**BÀI THỰC HÀNH – ĐỒ HỌA MÁY TÍNH**

Họ và tên: Đinh Thị Thảo

Lớp : D14CNPM4

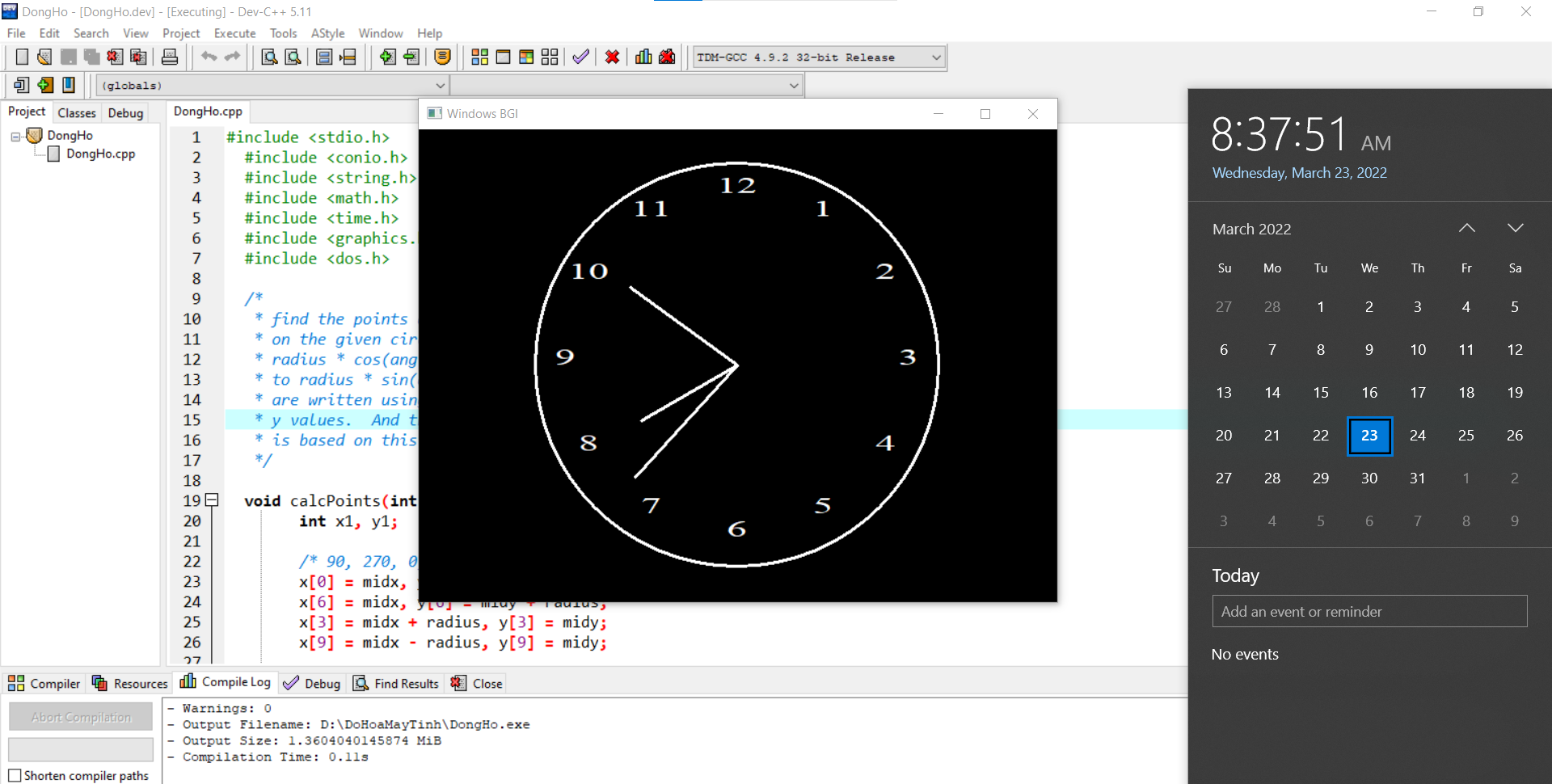
Mã sinh viên: 19810310256

Nhóm : 01

*Bài làm*

**Bài 27: Vẽ đồng hồ kim**

**Kết quả:**



**Code:**

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include <math.h>

#include <time.h>

#include <graphics.h>

#include <dos.h>

/\*

\* find the points at 0, 30, 60,.., 360 degrees

\* on the given circle. x value correponds to

\* radius \* cos(angle) and y value corresponds

\* to radius \* sin(angle). Numbers in the clock

\* are written using the above manipulated x and

\* y values. And the hour needle movement

\* is based on this

\*/

void calcPoints(int radius, int midx, int midy, int x[12], int y[12]) {

int x1, y1;

