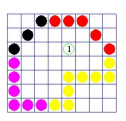
**Practical No 6.B**

**Implementation of FLOOD fill algorithm.**

**Aim: Write a program to implement a flood fill algorithm.**

**Theory:**

Sometimes we come across an object where we want to fill the area and its boundary with different colors. We can paint such objects with a specified interior color instead of searching for particular boundary color as in boundary filling algorithm. Instead of relying on the boundary of the object, it relies on the fill color. In other words, it replaces the interior color of the object with the fill color. When no more pixels of the original interior color exist, the algorithm is completed. Once again, this algorithm relies on the Four-connect or Eight-connect method of filling in the pixels. But instead of looking for the boundary color, it is looking for all adjacent pixels that are a part of the interior.



**Algorithm:**

floodFill 4(x,y,fillcolor,oldcolor:integer).

begin

If getpixel(x,y) = oldcolor then

begin

setpixel(x,y,fillcolor)

floodfill 4(x+1,y,fillcolor,oldcolor)

floodfill 4(x-1,y,fillcolor,oldcolor)

floodfill 4(x,y+1,fillcolor,oldcolor)

floodfill 4(x,y-1,fillcolor,oldcolor)

end

**Conclusion: We have implemented Flood Fill Algorithm.**

**Code: 4-Connected Polygon**

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<iostream.h>

void flood(int,int,int,int);

void main() {

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:/TURBOC3/BGI");

int a,b,c,d;

cout<<"Enter the X and Y positions of rectangle: \n";

cin>>a>>b;

cout<<"Enter the breadth and length of Rectangle: \n";

cin>>c>>d;

rectangle(a,b,c,d);

int x = (a+c)/2;

int y = (b+d)/2;

flood(x,y,10,0);

getch();

closegraph(); }

void flood(int x,int y,int fillColor, int defaultColor) {

if(getpixel(x,y)==defaultColor) {

delay(1.8);

putpixel(x,y,fillColor);

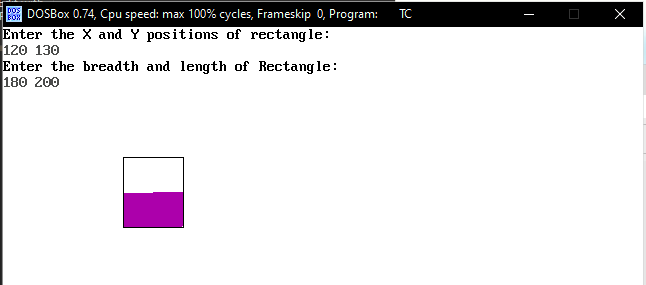
flood(x+1,y,fillColor,defaultColor);

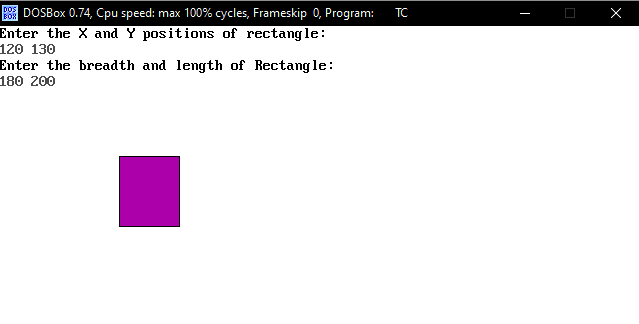
flood(x-1,y,fillColor,defaultColor);

flood(x,y+1,fillColor,defaultColor);

flood(x,y-1,fillColor,defaultColor); } }

**Output:**





**Code: 8-Connected Polygon**

#include<stdio.h>

#include<graphics.h>

#include<dos.h>

#include<conio.h>

#include<iostream.h>

void floodfill(int x,int y,int old,int newcol) {

int current;

current=getpixel(x,y);

if(current==old) {

delay(5);

putpixel(x,y,newcol);

floodfill(x+1,y,old,newcol);

floodfill(x-1,y,old,newcol);

floodfill(x,y+1,old,newcol);

floodfill(x,y-1,old,newcol);

floodfill(x+1,y+1,old,newcol);

floodfill(x-1,y+1,old,newcol);

floodfill(x+1,y-1,old,newcol);

floodfill(x-1,y-1,old,newcol); } }

void main() {

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:/TURBOC3/BGI");

int a,b,c,d;

cout<<"Enter the positions of rectangle: \n";

cin>>a>>b;

cout<<"Enter the length and breadth of rectangle: \n";

cin>>c>>d;

rectangle(a,b,c,d);

int x = (a+c)/2;

int y = (b+d)/2;

floodfill(x,y,0,15);

getch();

closegraph();

}

**Output:**

