Министерство образования и науки РФ
Государственное образовательное учреждение
высшего профессионального образования

Тульский государственный университет

КАФЕДРА АВТОМАТИКИ И ТЕЛЕМЕХАНИКИ

СРЕДСТВА ГРАФИКИ BORLAND PASCAL И BORLAND C

Контрольно-курсовая работа

по курсу «Программирование на ЯВУ»

Вариант № 4

Выполнил: студент группы 220601 _____ Белым А.А.
(подпись)

Проверил: к. ф.-м. н., доцент _____ Сулимова В.В.
(подпись)

Тула 2011

Цель работы

Цель работы заключается в том, чтобы научиться использовать видео функции для работы в графическом режиме на языках Borland Pascal и Borland C.

Задание

Написать программу, которая выводит на экран изображение идущих часов, имеющих секундную и минутную стрелки.

Теоретическая справка

1. Схема алгоритма

На рисунке 1 представлена схема алгоритма рисования анимированных аналоговых часов.

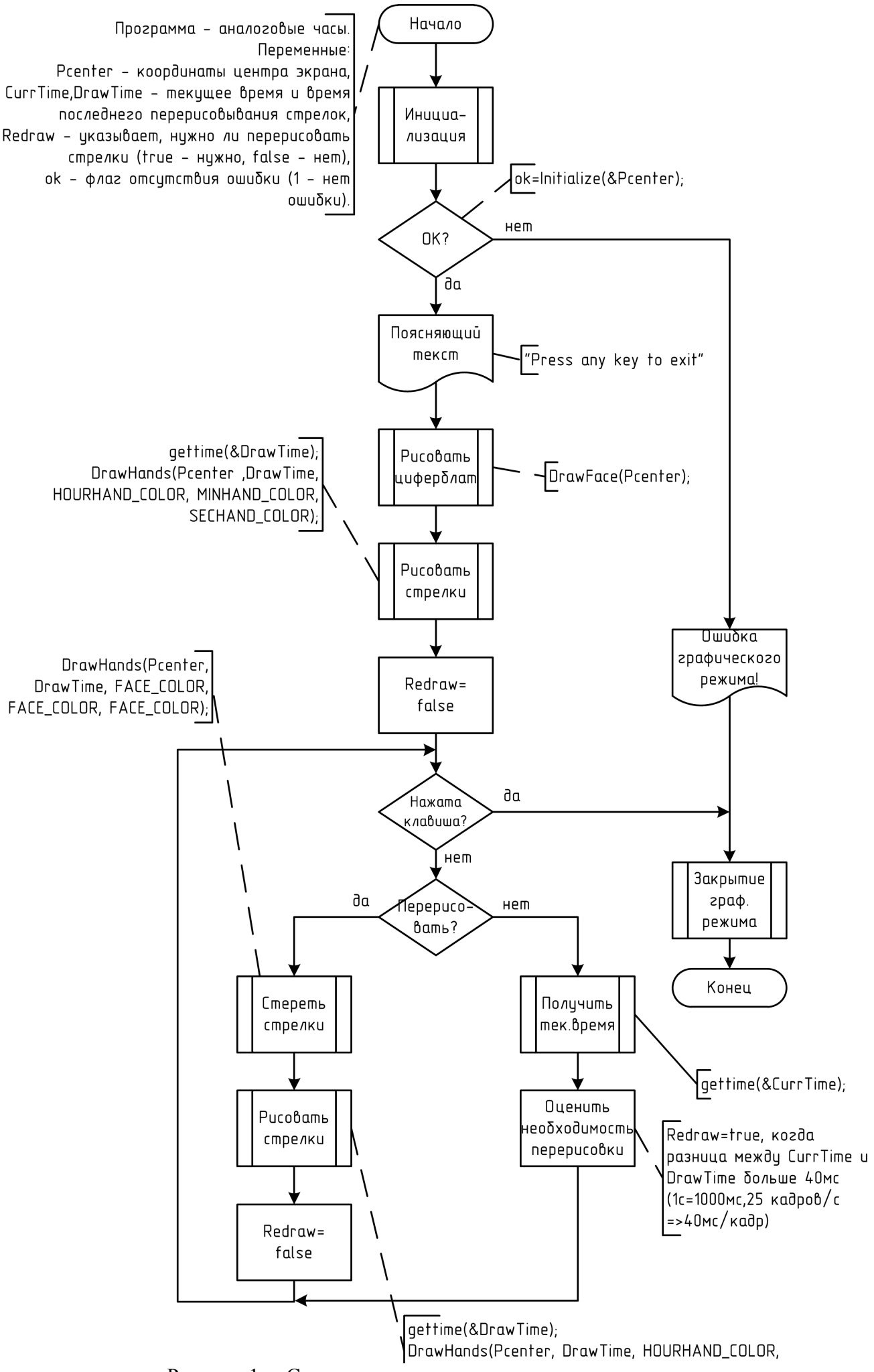

Рисунок 1 – Схема алгоритма рисования аналоговых часов

На рисунке 2 представлена схема алгоритма установки графического режима и нахождения центра экрана.

