электронный научно-технический журнал
ИНЖЕНЕРНЫЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51036. ISSN 2307-0595

Использование графического интерфейса в приложениях на C++ в решении задач вычислительной математики

 # 12, декабрь 2014

Шикуть А. В., Аристов Б. К.

УДК: 004.514

Россия, МГТУ им. Н.Э. Баумана
alla.shikut@rambler.ru

Введение

Приложения, разрабатываемые в визуальной среде, фактически, являются приложениями под Windows и этим принципиально отличаются от консольных приложений.

Характерной особенностью приложений на языке Visual C++ является то, что программист использует в них возможности *объектно-ориентированного программирования*, и все элементы программы, создаваемые в них являются объектами. При создании новых объектов, они порождаются из уже существующих объектов. Благодаря этому новые объекты наследуют свойства и возможности предшественников, и которые необходимо лишь дополнить необходимыми новыми функциями.

Приложения с графическим интерфейсом, использующие формы, создаются в управляемой среде CLR и являются приложениями Windows Forms Application – приложениями под Windows. Windows Forms помещает в управляемый код стандартные элементы управления Windows, доступные благодаря Win32 API. При этом код, который создается классами, реализующими API (Application Program Interface) для Windows Forms, не зависит от языка разработки. В результате этого возможно использование Windows Forms как при написании программы на языках программирования: C, C++, C#, так и на VB.NET и др. Внутри .NET Framework Windows Forms реализуется в рамках пространства имен System [5]. В результате этого среда CLR обеспечивает межъязыковую совместимость программ в процессе их выполнения, используя управляемый код, специальные операции и метаданные, благодаря чему становится возможным автоматическое преобразование данных в последовательную форму при их сохранении. Код, который предоставляется общеязыковой средой выполнения CLR и называется управляемым кодом. При этом память для управляемых данных распределяется и освобождается автоматически.

Основными задачами, которые решает управляемая среда CLR, являются:

- автоматическая сборка мусора;

- гарантированная инициализация переменных, контроль типов и проверка допустимости значений аргументов;
- сокрытие особенностей работы.

Возможности Windows Forms Application

В процессе создания приложений в визуальной среде программирования MS Visual C++.NET пользователю предоставляется возможность создавать графический интерфейс с помощью Windows Forms [10]. При этом в процессе создания графического интерфейса в приложении на языке C/C++ необходимо обеспечить поддержку взаимодействия пользователя с элементами управления интерфейса. Для достижения этой цели в ходе выполнения этой работы необходимо создать приложение, представляющее собой полностью функциональную программу Windows Forms. При этом необходимо знать:

- Как используя возможности конструктора форм **Form Design**, интерактивно строить графический интерфейс пользователя в приложении.
- Как добавлять в форму элементы управления.
- Как добавлять к элементам управления обработчики событий.

Разработка приложений для Windows в среде MS Visual C++.NET с помощью конструктора Windows Forms

Создание нового приложения начинается с создания формы [1,2]. Для этого сначала следует запустить среду MS Visual C++.NET и выполнить команду File/New/Project. Далее, надо выбрать язык программирования, например Visual C++, и установить режим Windows Application. Затем выбрать тип проекта - CLR и установить тип приложения - Windows Forms Application[5]. Далее надо ввести имя приложения с указанием полного пути к файлу проекта. При этом открывается конструктор Windows Forms, отображающий форму Form1 нового проекта.

В результате выполнения этих действий на рабочем столе открывается объект - окно формы Form1. Как и любой другой объект в визуальной среде, форма имеет свои свойства, которыми можно управлять с помощью окна свойств объектов – Properties. Для вызова окна свойств формы следует открыть контекстное меню формы, для чего надо поместить на форму курсор мыши, щелкнуть правой клавишей и выполнить команду Properties.

При создании объекта типа класс TForm вначале на экране появляется пустая форма, которая является окном дизайнера формы Form1.h [Design]. Дизайнер позволяет изменять размеры формы, помещать компоненты на форму, перетаскивать их с одного места на другое, удалять и выделять компоненты, изменять их размеры, делать видимыми или невидимыми и т.д.

Использование компонентов Windows Forms в приложениях

Создание интерфейса приложения в визуальной среде MS Visual C++.NET сводится к размещению на форме необходимых компонентов, инициализации событий и написанию необходимых обработчиков этих событий в соответствии с решаемой задачей [4].

