc	void

a	var	int
b	var	int
e	var	int

x	var	int
y	var	int
r	var	int

c	var	int
d	var	int

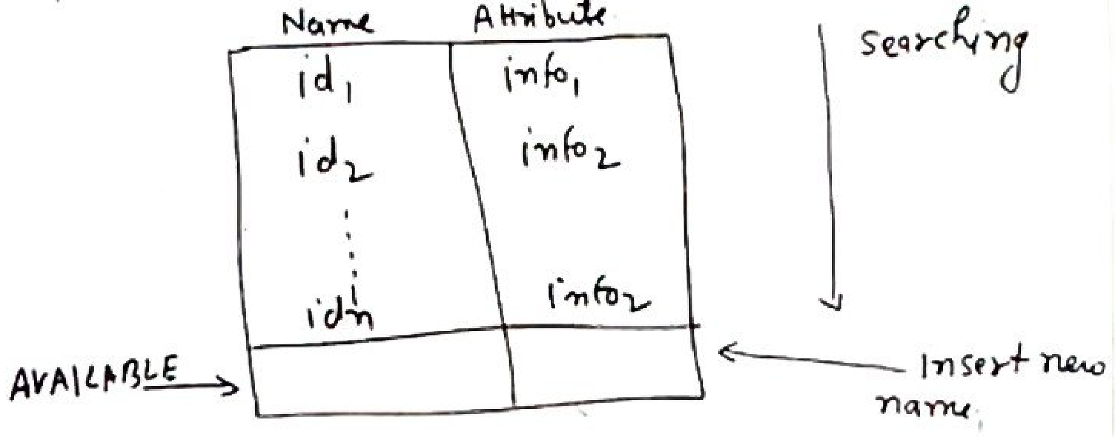
f	var	int
g	var	int

p	var	int
q	var	int

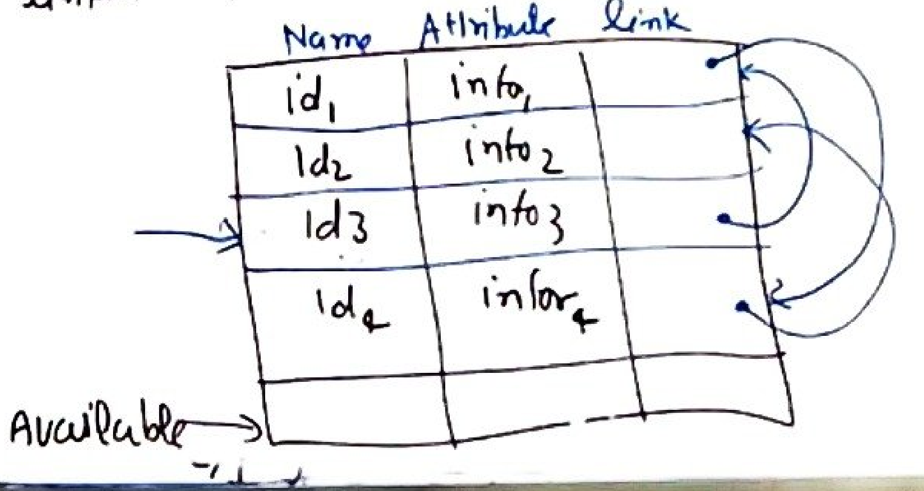
Symbol Table

Implimentation of symbol table.

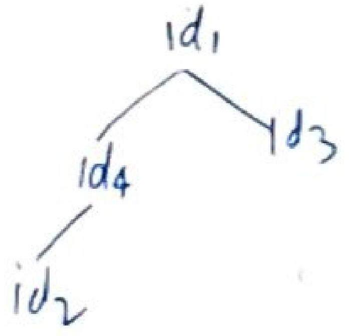
1. **linear list**: Linear list is simplest way to implement symbol table. An Array used to stor information

