УДК 004.43

Учебные демонстрационные программы на C++ в BORLAND C++ BUILDER 6

Л. В. Ландик, И. В. Пестренина
Пермский государственный национальный исследовательский университет
Россия, 614990, г. Пермь, ул. Букирева, 15
Ipestrenina@gmail.com; 8 (342) 2 396 375

Приведено несколько demo-программ на алгоритмическом языке C++ в объектно-ориентированной среде компилятора Borland C++ Builder 6, показывающие некоторые "современные" требования к программным продуктам и реализуемым возможностям. Это, во-первых, создание формы на экране дисплея для визуализации задачи, размещение на ней объектов-компонент, предлагающих интерактивное выполнение необходимых действий, а также возможности графики, сервисных функций для работы с файлами (поиск, чтение, запись) и запуск приложений.

Ключевые слова: *объектно-ориентированные языки программирования; C++.*

DOI: 10.17072/1993-0550-2018-4-60-73

В основе предлагаемых программ использованы некоторые demo-модули компилятора CBuilder 6, которые переработаны и дополнены для более полного представления начинающим программистам о возможностях применения различных компонент языка C++.

Рассмотрено применение следующих объектов и событий, это –

– графические компоненты (**PaintBox, Image, Chart**); при построении графика на объекте типа **Image** средствами **Canvas** приводятся формулы, связывающие истинные координаты графика с графическими координатами экрана дисплея;

– компоненты выбора цвета (**ColorGrid, ColorBox**); событие **OnChange** в объекте типа **ColorGrid** или событие **OnClick** в объекте **ColorBox** отслеживают и выполняют оперативное изменение выбранного цвета;

– группа быстрых кнопок **SpeedButton** для выбора инструмента последующего графического рисования с помощью курсора, управляемого "мышью": **PencilButton** – для рисования тонких линий; **FillButton** – для закрашивания области, ограниченной тонкими линиями; **EraseButton** – для стирания рисунка; **CircleButton и SquareButton** – для рисования тонких линий (окружность или эллипс и квадрат или прямоугольник); **SolidClrButton и SolidSqButton** – для рисования сплошных "закрашенных" геометрических тел (круг/ эллипс, квадрат/ прямоугольник); для оперативного отслеживания движения "мыши" предусмотрена обработка событий **OnMouseDown, OnMouseMove, OnMouseUp**, в которой определяются графические координаты "мыши";

– компонент **Shape** для отображения на форме различных геометрических фигур;

– компонент панель заголовков (**HeaderControl**) с помощью выполнения события **OnSectionTrack** настраивает размеры других объектов (например, **Shape**) под размеры разделов панели заголовков;

– компонент текстовой области **Memo** для ввода и хранения, например, справочной информации;

– невизуальный компонент **Timer** и событие **OnTimer** моделируют процессы, связанные с реальным временем (например, вращение нарисованных фигур);

– компонент главное меню (объект **MainMenu**) с помощью события **OnClick** запускает на выполнение выбранный элемент из **MainMenu,** это может быть некоторая функция или самостоятельный exe-модуль.

Предусмотрено копирование графических объектов в буфер (**Clipboard Windows**) для сохранения в Word-документе.

В приложениях часто нужно предусмотреть поиск файлов без online-выхода из выполняемого модуля в среду операционной системы. А так как предлагаемых функций в алгоритмических языках (в C++ это **FindFirst, FindNext, FileSearch**) недостаточно, то предложены демонстрационная программа, функции поиска файлов и запуска на выполнение найденного exe-модуля. Предлагаемые функции могут быть полезны начинающим программистам.

Все demo-программы работают в интерактивном режиме, для изменения параметров, выбора цвета, инструментов, конкретных действий используются объекты-компоненты – **LabelEdit**, панель группы переключателей **RadioGroup**, кнопки **Button**, панель страниц-вкладок **PageControl** и др. На панели типа **StatusBar** или в заголовке всех программ приведена справочная информация – на какой вкладке компилятора находятся необходимые объекты.

1. Модуль Canvas _new3.exe (проект **Canvas_new.bpr**, рис. 1, 2) демонстрирует построение графических фигур на объекте типа **PaintBox** (вкладка System) с помощью графического средства – **Canvas** (холст). Это прямоугольная поверхность, на которой выполняется рисование функцией-обработчиком события **OnPaint.** Для размещения на рисунке различных фигур нужно определить **центр рисунка** и **структурный массив их направляющих косинусов**. Реализовано построение и режим вращения 12 графических фигур. Выбор и активация нужного рисунка (**int num**) выполняется в интерактивном режиме с помощью объекта типа **ComboBox** (вкладка Standard), а режим вращения (**int koef**) – на объекте типа **RadioGroup**. Графические фигуры 1–11 состоят из окружностей и отрезков, соединяющих все базовые точки на этих окружностях, а 12-я демонстрирует движение "колеса" по большой круговой орбите (число базовых точек 16).

Для фигур 1–11 предусмотрено интерактивное определение числа базовых точек (**int PointCount**) на окружностях и рисующего цвета (**TColor col**) на объекте типа **ColorGrid** (вкладка Samples).

Предусмотрено сохранение активного рисунка в буфере Windows.

На рис. 1 приведен вид формы demo-модуля со всеми объектами и активным рисунком 12, а на рис. 2 – сохраненные копии рисунков 1 и 7.

