

INDEX

1. INTRODUCTION
2. THEORY
3. METHODOLOGY
4. CODE
5. RESULT
6. CONCLUSION
7. REFERENCES

Introduction

Langton's ant is a two dimensional universal turing machine with 4 states. It can be simulated using square grids or hexagonal grids and so on. Here we use square grids with two colours white and black.

The rules for Langton's ant are :

- Turn 90 degrees clockwise at a white square, flip the colour and move forward one grid.
- Turn 90 degrees anti-clockwise at a black square, flip the colour and move one grid forward.

According to Cohen Kong theorem the trajectory of the ant is always unbounded. All finite configurations tested have been known to converge suggesting that the highway is an attractor .
Langton's ant is also a cellular automaton with it's chaotic yet emergent behavior with a simple set of rules.

The universality of Langton's ant was proven by Gajardo et.al. It's universality implies it can simulate any computation. The significance of Langton's ant is that it unites the fields of physics, computer science and mathematics due to it's discrete space, chaotic nature and emergent behavior. It was also rediscovered by physicists during computational fluid dynamics simulation in that models of diffusion and percolation use a particle that moves on a lattice and upon collision with scatterers it's direction changes.

As was also shown by Gajardo et.al the prediction of the ant's trajectory is a P-hard problem and the trajectory grows exponentially as the number of grid increases.



Langton's ant after 11000 steps



Langton's ant after 16855 steps on 300 x 300 grid

METHODOLOGY

Here we first take the number of steps to be taken and the size of grid we want to simulate on. Then the steps are carried inside an array initialized with all values 0 representing white grid.

```
for(i=0;i<width;i++)
{
    for(j=0;j<width;j++)
    {
        a[i][j] = 0; // setting the entire grid to white
    }
}
```

Then the rules are carried out and written to array.

```
while(x<width && y<width && steps<moves) // while ant is within bounds of grid
{
    if(a[x][y] == 0)                // 0 = white so move clockwise
    {
        a[x][y] = 1 ;                // flips colour to black
        co = dir ;                    // gets the orientation of the ant
        switch(co)
        {
            case 0: //ant is oriented upwards
                ++x ;
                dir = 2; //move right
                break;
            case 1: //ant is oriented down
                --x;
                dir = 3; //move left
        }
    }
    ..
    ..
}
```

The final output is written to a PBM (Portable Bit Map) file and saved on the same directory.

```
photo = fopen("output.pbm","w"); // opening file and writing in ASCII format
fprintf(photo,"P1\n");           // P1 is the magic number for PBM file in ASCII
fprintf(photo,"%d %d\n",width,width); // width, height is necessary for PBM image
for(i=0;i<width;i++)
{
    for(j=0;j<width;j++)
    {
        temp = a[i][j];
        fprintf(photo,"%d",temp); // writing to PBM image from the array
    }
    fprintf(photo,"\n");
}
```

RESULTS

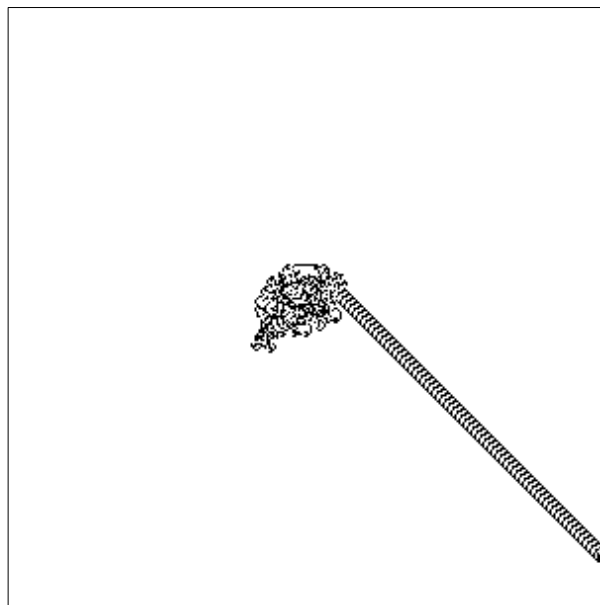
The outputs for various configuration are shown here.

```
Terminal - mounam@mounam:~/Desktop/CSIT/Projects/mine
File Edit View Terminal Tabs Help
[mounam@mounam mine]$ ls
lang lang.c
[mounam@mounam mine]$ ./lang
Enter size in pixels
300
Enter number of steps and orientation(up=0,down=1,right=2,left=3)
20000
3

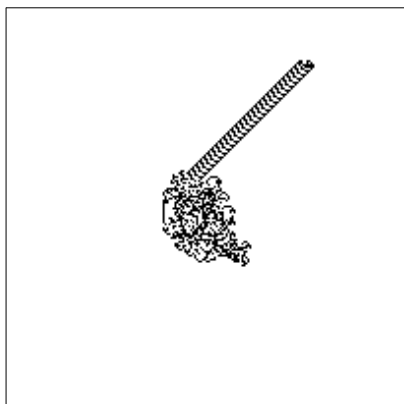
16855 steps taken
Writing to file

[mounam@mounam mine]$ ls
lang lang.c output.pbm
[mounam@mounam mine]$ |
```