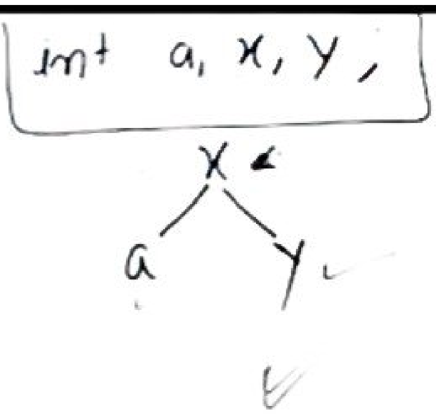


2. **Self organizing list** this symbol table Implementation using linked list. A link field added to each record

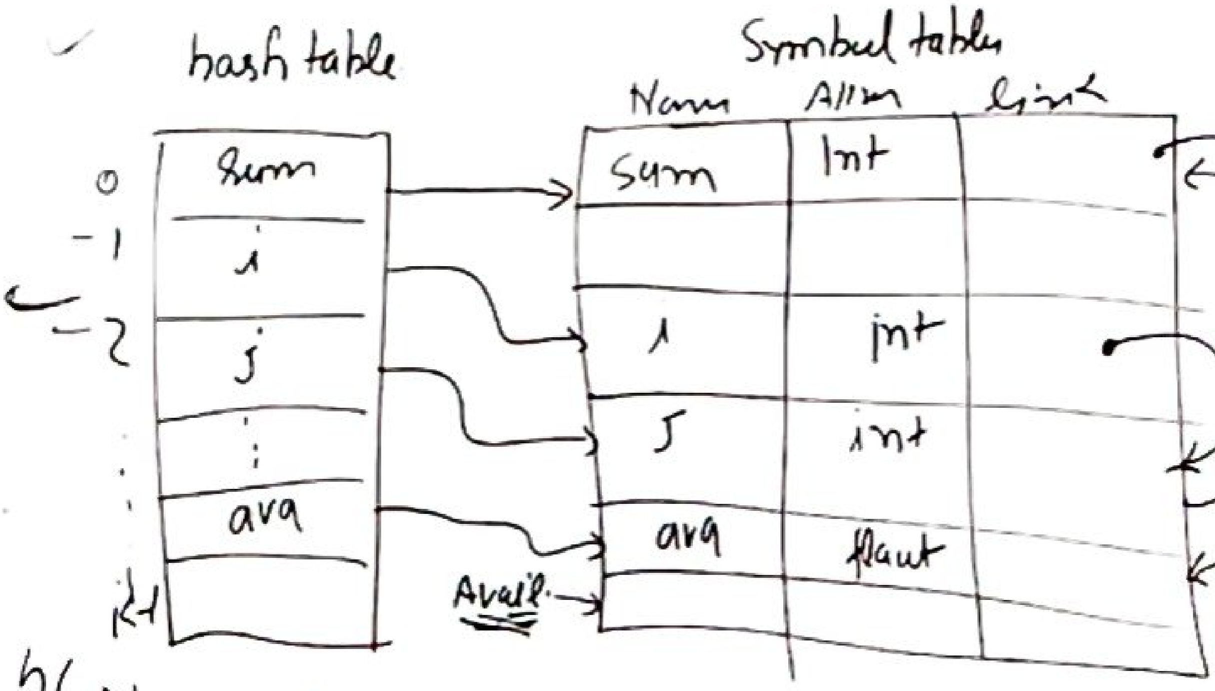