Компоненты Windows Forms. разбиты на функциональные группы в **палитре компонентов Toolbox** и выбираются из палитры компонентов по одному или группой. Для вызова этого окна следует выбрать пункт главного меню **View** и в раскрывшемся списке выбрать пункт ToolBox. При этом на рабочем столе раскрывается окно палитры компонентов, содержащее перечень визуальных компонентов и их пиктограммы.

Для помещения компонента на форму надо найти нужный компонент на палитре компонентов, щелкнуть на нем мышью, перевести курсор мыши в нужное место формы и снова щелкнуть мышью. Любой объект, расположенный на рабочем столе, обладает определенным набором свойств, таких как, имя, видимость, координаты расположения на форме, размеры и т.д. Значения свойств компонентов можно изменять с помощью окна свойств Properties, в котором отображаются все свойства активного объекта. Это окно становится видимым при выполнении команды контекстного меню Properties.

Windows Forms представляет собой целый набор средств создания Windows-приложений, выполняющихся в среде CLR. Форма представляет собой главный контейнер, в который помещаются визуальные компоненты (кнопки, метки, таблицы, листы прокрутки и т.д.), с помощью которых и реализуется конкретный алгоритм определённой задачи. Это компонент, который служит основой окна приложения или диалогового окна, в которое добавляются различные элементы управления, предназначенные для взаимодействия с пользователем. Пакет Visual C++ поставляется со стандартным набором, содержащим более 60 элементов управления, которые можно применять в формах. Каждый элемент управления и компонент Windows Forms имеет определённое назначение и выполняет определённые функции.

Ниже представлен список некоторых, наиболее широко используемых элементов управления и компонентов Windows Forms с указанием их основного назначения.

Элементы управления группы **Common Controls** палитры компонентов, позволяющие осуществлять ввод/вывод, отображение и редактирование элементов управления:

- Button - позволяет создавать элементы типа «кнопка» для запуска, остановки или прерывания некоторого процесса.
- Label – отображает текст, недоступный для непосредственного редактирования пользователем в процессе выполнения программы.
- TextBox - отображает текст, введённый в режиме разработки, который может быть изменён во время выполнения программы.
- ListBox – отображает список текстовых и графических элементов.
- ComboBox – отображает раскрывающийся список.
- CheckedListBox – отображает список с полосой прокрутки, состоящий из элементов с флажками.

- CheckBox – отображает флажок и надпись для текста.
- RadioButton - отображает кнопку, которая может быть включена или выключена.
- TrackBar - позволяет задавать значения на шкале, перемещая по ней ползунок.
- RichTextBox - позволяет представлять текст в обычном текстовом формате или в формате RTF.
- MaskedTextBox - ограничивает формат данных, вводимых пользователем.
- **WebBrowser** - позволяет пользователям перемещаться по веб-страницам внутри формы.
- **LinkLabel** – позволяет добавлять веб-ссылки на другое окно или на веб-узел.
- **DateTimePicker** - отображает графический календарь, позволяющий пользователю выбрать дату или время.
- **TabControl** – позволяет создать набор страниц с вкладками для эффективной организации доступа к сгруппированным объектам.
- **ProgressBar –** позволяет наблюдать за ходом выполнения некоторого процесса во времени.
- **DataGridView**- предоставляет собой настраиваемую таблицу для отображения данных; обеспечивает настройку ячеек, строк, столбцов и границ таблицы.

Группирующие элементы управления группы **Containers** палитры компонентов**:**

- **Panel** - группирует набор элементов управления для обеспечения общего поведения.
- **GroupBox** – групповой контейнер; используется для разделения компонентов на подгруппы.
- **PictureBox** – позволяет отображать графическое изображение нужного компонента.

Элементы управления группы **Menus & Toolbars** палитры компонентов, так или иначе связанные с организацией меню:

- **MenuStrip** – создаёт главное меню приложения, с помощью которого осуществляется управление всей работой приложения и его отдельных блоков.
- **ContextMenuStrip** – создаёт настраиваемые контекстные меню.
- **ListView** - отображает список элементов в одном из пяти возможных представлений: пиктограмма, только текст, текст с маленькими значками, текст с большими значками и только необходимые параметры.
- **ToolStrip** – позволяет создавать панели инструментов с использованием различных стилей и принципов.

Элементы управления группы **Components** палитры компонентов, связанные с временем**:**

- **Timer –** задаёт счётчик времени.

