Tugas Matrikulasi

Mata Kuliah Struktur Data



Nama: Fachrul Pralienka Bani Muhamad

NIM: 5115201044

JURUSAN TEKNIK INFORMATIKA (S2)

FAKULTAS TEKNOLOGI INFORMASI

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

SURABAYA

2015

1. Pendahuluan

Rekursif adalah konsep pengulangan yang penting. Konsep pengulangan tersebut dapat digunakan untuk merumuskan solusi sederhana dalam sebuah permasalahan yang sulit untuk diselesaikan secara iteratif. Pada kondisi tertentu konsep rekursif dapat digunakan untuk mendefinisikan permasalahan dengan konsisten dan sederhana. Pada kondisi lain, rekursif dapat membantu untuk mengekspresikan algoritma dalam sebuah rumusan yang dapat menyederhanakan proses analisis.

1. Ide Utama
2. Simbol X digunakan sebagai “tikus”
3. Simbol O digunakan sebagai “tembok/penghalang”
4. Simbol titik digunakan sebagai “track/jalur”
5. Simbol F digunakan sebagai pertanda jalur buntu
6. Tikus mulai berjalan dari array [0][0]
7. Tikus dinyatakan berhasil keluar dari maze ketika berada pada array [7][7]
8. Prioritas arah jalan tikus secara berurutan yaitu kanan, bawah, atas, dan kiri
9. Cuplikan Source Code

Method utama yang menginisialisasi array maze dengan ordo 7x7

int **main**()

{

char maze[ ROW ][ COL ]={

{'.','O','.','.','.','.','.'},

{'.','.','.','O','.','O','O'},

{'.','O','.','O','.','.','.'},

{'.','O','.','O','.','O','O'},

{'.','O','.','O','.','O','.'},

{'.','O','.','O','.','.','.'},

{'.','O','.','O','O','.','.'}

};

}

Method kondisi maze terkini, yang menampilkan dimana lokasi tikus berada

void **mazeSekarang**(char maze[ROW][COL], int a, int b,int x,int y,int i,int j ){

bool c = false;

maze[ x ][ y ] = 'x'; // Lokasi tikus sekarang

cout<<x;

cout<<y;

cout<<(" \n" );

cetakMaze( maze );

}

Tikus menemukan jalan keluar, dimana nilai ROW dan COL adalah 7.

if (x== ROW-1 && y== COL-1){

cetakMaze(maze);

cout <<endl<<"Alhamdulillah.. Tikus berhasil sampai tujuan :D \n\n";

return ; // maze is complete

}

Tikus mengambil jalur kanan

if ( validMove( maze, x , y + 1 ) ) {

if ( x != i || y + 1 != j ) {

mazeSekarang( maze, a, b, x , y + 1, x, y );

return;

}else {

if ( notStuck (maze,x,y,i,j)) {

}else{

maze[ x ][ y ] = 'F';

mazeSekarang( maze, a, b, x , y + 1, x, y );

return;

}

}

}

Tikus mengambil jalur bawah

if ( validMove( maze, x + 1 , y ) ) {

if ( x + 1 != i || y != j ) {

mazeSekarang( maze,a,b, x + 1, y, x , y );

return;

}else {

if ( notStuck (maze,x,y,i,j)) {

}else{

maze[ x ][ y ] = 'F';

mazeSekarang( maze , a , b , x + 1, y, x , y );

return;

}

}

}

Tikus mengambil jalur atas

if ( validMove( maze, x - 1 , y ) ) {

if ( x-1 != i || y != j ) {

mazeSekarang( maze,a,b, x - 1, y, x,y );

return;

}else {

if ( notStuck (maze,x,y,i,j)) {

}else{

maze[ x ][ y ] = 'F';

mazeSekarang( maze , a , b , x - 1, y, x,y );

return;

}

}

}

Tikus mengambil jalur kiri

if ( validMove( maze, x , y-1 ) ) {

if ( x != i || y-1 != j ) {

mazeSekarang( maze, a , b , x, y - 1,x,y );

return ;

}else {

if ( notStuck (maze,x,y,i,j)) {

}else{

maze[ x ][ y ] = 'F';

mazeSekarang( maze, a , b , x, y - 1,x,y );

return ;

}

}

}

Method valid move yang menentukan track mana saja yang dapat dilewati tikus

bool **validMove**( const char maze[ROW][ COL ], int r, int c ){

return ( r >= 0 && r <= ROW-1 && c >= 0 && c <= COL-1 &&

maze[ r ][ c ] != 'O' && maze[ r ][ c ] != 'F' );

}

Method cetak maze yang menampilkan maze pada layar

void **cetakMaze**( const char maze[ROW][ COL ] ) {

system("cls");

for ( int x = 0; x < ROW; ++x ) {

for ( int y = 0; y < ROW; ++y )