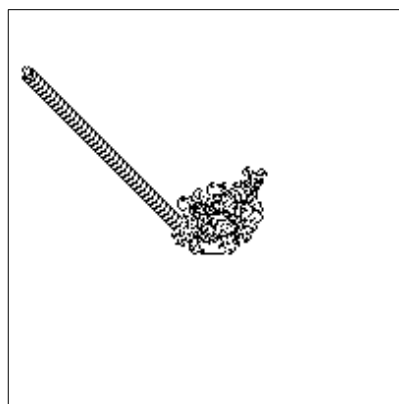
Executing the program named lang and giving parameters.



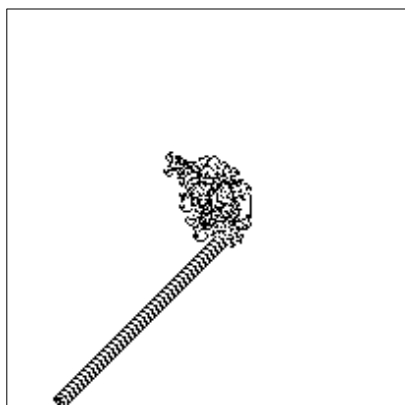
Output of the above execution on 300 x 300 grid and left orientation



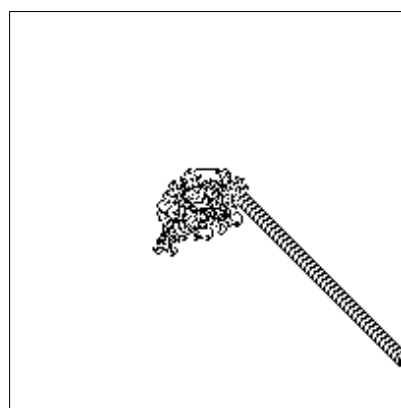
Langton's ant after 13000 steps on a 200 x 200 grid and up orientaion



Langton's ant after 14000 steps and right starting orientation.