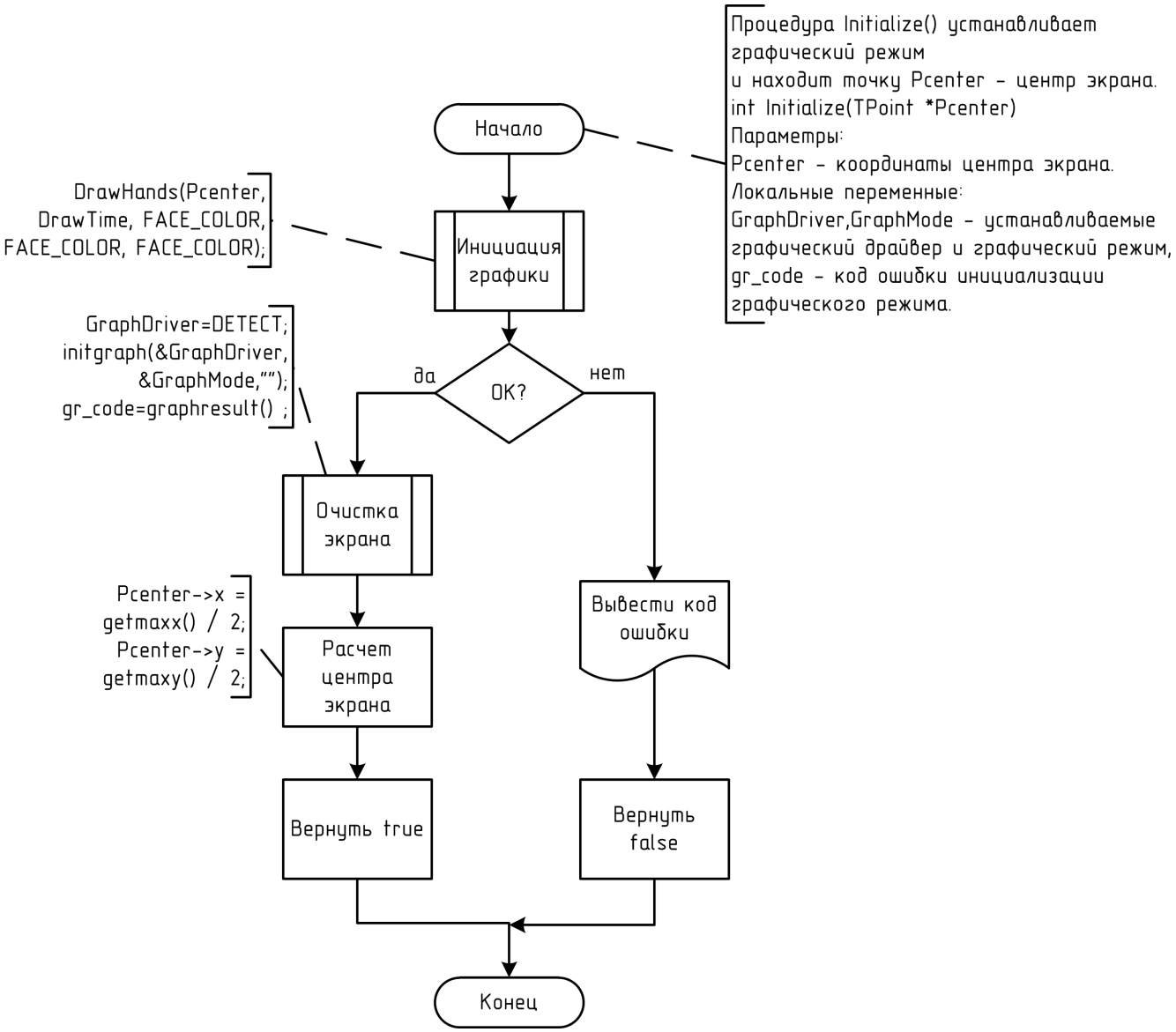

Рисунок 2 – Схема алгоритма установки графического режима

На рисунке 3 представлена схема алгоритма рисования циферблата.

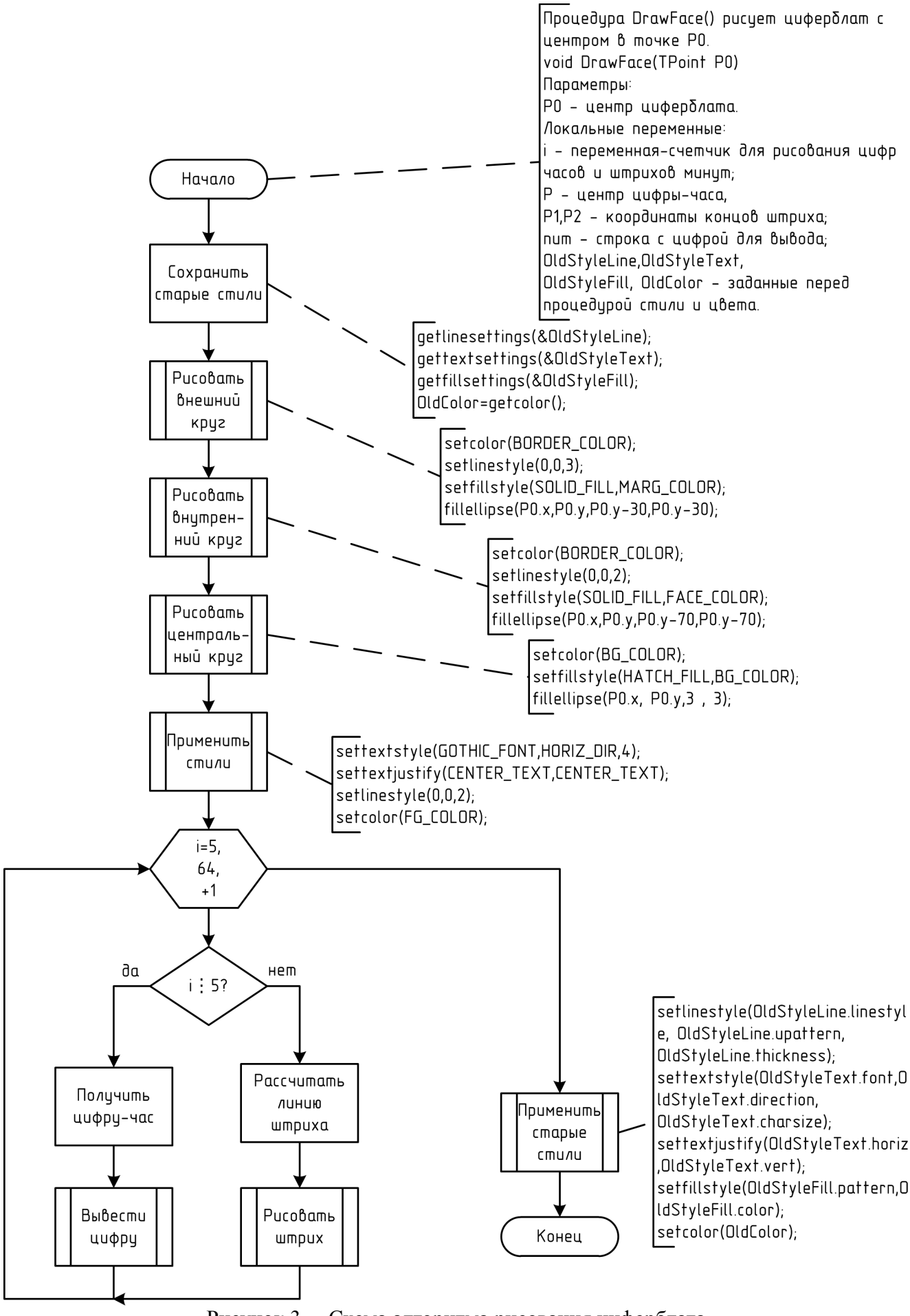

Рисунок 3 – Схема алгоритма рисования циферблата

На рисунке 4 представлена схема алгоритма рисования часовых стрелок.