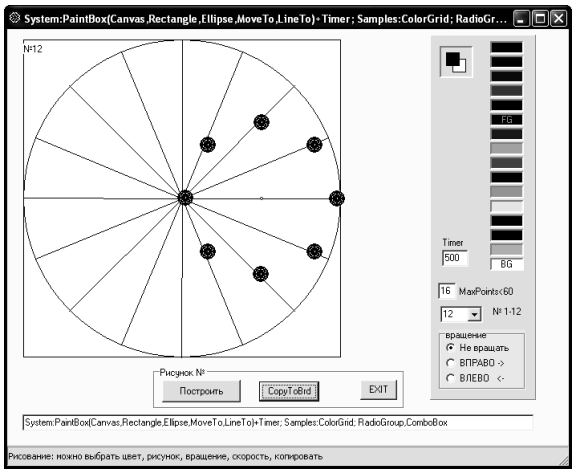

Рис. 1. *Копия формы demo-модуля (num=12)*

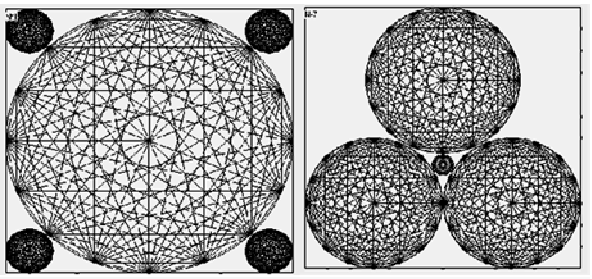

Рис. 2. *Копии рисунков (num=1, num=7)*

При выборе рисунка (объект **ComboBox1**, событие **OnClick**) управление передается на кнопку "**Построить**" (функция **Button3Click**), определяются направляющие косинусы соответствующих фигур рисунка и его полная перерисовка. Приведем фрагмент функции.

```
void __fastcall TFormMain::Button3Click(TObject *Sender)
{  int i,nn,angle;
   nn=ComboBox1->ItemIndex+1;
   switch  (nn)
```

```
 { case 1 :
   for (i = 0, angle = 45; i < 4; i++, angle += 90)
   { P8[i].X = cos(angle*M_PI/180.0); P8[i].Y = sin(angle*M_PI/180.0);  }
   break;
//……..
 case 11:
   for (i = 0, angle =0; i < 4; i++, angle += 90)
     { P8[i].X = cos(angle*M_PI/180.0); P8[i].Y = sin(angle*M_PI/180.0);  }
   for (i = 0, angle =45; i < 4; i++, angle += 90)
     { P11[i].X = cos(angle*M_PI/180.0); P11[i].Y = sin(angle*M_PI/180.0);  }
   break;
  }
  col = ColorGrid1->ForegroundColor;
   i=interval; interval=StrToIntDef(Led_time->Text,i);
   if(nn==12)
   {nc=0; nred=0; Rotation=0; MaxPoints=16; Led_max->Text=IntToStr(MaxPoints);}
    if(nn==13)   {nc=0;  Rotation=0;}
 Led_time->Text=IntToStr(interval);   Timer1->Enabled=false;
 Timer1->Interval=interval; Timer1->Enabled=true;
 PointCount=MaxPoints;   RotatePoints(); FormPaint(Sender);}
```

Для реализации эффекта вращения активного рисунка введен невизуальный объект – **TTimer *Timer1**, его основное свойство **Timer1–>Interval** содержит интервал времени между генерацией события **OnTimer**, т.е. скорость вращения активного рисунка. Приведем функцию **Led_timeClick**, определяющую в интерактивном режиме скорость вращения (в мс) и последующую работу модуля под управлением таймера.

```
void __fastcall TFormMain::Led_timeClick(TObject *Sender)
{  int i; i=interval;
   interval=StrToIntDef(Led_time->Text,i);
   if (interval<250) interval=max(i,250);
   Led_time->Text=IntToStr(interval);
   Timer1->Enabled=false;
   Timer1->Interval=interval; Timer1->Enabled=true;
   RotatePoints(); FormPaint(Sender);   }
```

Если входные параметры (номер рисунка, режим, цвет) не меняются, то работу модуля под управлением таймера автономно определяет событие **OnTimer** и соответствующая функция-обработчик **Timer1Timer.**

```
void __fastcall TFormMain::Timer1Timer(TObject *Sender)
{      RotatePoints();   if (RadioGroup1->ItemIndex==0) return;     Invalidate(); }
```

Для полной перерисовки рисунка здесь использован метод **Invalidate,** который позволяет избежать мерцания изображения. Кроме того, для правильной перерисовки на главной форме нужно предусмотреть событие **OnResize** и его функцию-обработчик **FormResize.**

```
void __fastcall TFormMain::FormResize(TObject *Sender)
{          Invalidate();   }
```

Каждый шаг работы таймера и всех случаев интерактивного изменения входных параметров (**num** – номер рисунка, **PointCount** – число базовых точек, **col** – выбранный цвет, **koef** – направление вращения) заканчивается обращением к двум функциям, это **RotatePoints()** и функция-обработчик **FormPaint (Sender)** для события **OnPaint.**

Для их выполнения введены геометрические параметры и структурные массивы для вычисления направляющих косинусов центров окружностей и их базовых точек на очередном шаге вращения:

а) угол поворота за 1 шаг вращения (**2*pi/32** для графических фигур 1–11 и **2pi/8** для фигуры 12);

б) центральный угол (**float StepAngle**) между базовыми точками, т.е. угловое расстояние между точками;

в) угол **(float Rotation)**, накапливается с учетом направления вращения на каждом следующем шаге возникновения события **OnTimer;**

г) **struct TRPoint { float X, Y; };**
TRPoint Points[200], P8[10], P11[10], P12[10]; – для направляющих косинусов.

Углы **StepAngle** и **Rotation** нужны для определения на каждом шаге вращения новых координат базовых точек.

Функция **RotatePoints()** – корректирует угол **Rotation**, вычисляет относительно центра вращения (центр рисунка) направляющие косинусы в массиве **Points** для определения координат базовых точек окружностей, а для фигуры 12 вычисляет в массиве **P12** направляющие косинусы мини-кругов относительно центра "колеса" и очередной точки касания. Мини-круги при движении "колеса" двигаются по хордам большой окружности.

