

Date: 24/May/2025  
Saturday.

Q. How memory Allocation can  
be done?

limitations 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,  
Queues, enum, Associative  
arrays.

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

data-type array 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.

if for a[] print a  
blank {}

⇒ 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

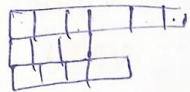
foreach (a[i])

a[i] = new [2]; → columns

end

end module

} 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}', '{0,0,0,0,0}'};

existing method.



For 3-D  
using foreach  $3 \times 2 \times 2$   
eg:  $\text{int } a[i][j][k];$

foreach ( $a[i]$ )

$a[i] = \text{new}[2];$

foreach ( $a[i, j]$ )

$a[i][j] = \text{new}[2];$

$\text{int } a[];$

initial begin

$a = \text{new}[10];$

$N \rightarrow \text{from } 0 \text{ to } N-1$

## # 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

module test3;

$\text{int } a[];$

initial begin

$a = \text{new}[5];$

foreach ( $a[i]$ )

$a[i] = (\$random) \% 15;$

$a = \text{new}[7](a);$  reallocate with prev value

5 | 12 | 13 | 10 | 15 | 0 | 0

$a = \text{new}[7];$  flush prev value

0 | 0 | 0 | 0 | 0 | 0 | 0

\* in dynamic array

insertion & deletion is not possible

by ready-made 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

## # 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  $\rightarrow$  limit in size

② unbounded queue

$\rightarrow$  no limit in value size



Declaration:

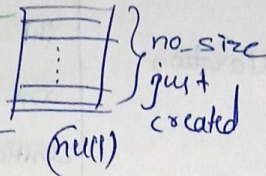
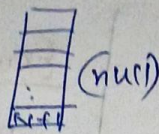
Queue-name [\$:N];

// bounded from 0 to N+1 maximum

// beyond entries are ignored

Queue name [\$];

// unbounded queue decl<sup>n</sup>



ex. q1[\$:5]; // stores six values

q2[\$]; // unbounded queue.

Methods	applicable in Queue
Method	Action
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 keeps  
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 ("relation of %p", a);

a = {a[0:1], 10, a[3:4]};

\$display ("Addition of 10 in %p", a);

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 (i, j)

\$display "a[%0d][%0d] = %0d", i, j, a[i][j];

end  
endmodule

prints

a[4][4] = value.

↑  
for fixed 2-D

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

prints from a[0][0] = 0

... last  
value

2D  
Dynamic  
Array

Example program

on Queue.

module queue-1;

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 initialization", q1, q2);

q2 = {1, 2, 3, 4, 5, 6, 10, 9};

\$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);  
// 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 queue after deletion.

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