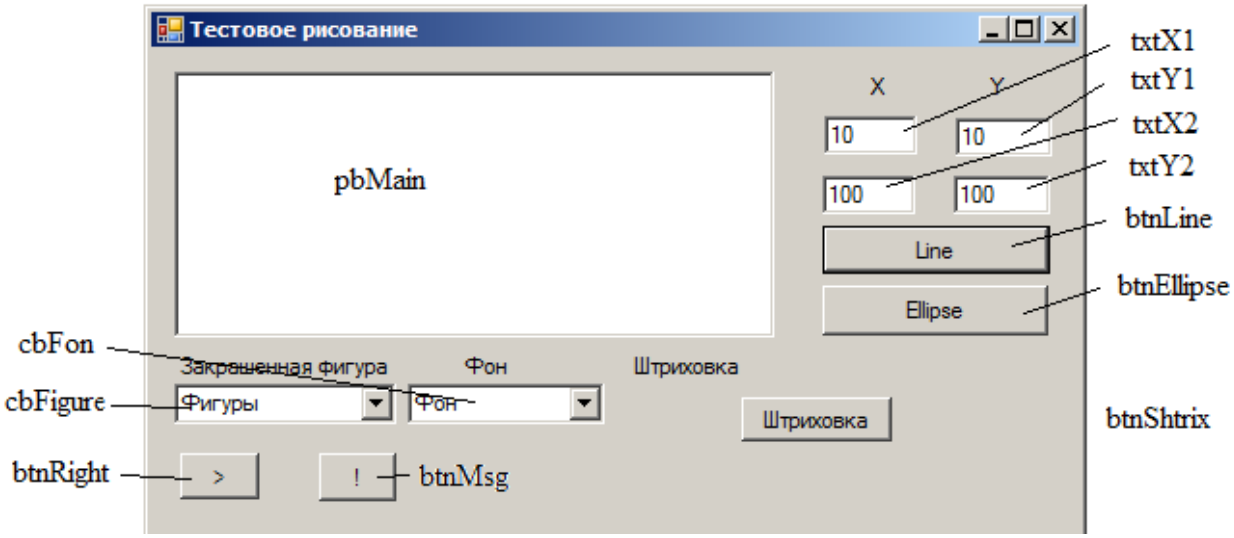
Доля П.Г.
Харьковский Национальный Университет
механико – математический факультет
2014 г.

Введение в графические возможности C++

Создаем проект Windows Forms.
Разместим на форме элементы PictureBox, Label – ы, TextBox – ы, Button – ы, ComboBox – ы. Меткам давать имя необязательно (ограничимся теми именами, которые им при создании дает VS). Остальным элементам дадим имена и установим свойство Text таким, как показано на рисунке.



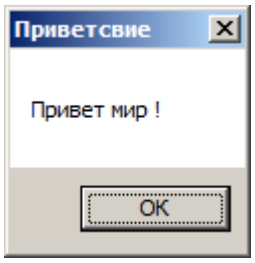
Приступим к программированию.
Кнопка с восклицательным знаком будет просто выводить окно сообщений. Выполните двойной щелчок по кнопке – вы попадете в файл Form1.h в тело процедуры `btnMsg_Click`. Заполните его следующим кодом

```
private: System::Void btnMsg_Click(System::Object^  sender,
System::EventArgs^  e)
{
//MessageBox::Show("Привет" + textBox1->Text + "!", "Приветсвите");
     MessageBox::Show("Привет мир !", "Приветсвие");
}
```

Проверьте, что в начале файла Form1.h имеется код

```
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
using namespace System::Drawing::Drawing2D; // нужно для рисования
```

Откомпилируйте и выполните программу. Нажмите на кнопку с восклицательным знаком. Вы увидите окно сообщений



Обратите внимание, что в файле Form1.h описывается класс формы. При этом большая часть этого кода создается средой автоматически.

Следующая кнопка  будет смещать предыдущую  вправо. Для этого сделайте двойной щелчок по кнопке `btnRight` и заполните отрывшуюся функцию следующим кодом

```
private: System::Void btnRight_Click(System::Object^  sender,
System::EventArgs^  e)
{
btnMsg->Location=Drawing::Point(btnMsg->Location.X+2,
                                    btnMsg->Location.Y);
 }
```

Здесь справа конструктору класса `Point` передаются координаты точки расположения кнопки (левого верхнего угла).

Переходим к рисованию.

Рисовать линии и дуги можно на любом элементе управления, но наилучшим для этого является элемент `PictureBox`, которому мы присвоили имя `pbMain`. Координаты начальной и конечной точек отрезка мы будем читать из текстовых элементов txtX1, txtY1, txtX2, txtY2 и хранить их в закрытых переменных x1,y1, x2, y2 класса Form1.

