TUTORIAL XII:

2-D Matrix Linked List Implementation

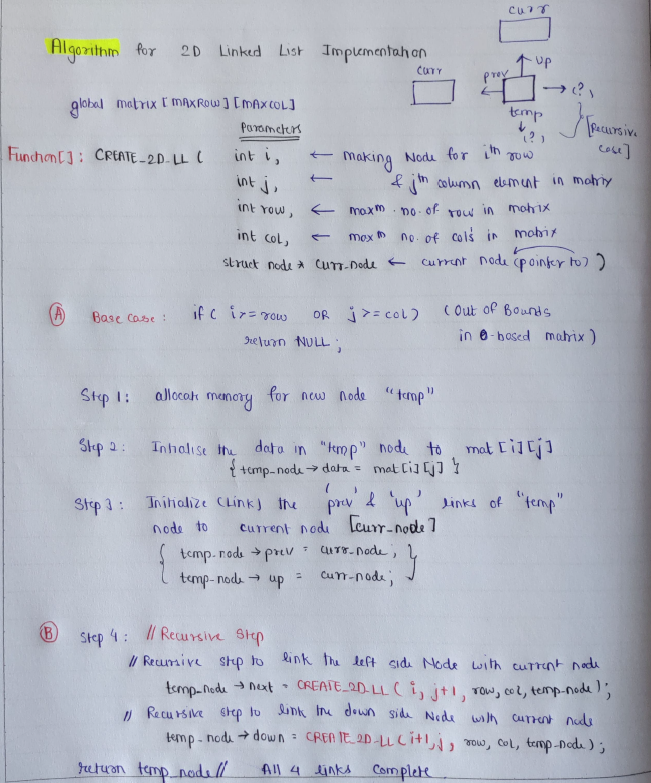
*U19CS012 [D-12]*

**Input**: A 2D matrix

**Aim**: Convert it into a *doubly linked list* which has *four pointers*; next, previous, up and down. Each node of this list should be **connected** to its next, previous, up and down nodes respectively.

Write an **algorithm** for the above task and implement the same.

(A)**Algorithm:**



(B)**Code:**

*#include* <stdio.h>

*#include* <stdlib.h>

*/\**

*Structure Of 4-Directional Node*

*up*

*prev <- [DATA] -> next*

*down*

*\*/*

struct node

{

    int data;

    struct node \*up;

    struct node \*down;

    struct node \*next;

    struct node \*prev;

};

*#define* MAXROW 10000

*#define* MAXCOL 10000

int mat[MAXROW][MAXCOL];

*// head Pointer -> head of Linked List*

struct node \*head = NULL;

*// Creation of the 2-D Doubly Linked List*

struct node \*CREATE\_2D\_LL(int i, int j, int row, int col, struct node \*curr\_node);

*// Display of the 2-D Doubly Linked List*

void DISPLAY\_2D\_LL(struct node \*head);

int main()

{

*// Number of Rows*

    int row;

*// Number of Columns*

    int col;

    printf("Enter the Number of Rows in Matrix : ");

    scanf("%d", &row);

    printf("Enter the Number of Columns in Matrix : ");

    scanf("%d", &col);

*for* (int i = 0; i < row; i++)

    {

*for* (int j = 0; j < col; j++)

        {

            printf("a[%d][%d] = ", i, j);

            scanf("%d", &mat[i][j]);

        }

    }

    head = CREATE\_2D\_LL(0, 0, row, col, NULL);

    DISPLAY\_2D\_LL(head);

*return* 0;

}

*// Creation of the 2-D Doubly Linked List*

struct node \*CREATE\_2D\_LL(int i, int j, int row, int col, struct node \*curr\_node)

{

*// Base Case*

*if* (i >= row || j >= col)

    {

*return* NULL;

    }

*// Create A Temporary Node*

    struct node \*temp\_node;

    temp\_node = (struct node \*)malloc(sizeof(struct node));

*// Assign the Value in the Node*

    temp\_node->data = mat[i][j];

*// Intialise the Prev & Up Pointer of node to Current*

*// Since temp\_node is Linked to Current Pointer*

    temp\_node->prev = curr\_node;

    temp\_node->up = curr\_node;