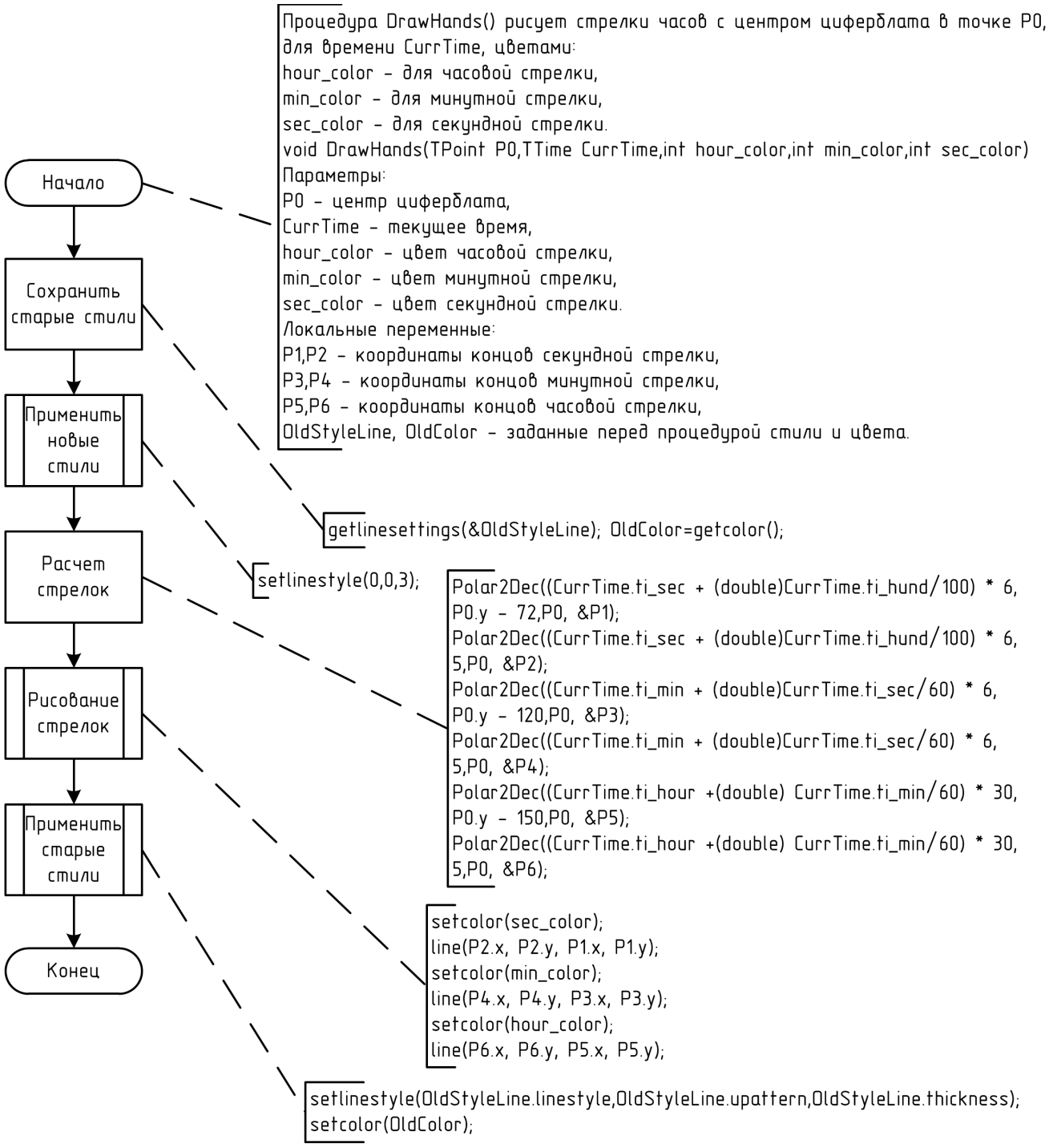

Рисунок 4 – Схема алгоритма рисования часовых стрелок

На рисунке 5 представлена схема алгоритма перевода координат точки из полярной системы в Декартову систему координат.