- **DateTimePicker** - отображает графический календарь, позволяющий пользователю выбрать дату или время.
- **MonthCalendar** - отображает графический календарь, позволяющий пользователю выбрать диапазон дат.

Элементы управления группы **Dialogs** палитры компонентов, использующие диалоговые окна:

- **ColorDialog** - отображает диалоговое окно выбора цвета, позволяющее задать цвет элемента интерфейса.
- **FontDialog** - отображает диалоговое окно, в котором можно указать шрифт и его атрибуты.
- **OpenFileDialog** - отображает диалоговое окно для поиска и выбора файла.
- **PrintDialog** - отображает диалоговое окно для выбора принтера и задания его атрибутов.
- **PrintPreviewDialog** - отображает диалоговое окно, показывающее, как будет выглядеть напечатанный компонент элемента управления PrintDocument.
- **FolderBrowserDialog** - отображает диалоговое окно для поиска, создания и выбора папки.
- **SaveFileDialog** - отображает диалоговое окно для сохранения файла.

Обработка событий

После того, как нужные компоненты интерфейса выбраны и помещены на форме, необходимо создать обработчики событий с помощью окна Properties. Для этого необходимо щёлкнуть на элементе управления, для которого требуется создать обработчик события. Затем в окне Properties нажать кнопку **Events.** В списке доступных событий щелкнуть на событие, для которого требуется создать обработчик события. Затем добавить соответствующий код для обработчика событий.

Список возможных событий, которые могут происходить с компонентом, содержит вкладка **Events.** Эта вкладка позволяет связывать каждое событие с программой-обработчиком этого события. Если дважды щёлкнуть мышью на окне с кнопкой рядом с именем события, то Visual C++ создаёт в модуле формы, в которую помещён компонент, программу – обработчик этого события. Такая программа-обработчик представляет собой функцию с заголовком и пустым телом, - что-то в виде заготовки - шаблона. Дальнейший процесс программирования состоит в том, чтобы в тело шаблона записать необходимые операторы, которые определяют реакцию компонента на данное событие с учетом передаваемых функции фактических значений ее параметров.

Например, пустой обработчик события OnClick кнопки Button1 имеет следующий вид:

```
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^  e) { }
```

При создании обработчика события происходит автоматическое переключение системы на вызов редактора кода, который устанавливает курсор на начало обработчика, для того чтобы можно было вводить нужные операторы программы.

Создать событие для кнопки можно и другим способом – двойным щелчком на ней.

Особенности редактирования кода

Таким образом, одновременно с созданием новой формы создаётся специальный программный модуль с именем формы и расширением h - Form1.h. Это h–файл (заголовочный файл), в котором находится описание формы и обработчики событий компонентов, используемых в проекте.

Попасть в редактор кода из окна формы можно с помощью комбинации клавиш **<Ctrl>+<Alt>+<0>,** или выполнив команду **Code** меню в опции главного меню **View**. Из режима написания кода в режим дизайна можно переключаться комбинацией клавиш <Shift>+<F7>.

Форма **Form1** (как программный объект) является **классом - наследником класса Form**. В Form1 попадают, в силу наследования, все члены класса Form, а также в неё попадают все компоненты, которые впоследствии добавляются в форму. В программном модуле все обработчики событий формируются в рамках одного класса-формы. Поэтому обращение к членам класса производится через указатель на экземпляр этого класса [3]. В данном случае таким указателем является указатель **this.** Этот указатель всегда содержит ссылку на текущий объект.

Так, например, если надо обратиться к свойству Text компонента label1 (однострочный редактор - поле ввода-вывода), то обращение должно выглядеть следующим образом:

this->label1->Text=s.ToString();

Здесь указатель this хранит адрес объекта label1.

Таким образом, имена экземпляров компонентов формируются как указатели на соответствующие экземпляры.