*// Recursive Step to Link the Left Sided Node with Current Node*

    temp\_node->next = CREATE\_2D\_LL(i, j + 1, row, col, temp\_node);

*// Recursive Step to Link the Down Sided Node with Current Node*

    temp\_node->down = CREATE\_2D\_LL(i + 1, j, row, col, temp\_node);

*// Return the temp\_node with all its 4 Links Complete*

*return* temp\_node;

}

*// Display of the 2-D Doubly Linked List*

void DISPLAY\_2D\_LL(struct node \*head)

{

*// pointer to move right*

    struct node \*rPtr;

*// pointer to move down*

    struct node \*dPtr = head;

    rPtr = head;

    {

        printf("         ");

*while* (rPtr != NULL)

        {

            printf("NULL  ");

            rPtr = rPtr->next;

        }

    }

    printf("\n");

    rPtr = head;

*// loop till struct node->right is not NULL*

    printf("         ");

*while* (rPtr != NULL)

    {

        printf("^     ");

        rPtr = rPtr->next;

    }

    printf("\n");

    rPtr = head;

*// loop till struct node->right is not NULL*

    printf("         ");

*while* (rPtr != NULL)

    {

        printf("|     ");

        rPtr = rPtr->next;

    }

    printf("\n");

    rPtr = head;

*// loop till struct node->right is not NULL*

    printf("         ");

*while* (rPtr != NULL)

    {

        printf("v     ");

        rPtr = rPtr->next;

    }

    printf("\n");

    int cnt = 0;

*// loop till struct node->down is not NULL*

*while* (dPtr != NULL)

    {

        rPtr = dPtr;

*// loop till struct node->right is not NULL*

        printf("NULL <-> ");

*while* (rPtr != NULL)

        {

            printf("%d <-> ", rPtr->data);

            rPtr = rPtr->next;

        }

        printf("NULL\n");

        rPtr = dPtr;

*// loop till struct node->right is not NULL*

        printf("         ");

*while* (rPtr != NULL)

        {

            printf("^     ");

            rPtr = rPtr->next;

        }

        printf("\n");

        rPtr = dPtr;

*// loop till struct node->right is not NULL*

        printf("         ");

*while* (rPtr != NULL)

        {

            printf("|     ");

            rPtr = rPtr->next;

        }

        printf("\n");

        rPtr = dPtr;

*// loop till struct node->right is not NULL*

        printf("         ");

*while* (rPtr != NULL)

        {

            printf("v     ");

            rPtr = rPtr->next;

        }

        printf("\n");

        dPtr = dPtr->down;

    }

    rPtr = head;

    {

        printf("         ");

*while* (rPtr != NULL)

        {

            printf("NULL  ");

            rPtr = rPtr->next;

        }

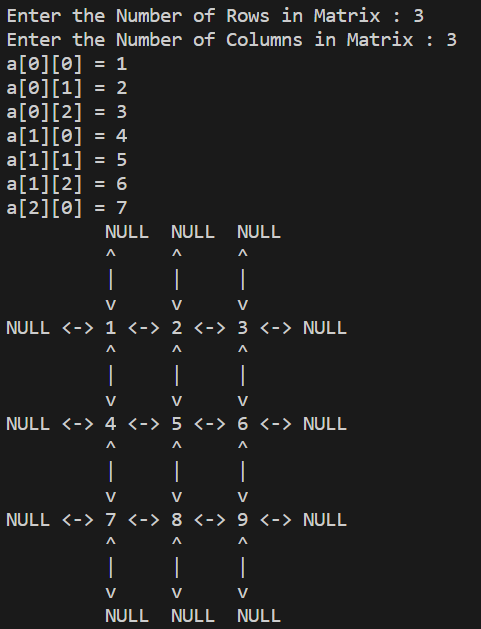
    }

}

(C)**Test Cases:**

1.) 3 x 3 Matrix

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |



2.) 3 x 4 Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| 19 | 22 | 43 | 25 |
| 66 | 36 | 39 | 11 |
| 78 | 63 | 94 | 88 |

