УДК 621.3

СОЗДАНИЕ ОКОННОГО ИНТЕРФЕЙСА В PASKALABC

Лугачёв В. М.
Научный руководитель – старший преподаватель Гапанюк С. Г.

Введение. Язык программирования Pascal получил широкое распространение в странах бывшего СССР. Его изучением занимаются в школе, часто даже на первых курсах высших учебных заведений. Как же получилось так, что язык, разработанный в 1970 году не теряет своей актуальности в начале 20-х годов XXI века?

Безусловно, сыграло огромную роль то что, язык имеет простой, интуитивно понятный синтаксис, строгую типизацию переменных, в нём присутствуют средства процедурного программирования, компилятор с комментариями ошибок и т.п. По своей структуре и сути Pascal схож с многими языками высокого уровня, базируется на той же логике, но является более простым и понятным начинающему программисту. Важной составлявшей успеха данного языка является его практически полная русификация, наличие русскоязычного справочника, сборника примеров и задачника, что позволяет овладеть материалом без знания иностранных языков. Для языка Pascal написано большое количество учебных пособий, вспомогательной литературы, методических материалов на русском языке, сделано большое количество переводов зарубежной литературы. К тому же, профессорско-преподавательского состава высших учебных заведений, обязательно сталкивались с данным языком в процессе своей ранней научной деятельности, т. к. на то время этот язык был прорывным и перспективным. И им оказалось проще поднять старые наработки, чем создать курс лекций и учебно-методический материал к лабораторным работам для более современного языка.

Однако, это широкое применение Pascal для обучения непрофильных специальностей, казалось подарившее ему вторую жизнь, сыграло с ним злую шутку: глубоко проработанными являются лишь самые простые, базовые пункты данного языка программирования. Более сложные вещи, такие как классы, ООП и т. п., хотя и реализованы в последних версиях программы, однако широко не освещены в специализированной литературе, форумах и сборниках примеров. Непрофильные специальности не занимаются изучением данных разделов, а профессиональные программисты предпочитают обретать знания по этим разделам в более совершенных языках программирования.

Данная работа имеет цель познакомит её читателя с возможностью создания оконных приложений на языке Pascal. Этот раздел широко не освещался, т. к. связан с глобальными понятиями информатики, которые не входят в спектр интересов непрофильных специалистов, но может быть им полезен.

FormsABC. Создание графического приложения на языке PascalABC, связано с подключаемым модулем FormsABC. Он предназначен для создания форм без специализированного дизайнера, с помощью потокового менеджера размещения. Дизайнер форм доступен, если при создании нового проекта выбрать тип «ПриложениеWindowsForms» (см. рис. 1). Дизайнер предоставляет

широкий выбор инструментов для работы с элементами интерфейса, позволяет просто, быстро, наглядно, интуитивно понятно изменять параметры формы, добавлять и убирать элементы, менять их параметры. Разработчик выбирает элементы из списка, применяет инструменты из палитры, а дизайнер автоматически формирует соответствующий «технический код», который будет отвечает за работу элементов в процессе работы программы. По завершению работы с визуальной составлявшей разработчик добавляет в автоматически созданный код процедуры, которые будут обрабатывать действия пользователей, выводить результаты.