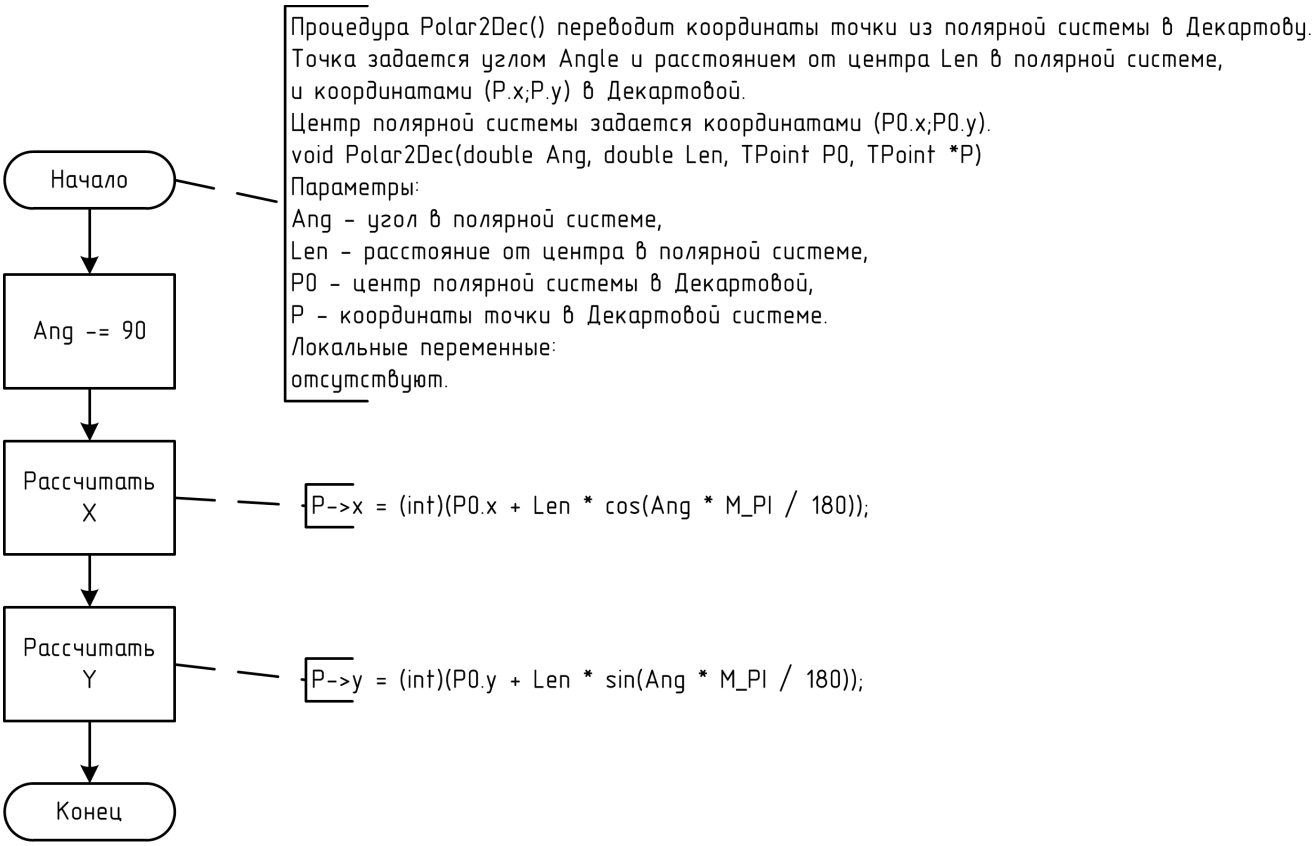

Рисунок 5 – Схема алгоритма преобразования координат

2. Инструкция пользователю

Данная программа изображает анимированные аналоговые часы.

Для использования просто запустите программу. Учтите, что в одном каталоге с программой должны находится файлы GOTHIC.CHR и драйвер EGAVGA.bgi (либо другой драйвер для вашего оборудования). Из-за отсутствия этих файлов, а также в некоторых других случаях программа завершит работу с сообщением об ошибке. Программа отображает время также как и любые другие аналоговые часы, например, механические. Чтобы завершить программу, нажмите любую клавишу.

3. Инструкция программисту

Для решения задачи рисования аналоговых часов были предприняты следующие действия.

Объявлены константы:

BORDER_COLOR - цвет границ циферблата,

BG_COLOR - цвет фона,

FG_COLOR - цвет надписей,

FACE_COLOR - цвет циферблата,

MARG_COLOR - цвет поля с цифрами и штрихами,

HOURHAND_COLOR - цвет часовой стрелки,

MINHAND_COLOR - цвет минутной стрелки,

SECHAND_COLOR - цвет секундной стрелки,

Подключены заголовочные файлы <stdio.h> для функций ввода-вывода, <dos.h> - для функций времени, <graphics.h> - для графического режима и <conio.h> - для управления клавиатурой в программе на языке Си.

Подключены модули dos - для функций времени, graph - для графического режима и crt - для управления клавиатурой в программе на языке Паскаль.

Обьявлены типы TTime – "время", его описание представлено в таблице 1, и Tpoint – координаты точки в Декартовой системе координат, его описание представлено в таблице 2.

Таблица 1 - Описание полей типа "время"

имя	**тип**		**предназначение**
	Pascal	**C**	
ti_hour, ti_min, ti_sec, ti_hund	word	unsigned char	Время: часы, минуты, секунды, сотые доли секунд.

Таблица 2 - Описание полей типа "координаты точки"

имя	**тип**		**предназначение**
	Pascal	**C**	
x,y	integer	int	координаты точки.

Объявлены следующие структуры данных, представленные в таблице 3:

Таблица 3 - Структуры данных, используемые в в основной части программы рисования часов

имя	**тип**		**предназначение**
	Pascal	**C**	
Pcenter	TPoint		координаты центра экрана,
CurrTime, DrawTime	TTime		текущее время и время последнего перерисовывания стрелок,
Redraw	boolean	int	указывает, нужно ли перерисовать стрелки (true - нужно, false - нет),
ok	boolean	int	флаг отсутствия ошибки (1 - нет ошибки).

Также программа была разбита на следующие подпрограммы:

1. Процедура Polar2Dec() переводит координаты точки из полярной системы в Декартову. Точка задается углом Angle и расстоянием от центра Len в полярной системе, и координатами (P.x;P.y) в Декартовой. Центр полярной системы задается координатами (P0.x;P0.y).

void Polar2Dec(double Ang, double Len, TPoint P0, TPoint *P)

procedure Polar2Dec(Ang, Len: Real;P0:TPoint; var P: Tpoint);

Параметры процедуры представлены в таблице 4:

Таблица 4 - Параметры процедуры преобразования координат

имя	**тип**		**предназначение**
	Pascal	**C**	
Ang	real	double	угол в полярной системе,
Len	real	double	расстояние от центра в полярной системе,
P0	TPoint		центр полярной системы в Декартовой,
P			координаты точки в Декартовой системе.

Локальные переменные процедуры отсутствуют.

2. Процедура DrawFace() рисует циферблат с центром в точке P0.

void DrawFace(TPoint P0);

procedure DrawFace(P0:TPoint);

Параметры процедуры представлены в таблице 5:

Таблица 5 - Параметры процедуры рисования циферблата

имя	**тип**		**предназначение**
	Pascal	**C**	
P0	TPoint		центр циферблата.

Локальные переменные процедуры представлены в таблице 6:

Таблица 6 - Локальные переменные процедуры рисования циферблата

имя	**тип**		**предназначение**
	Pascal	**C**	
i	integer	int	переменная-счетчик для рисования цифр часов и штрихов минут;
P	TPoint		центр цифры-часа,
P1,P2			координаты концов штриха;
num	string	char[5]	строка с цифрой для вывода;
OldStyleLine	linesettingstype	struct linesettingstype	заданные перед процедурой стили и цвета.
OldStyleText,	textsettingstype	struct textsettingstype	
OldStyleFill	fillsettingstype	struct fillsettingstype	
OldColor	integer	int	

3. Процедура DrawHands() рисует стрелки часов с центром циферблата в точке P0,для времени CurrTime, цветами: hour_color - для часовой стрелки, min_color - для минутной стрелки, sec_color - для секундной стрелки.

```
void DrawHands(TPoint P0,TTime CurrTime,
                                    int hour_color,int min_color,int sec_color);
Procedure DrawHands(P0:TPoint; CurrTime:TTime;
                                    hour_color, min_color, sec_color:word);
```

Параметры процедуры представлены в таблице 7:

Таблица 7 - Параметры процедуры рисования стрелок

имя	**тип**		**предназначение**
	Pascal	**C**	
P0	TPoint		центр циферблата,
CurrTime	TTime		текущее время,
hour_color	integer	int	цвет часовой стрелки,
min_color	integer	int	цвет минутной стрелки,
sec_color	integer	int	цвет секундной стрелки.