Для определения центров окружностей на каждом рисунке в **P8** и **P11** вычисляются их направляющие косинусы.

```
void __fastcall TFormMain::RotatePoints()
{     float angle,angl2;
      num=ComboBox1->ItemIndex+1;     if ( num==12)  PointCount=16;
      StepAngle = M_2PI /PointCount;      // angular distance between points
      switch (RadioGroup1->ItemIndex)   // режим направления вращения
      {case 0: koef=0; break;              // не вращать
       case 1: koef=1; break;              // вправо
       case 2: koef=-1; break;       }    // влево    // end switch
      if (num==12)  Rotation += koef*M_PI / 8;       // угол поворота рисунка  по  OnTimer
      if (num<12) Rotation += koef*M_PI / 32;        // Increment the angle of rotation of figure
         if ((Rotation > StepAngle)&&(num<12)) Rotation -= StepAngle;  angl2=Rotation;
         for (int i = 0, angle = Rotation; i < PointCount; i++, angle += StepAngle)
      {  Points[i].X = cos(angle);                  // These values will be multiplied by the
         Points[i].Y = sin(angle);                  // current radius at display time.
         if((num==12)&&(i<PointCount/2) )
            {  if (i>0) angl2=angl2+2*StepAngle;
              P12[i].X=cos(angl2);  P12[i].Y=sin(angl2);  }
         }
}  // end
```

Функция **FormPaint(TObject *Sender)** – обработчик события **OnPaint,** выполняет рисование активного рисунка на объекте **PaintBox1** с учетом его реальных размеров. Оператор **switch(num)** управляет выбором рисунка. Приведем фрагмент функции **FormPaint** для двух фигур (**num**=1 и 12).

```
void __fastcall TFormMain::FormPaint(TObject *Sender)
{     int centerX = PaintBox1->Width / 2;              // координаты центра вращения
      int centerY = PaintBox1->Height / 2;
     int radius = std::min(centerY, centerX);         // радиус большой окружности
      PaintBox1->Color=clWhite;                         // цвет фона
      PaintBox1->Canvas->Rectangle(0, 0, radius*2, radius*2); // рамка
      num=ComboBox1->ItemIndex+1;                  // номер рисунка
     switch  (num)
        { case 1: goto one;
//. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
        case 12: goto twelve;
         default: num=1; ComboBox1->ItemIndex=num-1; goto end;
        }
// 1 big circle+4 small. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
one:    r1=radius*(3.0-sqrt(8.0));  dr=(radius-r1)*sqrt(2.0);
```

```
      for (i = 0; i < 4; i++) {
      x0=radius + floor(P8[i].X * dr);
      y0=radius + floor(P8[i].Y * dr);
      y1=y0-r1; y2=y0+r1;   x1=x0-r1; x2=x0+r1;
       PaintBox1->Canvas->Ellipse(x1, y1, x2, y2);
        for (int i1 = 0; i1 < PointCount; i1++) {
            for (j = i1 + 1; j < PointCount; j++)
       {  PaintBox1->     Canvas->MoveTo(x0 + floor(Points[i1].X * r1),
                                         y0 + floor(Points[i1].Y * r1));
             PaintBox1-> Canvas->LineTo(x0 + floor(Points[j].X * r1),
                                         y0 + floor(Points[j].Y * r1));
          } } //j,i
       }
//------------------------------ big circle
 PaintBox1->Canvas->Ellipse(0, 0, radius*2, radius*2);
                    for (i = 0; i < PointCount; i++) {
                    for (j = i + 1; j < PointCount; j++) {
             PaintBox1-> Canvas->MoveTo(radius + floor(Points[i].X * radius),
                                         radius + floor(Points[i].Y * radius));
             PaintBox1-> Canvas->LineTo(radius + floor(Points[j].X * radius),
                                         radius + floor(Points[j].Y * radius));
          } }
goto end;
//. . . . . .wheel in circle . . . . . .
twelve:    PaintBox1->Canvas->Pen->Color = clBlack;
 PaintBox1->Canvas->Ellipse(0, 0, radius*2, radius*2);  npt=PointCount;
    for (i = 0; i < npt; i++)                                    //-------  big circle
{    PaintBox1->Canvas->MoveTo(radius + floor(Points[i].X * radius),
                                    radius + floor(Points[i].Y * radius));
     PaintBox1->Canvas->LineTo(radius,radius);   }
//  ---- центр колеса, касающегося большого круга
     xc0=radius+floor(P12[0].X * radius)/2;
     yc0=radius+floor(P12[0].Y * radius)/2;
   PaintBox1->Canvas->Ellipse(xc0-2, yc0-2, xc0+2, yc0+2);
   rad_1=radius/2;
// ------координаты центров мини-кругов колеса
    xc1=xc0+floor(P12[0].X * rad_1*0.9);
    yc1=yc0+floor(P12[0].Y * rad_1*0.9);
    if((0==nred)) PaintBox1->Canvas->Pen->Color = clRed;
    else  PaintBox1->Canvas->Pen->Color = clBlack;
    for ( k = 10; k >0; k--)
    {PaintBox1->Canvas->Ellipse(xc1-k, yc1-k, xc1+k, yc1+k);}
//    ------ остальные 7 мини-кругов колеса.
     PaintBox1->Canvas->Pen->Color = clBlack;
     for (i = 1; i < npt/2; i++)
     { if((i==nred)) PaintBox1->Canvas->Pen->Color = clRed;
       else  PaintBox1->Canvas->Pen->Color = clBlack;
       xc1=xc0+floor(P12[i].X * rad_1*0.9);      //nn
       yc1=yc0+floor(P12[i].Y * rad_1*0.9);
       for (k = 10; k >0; k--)
       {PaintBox1->Canvas->Ellipse(xc1-k, yc1-k, xc1+k, yc1+k);}
     }
 PaintBox1->Canvas->Pen->Color = clBlack;
   if(koef>0) {nc=nc+1; if (nc==8) nc=0;nred=nred-1; if(nred<0)nred=7; }
```

```
   if(koef<0) {nc=nc-1; if (nc<0) nc=7; nred=nred+1; if(nred==8)nred=0;}
    StatusBar1->SimpleText=IntToStr(nc)+"+"+IntToStr(nred);
  goto end ;
// . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
 end:   PaintBox1->Canvas->Pen->Color = col;
PaintBox1->Canvas->TextOut(0,5,"¹"+IntToStr(num));
  Application->ProcessMessages();   }
```