/\* 90, 270, 0, 180 degrees \*/

x[0] = midx, y[0] = midy - radius;

x[6] = midx, y[6] = midy + radius;

x[3] = midx + radius, y[3] = midy;

x[9] = midx - radius, y[9] = midy;

/\* 30, 150, 210, 330 degrees \*/

x1 = (int) ((radius / 2) \* sqrt(3));

y1 = (radius / 2);

x[2] = midx + x1, y[2] = midy - y1;

x[4] = midx + x1, y[4] = midy + y1;

x[8] = midx - x1, y[8] = midy + y1;

x[10] = midx - x1, y[10] = midy - y1;

/\* 60, 120, 210, 300 degrees \*/

x1 = radius / 2;

y1 = (int) ((radius / 2) \* sqrt(3));

x[1] = midx + x1, y[1] = midy - y1;

x[5] = midx + x1, y[5] = midy + y1;

x[7] = midx - x1, y[7] = midy + y1;

x[11] = midx - x1, y[11] = midy - y1;

return;

}

/\*

\* Calculates position for minute and second needle movement

\* Each quadrant has 90 degrees. So, we need to split each

\* quadrant into 15 parts(6 degree each) to get the minute

\* and second needle movement

\*/

void minSecCalc(int radius, int midx, int midy, int secx[60], int secy[60]) {

int i, j = 0, x, y;

char str[32];

/\* 15 position(min/sec - 12 to 3) in first quadrant of clock \*/

secx[j] = midx, secy[j++] = midy - radius;

for (i = 96; i < 180; i = i + 6) {

secx[j] = midx - (radius \* cos((i \* 3.14) / 180));

secy[j++] = midy - (radius \* sin((i \* 3.14) / 180));

}

/\* 15 positions(min or sec - 3 to 6) in second quadrant of clock \*/

secx[j] = midx + radius, secy[j++] = midy;

for (i = 186; i < 270; i = i + 6) {

secx[j] = midx - (radius \* cos((i \* 3.14) / 180));

secy[j++] = midy - (radius \* sin((i \* 3.14) / 180));

}

/\* 15 positions(min or sec - 6 to 9) in third quadrant of clock \*/

secx[j] = midx, secy[j++] = midy + radius;

for (i = 276; i < 360; i = i + 6) {

secx[j] = midx - (radius \* cos((i \* 3.14) / 180));

secy[j++] = midy - (radius \* sin((i \* 3.14) / 180));

}

/\* 15 positions(min or sec - 9 to 12) in fourth quadrant of clock \*/

secx[j] = midx - radius, secy[j++] = midy;

for (i = 6; i < 90; i = i + 6) {

secx[j] = midx - (radius \* cos((i \* 3.14) / 180));

secy[j++] = midy - (radius \* sin((i \* 3.14) / 180));

}

return;

}

int main() {

/\* request auto detection \*/

int gdriver = DETECT, gmode, err, tmp;

int i, j, midx, midy, radius, hr, min, sec;

int x[12], y[12], minx[60], miny[60];

int hrx[12], hry[12], secx[60], secy[60];

int secx1, secy1;

char str[256];

time\_t t1;

struct tm \*data;

/\* initialize graphic mode \*/

initgraph(&gdriver, &gmode, "C:/TURBOC3/BGI");

err = graphresult();

if (err != grOk) {

/\* error occurred \*/

printf("Graphics Error: %s",

grapherrormsg(err));

return 0;

}

/\* mid position in x and y -axis \*/

midx = getmaxx() / 2;

midy = getmaxy() / 2;

radius = 200;

/\* position to locate numbers in clock \*/

calcPoints(radius - 30, midx, midy, x, y);

/\* gets position for hour needle \*/

calcPoints(radius - 90, midx, midy, hrx, hry);

/\* gets position for minute needle \*/

minSecCalc(radius - 50, midx, midy, minx, miny);

/\* gets position for second needle \*/

minSecCalc(radius - 70, midx, midy, secx, secy);

while (!kbhit()) {

setlinestyle(SOLID\_LINE, 1, 3);

settextstyle(TRIPLEX\_FONT, 0, 3);

/\* draws frame of the clock \*/

circle(midx, midy, radius);

/\* place the numbers in clock \*/

for (j = 0; j < 12; j++) {

if (j == 0) {

sprintf(str, "%d", 12);

} else {

sprintf(str, "%d", j);