3 Binary search tree.



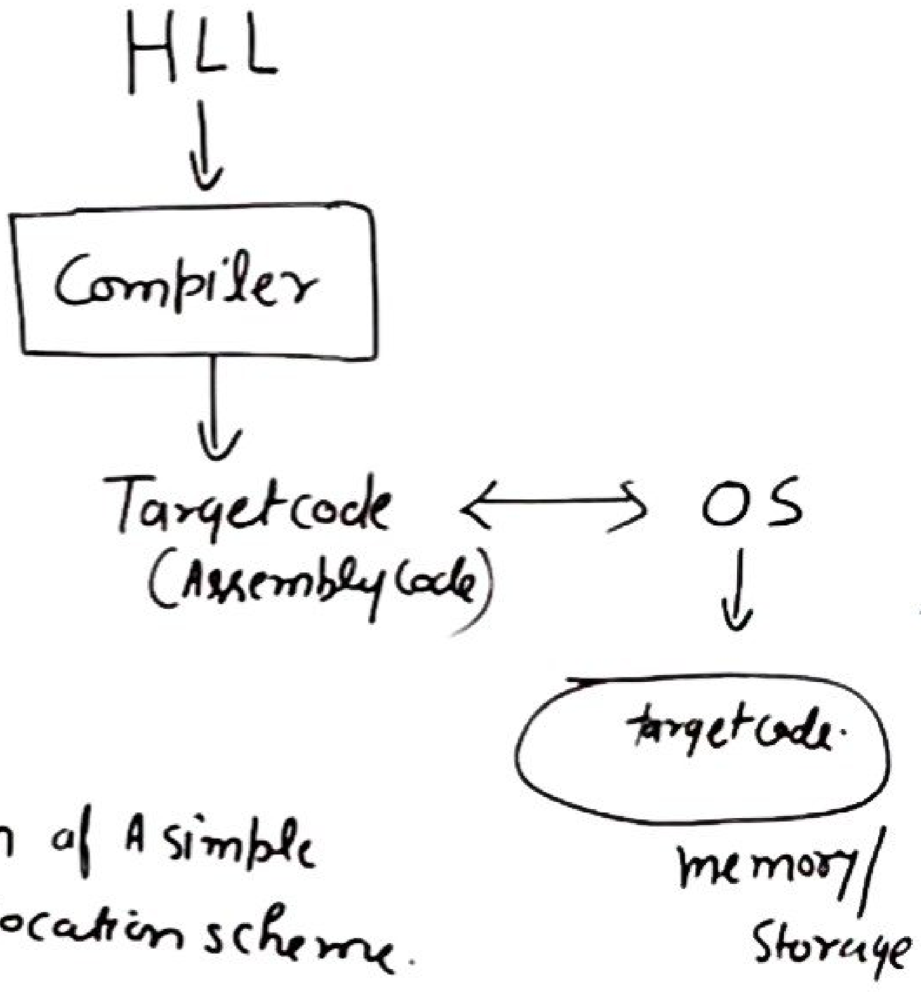


4 Hash table. hashing is most powerfull Implementation Technique in Symbol table. in hashing scheme two table are maintained.



