

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(void)
5  {
6      //variable declarations
7      int **ptr_iArray = NULL; //A pointer-to-pointer to integer ... but can also ↗
          hold base address of a 2D Array which will can have any number of rows ↗
          and any number of columns ...
8      int i, j;
9      int num_rows, num_columns;
10
11     //code
12
13     // *** ACCEPT NUMBER OF ROWS 'num_rows' FROM USER ***
14     printf("\n\n");
15     printf("Enter Number Of Rows : ");
16     scanf("%d", &num_rows);
17
18     // *** ACCEPT NUMBER OF COLUMNS 'num_columns' FROM USER ***
19     printf("\n\n");
20     printf("Enter Number Of Columns : ");
21     scanf("%d", &num_columns);
22
23     // *** ALLOCATING MEMORY TO 1D ARRAY CONSISTING OF BASE ADDRESS OF ROWS ***
24     printf("\n\n");
25     printf("***** MEMORY ALLOCATION TO 2D INTEGER ARRAY *****\n\n");
26     ptr_iArray = (int **)malloc(num_rows * sizeof(int *));
27     if (ptr_iArray == NULL)
28     {
29         printf("FAILED TO ALLOCATE MEMORY TO %d ROWS OF 2D INTEGER ARRAY !!! ↗
          EXITTING NOW...\n\n", num_rows);
30         exit(0);
31     }
32     else
33         printf("MEMORY ALLOCATION TO %d ROWS OF 2D INTEGER ARRAY SUCCEEDED !!! ↗
          \n\n", num_rows);
34
35     // *** ALLOCATING MEMORY TO EACH ROW WHICH IS A 1D ARRAY CONTAINING ↗
          CONSISTING OF COLUMNS WHICH CONTAIN THE ACTUAL INTEGERS ***
36     for (i = 0; i < num_rows; i++)
37     {
38         ptr_iArray[i] = (int *)malloc(num_columns * sizeof(int)); //ALLOCATING ↗
          MEMORY (Number Of Columns * size of 'int') TO ROW 'i'
39         if (ptr_iArray[i] == NULL) //ROW 'i' MEMORY ALLOCATED ?
40         {
41             printf("FAILED TO ALLOCATE MEMORY TO COLUMNS OF ROW %d OF 2D ↗
          INTEGER ARRAY !!! EXITTING NOW...\n\n", i);
42             exit(0);
43         }
44         else
45             printf("MEMORY ALLOCATION TO COLUMNS OF ROW %d OF 2D INTEGER ARRAY ↗
          SUCCEEDED !!!\n\n", i);
46     }
47
48     // *** FILLING UP VALUES ***
```

```
49     for (i = 0; i < num_rows; i++)
50     {
51         for (j = 0; j < num_columns; j++)
52         {
53             ptr_iArray[i][j] = (i * 1) + (j * 1); // can also use : *(
54             (ptr_iArray + i) + j) = (i * 1) + (j * 1)
55         }
56     }
57     // *** DISPLAYING VALUES ***
58     for (i = 0; i < num_rows; i++)
59     {
60         printf("Base Address Of Row %d : ptr_iArray[%d] = %p \t At Address : %p
61             \n", i, i, ptr_iArray[i], &ptr_iArray[i]);
62     }
63     printf("\n\n");
64
65     for (i = 0; i < num_rows; i++)
66     {
67         for (j = 0; j < num_columns; j++)
68         {
69             printf("ptr_iArray[%d][%d] = %d \t At Address : %p\n", i, j,
70                 ptr_iArray[i][j], &ptr_iArray[i][j]); // can also use *(
71                 (ptr_iArray + i) + j) for value and *(ptr_iArray + i) + j for
72                 address ...
73         }
74         printf("\n");
75     }
76
77     // *** FREEING MEMORY ALLOCATED TO EACH ROW ***
78     for (i = (num_rows - 1); i >= 0; i--)
79     {
80         if (ptr_iArray[i])
81         {
82             free(ptr_iArray[i]);
83             ptr_iArray[i] = NULL;
84             printf("MEMORY ALLOCATED TO ROW %d HAS BEEN SUCCESSFULLY FREED !!!
85                 \n\n", i);
86         }
87     }
88
89     // *** FREEING MEMORY ALLOCATED TO 1D ARRAY CONSISTING OF BASE ADDRESSES OF
90     ROWS ***
91     if (ptr_iArray)
92     {
93         free(ptr_iArray);
94         ptr_iArray = NULL;
95         printf("MEMORY ALLOCATED TO ptr_iArray HAS BEEN SUCCESSFULLY FREED !!!
96             \n\n");
97     }
98
99     return(0);
100 }
```