Позже нам потребуется признак, который будет определять, что мы будем рисовать – отрезок или эллипс. Для этого создадим перечисление

```
enum whatWhantDraw {
     Line=0,
     Circle=1
};
```

которое разместим сразу после операторов описания пространств имен `using namespace` и перед описанием класса `Form1`. А в классе формы опишем переменные

```
public ref class Form1 : public System::Windows::Forms::Form
{
     private:  int x1,y1,x2,y2;
     private:  whatWhantDraw whatToDraw;
     ....
```

Читать координаты нам нужно будет несколько раз, поэтому их чтение запрограммируем в теле функции getTwoPoints(), которую разместим сразу после команды препроцессора #pragma endregion, которая завершает область кода, создаваемую средой разработки.

```
#pragma endregion
// код создаваемый пользователем
private: System::Void getTwoPoints() {
     x1=Convert::ToInt32(txtX1->Text);
     y1=Convert::ToInt32(txtY1->Text);
     x2=Convert::ToInt32(txtX2->Text);
     y2=Convert::ToInt32(txtY2->Text);
}
```

Напомним, что функции Convert::ToInt32(текст) преобразуют текстовое представление числа в числовое.

Само рисование рекомендуется выполнять в процедуре обработки сообщения Paint соответсвующего элемента управления, в данном случае элемента pbMain. Чтобы создать код этой процедцры надо в окне свойств элемента pbMain выделить закладку 'Events' (щелкнуть по кнопке с молнией) и в списке процедур выбрать имя Paint. Будет создан и откроется код этой процедуры. Напишите в нем следующее

```
private: System::Void pbMain_Paint(System::Object^  sender,
            System::Windows::Forms::PaintEventArgs^  e)
{
e->Graphics->DrawLine (System::Drawing::Pens::Green,
                              x1, y1, x2, y2);
}
```

Теперь в коде функции btnLine_Click напишем

```
private: System::Void btnLine_Click(System::Object^  sender,
               System::EventArgs^  e) {
     whatToDraw=Line;
     getTwoPoints();
     pbMain->Refresh();
}
```

Функция pbMain->Refresh() вызывает процедуру обновления элемента pbMain и в частности выполняется процедура pbMain_Paint(...), которая рисует линию. Откомпилируйте и выполните программу – меняйте координаты начальной и конечной точек отрезка и рисуйте снова.

Заметим, что функция pbMain_Paint(...) вызывается автоматически также в тех случаях, когда элемент pbMain перекрывается другим окном и снова открывается. Если код рисования линии записать внутри процедуры обработки щелчка по кнопке, то автоматического перерисовывания не будет.

Теперь запрограммируем код кнопки, рисующий эллипс.

```
private: System::Void btnEllipse_Click(System::Object^  sender,
                    System::EventArgs^  e) {
```

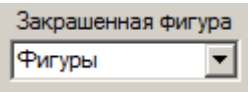
```
     whatToDraw=Circle;
     getTwoPoints();
     pbMain->Refresh();
}
```

Также следует изменить код процедуры обработки сообщения `Paint`.

```
private: System::Void pbMain_Paint(System::Object^  sender,
     System::Windows::Forms::PaintEventArgs^  e) {
if (whatToDraw==0) {
     e->Graphics->DrawLine (System::Drawing::Pens::Green,
          x1, y1, x2, y2); //Рисуем линию
}
else {
     Pen^ mypen= gcnew Pen(Color::Red);
     e->Graphics->DrawEllipse(mypen, x1, y1, x2, y2);
     //рисуем эллипс
}
```

Эллипс вписывается в прямоугольник две противоположные вершины которого определяются числами x1,y1, x2, y2.

Научимся работать с выпадающим списком `cbFigure`.



Над списком размещена метка с текстом *<Закрашенная фигура>*. Один способ заполнения содержимого списка состоит в заполнении его в процедуре загрузки формы. Выделите форму, перейдите на закладку ее событий и выберите процедуру `Load`.

```
private: System::Void Form1_Load(System::Object^  sender,
          System::EventArgs^  e) {
// Заполнение ComboBox названий фигур
cbFigure->Text = "Фигуры";
array<String^>^ Figure ={"Прямоугольник","Эллипс","Окружность"};
cbFigure->Items->AddRange(Figure);
}
```

Откомпилируйте и выполните программу. Список будет заполнен, но пока программа никак не реагирует на выбор элемента в нем. Для реакции на выбор надо запрограммировать процедуру `SelectedIndexChanged(...)`.

```
private: System::Void
cbFigure_SelectedIndexChanged(System::Object^  sender,
          System::EventArgs^  e) {
     Graphics ^ g= pbMain->CreateGraphics();
     Brush ^ brsh = gcnew SolidBrush(Color::Orange);
     g->Clear(SystemColors::Control);

     getTwoPoints();
     switch (cbFigure->SelectedIndex)
     {
     case 0:  //прямоугольник
          g->FillRectangle(brsh, x1, y1, x2, y2); break;
```

```
     case 1:  // эллипс
          g->FillEllipse(brsh, x1, y1, x2, y2); break;
     case 2:  //окружность
          g->FillEllipse(brsh, x1, y1, x2, x2-x1); break;
          }
}
```

Здесь вначале мы создаем графический объект `Graphics^ g`

Замечание. В C++/CLI для обозначения ссылок на управляемые объекты (находящиеся в среде NET) используется «^» вместо «». Например,*

```
Object  ^ obj = gcnew Object();
```

Затем создаем объект `Brush^ brsh`

Затем очищаем PictureBox системным цветом `SystemColors::Control`

Потом читаем координаты точек и, в зависимости отномера `cbFigure->SelectedIndex` выбранного элемента списка, рисуем закрашенный прямоугольник, эллипс или окружность.