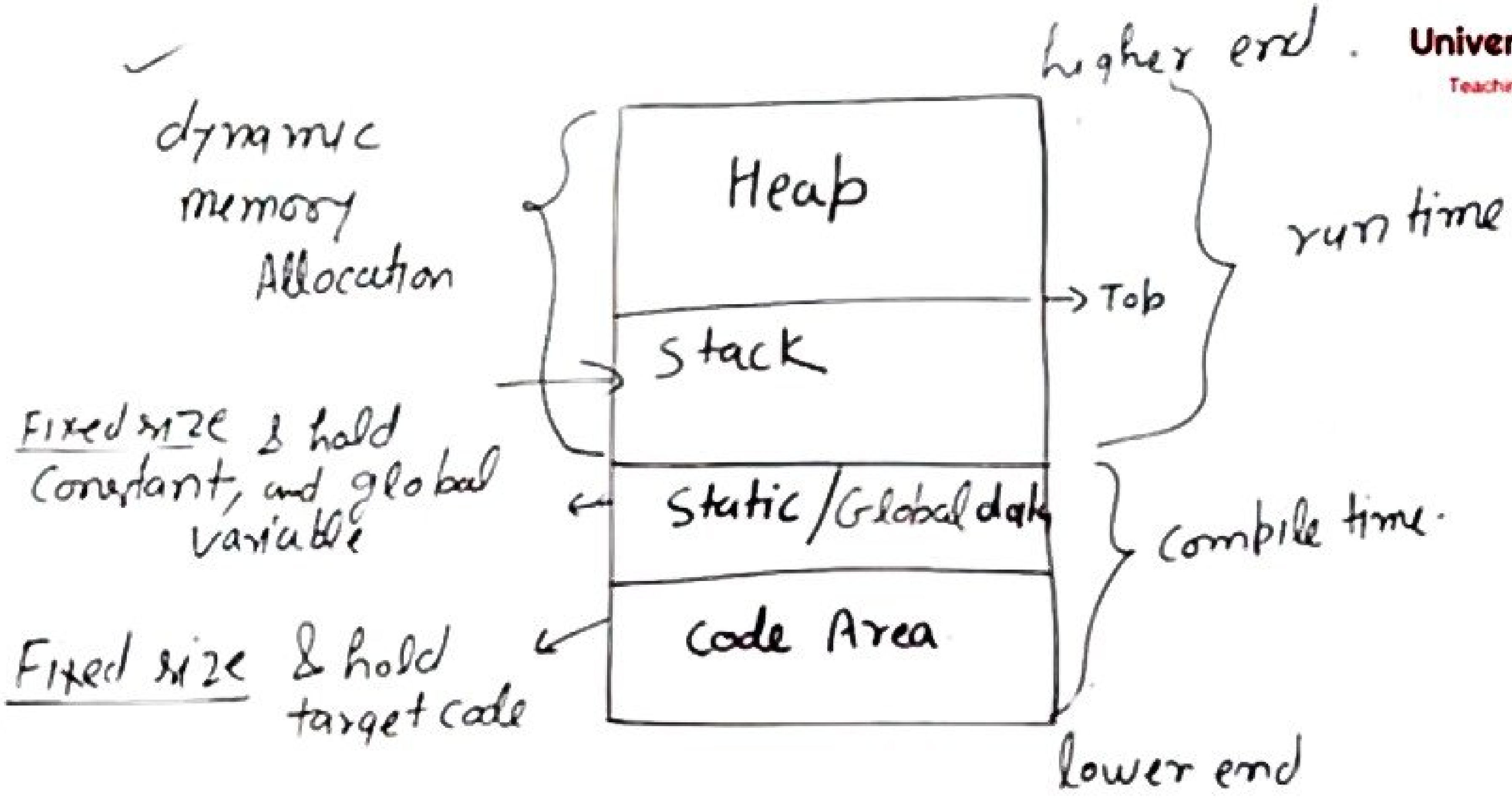
h(j) = 2 Hash funct h(name) returns integer value 0 ... k

Run time storage Administration.



① Implementation of A simple Stack Allocation scheme.

② Implementation of Block structured lang.


higher end.
dynamic memory Allocation
Heap
run time
Top
Stack
Fixed size & hold Constant, and global variable
Static/Global data
compile time.
Code Area
Fixed size & hold target code
lower end

Run time storage Administration.

① Simple stack Allocation scheme:

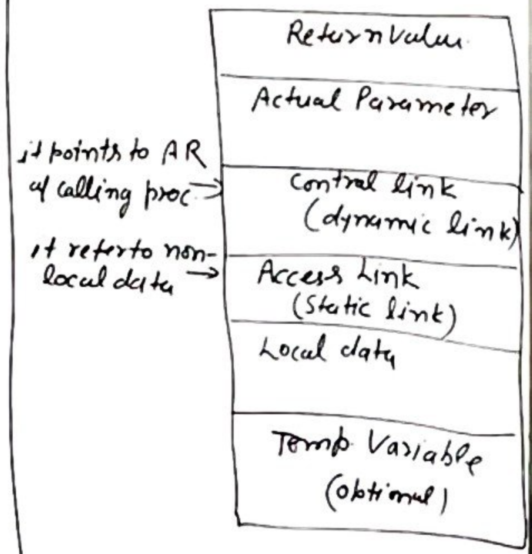
```
main()
{
   one();
}
one()
{
   two();
}
two()
{
   ...
}
```