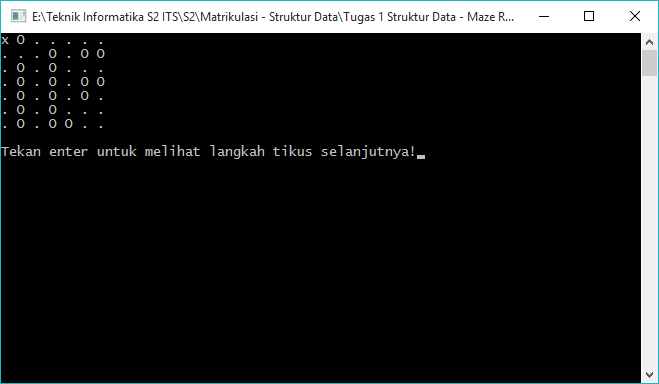
cout << maze[ x ][ y ] << ' ';

cout << '\n';

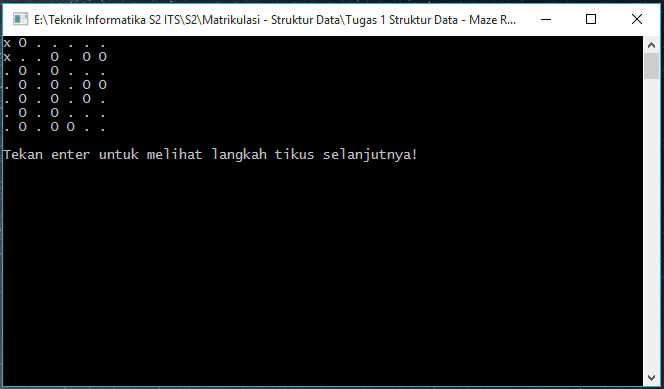
}

}

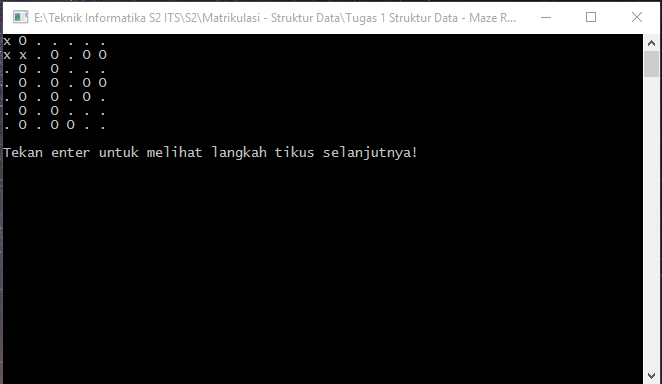
1. Hasil Output Program



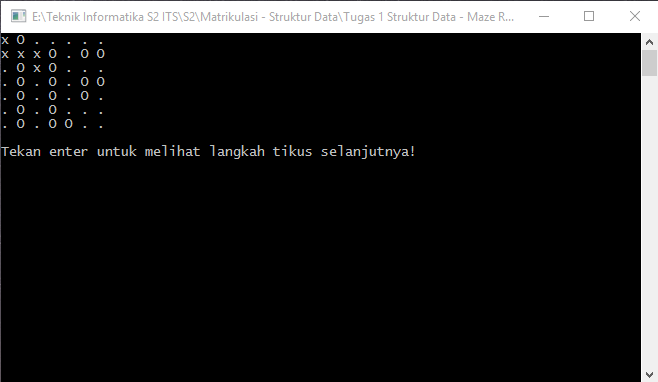
**Gambar 1.** Inisialisasi awal, tikus berada pada array [0][0]



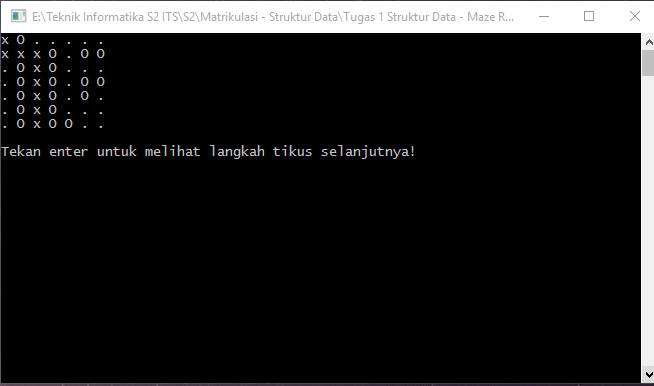
**Gambar 2.** Tikus menuju ke arah bawah