Выбор цвета для фигур 1–11 выполняет функция **CColorGrid1Change** (приведем фрагмент)

```
void __fastcall TFormMain::CColorGrid1Change(TObject *Sender).
{   TColor col = ColorGrid1->ForegroundColor;      //рисующий цвет
//. . . . . .
    PaintBox1->Canvas->Pen->Color = col;
//. . . . . .
}
```

Сохранение рисунка из **PaintBox1** в **ClipBoard Windows** выполняет функция **Button2Click**.

```
void __fastcall TFormMain::Button2Click(TObject *Sender)
{ int x0,y0,d_x,d_y;
  Graphics::TBitmap* Bitmap=new Graphics::TBitmap;
//левый верхний угол
// P=PaintBox1->ClientToScreen(P);
    x0 = P.x; y0  = P.y; d_x=PaintBox1->Width;  d_y=PaintBox1->Height;
   Bitmap->Width =d_x;    Bitmap->Height=d_y;
   BitBlt(Bitmap->Canvas->Handle,0,0,d_x,d_y,GetDC(0),x0,y0,SRCCOPY);
   Clipboard()->Assign(Bitmap);
   StatusBar1->SimpleText="Копирование рисунка в ClipBoard";
   delete  Bitmap; }
```

2. Модуль Doodle_new.exe (проект **Doodle_new.bpr**, рис. 3) демонстрирует процесс графического рисования на экране движением "мыши" на объекте типа **Image.** Выбор "инструмента" (линии, примитивы) выполняют быстрые кнопки **SpeedButton** (вкладка Additional), расположенные на форме demo-модуля в объекте **PageControl** на вкладке "**Tool**" (рис. 3, справа).

Реализованы быстрые кнопки-"инструменты": **PencilButton** – рисование тонких линий; **FillButton** – закрашивание области, ограниченной тонкими линиями; **EraseButton** – стирание рисунка; **CircleButton и SquareButton** – рисование тонкими линиями фигур-примитивов (круг/эллипс, квадрат/прямоугольник); **SolidClrButton и SolidSqButton** – рисование **сплошных закрашенных геометрических тел** (круг/ эллипс, квадрат/ прямоугольник). Выбор быстрой кнопки на вкладке "**Tool**" (событие **OnClick**) – с помощью курсора, управляемого "мышью", а для оперативного отслеживания ее движения (т.е. процесса рисования) предусмотрены события **OnMouseDown, OnMouseMove, OnMouseUp и их обработчики**, которые определяют графические координаты "мыши".

Цвет рисования определяется на вкладке "**Color**" на форме demo-модуля. Для создания этой вкладки использован объект типа **ColorGrid** (вкладка Samples).

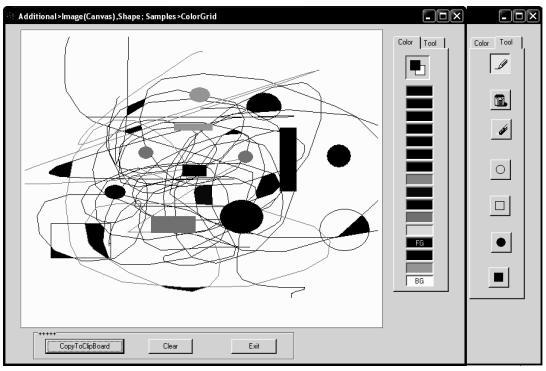

Рис. 3. *Копия формы demo-модуля Doodle_new*

Чтобы увидеть особенности реализации нужно установить соответствие – **события** и их **функции-обработчики**:

а) **OnCreate, OnActivate, OnClose** для всей формы – функции **FormCreate, FormActivate, FormClose** для создания базовых средств рисования, активации размеров формы, закрытия формы;

б) **OnChange** для объекта **ColorGrid1** на вкладке **Color** формы demo-модуля – функция **ColorGrid1Change** (см. п. 1), выбирает рисующий цвет;

в) **OnClick** быстрых кнопок **FillButton, PencilButton, EraseButton** на вкладке **Tool** формы – функции **FillButtonClick, PencilButtonClick, EraseButtonClick** для активации выбранного простейшего рисующего "примитива";

г) **OnClick** быстрых кнопок **CircleButton, SquareButton, SolidClrButton, SolidSqButton** – функция **ShapeButtonClick** для активации рисующих фигур (круг, эллипс, квадрат, прямоугольник и др.);

д) **OnMouseDown, OnMouseMove, OnMouseUp** для объекта **Image1** – функции **Image1MouseDown, Image1MouseMove, Image1MouseUp** для отслеживания движения курсора-мыши и рисования линий стандартными графическими функциями **MoveTo, LineTo** в C++, фигур с помощью функции **DrawShape**, созданной в проекте demo-модуля.

