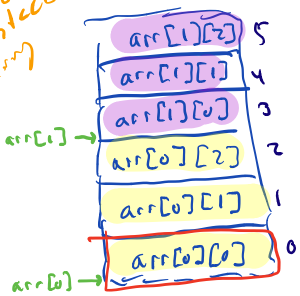
TWO-DIMENSIONAL ARRAYS ON THE STACK

```
const unsigned int   noRows = 2;
const unsigned int   noCols = 3;

int arr[noRows][noCols];
```

How we think about a 2D array

arr[0][0]	arr[0][1]	arr[0][2]
arr[1][0]	arr[1][1]	arr[1][2]

How a 2D array on stack is in memory



what's happening behind the scenes

arr[1] → arr + 1 * sizeof(int) * noCols

↑ row number, here 1

Must know length of 'y' dimension to offset into the next row

	col 0	col 1	col 2
row 0	0	1	2
row 1 →	3	4	5

i is current Row

j is the current Column

noCols is max no Columns

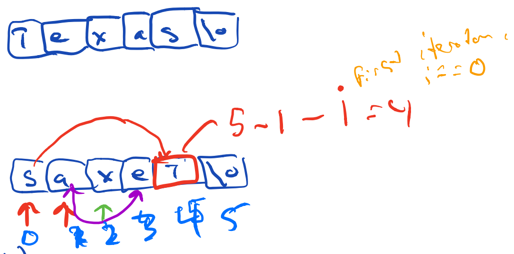
noRows is max no Rows

Assigns as shown above

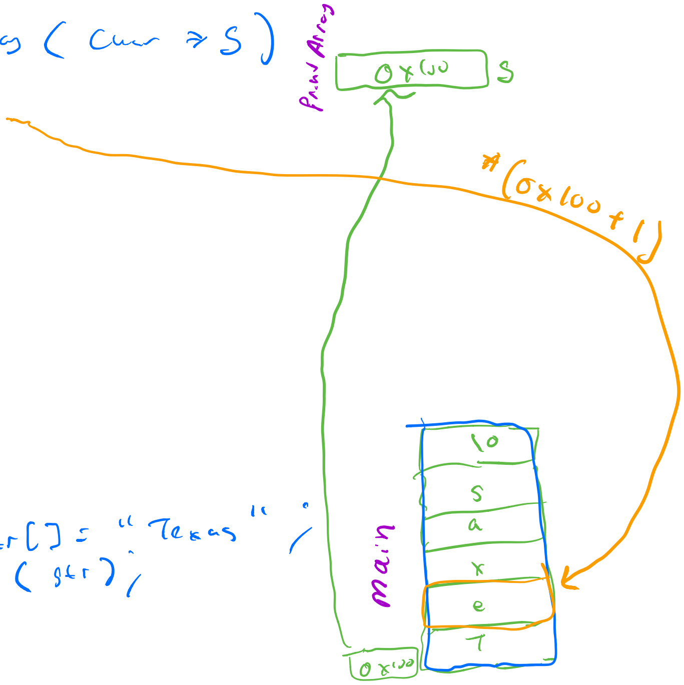
```
for (int i = 0; i < noRows; ++i)
{
    for (int j = 0; j < noCols; ++j)
    {
        arr[i][j] = i * noCols + j;

    }
}
```

Reverse String

"Texas" → "saxeT"



```
Void reverseString (char *str)
{
        int len=0;
        for (int i=0; str[i] != 0; ++i)
              len += 1;

        for (int i=0; i < len/2; ++i)
        {
              char temp = str[i];
              str[i] = str[len-i-1]      ;
              str[len-i-1] = temp;
        }
void   printArray ( char *s )

          s[1]

}



int   main ( )

{

      char   str[] = "Texas" ;
      printArray ( str ) ;

}
```


printArray
0x100
s
*(0x100+1)
\0
s
a
x
e
T
main
0x100