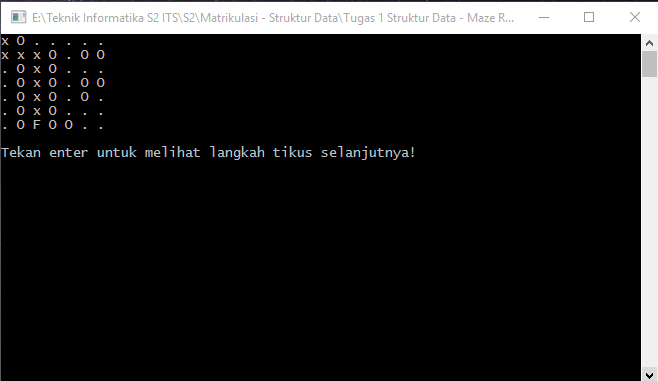
**Gambar 3.** Tikus menuju ke arah kanan



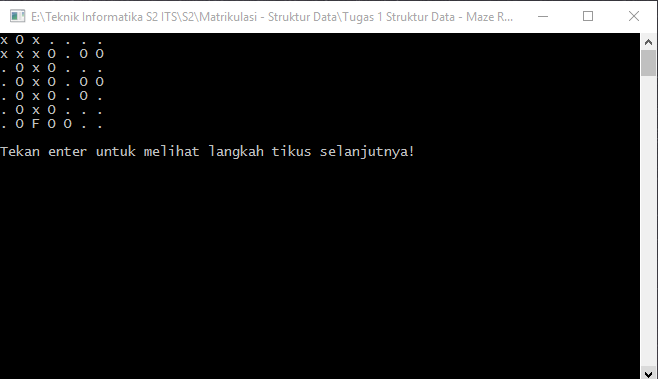
**Gambar 4.** Tikus tidak dapat menembus dinding/tembok



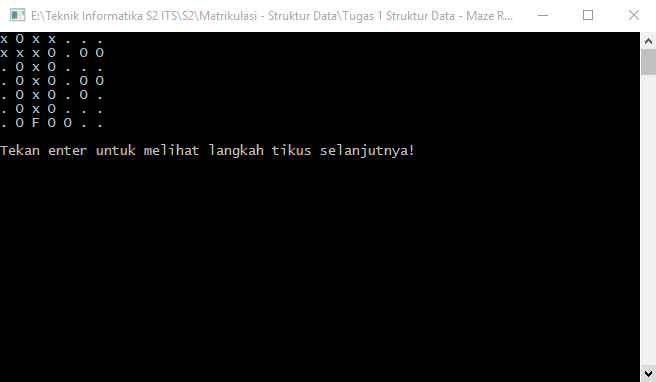
**Gambar 5.** Tikus berada pada kondisi stuck



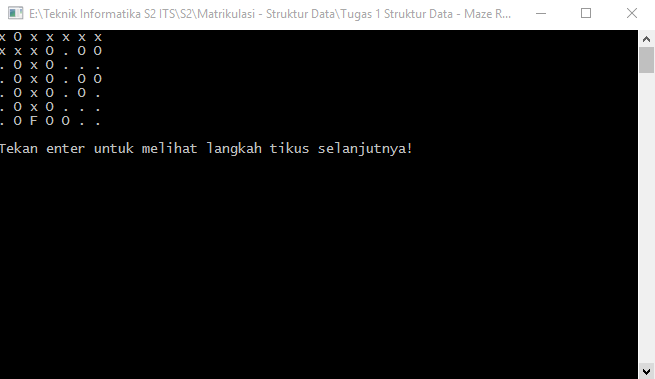
**Gambar 6.** Tikus kembali ke arah track sebelumnya yang pernah dilewati