Сохранение рисунка в буфере Windows выполняет кнопка "**CopyToClipBoard**" (функция **Button1Click).** Приведем некоторые функции.

```
void __fastcall TDoodleForm::FormCreate(TObject *Sender)
{
   HINSTANCE HInst;
   HInst = reinterpret_cast<HINSTANCE>(HInstance);
   // Load custom cursors for tools from extrares.res
   Screen->Cursors[crFill] = LoadCursor(HInst, "FILL");
   Screen->Cursors[crPlus] = LoadCursor(HInst, "PLUS");
   Screen->Cursors[crDraw] = LoadCursor(HInst, "DRAW");
   Screen->Cursors[crErase] = LoadCursor(HInst, "ERASE");
   Cursor = TCursor(crDraw);
   DoodleForm->Caption="Additional>Image(Canvas),Shape; Samples>ColorGrid";
}
//---------------------------------------------------------------------------
void __fastcall TDoodleForm::FormActivate(TObject *Sender)
{
    if (DoodleForm->Visible)       return;
   DoodleForm->Top = DoodleForm->Top + (DoodleForm->Height / 6);
   DoodleForm->Left = DoodleForm->Left - ((DoodleForm->Width / 5));
   DoodleForm->Show();}
//---------------------------------------------------------------------------
void __fastcall TDoodleForm::PencilButtonClick(TObject *Sender)
{   DoodleForm->Cursor = TCursor(crDraw);   }
//---------------------------------------------------------------------------
void __fastcall TDoodleForm::ShapeButtonClick(TObject *Sender)
{   DoodleForm->Cursor = TCursor(crPlus);    }
//---------------------------------------------------------------------------
void __fastcall TDoodleForm::Image1MouseUp(TObject *Sender, TMouseButton Button,
   TShiftState Shift, int X, int Y)
{   // Only the left mouse button is of interest
   if (Button != mbLeft)       return;
   if ((DoodleForm->FillButton->Down) || (DoodleForm->PencilButton->Down))     return;
   if (DoodleForm->EraseButton->Down)
   {       Image1->Canvas->Pen->Width = 1;     return;     }
   DrawShape(X, Y);     delete TmpImage;}
//---------------------------------------------------------------------------
```

```
void __fastcall TDoodleForm::DrawShape(int X, int Y)
{   TRect bounds;  // for graphics functions which require a rect
    Image1->Picture = TmpImage->Picture;
   Image1->Canvas->Brush->Color = DoodleForm->FGShape->Brush->Color;
   Image1->Canvas->Pen->Color = DoodleForm->FGShape->Brush->Color;
   if (X < InitialX)
   {  bounds.Left = X;      bounds.Right = InitialX;    }
   else    {  bounds.Right = X;     bounds.Left = InitialX;     }
   if (Y < InitialY)
   {  bounds.Top = Y;        bounds.Bottom = InitialY;   }
   else
   {   bounds.Bottom = Y;    bounds.Top = InitialY;     }
   // Draw the circle or square using the corresponding function.
   if (DoodleForm->CircleButton->Down)
      Image1->Canvas->Arc(InitialX, InitialY, X, Y, X, Y, X, Y);
   else if (DoodleForm->SolidCirButton->Down)
      Image1->Canvas->Ellipse(InitialX, InitialY, X, Y);
   else if (DoodleForm->SquareButton->Down)
      Image1->Canvas->FrameRect(bounds);
   else if (DoodleForm->SolidSqButton->Down)
      Image1->Canvas->FillRect(bounds);
}
//---------------------------------------------------------------------------
void __fastcall TDoodleForm::Button1Click(TObject *Sender)
{  Clipboard()->Assign(DoodleForm->Image1->Picture->Bitmap); }
```

3. Модуль Function_new.exe (проект Function_new.bpr, рис. 4) демонстрирует построение графика функции **Y=F(X,m)** на объекте типа **Image** (вкладка Additional) с помощью графического средства – **Canvas.** На рис. 4 – копия формы demo-модуля.

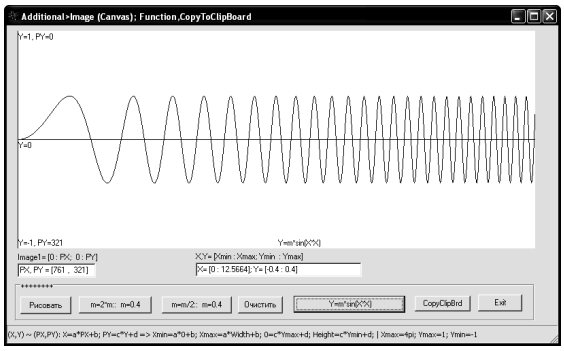

Рис. 4. *Копия формы demo-модуля Function*

Связь истинных координат **(X,Y)** графика с графическими координатами **(PX,PY)** объекта типа **Image** имеет вид:

X=a*PX+b; PY=c*Y+d .

Параметры связи **a, b, c, d** определяются из очевидных условий

Xmin=a*0+b; Xmax=a*Width+b;
0=c*Ymax+d; Height=c*Ymin+d.

В demo-примерах принято:
Xmin=0; Xmax=4pi; Ymax=1; Ymin=-1,
m – интерактивный масштабный коэффициент.

Рассмотрим некоторые функции-кнопки demo-модуля.

Функция **Button1Click** (**Button1 -> Caption ="Рисовать")** реализует рисование выбранного графика с учетом обрезания, если он выходит за выбранный диапазон.

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{  float X, Y,ymin0,ymax0;              // Переменные функции
int PX, PY,kk;                          // Координаты пикселов
Form1->Button4Click(0);  kk=0;       // Очистить область графика
Image1->Canvas->MoveTo(0, Image1->Height/2);          // ось OX
Image1->Canvas->LineTo(Image1->Width, Image1->Height/2);
Image1->Canvas->MoveTo(0, 0);                         //ось OY
Image1->Canvas->LineTo(0,Image1->Height);
Image1->Canvas->MoveTo(0, Image1->Height/2);
Xmin=0;Ymin=100; Xmax=0; Ymax=0;
```

```
for (PX=0; PX<=Image1->Width; PX++)
{  X=PX*4*PI/Image1->Width;
 switch (n)      //номер функции
 {  case 1: Y=m*sin(X);        break;
//. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
  case 5: Y=m*(cos(X)+sin(X)); break;
  default: break;
 }
 PY=Image1->Height-(Y+1)*Image1->Height/2;
 if (PY<0)  { PY=0; ymax0=1.0; kk=1;}
if (PY>Image1->Height )
 { PY=Image1->Height-1; ymin0=2.0/Image1->Height-1; kk=1;}
 Image1->Canvas->LineTo(PX, PY);
Xmin=std::min(Xmin,X);Xmax=std::max(Xmax,X);
Ymin=std::min(Ymin,Y);Ymax=std::max(Ymax,Y); }
//. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
}
```

Функция-кнопка **Button4Click** выполняет очистку области рисования, т.е. **Image1**:

```
void __fastcall TForm1::Button4Click(TObject *Sender)
{ Image1->Canvas->FillRect(Rect (0,0,Image1->Width, Image1->Height));}
```

Остальные кнопки – это изменение **m**-параметра, выбор функции и построение графика прямым переключением на кнопку "**Рисовать**" одним оператором **Form1->Button1Click(0);**

Функция **Button6Click** копирует график из объекта **Image1** в буфер **ClipBoard Windows**