}

settextjustify(CENTER\_TEXT, CENTER\_TEXT);

moveto(x[j], y[j]);

outtext(str);

}

/\* get the current time using time() API \*/

t1 = time(NULL);

data = localtime(&t1);

/\* draw the second needle in clock \*/

sec = data->tm\_sec % 60;

line(midx, midy, secx[sec], secy[sec]);

/\* draw the minute needle in clock \*/

min = data->tm\_min % 60;

line(midx, midy, minx[min], miny[min]);

/\* draw the hour needle in clock \*/

hr = data->tm\_hour % 12;

line(midx, midy, hrx[hr], hry[hr]);

/\* sleep for a second \*/

delay(1000);

cleardevice();

}

getch();

/\* deallocate memory allocated for graphic screen \*/

closegraph();

return 0;

}

**Bài 1: Viết chương trình vẽ quả bóng nảy**

**Code:**

#include <stdio.h>

#include <conio.h>

#include <graphics.h>

#include <dos.h>

int main() {

int gd = DETECT, gm;

int i, x, y, flag=0;

initgraph(&gd, &gm, "C:\\TC\\BGI");

/\* get mid positions in x and y-axis \*/

x = getmaxx()/2;

y = 30;

while (!kbhit()) {

if(y >= getmaxy()-30 || y <= 30)

flag = !flag;

/\* draws the gray board \*/

setcolor(RED);

setfillstyle(SOLID\_FILL, BLUE);

circle(x, y, 30);

floodfill(x, y, RED);

/\* delay for 50 milli seconds \*/

delay(50);

/\* clears screen \*/

cleardevice();

if(flag){

y = y + 5;

} else {

y = y - 5;

}

}

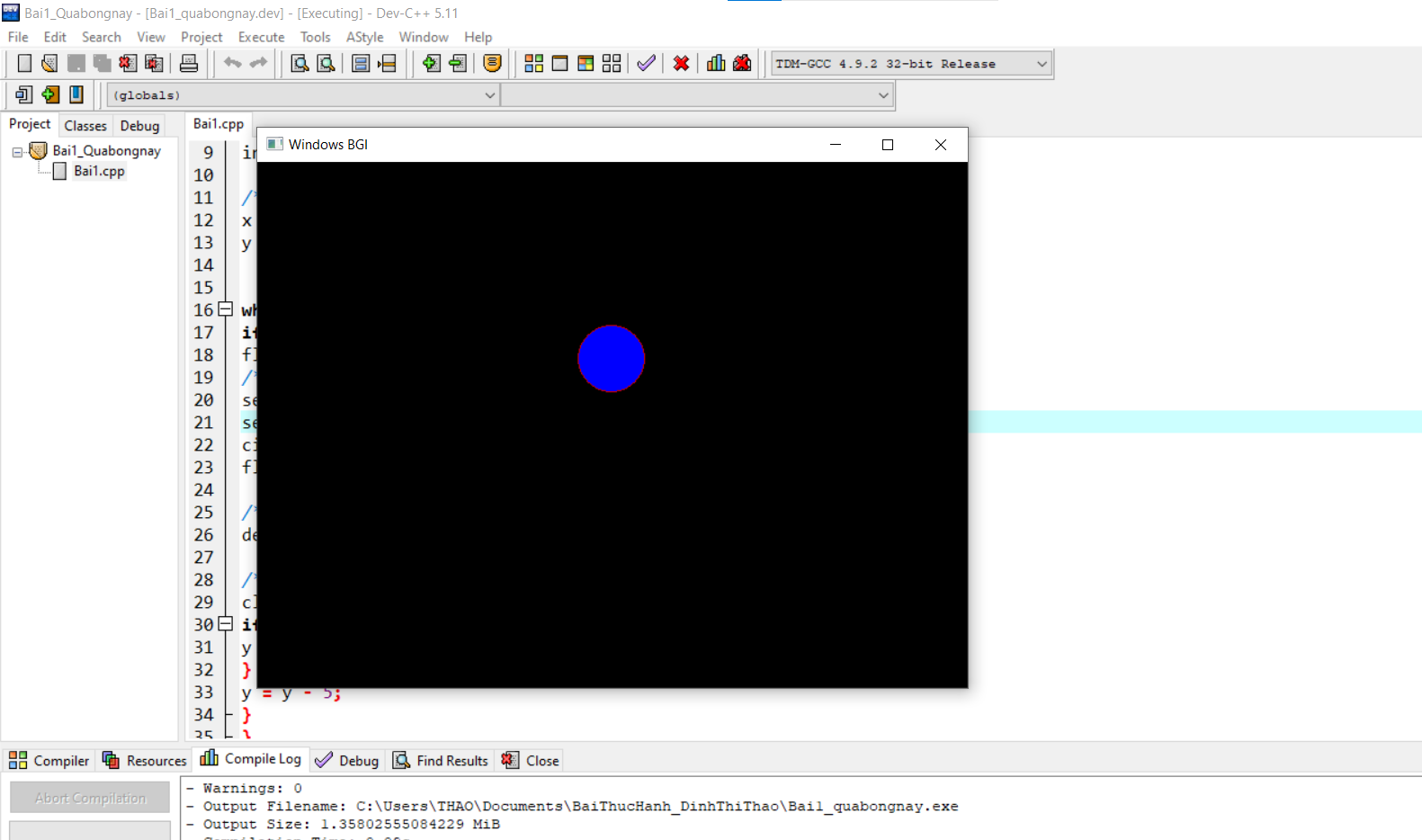
getch();