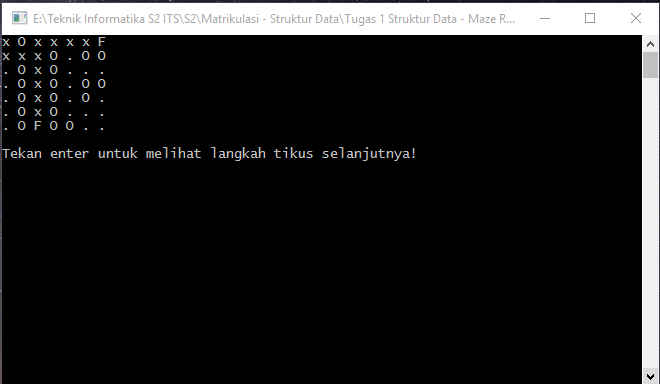
**Gambar 7.** Tikus menuju ke arah atas



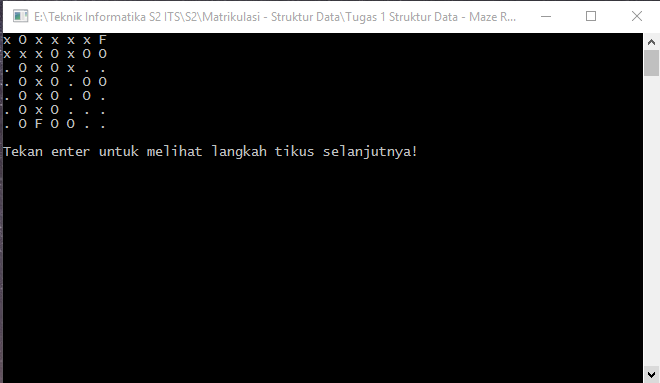
**Gambar 8.** Tikus keluar dari kondisi stuck



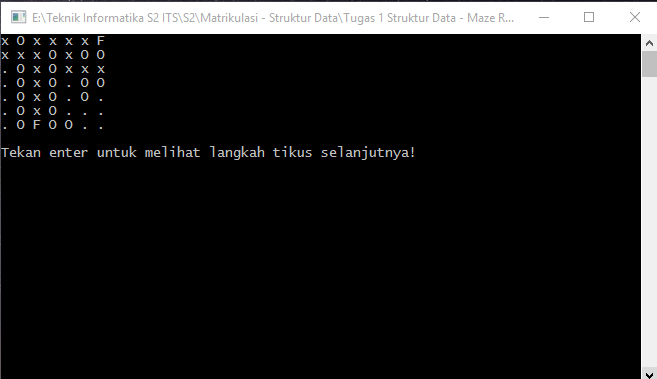
**Gambar 9.** Tikus kembali mengalami kondisi stuck



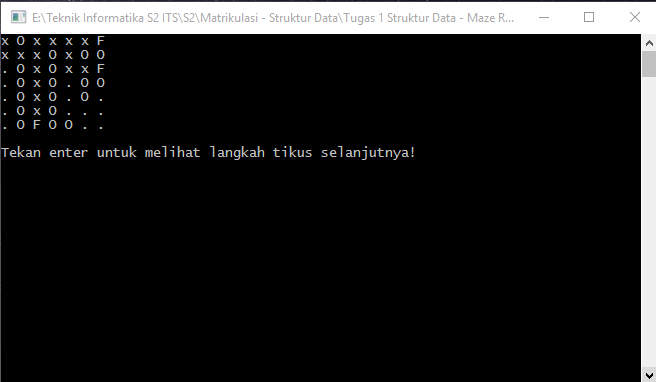
**Gambar 10.** Tikus menggunakan track yang sebelumnya pernah dilewati



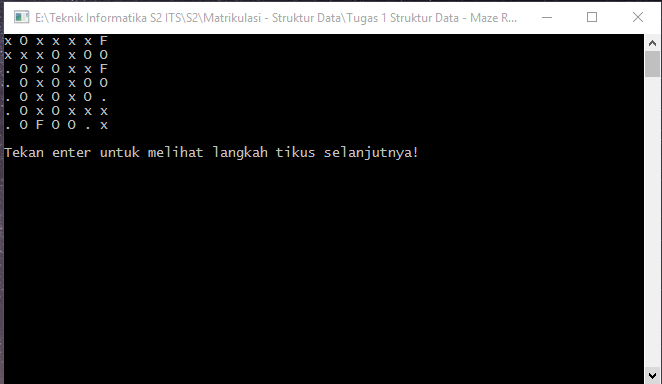
**Gambar 11.** Tikus kembali keluar dari kondisi stuck



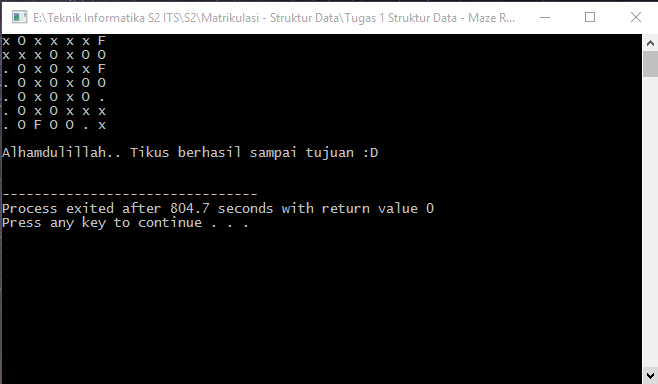
**Gambar 12.** Tikus kembali lagi pada kondisi stuck di lokasi yang berbeda



**Gambar 13.** Tikus berusaha kembali ke arah yang pernah dilewati



**Gambar 14.** Tikus berada pada array [7][7]



**Gambar 15.** Tikus dinyatakan berhasil keluar dari maze