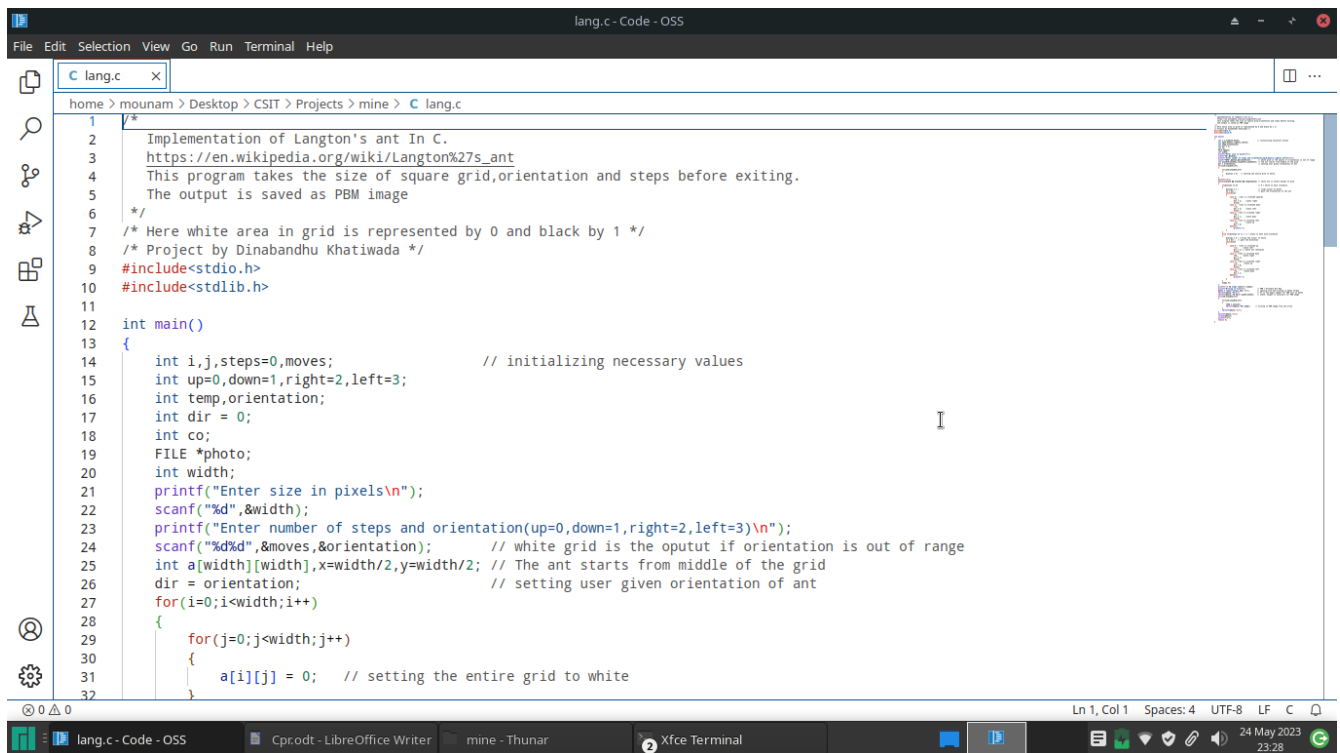


Langton's ant on 200 x 200 grid after 14255 steps with down orientation



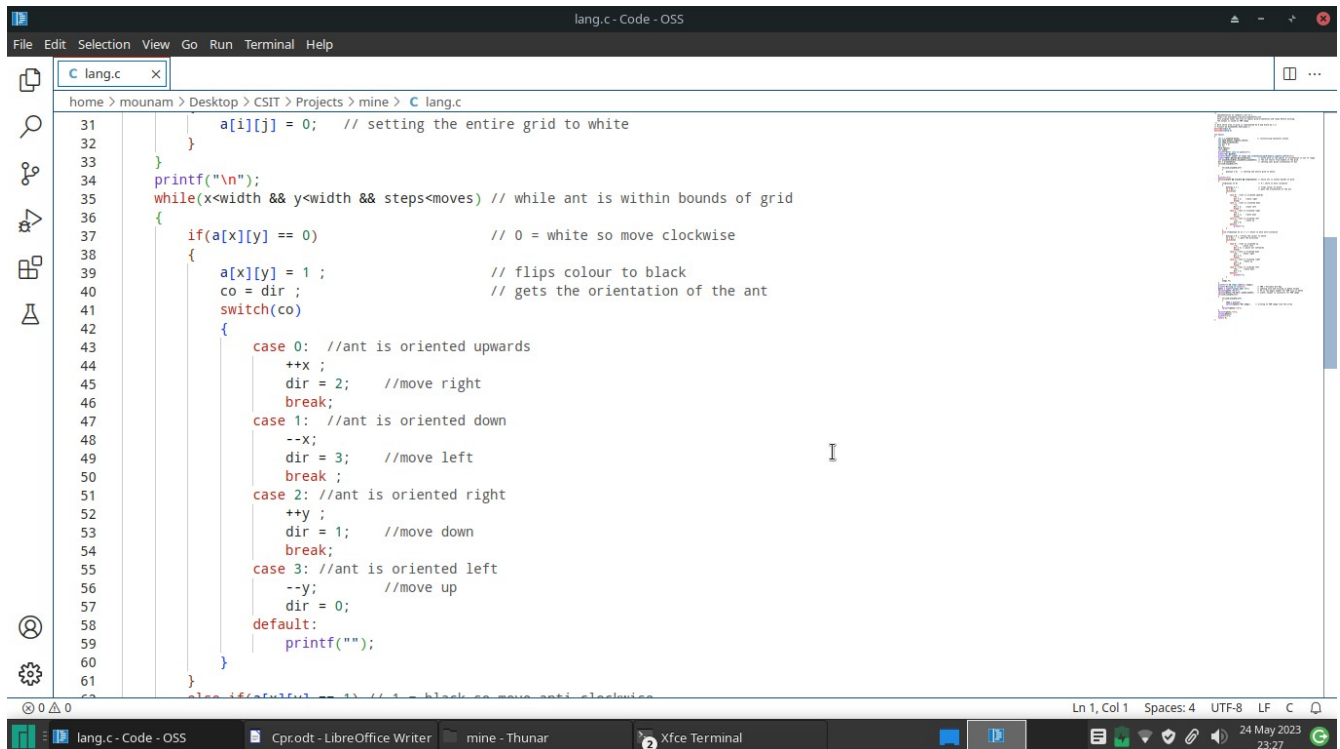
Langton's ant on 200 x 200 grid after 14255 steps with left orientation