В качестве примера создания и использования графического интерфейса в приложениях на языке Си приводится фрагмент кода, используемого в процессе обработки матриц (сложение, перемножение, транспонирование). Основными объектами в программе являются объекты dgA, dgB и dgC, порождённые из визуального компонента DataGridView.

```
#pragma endregion
  //считывание матрицы из таблицы
  Void Read(int *A, DataGridView ^dgA, bool* err)
  {
    for (int i=0; i<N; i++)
        for (int j=0; j<N; j++)
              if (dgA[j,i]->Value=="")//обработка ситуации пустого ввода
              {
                  MessageBox::Show( "Неверно введены матрицы!", "Ошибка
ввода",
            MessageBoxButtons::OK, MessageBoxIcon::Exclamation );
```

```
                *err=true;
               return;
             }
            else
         *(A+i*N+j)=Convert::ToInt32(dgA[j,i]->Value);
  }

  //вывод матрицы в таблицу
  Void Show(int *A, DataGridView ^dgA)
  {
     for (int i=0; i<N; i++)
        for (int j=0; j<N; j++)
        {
        int r= *(A+i*N+j);
          dgA[j,i]->Value=r.ToString();
        }

  }

 //очистка таблицы
  Void Clear( DataGridView ^dgA)
  {
    for (int i=0; i<N; i++)
        for (int j=0; j<N; j++)
            dgA[i,j]->Value="";
  }

 //сложение матриц
      private:    System::Void    btnPlus_Click(System::Object^        sender,
System::EventArgs^  e) {
                        bool err=false;
                  Read(A, dgA, &err);
                  Read(B, dgB, &err);
                  for (int i=0; i<N; i++)
                    for (int j=0; j<N; j++)
                          *(C+i*N+j)=*(A+i*N+j)+*(B+i*N+j);
                  lblRes->Text="Сумма матриц:";
                  Show(C, dgC);
                  }

 //транспонирование матрицы
 Void Transp(int* A)
 {
   int r;
   for (int i=0; i<N; i++)
     for (int j=i+1; j<N; j++)
       {

         r=*(A+i*N+j);
```

```
          *(A+i*N+j)=*(A+j*N+i);
          *(A+j*N+i)=r;
        }
 }
//выделение места под матрицы размерности N  и вывод пустых таблиц
private: System::Void btnOk_Click(System::Object^  sender, System::EventArgs^
e) {
                    N=Convert::ToInt32(tbN->Text);
                    dgA->ColumnCount=N; dgA->RowCount=N;
                    dgB->ColumnCount=N; dgB->RowCount=N;
               dgC->ColumnCount=N; dgC->RowCount=N;
                    Clear(dgA); Clear(dgB); Clear(dgC);
                    for (int i=0; i<N; i++)  //задаем ширину столбцов
                    {
                      dgA->Columns[i]->Width=50;
                      dgB->Columns[i]->Width=50;
                      dgC->Columns[i]->Width=50;
                    }
                    if (N<=4) //определение размера таблицы
                    {
                      dgA->Width=52*N; dgA->Height=30*N;
                  dgB->Width=52*N; dgB->Height=30*N;
                  dgC->Width=52*N; dgC->Height=30*N;
                    }

                    A=new int [N*N];
                    B=new int [N*N];
                    C=new int [N*N];
               }

//умножение матриц
private:     System::Void     btnMult_Click(System::Object^           sender,
System::EventArgs^  e) {
                     bool err=false;
                     Read(A, dgA, &err);
                Read(B, dgB, &err);
                     if (!err)
                     {
                     int k;
                     for (int i=0; i<N; i++)
                       for (int j=0; j<N; j++)
                         {
                           *(C+i*N+j)=0;
                           for (k=0; k<N; k++)
                             *(C+i*N+j)=*(C+i*N+j)+*(A+i*N+k)*(*(B+N*k+j));
                         }
                     lblRes->Text="Матрица-произведение:";
                     Show(C, dgC);
                     }
```

```
            }

//транспонирование матрицы
private: System::Void btnTA_Click(System::Object^  sender, System::EventArgs^
e) {
                    bool err=false;
                    Read(A, dgA, &err);
                    Transp(A);
                    Show(A, dgA);

            }
private: System::Void btnTB_Click(System::Object^  sender, System::EventArgs^
e) {
                    bool err=false;
                     Read(B, dgB, &err);
                    Transp(B);
                    Show(B, dgB);
            }

};
}
```

Возможности кроссплатформенной среды GTK+

Большие возможности для создания интерфейса открываются пользователю при использовании библиотек элементов интерфейса. При этом на практике используется несколько видов библиотек элементов интерфейса. В каждой оконной системе существует свой набор «родных» элементов с интерфейсом низкого уровня для работы с ними. Одни библиотеки элементов управления — представляют собой высокоуровневые «обертки» к имеющимся стандартным элементам, которые упрощают работу с ними и расширяют их функциональность. Другие библиотеки предоставляют свой единый программный интерфейс для программирования интерфейса пользователя сразу в нескольких платформах или оконных системах, и с целью обеспечения кроссплатформенности приводят их к единому для всех платформ API. Третьи предоставляют кроссплатформенные возможности за счёт собственной, платформонезависимой реализации элементов управления [8,9].