```
void __fastcall TForm1::Button6Click(TObject *Sender)
{   Clipboard()->Assign(Form1->Image1->Picture->Bitmap);  }
```

4. Модуль My_Prim.exe (проект My_prim.bpr, рис. 5) демонстрирует часто применяемые объекты и их возможности, это **PageControl** (вкладка Win32) с набором вложенных страниц-вкладок; компонент текстовая область **Memo** (вкладка Standard) – для информации о модуле и предлагаемых действиях; объекты типа **LabelEdit** (вкладка Additional) для интерактивного ввода параметров; стандартный графический объект **Chart-диаграммы** (вкладка Additional) для эффективного и удобного построения графиков; невизуальные компоненты **SaveDialog и OpenDialog** (вкладка Dialogs) для стандартного online-выбора файла при выполнении операций чтения-записи массивов данных; набор объектов **Button** для выполнения предусмотренных действий.

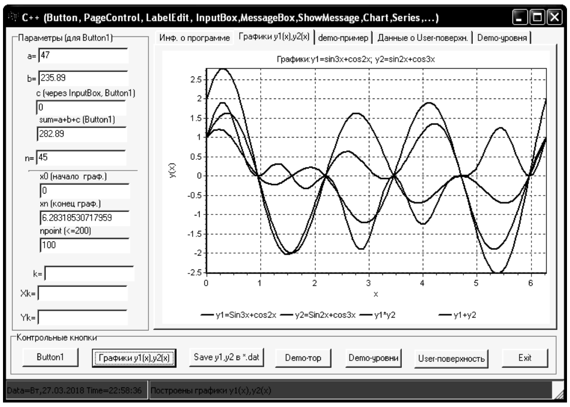

Рис. 5. *Копия формы demo-модуля **My_prim***

Реализованы некоторые технические возможности стандартных средств **Chart**-диаграмм [1–4], например, оперативное определение координат точки, выбранной курсором "мыши" на графике (событие **OnClickSeries**), программное оперативное изменение цвета графика или его части.

Сохранение **Chart**-диаграммы в буфере Windows (событие **OnClick**) выполняет функция-обработчик **Chart1Click.**

```
void __fastcall TForm1::Chart1Click(TObject *Sender)
{  OpenClipboard;
   switch ( PageControl1->TabIndex )
  { case 0:    Chart1->CopyToClipboardBitmap();   break;    }
  CloseClipboard;  }
```

5. Модуль Header_new.exe (проект Header_new.bpr) демонстрирует применение объекта панель заголовков типа **HeaderControl** (вкладка Win32) и объектов типа **Shape (**вкладка Additional). На рис. 6 приведена копия формы demo-модуля.

Размеры рисуемых фигур подгоняются в demo-модуле под размеры разделов заголовков панели **HeaderControl** или определяются в интерактивном режиме (события **OnSectionTrack и OnChange**). Для этого введены массивы **int H[3],W[3].**

Движение "мыши" по панели заголовка порождает событие **OnSectionTrack,** которое вызывает функцию

HeaderControl1SectionTrack

для корректировки ширины полей как панели заголовка, так и объектов типа **Shape**, полей массива **W[3]** и соответствующих объектов типа **LabelEdit (Led_W0,Led_W1,Led_W2).**

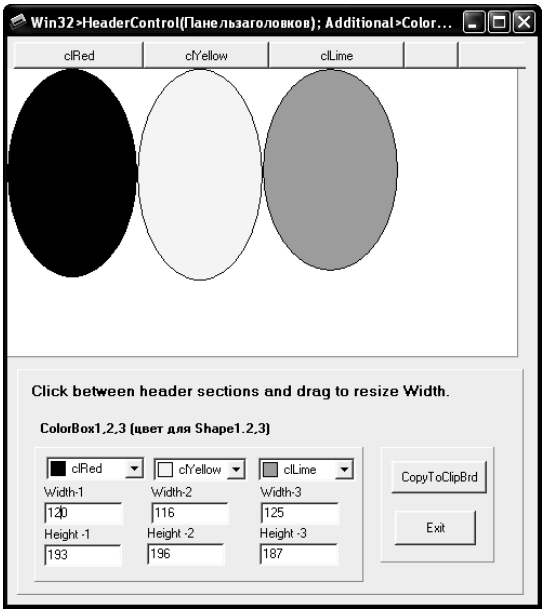

Рис. 6. *Копия формы модуля Header new*

```
void __fastcall TFormMain::HeaderControl1SectionTrack(
    THeaderControl *HeaderControl, THeaderSection *Section,
    int Width, TSectionTrackState State)
{     Section->Width = Width;
      Shape1->Width = HeaderControl1->Sections->Items[0]->Width;
      Shape2->Width = HeaderControl1->Sections->Items[1]->Width;
      Shape2->Left = HeaderControl1->Sections->Items[1]->Left;
      Shape3->Width = HeaderControl1->Sections->Items[2]->Width;
      Shape3->Left = HeaderControl1->Sections->Items[2]->Left;
      W[0]=Shape1->Width; W[1]=Shape2->Width; W[2]=Shape3->Width;
      Led_W0->Text=IntToStr(W[0]);       Led_W1->Text=IntToStr(W[1]);
      Led_W2->Text=IntToStr(W[2]);    }
```

Любое интерактивное изменение ширины графических объектов типа **Shape** (показанной на форме в объектах **Led_W0, Led_W1, Led_W2**) приводит к выполнению функции **Led_W0Change,** т.е. к перерисовке **Shape**-объектов и перенастройке панели заголовков.