Обратите внимание, что здесь мы создаем графический объект `Graphics^g` с помощью которого выполняется рисование, например,

```
g->FillRectangle(...)
```

Для рисования линии или эллипса раньше мы также использовали этот тип в виде

```
e->Graphics->DrawEllipse(...)
```

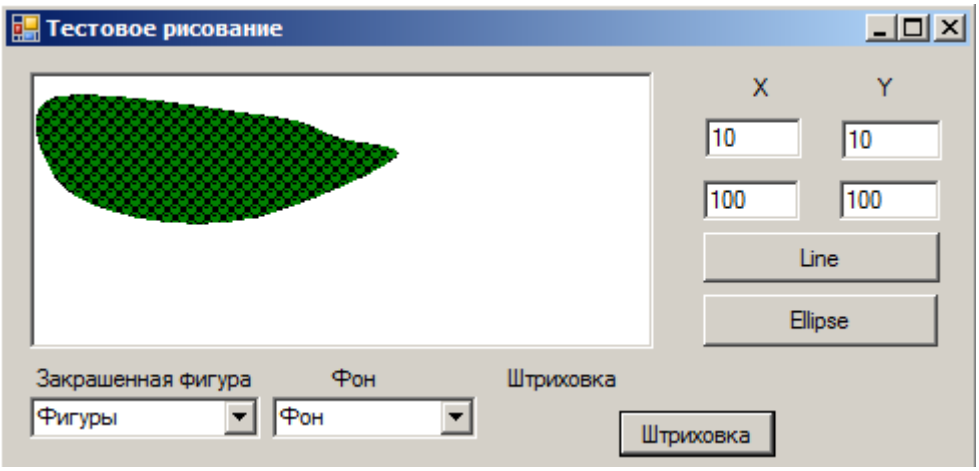
Теперь приведем пример отображения графического файла в элементе `PictureBox`. Для этого найдите два графических файла, разместите их в каталоге проекта в подкаталоге `\Graphics` и переименуйте их в `01.bmp` и `02.bmp` и перепишите процедуру `cbFon_SelectedIndexChanged` следующим образом

```
private: System::Void cbFon_SelectedIndexChanged(
          System::Object^  sender, System::EventArgs^  e) {
     switch (cbFon->SelectedIndex)
     {
     case 0:
     pbMain->Image=Image::FromFile("..\\Graphics\\01.bmp");
     break;
     case 1:
     pbMain->Image=Image::FromFile
     ("D:\\Work\\C_Plus_Plus\\Examples\\WinForms01\\Graphics\\02.
bmp");
     break;
     }
}
```

Можно использовать либо полный путь к файлу, либо относительный

Кнопка *<Штриховка>* заполняет многоугольную область (см. следующий рисунок). Создадим код соответствующей кнопки. Вначале добавьте строку в процедуру `Form1_Load`

```
private: System::Void Form1_Load(System::Object^  sender,
           System::EventArgs^  e) {
...
     cbFon->Text = "Фон";
...
}
```



Код процедуры обработки сообщения щелчка по кнопке

```
private: System::Void btnShtrix_Click(System::Object^  sender,
           System::EventArgs^  e) {
Graphics^ graf = pbMain->CreateGraphics();
Pen^ gPen = gcnew Pen( Color::Black,1 );
HatchBrush^ HBrush = gcnew HatchBrush( HatchStyle::Sphere,
                Color::Green, Color::Black );
//System.Drawing.Drawing2D::HatchBrush HBrush = gcnew HatchBrush(
     HatchStyle::Sphere, Color::Green, Color::Black );
// Координаты вершин многоугольника
Point point1 = Point(10,10);
Point point2 = Point(10,50);
Point point3 = Point(50,70);
Point point4 = Point(110,70);
Point point5 = Point(180,40);
Point point6 = Point(150,30);
Point point7 = Point(120,20);
array<Point>^ Points =
     {point1,point2,point3,point4,point5,point6,point7};
graf->DrawPolygon( gPen, Points );
graf->FillClosedCurve( HBrush, Points);
}
```

Еще один полезный элемент – подсказка `ToolTip`.

Разместите его на форме (он появится внизу формы). В процедуру `Form1_Load` добавьте строки

```
private: System::Void Form1_Load(System::Object^  sender,
           System::EventArgs^  e) {
...
toolTip1->SetToolTip(btnMsg, "Сообщение\nподсказка");
toolTip1->IsBalloon = true;
...
}
```

Откомпилируйте и выполните программу. При подведении курсора мыши к элементу `btnMsg` автоматически всплывает подсказка.

В C++/CLI для обозначения ссылок на управляемые объекты (находящиеся в среде NET) используется «^» вместо «*».

```
Object  ^ obj = gcnew Object();
```

Директива препроцессора вначале файла Form1.h

```
#pragma once
```

приказывает препроцессору включать в проект описания и тела функций только один раз.