CODE



The screenshot shows a code editor window titled 'lang.c - Code - OSS'. The file path is 'home > mounam > Desktop > CSIT > Projects > mine > C > lang.c'. The code is in C and implements the first part of Langton's ant algorithm. It includes comments in Hindi and English. The code initializes variables for grid size, orientation, and ant position. It prompts the user to enter the grid size and orientation. It then initializes the grid to white (0) and sets the ant's starting position in the middle of the grid.

```
1  /*
2  Implementation of Langton's ant In C.
3  https://en.wikipedia.org/wiki/Langton%27s_ant
4  This program takes the size of square grid,orientation and steps before exiting.
5  The output is saved as PBM image
6  */
7  /* Here white area in grid is represented by 0 and black by 1 */
8  /* Project by Dinabandhu Khatiwada */
9  #include<stdio.h>
10 #include<stdlib.h>
11
12 int main()
13 {
14     int i,j,steps=0,moves;           // initializing necessary values
15     int up=0,down=1,right=2,left=3;
16     int temp,orientation;
17     int dir = 0;
18     int co;
19     FILE *photo;
20     int width;
21     printf("Enter size in pixels\n");
22     scanf("%d",&width);
23     printf("Enter number of steps and orientation(up=0,down=1,right=2,left=3)\n");
24     scanf("%d",&moves,&orientation);    // white grid is the output if orientation is out of range
25     int a[width][width],x=width/2,y=width/2; // The ant starts from middle of the grid
26     dir = orientation;                    // setting user given orientation of ant
27     for(i=0;i<width;i++)
28     {
29         for(j=0;j<width;j++)
30         {
31             a[i][j] = 0; // setting the entire grid to white
32         }
33     }
```