AR - Main

AR - one()

AR - two()

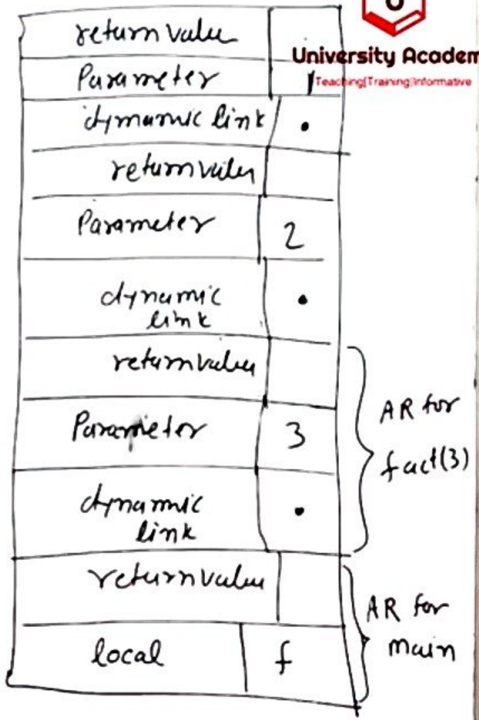
Activation Record



Return Value
Actual Parameter
it points to AR of calling proc
control link (dynamic link)
it refer to non-local data
Access Link (Static link)
Local data
Temp Variable (optional)

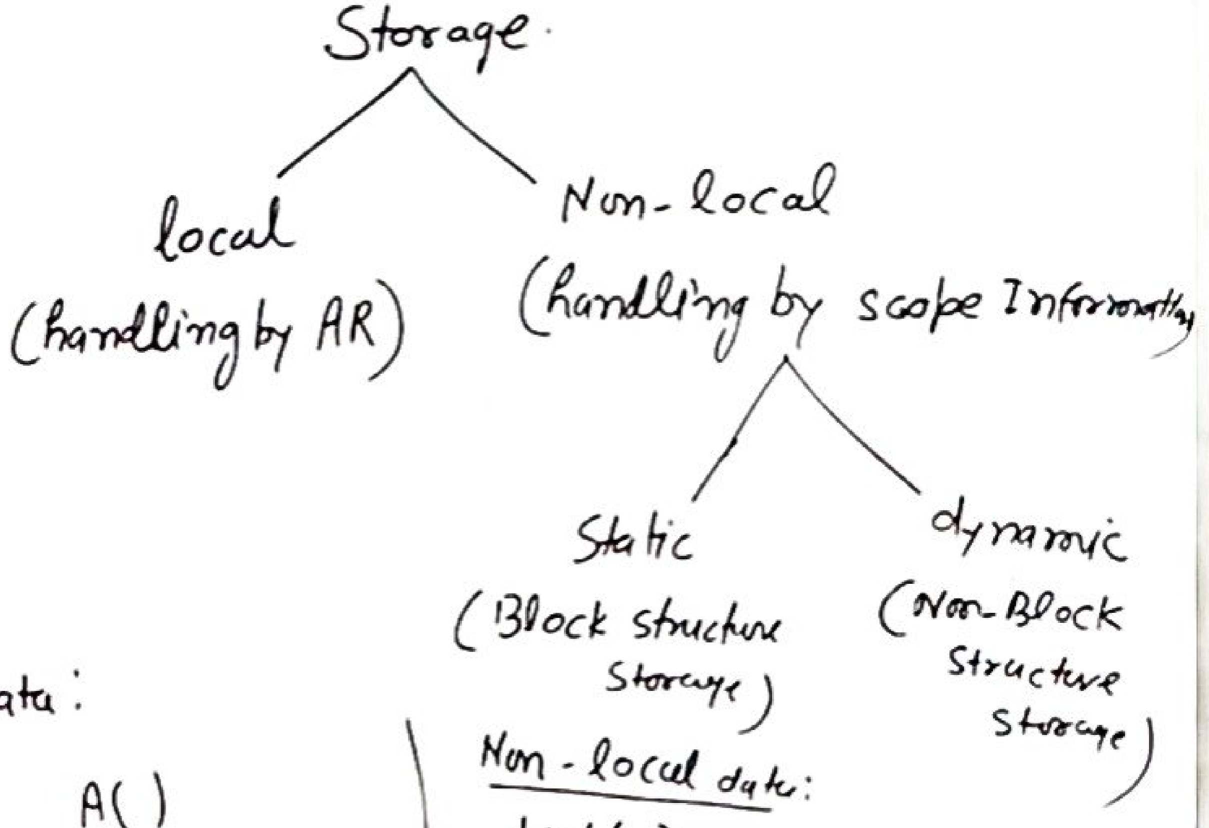
eg.

```
main()
{
 int f;
 f = fact(3);
}
int fact (int n)
{
  if(n==1)
   return 1;
  else
   return(n*fact(n-1));
}
```