```
void __fastcall TFormMain::Led_W0Change(TObject *Sender)
{  W[0]=StrToInt(Led_W0->Text); W[1]=StrToInt(Led_W1->Text);
   W[2]=StrToInt(Led_W2->Text);
 Shape1->Width=W[0];
 HeaderControl1->Sections->Items[0]->Width=Shape1->Width;
 Shape2->Width=W[1];  Shape2->Left=Shape1->Left+W[0];
 HeaderControl1->Sections->Items[1]->Width=Shape2->Width;
```

```
Shape3->Width=W[2]; Shape3->Left=Shape2->Left+W[1];
HeaderControl1->Sections->Items[2]->Width=Shape3->Width;  }
```

Изменение высоты всех фигур (например, **Shape1**) контролируется функцией-обработчиком события **OnChange**, т.е. функцией **Led_H0Change** и приводит к перерисовке **Shape**-объектов.

```
void __fastcall TFormMain::Led_H0Change(TObject *Sender)
{ H[0]=StrToInt(Led_H0->Text); Shape1->Height=H[0];
}
```

Цвет фигур определяется с помощью объектов типа **ColorBox** (вкладка Additional).

Приведем функцию, определяющую цвет, например, для фигуры **Shape1** и соответствующую корректировку на панели заголовков.

```
void __fastcall TFormMain::ColorBox1Click(TObject *Sender)
{ Shape1->Brush->Color=ColorBox1->Selected;
  HeaderControl1->Sections->Items[0]->Text=
       ColorBox1->Items->Strings[ColorBox1->ItemIndex];
}
```

Предусмотрено копирование в буфер (функция-кнопка **Button1Click**). Возможны варианты.

```
void __fastcall TFormMain::Button1Click(TObject *Sender)
{  Graphics::TBitmap*  Bitmap = new Graphics::TBitmap;
//левый верхний угол -1й вариант
   P= Panel1->ClientToScreen(P);    x0 = P.x;   y0  = P.y;
// 2-й вариант
 //  x0=Left+Panel1->Left;   y0=Top+Panel1->Top;
 //3-й вариант
// x0=Left+HeaderControl1->Left;  y0=Top+HeaderControl1->Height;
    dx=Panel1->Width-Panel1->BevelWidth;
    dy=Panel1->Height-Panel1->BevelWidth;
   Bitmap->Width =dx;   Bitmap->Height=dy;
   BitBlt(Bitmap->Canvas->Handle,0,0,dx,dy,GetDC(0),x0,y0,SRCCOPY);
   Clipboard()->Assign(Bitmap);
 }
```

6. Модуль Find_File_c.exe (рис. 7) – выполняет внутрипрограммный поиск файлов без online-выхода из выполняемого модуля в среду Windows, а также запуск на выполнение найденного exe-модуля. В данном модуле выполняется, например, запуск графического пакета **tecplot.**

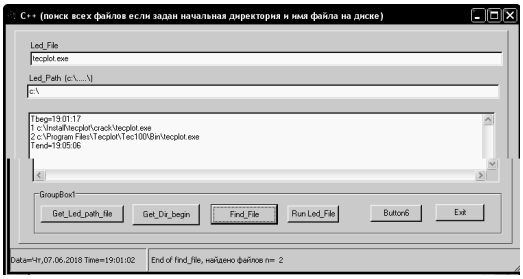

Рис. 7. *Копия формы demo-модуля поиска файла*

Приведем главную рекурсивную функцию **ListFiles_n** для поиска файлов и несколько вспомогательных функций, которые могут быть полезны начинающим программистам.

Входные параметры функции **ListFiles_n:**

AnsiString path – строка-директория начального поиска,

AnsiString filename – искомый файл,

TStrings* List – для результатов поиска.

Для хранения и визуального обзора найденных результатов поиска на форме предусмотрен объект типа **Memo**.

```
void ListFiles_n(AnsiString path,AnsiString filename, TStrings*  List)
{  TSearchRec sr;
if (FindFirst(path+"*.*",  faAnyFile, sr) == 0)
{   do
    {   if (sr.Attr &  faDirectory)
        {   if (sr.Name!=".")
            if  (sr.Name!="..")
             {   ListFiles_n(path+sr.Name+"\\",filename,List);     }
        }
     else
    {   if (sr.Name.UpperCase()==filename.UpperCase())
         { n=n+1;  outfile=path+sr.Name;
          List->Add(IntToStr(n)+" "+outfile);}      }
    }
  while  (FindNext(sr) ==  0);
  FindClose(sr);
}
   Application->ProcessMessages();
}
//  функция-кнопка Button2Click  – поиск файла и запись результатов в область Memo1
void __fastcall TForm1::Button2Click(TObject *Sender)
{  int kk2=10; Memo1->Clear();
 GetTimeFormat (NULL,NULL,NULL,"HH':'mm':'ss",lptime,kk2);
 strcpy(s,""); strcat(s,"Tbeg="); strcat(s,lptime);
 Memo1->Lines->Add (s);
n=0;  StatusBar1->Panels->Items[1]->Text="Поиск...";
  path=Led_path->Text;  filename=Led_file->Text;
ListFiles_n(path,filename,Memo1->Lines);
 GetTimeFormat (NULL,NULL,NULL,"HH':'mm':'ss",lptime,kk2);
 strcpy(s,"");strcat(s,"Tend="); strcat(s,lptime);
 Memo1->Lines->Add (s);
StatusBar1->Panels->Items[1]->Text=
" End of find_file, найдено файлов n=  "+IntToStr(n);
}
//-----------------------------------------------
AnsiString Get_Path(AnsiString mes)
{  AnsiString Dir,Root,get_path;
  Root = "c:\\";
  get_path =  SelectDirectory(mes, Root, Dir) ? Dir+"\\" : String("");
return (get_path);
}
//. . . . . .
AnsiString SlashSep (AnsiString Path, AnsiString FName)
{     if(Path[Path.Length()-1]!='\\')       return(Path+"\\"+FName);
    else return(Path+FName);
}
```

Для запуска найденного файла (или необходимого приложения) используется вспомогательная функция **ExecuteFile.**

```
 int ExecuteFile
(AnsiString FileName, AnsiString Params, AnsiString DefaultDir,  int  ShowCmd)
{ char* zFileName;
  char* zDir;
  char zParams[256];
```

```
 zFileName=(char *) calloc(256,sizeof(char));
 zDir=     (char *) calloc(256,sizeof(char));
 StrPCopy(zFileName, FileName);       //имя файла
 StrPCopy(zParams, Params);           //передаваемые параметры
 StrPCopy(zDir, DefaultDir);          //директория
 int Result= (UINT) ShellExecute(Application->MainForm->Handle,NULL,
             zFileName,zParams,zDir,ShowCmd);
 return (Result);
}
```

Функция-кнопка **Button5Click** выполняет запуск найденного файла и интерактивно выделенного в области объекта типа **Memo** (в общем случае это может быть самостоятельное приложение).