Локальные переменные процедуры представлены в таблице 8:

Таблица 8 - Локальные переменные процедуры рисования стрелок

имя	**тип**		**предназначение**
	Pascal	**C**	
P1,P2	TPoint		координаты концов секундной стрелки,
P3,P4			координаты концов минутной стрелки,
P5,P6			координаты концов часовой стрелки,
OldStyleLine	linesettingstype	struct linesettingstype	заданные перед процедурой стили и цвета.
OldColor	integer	int	

4. Функция Initialize() устанавливает графический режим и находит точку Pcenter - центр экрана.

Возвращает 1(true), если операция прошла успешно, иначе возвращается 0(false).

int Initialize(TPoint *Pcenter);

function Initialize(var Pcenter:TPoint):boolean;

Параметры функции представлены в таблице 9:

Таблица 9 - Параметры функции установки графического режима

имя	**тип**		**предназначение**
	Pascal	**C**	
Pcenter	TPoint		координаты центра экрана.

Локальные переменные функции представлены в таблице 10:

Таблица 10 - Локальные переменные функции установки графического режима

имя	**тип**		**предназначение**
	Pascal	**C**	
GraphDriver, GraphMode	integer	int	устанавливаемые графический драйвер и графический режим,
gr_code	integer	int	код ошибки инициализации графического режима.