Run time storage Administration.

(2) Implementation of Block-structured Language:



Storage.
local
(handling by AR)
Non-local
(handling by scope Information)
Static
(Block structure Storage)
dynamic
(Non-Block Structure Storage)

(1) local data:

```
A( )
{  int a;
}
```

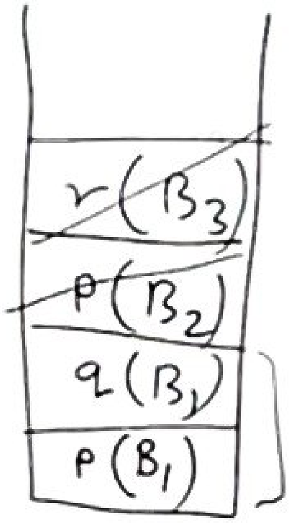
Non-local data:

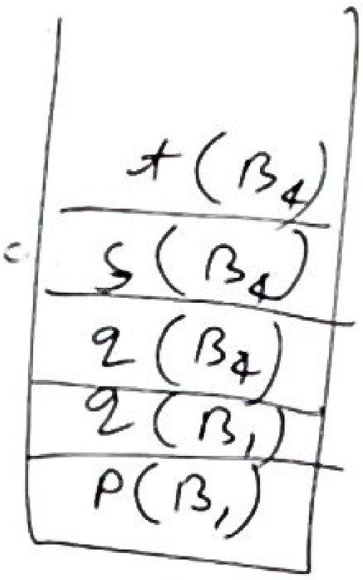
```
test( )
{
   int a, b;
      {
       int x, y;
      }
      {
       int c, d;
      }
```

static scope rule (Lexical scope) : eg C, ADA, PASCAL

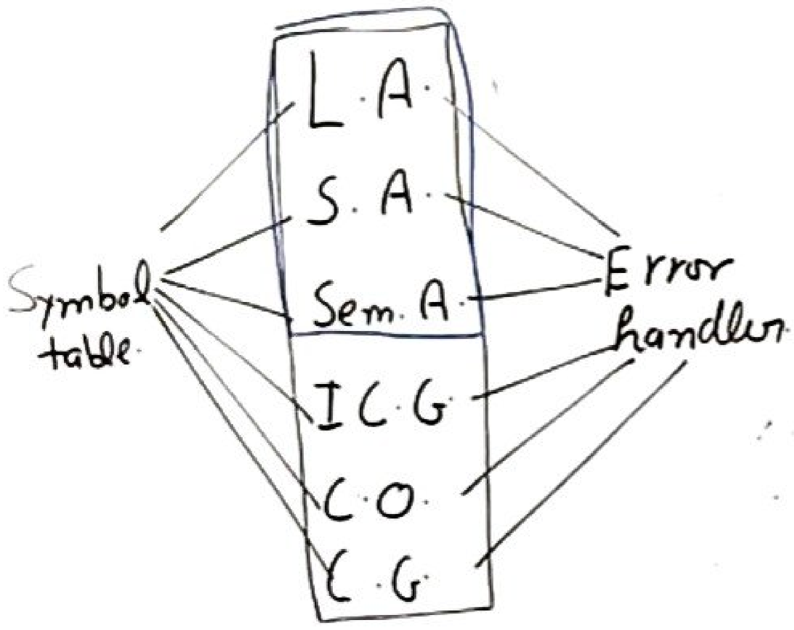
eg:

```
scop-test( )
B1 {
       int p, q;
   B2 {
          int p,
      B3 {
             int r;
             . . .
         }
      }
      . . . .
      . . . .
   B4 {
          int q, s, t;
          . . .
      }
   }
```