```
void __fastcall TForm1::Button5Click(TObject *Sender)
{  AnsiString F_exe,Params,DefaultDir; int i;
   i=Memo1->SelStart; int j= Memo1->SelLength;
   if(j==0)  {ShowMessage ("Exe-file не выделен"); goto eee;}
   F_exe=Memo1->SelText;
   StatusBar1->Panels->Items[1]->Text="Выделен для запуска exe-file: "+F_exe;
   i= MessageBox
     (NULL,"Выполнить запуск  exe-file","Exe-file выделен верно?",MB_OKCANCEL);
 if (i==IDCANCEL) goto eee;
  StatusBar1->Panels->Items[1]->Text="Запуск "+F_exe;
   Params="";   DefaultDir= "c:\\";   Application->ProcessMessages();
   i=ExecuteFile(F_exe,Params,DefaultDir,SW_SHOW);
  StatusBar1->Panels->Items[1]->Text=IntToStr(i)+": End of pgm="+F_exe;  eee:  }
```

7. Модуль Square.exe (рис. 8) – в базовом варианте demo-программа демонстрирует применение объекта главное меню (**MainMenu**), в котором реализовано меню 0-уровня, содержащее 2 элемента. Это – элемент "О программе", который на отдельной форме выполняет вывод некоторой объектной информации (календарь в виде объекта **MonthCalendar**, примеры "входа" в **Internet**, кнопка выхода) и элемент "Exit", выполняющий выход из модуля.

Решение квадратного уравнения выполняется простейшей кнопкой "Решить" типа **Button**.

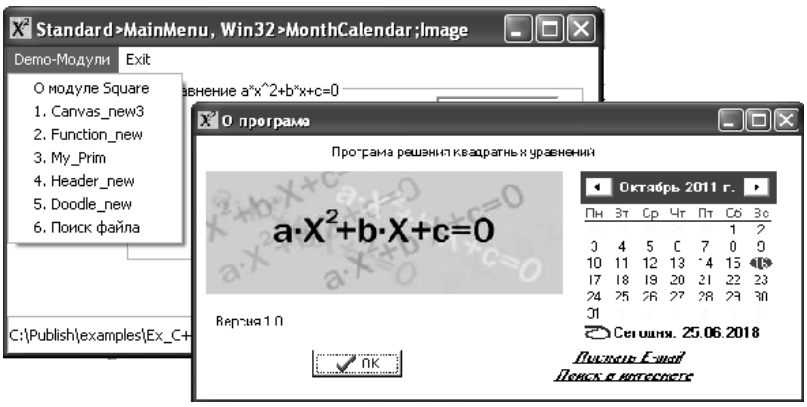

Рис. 8. *Копия MainForm (с MainMenu) и формы Form2About*

В данном demo-модуле реализованы 2 уровня главного меню (0-уровень – это элементы "Demo-модули" и "Exit"). В подменю 1-го уровня для элемента "Demo-модули" включены элементы, которые запускают на выполнение самостоятельные exe-модули, расположенные в поддиректории "pgm" (здесь это все рассмотренные выше учебные demo-модули).

Для запуска exe-модулей используется функция **ExecuteFile (см. п. 6)**: **int Execute-File (AnsiString FileName, AnsiString Params, AnsiString DefaultDir, int ShowCmd).**

Приведем функцию **N61Click**, которая выполняет запуск **exe**-модуля при выборе из **MainMenu** элемента "**Поиск файла**".

```
void __fastcall TFormMain::N61Click(TObject *Sender)
{  AnsiString F_exe,Params,DefaultDir;
  F_exe=path+"pgm\\find_file_c.exe";   StatusBar1->SimpleText=F_exe;
  Params=""; DefaultDir= "c:\\";       Application->ProcessMessages();
  int i=ExecuteFile(F_exe,Params,DefaultDir,SW_SHOW);
 StatusBar1->SimpleText=F_exe+", Err="+IntToStr(i);
}
```

Проекты **(*.bpr**), реализующие создание рассмотренных demo-программ, можно изучить, а затем вносить в проект и исходные модули **(*.cpp**), изменения, исправления, добавлять новые возможности.

Список литературы

1. *Архангельский А.Я.* Программирование в C++ Builder 6. М.: Бином, 2003. 1152 с.
2. *Архангельский А.Я.* C++ Builder 6. Кн. 1. Язык C++: справ. пособие. М.: Бином, 2002. 544 с.
3. *Романчик В.С., Люлькин А.Е.* Программирование в C++ Builder: пособие для студентов мех.-мат. фак. Минск: БГУ, 2007. 128 с.
4. *Шамис В.А.* Borland C++ Builder 6. Для профессионалов. СПб.: Питер, 2004. 800 с.

Training demonstration programs IN C ++ IN BORLAND C++ BUILDER 6

L. V. Landik, I. V. Pestrenina
Perm State University; 15, Bukireva st., Perm, 614990, Russia
Ipestrenina@gmail.com; 8 (342) 2 396 375

Several demo programs are presented in the C ++ algorithmic language in the object-oriented environment of the compiler Borland C++ Builder 6, showing some "modern" requirements for software products and realizable features. This, firstly, creates a form on the display screen to visualize the task, placing component objects on it that offer interactive execution of necessary actions, as well as graphics, service functions for working with files (search, read, write) and launching applications.

Keywords: *object-oriented programming language; C++.*