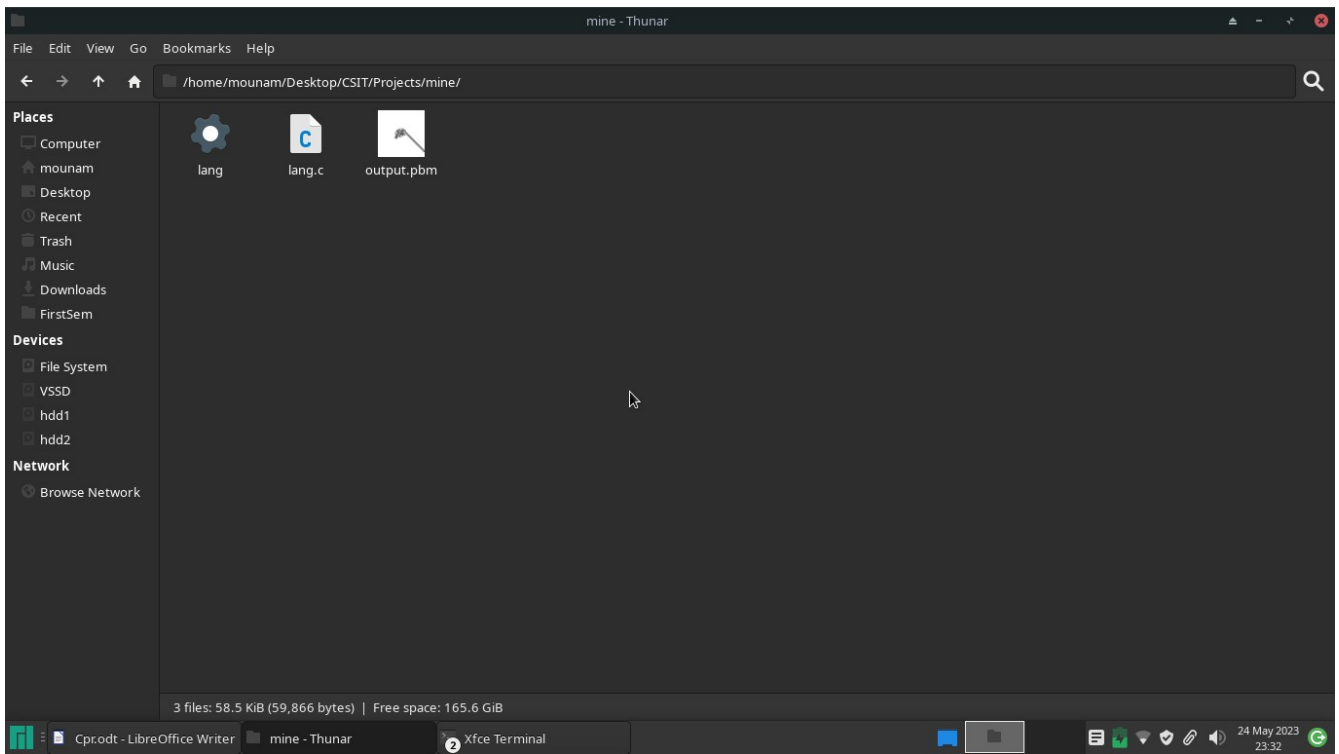


The screenshot shows the same code editor window, but now displaying the second part of the C program. This section contains a while loop that moves the ant across the grid. It checks if the ant is within the bounds of the grid. If it is, it checks the current cell's color (0 for white, 1 for black). Based on the color, it determines the ant's new orientation and moves it in that direction. It also flips the color of the current cell. The loop continues until the ant has taken the specified number of steps.

```
31     a[i][j] = 0; // setting the entire grid to white
32 }
33 }
34 printf("\n");
35 while(x<width && y<width && steps<moves) // while ant is within bounds of grid
36 {
37     if(a[x][y] == 0)                // 0 = white so move clockwise
38     {
39         a[x][y] = 1 ;                // flips colour to black
40         co = dir ;                    // gets the orientation of the ant
41         switch(co)
42         {
43             case 0: //ant is oriented upwards
44                 ++x ;
45                 dir = 2; //move right
46                 break;
47             case 1: //ant is oriented down
48                 --x;
49                 dir = 3; //move left
50                 break ;
51             case 2: //ant is oriented right
52                 ++y ;
53                 dir = 1; //move down
54                 break;
55             case 3: //ant is oriented left
56                 --y; //move up
57                 dir = 0;
58             default:
59                 printf("");
60         }
61     }
62     // if(a[x][y] == 1) // 1 = black so move anti clockwise
63 }
```

```
lang.c - Code - OSS
File Edit Selection View Go Run Terminal Help
C lang.c x
home > mounam > Desktop > CSIT > Projects > mine > C lang.c
60     }
61     }
62     else if(a[x][y] == 1) // 1 = black so move anti-clockwise
63     {
64         a[x][y] = 0; //flips the colour to white
65         co = dir; // gets the direction
66         switch(co)
67         {
68             case 0: //ant is oriented up
69                 --x; //move left
70                 dir = 3; // point ant leftwards
71                 break;
72             case 1: //ant is oriented down
73                 ++x; //move right
74                 dir = 2;
75                 break;
76             case 2: //ant is oriented right
77                 --y; //move up
78                 dir = 0;
79                 break;
80             case 3: //ant is oriented left
81                 ++y; //move down
82                 dir = 1;
83             default:
84                 printf("");
85         }
86     }
87     steps ++;
88 }
89 printf("\n %d steps taken\n",steps);
90 printf("Writing to file\n"); // PBM = Portable Bit Map
Ln 1, Col 1 Spaces: 4 UTF-8 LF C
lang.c - Code - OSS Cpr.odt - LibreOffice Writer mine - Thunar Xfce Terminal 24 May 2023 23:27
```

```
lang.c - Code - OSS
File Edit Selection View Go Run Terminal Help
C lang.c x
home > mounam > Desktop > CSIT > Projects > mine > C lang.c
78         --y; //move up
79         dir = 0;
80         break;
81     case 3: //ant is oriented left
82         ++y; //move down
83         dir = 1;
84     default:
85         printf("");
86     }
87     steps ++;
88 }
89 printf("\n %d steps taken\n",steps);
90 printf("Writing to file\n"); // PBM = Portable Bit Map
91 photo = fopen("output.pbm","w"); // opening file and writing in ASCII format
92 fprintf(photo,"P1\n"); // P1 is the magic number for PBM file in ASCII
93 fprintf(photo,"%d %d\n",width,width); // width, height is necessary for PBM image
94 for(i=0;i<width;i++)
95 {
96     for(j=0;j<width;j++)
97     {
98         temp = a[i][j];
99         fprintf(photo,"%d",temp); // writing to PBM image from the array
100     }
101     fprintf(photo,"\n");
102 }
103 fprintf(photo,"\n");
104 fclose(photo);
105 printf("\n");
106 return 0;
107 }
108
Ln 1, Col 1 Spaces: 4 UTF-8 LF C
lang.c - Code - OSS Cpr.odt - LibreOffice Writer mine - Thunar Xfce Terminal 24 May 2023 23:27
```