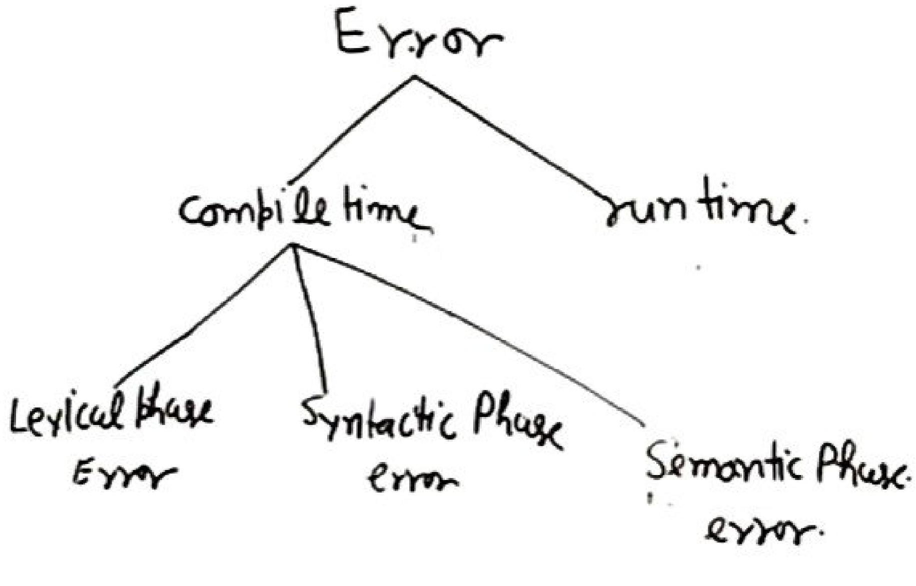




Error Detection and Recovery



= Error Detection + Error Reporting + Error Recovery.



Error / Recovery method	Lexical Phase Error	Syntactic Phase Error	Semantic Phase Error
Panic Mode	✓	✓	×
Phrase level	×	✓	×
Error production	×	✓	×
Global production	×	✓	×
Using Symbol table	×	×	✓

① Lexical Phase Error :-

int sum;

(i) Exceeding length of identifier
(ii) Appearance of illegal character
(iii) Unmatched string or comment

ex.
```
void main( )
 {
  int a, ©, $;   variable declaration */
   a=10;
   printf("%d", a); $
  }
```

Error Detection and Recovery

Syntactic Phase Error.

(i) missing parenthesis eg. printf("hello" ;

(ii) missing operator $\boxed{a+b\ c}$

(iii) Misspelled keyword ~~swicth~~(ch)
{
- - -
}

(iv) Colon in place of semicolon $\boxed{a=1:}$ × $\boxed{a=1;}$ ✓

(v) Extra Blank space. /* comment* /

① using symbol table.

③ Error production. Add exta grammar production and make an Augmented grammar and parse the input.

④ global correction. The parser examine the whole program and tries to find out closest match for it which is error free. due to high space and time complexity it is not implemented practically.

Semantic Phaser error:

(i) Incompatible type of operands. $\underset{int}{a} * \underset{float}{b}$

(ii) Undeclared Variable.

(iii) Not matching actual argument with formal argument.

e.g. int $\boxed{a[10]}$, b;
- - -
$\boxed{a}$ = b;

int a, b
:
$\boxed{sum}$ = a+b
int int