Date: 24/May/2025
Saturday.

limitation of packed fixed arrays
- no flexibility.
- size constraint
- deletion, insertion not that easy
- no dynamic nature

Solution is "Dynamic Array" unpacked.

which has Dynamic arrays, Queue, enum, Associative arrays.

definition:- size allocation is dynamic, done at runtime not compile time.

data-type arr_name [ ] ← kept empty
↓ for 1-D

```
int a[];  //1-D
int b[][]; //2-D
int c[][][]; //3-D
```

allocation can be done using "new" keyword.

%p for a[] prints a blank '{}

Q. How memory Allocation can be done?

⇒ arr_name = new [size];

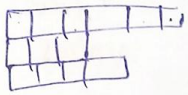
example

```
module test2;
 int a[];
initial begin
 $display ("%p",a); //prints '{}
 a = new[5];
$display ("%p", a); //prints '{0,0,0,0,0}
```

Q. How to allocate memory to 2-D dynamic array?

```
int a[][];
initial begin
 a=new[3];  → rows
 foreach (a[i])
  a[i]=new[2]; → columns
end
endmodule
```

} regular array



} irregular allocation? 2

```
a=new [3];
a[0]= new[6];
a[1] = new[3]
a[2] = new[4];
```

} Explicit declaration.

Without new keyword

a = '{'{0,0,0,0,0,0}, '{0,0,0}, '{0,0,0,0}};

listing method.

for 3-D

using foreach 3×2×2

eg: int a[][][];

```
foreach (a[i])
  a[i] = new[2];
foreach (a[i,j])
  a[i][j] = new[2];
```

```
int a[];
initial begin
  a = new[10];
```

N → from '0' to 'N-1'

★ in dynamic array

insertion & deletion is not possible by readymade method

instead we can do perform it like this

```
a = '{1,2,3,4};
a = {a[0:1], 5, a[2:3]};
```

insertion

```
a = {a[0:1], a[3:4]};
```

deletion

but entire array can be deleted using a.delete();

a.delete(5);

↓

performs entire array deletion not an element

Reallocation :-

Scenario:-

1) Declare a 1-D dynamic array.

2) Allocate the '5' locations and initialize with user-defined values

3) Reallocate the size to '7' locations

- keep prev values
- flush prev values

```
module test3;
int a[];
initial begin
a = new[5];
  foreach (a[i])
    a[i] = {$random} %15;
a = new[7](a);
```

reallocate with prev value

5	12	13	10	15	0	0

```
a = new[7];
```

flush prev value

0	0	0	0	0	0	0

QUEUES :-

① are growing, shrinking dynamically.

② More flexible than dynamic arrays

③ Has deletion, insertion etc

④ they don't have memory allocation

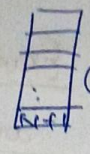
⑤ limitations it can be only used in 1-Dimension

Types

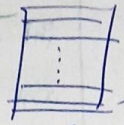
① Bounded queue → limit in size

② unbounded queue → no limit in values size

Declaration :



Queue_name [$:N];
//bounded from 0 to N+1 maximum
//beyond entries are ignored



Queue name [$];
//unbounded queue decl^n

ex q1 [$:5]; //stores six values
q2 [$]; //unbounded queue.

Methods applicable in Queue

Method	Action	Syntax
size	to get no. of elements	q_name.size;
insert	to insert an element	q_name (index, value);
delete	to delete an element	q_name.delete (index) q1.delete(2);
first index	always zero	it's '0' first index. q_name [$];
last index	last value access	

Queue can't be initialized using foreach loop

Reason : Since they have sizes which are unknown to us, Hence loop will get to know where to end, error is encountered.

Example program :

For dynamic array :

```
module dyn_array;
int a[];
initial begin
$display ("Array is = %p", a);
// '{} is printed
a=new[5];
$display ("Array after reallocation
          %p", a);
a = '{5,10,15,25,12};
$display ( %p after initializing", a);

a= new[7](a);
$display("%p after reallocating keepin
          prev. value", a);
a=new[10];
$display ("%p", a);
// flushes a
a = '{1,2,3,4,5};
$display ("new array %p", a);
a = {a[0:1], a[3:4]};
$display (" deletion of 3' %p", a);
a= {a[0:1], 10, a[3:4]};
$display ("Addition of 10 in %p");
a.delete();
$display ("%p", a);
end
endmodule
```

Q. program for 2D dynamic array :- with functionalities

Ans:-

```
module two_array_dyn;
int a[4:2][4:2];// Verbose declaration
 initial begin
  $display("the array is %p",a);
foreach(a[i,j])
  $display "a[%0d][%0d] = %0d",i,j,a[i][j]);
end
endmodule
```

↑ for fixed 2-D

prints a[4][4] = value.

for Dynamic array

```
module two_ary;
int a[][];
initial begin
  a = new[3];
  foreach (a[i])
    a[i] = new[3];
  foreach (a[i][j])
   $display ("the array is %p", a);
end
endmodule
```

2-D Dynamic Array

prints from a[0][0] = 0 ... last value

Example program on Queue :-

```
module queue1;
int q1[$];
int q2[$:5];
initial begin
$display("print the q1 : %p,
        q2 : %p", q1, q2);
q1 = '{1,2,3,4,5,6,10,9};
q2 = '{1,2,3,4,5,6};
$display("printing q1 = %p, q2 =
        %p after initializing", q1,q2);
q2 = '{1,2,3,4,5,6,10,4};
$display("inserting 10,9 in q2 = %p",
        q2);
$display("the size of q1 = %0d,
        size of q2 = %0d", q1.size,
        q2.size);
q1.insert(3,50);
// display statement
// insert 50 at 3rd index
$display("the last element
        q = %0d", q1[$]);
q1.delete(); // delete q1 fully
q2.delete(2); // delete 2nd element
$display("%p is q1, %p, is
        q2 after deletion", q1,
        q2);
```

Output:

print the q1 = '{}

print the q2 = '{}

print q1 after initialize = '{1,2,3,4,5,6,10,9};

q2 = '{1,2,3,4,5,6};

Inserting 10, 9 in q2 ignore it =
'{1,2,3,4,5,6}

the size of q1 = 8, size of q2 = 6

last element = 9.

q1 after insertion of 50 at 2 =
'{1,2,3,50,4,5,6,10,9}

print the queue1 after deletion.

'{}, '{1,2,4,5,6} after deletion method.