closegraph();

return 0;

}

**Kết quả:**



**Bài 14: Viết chương trình ngôi sao 5 cánh vừa quay vừa di chuyển**

**Code:**

#include <conio.h>

#include <iostream>

#include <graphics.h>

#include <stdlib.h>

using namespace std;

struct ToaDo

{

int x,y;

};

//==============================================

void nhapDaGiac(ToaDo p[], int v)

{

int i;

for(i=0;i<v; i++){

cout<<"\nNhap toa do dinh "<<i+1<<" : ";

cout<<"\n\tx["<<(i+1)<<"] = "; cin>>p[i].x;

cout<<"\n\ty["<<(i+1)<<"] = "; cin>>p[i].y;

}

p[i].x=p[0].x;

p[i].y=p[0].y;

}

//==============================================

void veDaGiac(ToaDo p[], int v)

{

for(int i=0;i<v;i++)

line(p[i].x,p[i].y,p[i+1].x,p[i+1].y);

}

//==============================================

void ScanLine(ToaDo p[], int v)

{

int xmin,xmax,ymin,ymax,c,mang[50];

xmin=xmax=p[0].x;

ymin=ymax=p[0].y;

for(int i=0;i<v;i++){ if(xmin>p[i].x) xmin=p[i].x;

if(xmax<p[i].x) xmax=p[i].x; if(ymin>p[i].y) ymin=p[i].y;

if(ymax<p[i].y) ymax=p[i].y;

}

float y;

y=ymin+0.01;

while(y<=ymax){ //v?i y tang d?n t? ymin > ymax,tìm các giao di?m c?a t?ng y v?i các c?p c?nh

int x,x1,x2,y1,y2,tg;

c=0; //ch? s? c?a m?ng ph?n t?

for(int i=0;i<v;i++){ //xét trên t?t c? các d?nh

//xét 2 d?nh li?n k? nhau

x1=p[i].x;

y1=p[i].y;

x2=p[i+1].x;

y2=p[i+1].y;

if(y2<y1){ //s?p x?p l?i y c?a 2 di?m liên ti?p

tg=x1;x1=x2;x2=tg;

tg=y1;y1=y2;y2=tg;

}

//m?ng giao di?m

if(y<=y2&&y>=y1){

if(y1==y2) x=x1; //n?u y c?a 2 d?nh liên ti?p trùng nhau => b? qua

else{

x=((y-y1)\*(x2-x1))/(y2-y1); //h? s? góc

x+=x1; //300

}

if(x<=xmax && x>=xmin)

mang[c++]=x; //cho ph?n t? c = x sau dó c++

}

}

//v?i t?ng y tang d?n ta v? luôn du?ng th?ng n?i 2 giao di?m

for(int i=0; i<c;i+=2){

delay(30);

line(mang[i],y,mang[i+1],y);

} //line(302,91,300,91

y++;

}

}

//nhap 10 dinh:

//1.(75,250),2.(210,250),3.(250,128),4.(291,250),5.(425,250)

//6.(318,331),7.(360,460),8.(249,380),9.(140,460),10.(182,331)

//3 7 1 5 9

int main()

{

int cl,v;

do{

cout<<"\n Nhap so dinh da giac:"; cin>>v;

}while(v<3);

ToaDo p[v];

nhapDaGiac(p,v);

cout<<"\nChon mau (0-15) : "; cin>>cl;

initwindow(500,600);

veDaGiac(p,v);