THEORY

In C an array is stored as a block of contiguous memory with each block being the size of specified data type. For example : `int array[10]` creates an array of 10 bytes with each block being 2 bytes.

The index of array starts with 0 and ends at $n-1$ if n is the size of the array.

For example : `int array[5] = {23,45,88,11,37}` , here `a[0] = 23` and `a[4] = 37`

A two dimensional array can be declared and used similarly. Example : `int array[2][2]` creates a 2D array of 2×2 . It's also called a matrix.

In C files can be handled using file handling functions.

`fopen()` returns a NULL pointer if the file couldn't be opened properly else it points at beginning of the file.

Our desired output is PBM (Portable Bit Map) file in ASCII so it's magic number i.e P1 and height and width are required before writing data to file. PBM has only two values 1 and 0. 0 represents white and 1 represents black.

We have also used looping conditions in this project and the theory behind them is outlined briefly.

while loop executes the statement until its parameter is true. It also accepts 0 as false and nonzero value as true .

```
Syntax : while(condition)
        { statements ..
        }
```

for loop executes statement for defined number of iterations.

```
Syntax : for(condition)
        { statements ..
        }
```

if is a conditional statement that executes a statement if condition is true. Nested if, else if, else ladder can be used for complex use cases.

```
Syntax : if(condition 1)
        {statement 1
        }
        else if(condition 2)
        {statement 2
        }
        ...
        else
        statement n;
```

switch case is a statement that transfers control to desired statement.

```
Syntax : switch(choice)
        { case 1:
          statement 1;
          break;
          ...
          default:
          statement n;
```

REFERENCES

- Simulate langton's ant :
<https://personal.math.ubc.ca/~cass/www/ant/ant.html>
- Hardness of Approximation for Langton's Ant on a Twisted Torus
by **Takeo Hagiwara** and Tatsuie Tsukiji
<https://www.mdpi.com/1999-4893/13/12/344>
- A. Gajardo, A. Moreira, E. Goles Complexity of Langton's ant:
<https://www.sciencedirect.com/science/article/pii/S0166218X00003346>
- Rosetta Code :
https://rosettacode.org/wiki/Langton%27s_ant
- Turing Machine :
<https://introcs.cs.princeton.edu/java/52turing/>
- Wolfram Mathworld :
<https://mathworld.wolfram.com/LangtonsAnt.html>
- Langton's Ant. Aldoaloz :
<https://www.youtube.com/watch?v=1X-gtr4pEBU>
- Dynamics of Langton's ant allowed to periodically go straight Paweł Tokarza :
<https://arxiv.org/pdf/1807.08789.pdf>
- Implementation of langton's ant in C :
<https://github.com/keisentraut/langtons-ant>
- Langton's ant: a mystery cellular automaton :
<https://habr.com/en/articles/726288/>
- Langton's Ant from the Ant's referential by Thomas Cailleteau :
<https://arxiv.org/pdf/1707.00255.pdf>
- Langton's ant software :
<https://theory.org/software/ant/>

CONCLUSION

Langton's ant was successfully simulated and the output was saved to a PBM file for visual inspection. The project may be simplistic and may be improved by :

- Adding multiple colors
- Extension to any finite ruleset (here the default is 10)
- Use of Dynamic memory allocation for handling very large grids
- Refactoring the code into individual functions
- Use of `main(int argc, char *argv[])` for better interface and usability
- Support for hexagonal and other grids
- Extension to multiple ants