4. Текст программы на языке Pascal

Ниже представлен текст программы на языке Borland Pascal 7.0 для рисования аналоговых часов.

```
uses graph, crt, dos;
Const BORDER_COLOR=Black;{цвет границ циферблата}
    BG_COLOR =DarkGray;{цвет фона}
      FG_COLOR=Black;{цвет надписей}
      FACE_COLOR=LightGray;{цвет циферблата}
      MARG_COLOR=White;{цвет поля с цифрами и штрихами}
      HOURHAND_COLOR=Blue;{цвет часовой стрелки}
      MINHAND_COLOR=Green;{цвет минутной стрелки}
      SECHAND_COLOR=Black;{цвет секундной стрелки}

type
      TPoint = record {точка}
            x, y: Integer; {координаты}
      end;
      TTime=record {время}
            ti_hour, ti_min, {часы, минуты,}
            ti_sec, ti_hund {секунды, миллисекунды}
            : Word;
      end;

(*
Процедура Polar2Dec() переводит координаты точки из полярной системы в Декартову.
Точка задается углом Angle и расстоянием от центра Len в полярной системе,
и координатами (P.x;P.y) в Декартовой.
Центр полярной системы задается координатами (P0.x;P0.y).
Параметры:
Ang - угол в полярной системе,
Len - расстояние от центра в полярной системе,
P0 - центр полярной системы в Декартовой,
P - координаты точки в Декартовой системе.
Локальные переменные:
отсутствуют.
*)
procedure Polar2Dec(Ang, Len: Real;P0:TPoint; var P: TPoint);
begin
      Ang := Ang - 90; { Correlation for our coord system }
      P.x := Round(P0.x + Len * cos(Ang * Pi / 180));
      P.y := Round(P0.y + Len * sin(Ang * Pi / 180));
end;
```

```
(*
Процедура DrawFace() рисует циферблат с центром в точке P0.
Параметры:
P0 - центр циферблата.
Локальные переменные:
i - переменная-счетчик для рисования цифр часов и штрихов минут;
P - центр цифры-часа,
P1,P2 - координаты концов штриха;
num - строка с цифрой для вывода;
OldStyleLine,OldStyleText,
OldStyleFill, OldColor - заданные перед процедурой стили и цвета.
*)
procedure DrawFace(P0:TPoint);
var i:byte; P,P1,p2:TPoint; num:string;
    OldStyleLine:LineSettingsType; OldStyleText:TextSettingsType;
    OldStyleFill:FillSettingsType; OldColor:integer;
begin
      GetLineSettings(OldStyleLine); GetTextSettings(OldStyleText);
      GetFillSettings(OldStyleFill); OldColor:=GetColor;

      SetColor(BORDER_COLOR);
      SetLineStyle(0,0,3);
      SetFillStyle(SolidFill,MARG_COLOR);
      FillEllipse(P0.x,P0.y,P0.y-30,P0.y-30);

      SetColor(BORDER_COLOR);
      SetLineStyle(0,0,2);
      SetFillStyle(SolidFill,FACE_COLOR);
      FillEllipse(P0.x,P0.y,P0.y-70,P0.y-70);

      SetColor(BG_COLOR);
      SetFillStyle(HatchFill,BG_COLOR);
      FillEllipse(P0.x, P0.y,3 , 3);

      SetTextStyle(GothicFont,HorizDir,4);
      SetTextJustify(CenterText,CenterText);
      SetLineStyle(0,0,2);
      SetColor(FG_COLOR);

      for i := 5 to 64 do    begin
            if i mod 5 =0 then begin
            Polar2DEc(i*6, P0.y - 50,P0, P);
            str(i div 5,num);
            OutTextXY(Round(P.x),Round(P.y)-7,num);
            end else begin
```

```
            Polar2Dec(i * 6, P0.y - 70,p0, P1);
            Polar2Dec(i * 6, P0.y-60,p0, P2);
            Line(p1.x,p1.y,p2.x,p2.y);
            end;
      end;

      with oldStyleLine do
         setLineStyle(LineStyle,Pattern,Thickness);
      with oldStyleText do begin
         setTextStyle(Font,Direction,CharSize);
         setTExtJustify(horiz,vert);
      end;
      with oldStyleFill do
         setFillStyle(Pattern,Color);
      setColor(OldColor);
end;

(*
Процедура DrawHands() рисует стрелки часов с центром циферблата в точке P0,
для времени CurrTime, цветами:
hour_color - для часовой стрелки,
min_color - для минутной стрелки,
sec_color - для секундной стрелки.
Параметры:
P0 - центр циферблата,
CurrTime - текущее время,
hour_color - цвет часовой стрелки,
min_color - цвет минутной стрелки,
sec_color - цвет секундной стрелки.
Локальные переменные:
P1,P2 - координаты концов секундной стрелки,
P3,P4 - координаты концов минутной стрелки,
P5,P6 - координаты концов часовой стрелки,
OldStyleLine, OldColor - заданные перед процедурой стили и цвета.
*)
Procedure DrawHands(P0:TPoint;CurrTime:TTime;hour_color,min_color,sec_color:word);
var P1,p2,p3,p4,p5,p6:TPoint;
OldStyleLine:LineSettingsType; oldColor:integer;
begin

      GetLineSettings(OldStyleLine); OldColor:=GetColor;
      SetLineStyle(0,0,3);
      { Second arrow }
      Polar2Dec((CurrTime.ti_sec + CurrTime.ti_hund/100) * 6, p0.y - 72,p0, P1);
      Polar2Dec((CurrTime.ti_sec + CurrTime.ti_hund/100) * 6, 5,p0, P2);
```

```
      { Minute arrow }
      Polar2Dec((CurrTime.ti_min + CurrTime.ti_sec/60) * 6, p0.y - 120,p0, P3);
      Polar2Dec((CurrTime.ti_min + CurrTime.ti_sec/60) * 6, 5,p0, P4);

      { Hour arrow }
      Polar2Dec((CurrTime.ti_hour + CurrTime.ti_min/60) * 30, p0.y - 150,p0, P5);
      Polar2Dec((CurrTime.ti_hour + CurrTime.ti_min/60) * 30, 5,p0, P6);

      { Draw }
      SetColor(sec_color);
      SetColor(sec_color);
      Line(P2.x, P2.y, P1.x, P1.y);

      SetColor(min_color);
      Line(P4.x, P4.y, P3.x, P3.y);

      SetColor(hour_color);
      Line(P6.x, P6.y, P5.x, P5.y);

      with oldStyleLine do
         setLineStyle(LineStyle,Pattern,Thickness);
      SetColor(oldColor);
end;

(*
Процедура Initialize() устанавливает графический режим
и находит точку Pcenter - центр экрана.
Параметры:
Pcenter - координаты центра экрана.
Локальные переменные:
GraphDriver,GraphMode - устанавливаемые графический драйвер и графический режим,
gr_code - код ошибки инициализации графического режима.
*)
function Initialize(var Pcenter:TPoint):boolean;
var GraphDriver,GraphMode,gr_code:integer;
begin
      GraphDriver:=DETECT;
      InitGraph(GraphDriver, GraphMode, '');
      gr_code:=GraphResult;
      if gr_code=grOK then begin
            ClearDevice;
            SetFillStyle(SolidFill,BG_COLOR);
            floodfill(0,0,BG_COLOR);
            Pcenter.x := GetMaxX div 2;
```

```
            PCenter.y := GetMaxY div 2;
            Initialize:=true;
      end else begin
            WriteLn('Ошибка инициализации графического режима #',gr_code);
            Initialize:=false;
      end;
end;

(*
Программа - аналоговые часы.
Переменные:
Pcenter - координаты центра экрана,
CurrTime,DrawTime - текущее время и время последнего перерисовывания стрелок,
Redraw - указывает, нужно ли перерисовать стрелки (true - нужно, false - нет),
ok - флаг отсутствия ошибки (1 - нет ошибки).
*)
var ok:boolean;
      Pcenter:TPoint;
      Redraw:boolean;
      CurrTime,DrawTime:TTime;
begin
      ok:=Initialize(Pcenter);

      if ok then begin
            SetTextStyle(GothicFont,HorizDir,4);
            SetTextJustify(CenterText,CenterText);
            SetColor(FG_COLOR);
            OutTextXY(Pcenter.x,7, 'Press any key to exit');
            OutTextXY(Pcenter.x,GetMaxY-21, 'Press any key to exit');

            DrawFace(Pcenter);

            GetTime(DrawTime.ti_hour, DrawTime.ti_min,
                              DrawTime.ti_sec, DrawTime.ti_hund);
            DrawHands(Pcenter,DrawTime,HOURHAND_COLOR,
                                 MINHAND_COLOR, SECHAND_COLOR);
            Redraw:=false;

            while not keypressed do begin
                  if Redraw then begin
                        { Erase }
                        DrawHands(Pcenter,DrawTime,FACE_COLOR,
                                    FACE_COLOR,FACE_COLOR);

                        GetTime(DrawTime.ti_hour, DrawTime.ti_min,
```

```
                                    DrawTime.ti_sec, DrawTime.ti_hund);
                        DrawHands(Pcenter,DrawTime,HOURHAND_COLOR,
                                    MINHAND_COLOR,SECHAND_COLOR);

                        Redraw:=False;
                    end else begin
                        GetTime(CurrTime.ti_hour, CurrTime.ti_min,
                                    CurrTime.ti_sec, CurrTime.ti_hund);
                        Redraw:=((CurrTime.ti_hour-DrawTime.ti_hour)<>0)
                              or((CurrTime.ti_min-DrawTime.ti_min)<>0)
                               or ((CurrTime.ti_sec-DrawTime.ti_sec)<>0)
                               or(abs(CurrTime.ti_hund- DrawTime.ti_hund)>4);
                    end;
                end;
                CloseGraph;
            end else WriteLn('Программа прервана!');
end.
```