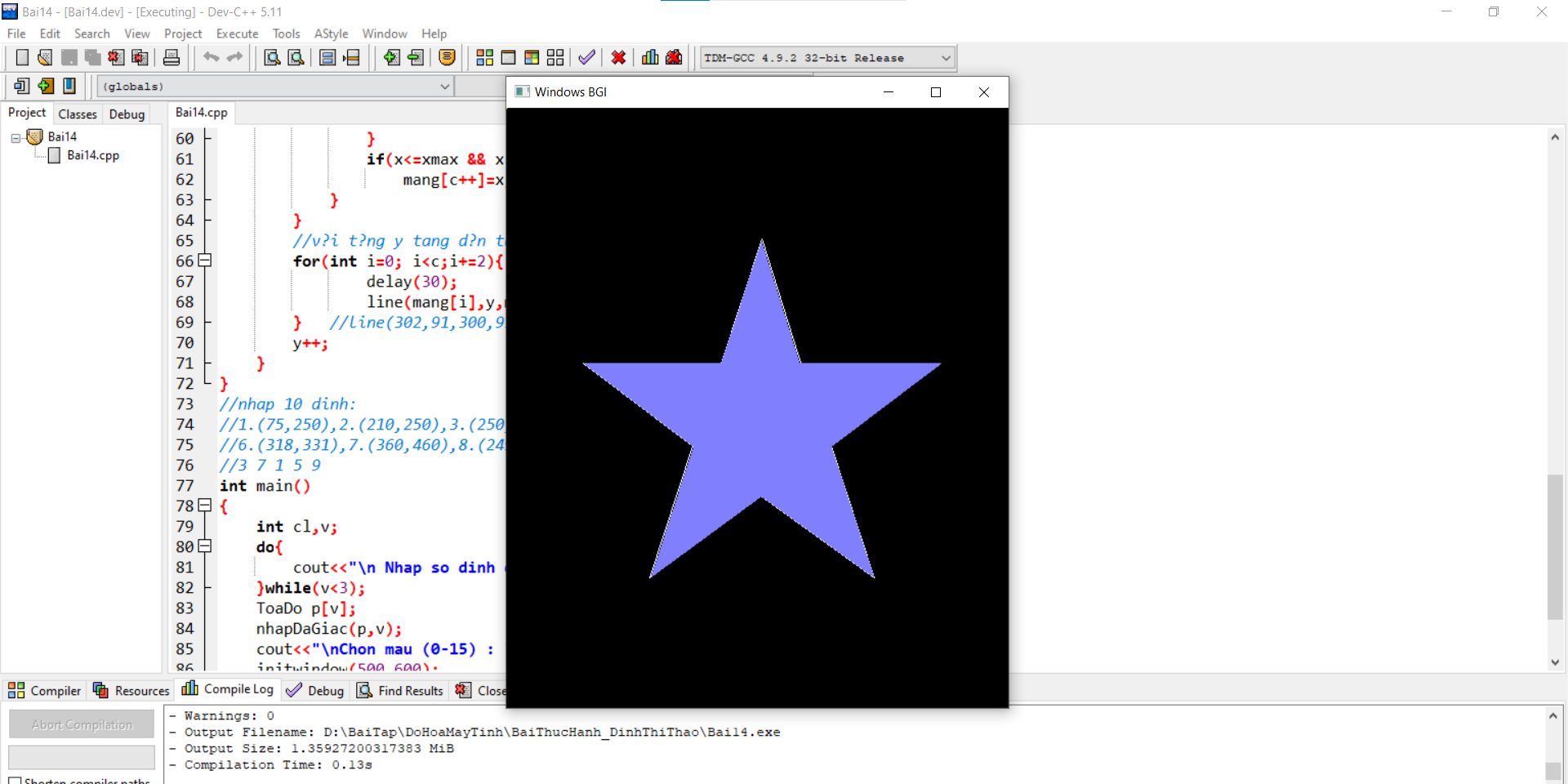
setcolor(cl);

ScanLine(p, v);

getch();

}

**Kết quả:**



**Bài 25: Viết chương trình vẽ đường tròn, tô màu đường tròn bằng giải thuật tô màu đơn giản**

**Code:**

#include <stdio.h>

#include <conio.h>

#include <graphics.h>

int main(){

initwindow(800,600);//tao cua so do hoa, size 800x600

int x=400,y=300,r=200;

int d=1;

int dl=200;

int n=15;

int color=3;

while(1){

for(int i=0;i<=n;i++){

setcolor(color-i);

circle(x,y,r-d\*i);

delay(dl);

}

setcolor(3);

// for(int i=n;i>=0;i--){

// circle(x,y,r-d\*i);

// delay(dl);

// }

}

getch();

return 0;

}

**Kết quả:**

