Write a C program to implement traversal of a directed graph through BFS and DFS.
Code:

```c
#include <stdio.h>
#define MAX 5
void breadth_first_search(int adj[][MAX], int visited[], int start);
void depth_first_search(int adj[][MAX], int visited[], int start);
int main()
{
  int visited[MAX] = {0};
  int adj[MAX][MAX], i, j;
  int option, size;
  // clrscr();
  do
  {
    printf("\n******MAIN MENU******* \n");
    printf("\n1. Enter values in graph");
    printf("\n2. BFS Traversal ");
    printf("\n3. DFS Traversal ");
    printf("\n4. Exit ");
    printf("\n\nEnter your option : ");
    scanf("%d", &option);
    switch (option)
    {
    case 1:
      printf("\nEnter the adjacency matrix: \n");
      for (i = 0; i < MAX; i++)
      {
        for (j = 0; j < MAX; j++)
        {
          printf("a[%d][%d]=", i, j);
          scanf("%d", &adj[i][j]);
        }
      }

      printf("\nGraph with adjacency matrix representation: \n");
      for (i = 0; i < MAX; i++)
        printf("\t%c ", i + 65); // print characters in rows
      for (i = 0; i < MAX; i++)
      {
        printf("\n");
        printf("%c\t", i + 65); // print characters in columns
        for (j = 0; j < MAX; j++)
          printf("%d \t", adj[i][j]);
      }
      break;
    case 2:
      printf("BFS Traversal: ");
      breadth_first_search(adj, visited, 0);
      break;

    case 3:
      printf("DFS Traversal: ");
      depth_first_search(adj, visited, 0);
      break;
    }
  } while (option != 4);
  // getch();
  return 0;
}
void breadth_first_search(int adj[][MAX], int visited[], int start)
{
  int queue[MAX], rear = -1, front = -1, i;
  queue[++rear] = start;
  visited[start] = 1;
  while (rear != front)
  {
    start = queue[++front];
    printf("%c \t", start + 65);
    for (i = 0; i < MAX; i++)
    {
      if (adj[start][i] == 1 && visited[i] == 0)
      {
        queue[++rear] = i;
        visited[i] = 1;
      }
    }
  }
  for (int i = 0; i < MAX; i++)
  {
    visited[i] = 0;
  }
}
void depth_first_search(int adj[][MAX], int visited[], int start)
{
  int stack[MAX];
  int top = -1, i;
  printf("%c \t", start + 65);
  visited[start] = 1;
  stack[++top] = start;
  while (top != -1)
  {
    start = stack[top];
    for (i = 0; i < MAX; i++)
    {
      if (adj[start][i] == 1 && visited[i] == 0)
      {
        stack[++top] = i;
        printf("%c \t", i + 65);
        visited[i] = 1;
        break;
      }
    }
    if (i == MAX)
      top--;
  }
  for (int i = 0; i < MAX; i++)
  {
    visited[i] = 0;
  }
}
```

Output:

```
PS C:\Users\Ajay kumar\Desktop\SEIT-B> cd "c:\Users\Ajay
kumar\Desktop\SEIT-B\DSA\Lab\10\" ; if ($?) { gcc main.c -o main } ; if
($?) { .\main }

******MAIN MENU*******

1. Enter values in graph
2. BFS Traversal
3. DFS Traversal
4. Exit

Enter your option : 1

Enter the adjacency matrix:
a[0][0]=1
a[0][1]=0
a[0][2]=1
a[0][3]=1
a[0][4]=0
a[1][0]=1
a[1][1]=1
a[1][2]=1
a[1][3]=0
a[1][4]=0
a[2][0]=1
a[2][1]=0
a[2][2]=0
```

```
a[2][3]=0
a[2][4]=1
a[3][0]=1
a[3][1]=1
a[3][2]=0
a[3][3]=0
a[3][4]=0
a[4][0]=1
a[4][1]=1
a[4][2]=1
a[4][3]=0
a[4][4]=1

Graph with adjacency matrix representation:
        A       B       C       D       E
A       1       0       1       1       0
B       1       1       1       0       0
C       1       0       0       0       1
D       1       1       0       0       0
E       1       1       1       0       1
******MAIN MENU*******

1. Enter values in graph
2. BFS Traversal
3. DFS Traversal
4. Exit

Enter your option : 2
BFS Traversal: A        C       D       E       B
******MAIN MENU*******

1. Enter values in graph
2. BFS Traversal
3. DFS Traversal
4. Exit

Enter your option : 3
DFS Traversal: A        C       E       B       D
******MAIN MENU*******

1. Enter values in graph
2. BFS Traversal
3. DFS Traversal
4. Exit

Enter your option : 4

PS C:\Users\Ajay kumar\Desktop\SEIT-B\DSA\Lab\10>
```