5. Текст программы на языке Си

Ниже представлен текст программы на языке Borland C 3.1 для рисования аналоговых часов.

```
#include <stdio.h>
#include <graphics.h>
#include <conio.h>
#include <dos.h>
#include <math.h>
const int BORDER_COLOR=BLACK, // цвет границ циферблата
    BG_COLOR=DARKGRAY, // цвет фона
      FG_COLOR=BLACK, // цвет надписей
      FACE_COLOR=LIGHTGRAY, // цвет циферблата
      MARG_COLOR=WHITE, // цвет поля с цифрами и штрихами
      HOURHAND_COLOR=BLUE, // цвет часовой стрелки
      MINHAND_COLOR=GREEN, // цвет минутной стрелки
      SECHAND_COLOR=BLACK; // цвет секундной стрелки
typedef struct { //точка
      int x, y; //координаты
} TPoint;
typedef struct time TTime; //время
/*
Процедура Polar2Dec() переводит координаты точки из полярной системы в Декартову.
Точка задается углом Angle и расстоянием от центра Len в полярной системе,
и координатами (P.x;P.y) в Декартовой.
Центр полярной системы задается координатами (P0.x;P0.y).
Параметры:
Ang - угол в полярной системе,
Len - расстояние от центра в полярной системе,
P0 - центр полярной системы в Декартовой,
P - координаты точки в Декартовой системе.
Локальные переменные:
отсутствуют.
*/
void Polar2Dec(double Ang, double Len, TPoint P0, TPoint *P){
      Ang -= 90;
      P->x = (int)(P0.x + Len * cos(Ang * M_PI / 180));
      P->y = (int)(P0.y + Len * sin(Ang * M_PI / 180));
};
/*
Процедура DrawFace() рисует циферблат с центром в точке P0.
Параметры:
P0 - центр циферблата.
Локальные переменные:
```

```
i - переменная-счетчик для рисования цифр часов и штрихов минут;
P - центр цифры-часа,
P1,P2 - координаты концов штриха;
num - строка с цифрой для вывода;
OldStyleLine,OldStyleText,
OldStyleFill, OldColor - заданные перед процедурой стили и цвета.
*/
void DrawFace(TPoint P0){
  int i; TPoint P,P1,P2; char num[5];
  struct linesettingstype OldStyleLine; struct textsettingstype OldStyleText;
  struct fillsettingstype OldStyleFill; int OldColor;
      getlinesettings(&OldStyleLine); gettextsettings(&OldStyleText);
      getfillsettings(&OldStyleFill); OldColor=getcolor();

      setcolor(BORDER_COLOR);
      setlinestyle(0,0,3);
      setfillstyle(SOLID_FILL,MARG_COLOR);
      fillellipse(P0.x,P0.y,P0.y-30,P0.y-30);

      setcolor(BORDER_COLOR);
      setlinestyle(0,0,2);
      setfillstyle(SOLID_FILL,FACE_COLOR);
      fillellipse(P0.x,P0.y,P0.y-70,P0.y-70);

      setcolor(BG_COLOR);
      setfillstyle(HATCH_FILL,BG_COLOR);
      fillellipse(P0.x, P0.y,3 , 3);

      settextstyle(GOTHIC_FONT,HORIZ_DIR,4);
      settextjustify(CENTER_TEXT,CENTER_TEXT);
      setlinestyle(0,0,2);
      setcolor(FG_COLOR);

      for (i = 5; i<65;i++) {
            if (!(i%5)){
                  Polar2Dec(i*6, P0.y - 50,P0, &P);
                  sprintf(num,"%d",i/5);
                  outtextxy((int)(P.x),(int)(P.y)-7,num);
            } else {
                  Polar2Dec(i * 6, P0.y - 70,P0,&P1);
                  Polar2Dec(i * 6, P0.y-60,P0, &P2);
                  line(P1.x,P1.y,P2.x,P2.y);
            };
      };
```

```
      setlinestyle(OldStyleLine.linestyle,
                           OldStyleLine.upattern,OldStyleLine.thickness);
      settextstyle(OldStyleText.font,OldStyleText.direction,
                        OldStyleText.charsize);
      settextjustify(OldStyleText.horiz,OldStyleText.vert);
      setfillstyle(OldStyleFill.pattern,OldStyleFill.color);
      setcolor(OldColor);
};
/*
Процедура DrawHands() рисует стрелки часов с центром циферблата в точке P0,
для времени CurrTime, цветами:
hour_color - для часовой стрелки,
min_color - для минутной стрелки,
sec_color - для секундной стрелки.
Параметры:
P0 - центр циферблата,
CurrTime - текущее время,
hour_color - цвет часовой стрелки,
min_color - цвет минутной стрелки,
sec_color - цвет секундной стрелки.
Локальные переменные:
P1,P2 - координаты концов секундной стрелки,
P3,P4 - координаты концов минутной стрелки,
P5,P6 - координаты концов часовой стрелки,
OldStyleLine, OldColor - заданные перед процедурой стили и цвета.
*/
void DrawHands(TPoint P0,TTime CurrTime,int hour_color,int min_color,int
sec_color){
  TPoint P1,P2,P3,P4,P5,P6;
  struct linesettingstype OldStyleLine; int OldColor;
      getlinesettings(&OldStyleLine); OldColor=getcolor();

      setlinestyle(0,0,3);

      Polar2Dec((CurrTime.ti_sec + (double)CurrTime.ti_hund/100) * 6,
                                                          P0.y - 72,P0, &P1);
      Polar2Dec((CurrTime.ti_sec + (double)CurrTime.ti_hund/100) * 6, 5,P0, &P2);

      Polar2Dec((CurrTime.ti_min + (double)CurrTime.ti_sec/60) * 6,
                                                       P0.y - 120,P0, &P3);
      Polar2Dec((CurrTime.ti_min + (double)CurrTime.ti_sec/60) * 6, 5,P0, &P4);

      Polar2Dec((CurrTime.ti_hour +(double) CurrTime.ti_min/60) * 30,
                                                     P0.y - 150,P0, &P5);
      Polar2Dec((CurrTime.ti_hour +(double) CurrTime.ti_min/60) * 30, 5,P0, &P6);
```

```
      setcolor(sec_color);
      line(P2.x, P2.y, P1.x, P1.y);

      setcolor(min_color);
      line(P4.x, P4.y, P3.x, P3.y);

      setcolor(hour_color);
      line(P6.x, P6.y, P5.x, P5.y);
      setlinestyle(OldStyleLine.linestyle,OldStyleLine.upattern,
                        OldStyleLine.thickness);
      setcolor(OldColor);
};

/*
Процедура Initialize() устанавливает графический режим
и находит точку Pcenter - центр экрана.
Параметры:
Pcenter - координаты центра экрана.
Локальные переменные:
GraphDriver,GraphMode - устанавливаемые графический драйвер и графический режим,
gr_code - код ошибки инициализации графического режима.
*/
int Initialize(TPoint *Pcenter){
int GraphDriver,GraphMode,gr_code;
      GraphDriver=DETECT;
      initgraph(&GraphDriver, &GraphMode,"");
    gr_code=graphresult() ;
      if (gr_code==grOk) {
            cleardevice();
            setfillstyle(SOLID_FILL,BG_COLOR);
            floodfill(0,0,BG_COLOR);
            Pcenter->x = getmaxx() / 2;
            Pcenter->y = getmaxy() / 2;
            return 1;
      } else {
            printf("Ошибка инициализации графического режима #%d",gr_code);
            return 0;
      }
};

/*
Программа - аналоговые часы.
Переменные:
Pcenter - координаты центра экрана,
```

```
CurrTime,DrawTime - текущее время и время последнего перерисовывания стрелок,
Redraw - указывает, нужно ли перерисовать стрелки (true - нужно, false - нет),
ok - флаг отсутствия ошибки (1 - нет ошибки).
*/
int main(void){
  int ok;
  TPoint Pcenter;
  int WaitEnded;
  TTime CurrTime,DrawTime;
      ok=Initialize(&Pcenter);
      if (ok)   {
            settextstyle(GOTHIC_FONT,HORIZ_DIR,4);
            settextjustify(CENTER_TEXT,CENTER_TEXT);
            setcolor(FG_COLOR);
            outtextxy(Pcenter.x,7, "Press any key to exit");
            outtextxy(Pcenter.x,getmaxy()-21, "Press any key to exit");

            DrawFace(Pcenter);

            gettime(&DrawTime);
            DrawHands(Pcenter,DrawTime,HOURHAND_COLOR,
                        MINHAND_COLOR,SECHAND_COLOR);
            WaitEnded=0;

            while (!kbhit()){
                  if (WaitEnded){
                        DrawHands(Pcenter,DrawTime,FACE_COLOR,
                                    FACE_COLOR,FACE_COLOR);

                        gettime(&DrawTime);
                        DrawHands(Pcenter,DrawTime,HOURHAND_COLOR,
                                     MINHAND_COLOR,SECHAND_COLOR);
                        WaitEnded=0;
                  } else {
                        gettime(&CurrTime);
                        WaitEnded=(CurrTime.ti_hour-DrawTime.ti_hour)||
                                    (CurrTime.ti_min-DrawTime.ti_min)
                                    ||(CurrTime.ti_sec-DrawTime.ti_sec)
                                      ||(abs(CurrTime.ti_hund- DrawTime.ti_hund)>4);
                  };
            };
            closegraph();
      } else printf("Программа прервана!");
    return 0;
}
```