Широкие возможности для создания графического интерфейса открываются пользователю при использовании библиотек элементов интерфейса кроссплатформенной среды **GTK+** [7].

Кроссплатформенная среда GTK+ (сокращение от GIMP ToolKit) представляет собой библиотеку функций графического интерфейса, так *называемых виджетов* [11]. GTK+ написана на языке Си и предназначена для работы в операционной системе Linux, Mac OS X, других Unix - подобных системах, а также – в операционной среде Windows. Наряду с Qt GTK+ является одной из наиболее популярных на сегодняшний день библиотек для X Window System.

GTK+ представляет собой свободное программное обеспечение и позволяет создавать как свободное , так и проприетарное программное обеспечение [6]. Кроме того, GTK+ является официальной библиотекой для создания графического интерфейса проекта GNU. Несмотря на то, что GTK+ написана на языке Си, она является объектно-ориентированной средой и позволяет легко создавать интерфейсы для других языков программирования таких как Ada, C++, C# и других языков программирования платформы .NET, а также Fortran, Free BASIC, Free Pascal, Java, JavaScript, Pure Basic, Python, Smalltalk …

Заключение

Таким образом рассмотрены основные проблемные вопросы, связанные с созданием и использованием графического интерфейса в приложениях на языке C/C++ в среде MS Visual C++.NET. Вопросы разработки и использования графического интерфейса являются весьма важными и актуальными для пользователей в решении задач вычислительной математики и компьютерной графики, в том числе в решении задач линейной алгебры, численного дифференциального и интегрального исчисления. В статье рассмотрены особенности управляемой среды CLR и основные возможности создания управляемого кода в управляемой среде CLR , а также основные вопросы создания приложений с графическим интерфейсом. Приведены основные возможности Windows Forms. Рассмотрены принципиальные особенности **р**азработки приложений для Windows в среде MS Visual C++.NET с помощью конструктора Windows Forms. Приведен конкретный пример создания и использования графического интерфейса в приложении MS Visual C++.NET. Показано, что графический интерфейс приложения на языке C++ с большим успехом может быть создан с помощью кроссплатформенной библиотеки GTK+. На практике при использовании ее в лабораторных работах библиотека показала себя быстрой в работе и легко переносимой на разные платформы. Делать интерфейсы пользователя достаточно легко, особенно при использовании дизайнера формы.

Список литературы

1. Давыдов В.Г. Visual C++. Разработка Windows-приложений с помощью MFS и API-функций– СПб.: БХВ – Петербург, 2008 . – 576 с.
2. Культин Н.Б. Delphi 6. Программирование на Object Pascal. – СПб.: БХВ – Петербург, 2002 . – 528 с.
3. Шилдт Герберт. Полный справочник по C, 4-ое издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2005.– 704 с.
4. Яцюк О. Основы графического дизайна на базе компьютерных технологий (+CD), 2004, 240 с.

5. Джеффри Рихтер CLR via C#. Программирование на платформе .NET Framework 4.0 на языке C#. 3-е изд. = CLR via C#, 3 ed. — СПб.: Питер, 2011. — ISBN 978-5-459-00297-3
6. Костельцев А. GTK+. Разработка переносимых графических интерфейсов. – BHV – Санкт-Петебург, 2003. – 368 с. ISBN: 5941571615
7. GTK+ 2.0 Tutorial (рус.) — официальный учебник по GTK+ 2.0.
8. Richard Coyne. The Tuning of Place: Sociable Spacer and Pervasive Digital Media - MIT Press. 2010. – p.52 – 344 p. ISBN 9780262013918.
9. Mac OS X Human. Interface Guidelines: UL Element Guidelines: Controles – https://developer.apple.com/.../**mac**/.../applehi**guidelines**/UE**Guidelines**/ UE**Guidelines**.html
10. Свободная энциклопедия - http://ru.wikipedia.org
11. **Виджет для Windows 7** с использованием ActiveX компонентов **...** habrahabr.ru/sandbox/37802/