6. Тестовый пример

Ниже на рисунке 6 приведен пример работы программы, рисующей анимированные аналоговые часы.

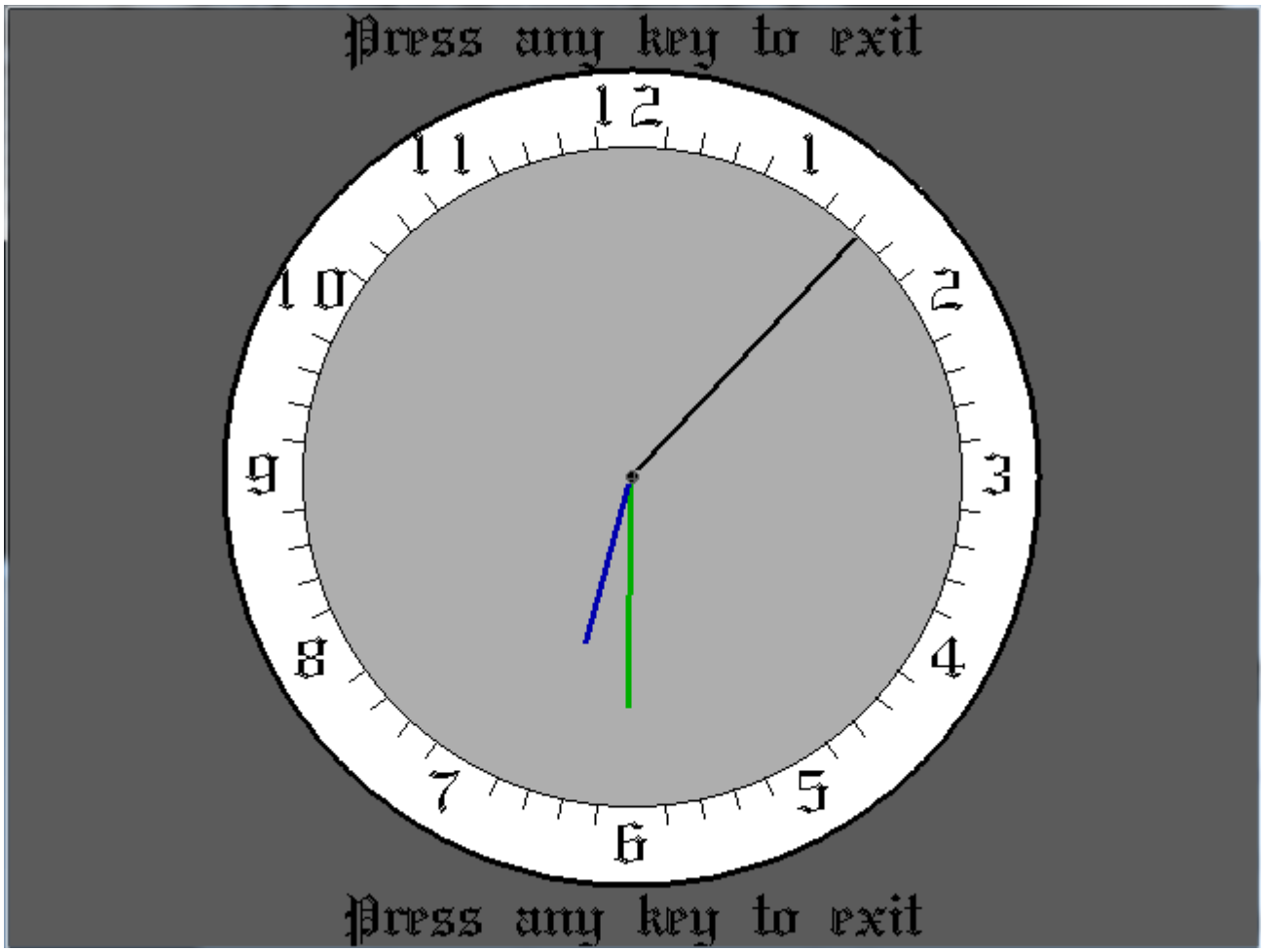

Рисунок 6 — Пример работы программы рисования аналоговых часов

Вывод

В ходе выполнения данной лабораторной работы я научился использовать функции и процедуры графического режима в программах на языках Си и Паскаль. Графический режим позволяет программисту управлять каждым пикселом на экране, что позволяет строить изображения различной степени сложности — от графиков функций и графических окон до анимированных интерактивных изображений. На сегодняшний день графический режим является основным режимом работы видеоадаптера. Сейчас многие вообще не могут представить компьютер без графической оконной среды. Конечно, для работы с графическим режимом требуется более высокая квалификация программиста.