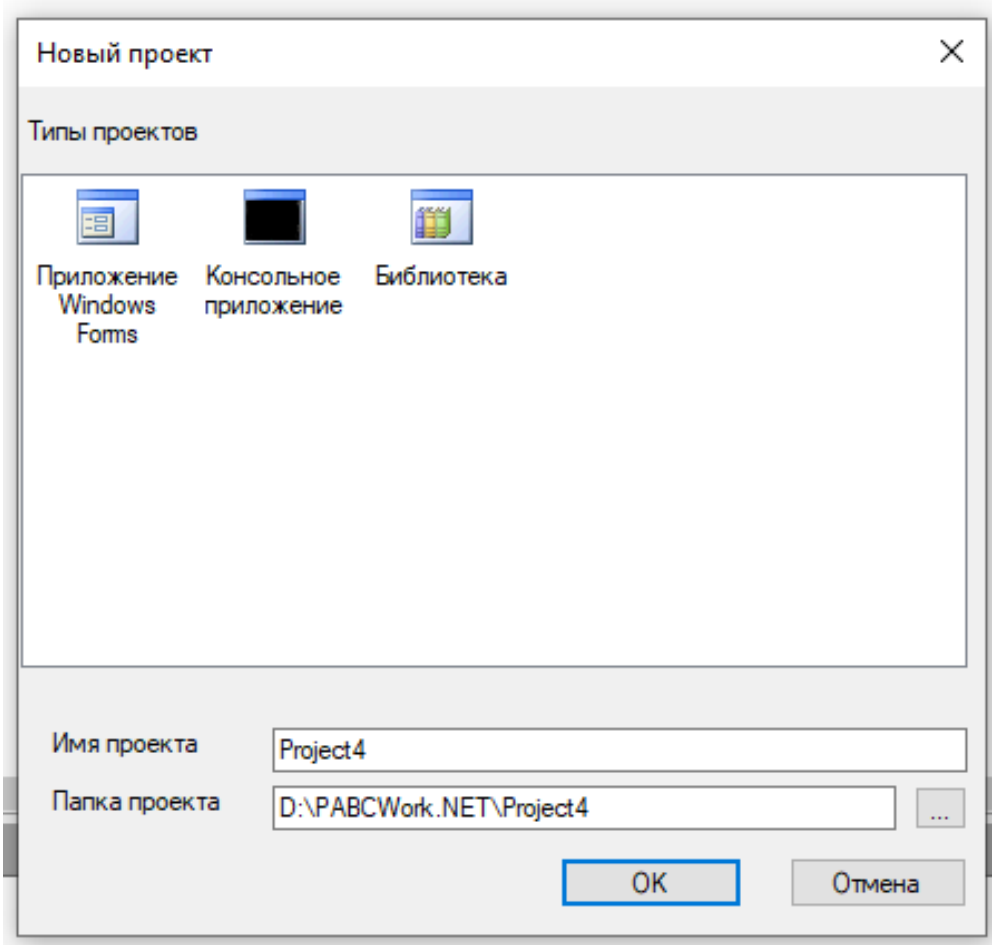

Рисунок 1 – Окно выбора, при создании нового проекта

Более подробно разберём реализацию модуля FormsABC в режиме создания консольного приложения. Т. к. большинство учебных программ связано с работой в этом режиме, изучив базовые понятия на этом, более понятном примере, далее будет проще работать в Дизайнере форм.Для начала разберём основные классы виджетов, которые можно добавлять на форму, т. е. окно приложения.

Виджеты. FormsABC. Говоря простыми словами, виждеты это те иконки, кнопки, галочки, значки, которые можно помещать на форму. Пользователь, при работе с программой будет взаимодействовать именно с ним: вписывать данные в ячейки, устанавливать галочки, выбирать пункты выплывающего меню, программа, в свою очередь, считает эти данные в память, произведёт с ними какие-либо действия и может вывести результат, например, в ту же форму.

Любой виджет должен быть описан в блоке описаний var, описание состоит из имени виджета и описания типа (типы описаны ниже). Для возможности вывода виджета на экран и работы с ним необходимо проинициализировать его в основном теле программы. Для этого используется функцияnew, в параметре которой указан класс виджета. Каждому виджету свойственен определённый набор событий, т. е. действий, которые пользователь может совершать над ними (кликать по ним, выбирать пункт выпадающего списка и т.п.), эти функции задаются в теле программы и о них будет сказано ниже. Пример инициализации окна для ввода/вывода текста:

```
tb: TextBox;// в разделе var
tb := newTextBox;// в теле программы
```

Основные классы виждетов:

Button–кнопка, она имеет прямоугольную форму, внутри которой помещено её название. Объявление и вызов происходят с помощью ключевого слова Button. Пример:

```
name:Button; //bvar
name:=new Button('name'); //в begin-end
name.Click += MyClick_1;
```

В скобках указывают название кнопки или функции, за которую она отвечает. Единственным событием, которое может произойти с кнопкой является нажатие на неё, оно обозначается словом Click, т. е. этот оператор содержит ссылку на действие, процедуру которая будет исполнятся при нажатии на кнопку. Если рассмотреть пример, то там при нажатии на кнопку namebудет выполнятся процедура MyClick_1. Привязка процедуры к нажатию описано в примере и обозначается значком «+=».

Checkbox– флажок, маркер, служит для возможности выбора, подтверждения параметра, указанного в описании флажка. Создание и вывод на экран осуществляется командами, приведёнными ниже:

```
name:= CheckBox; // bvar
name := new CheckBox ('Variant'); //в begin-end
```

В скобках, вместо слова variant указывают название параметра, за который отвечает флажок. Обращение к основной функции флажка – значению (нажат/не нажат) происходит через процедуру Checked, с указанием имени виджета, она возвращает логическое true, если флажок нажат и false, если флажок не нажат. Пример использования виджета:

```
if(name.Checked=true) then c:=a+b elsea-b;
```

IntegerField– однострочное поле для ввода целых чисел. Оно представляет собой прямоугольник, внутрь которого пользователь может вписать целое число. Здание поля происходит с помощью команд, приведённых ниже:

```
name:Integerfield;// bvar
name:= new Integerfield('param', x); // в begin-end
```

Где name – имя поля, x – длинна поля, а вместо param указывают заголовок. У данноговиджета существует ряд свойств: FieldWidth – позволяет изменить ширину поля, text – позволяет изменить заголовок поля, value – позволяет обратится к значению поля (с помощью данной процедуры можно передать

значение поля в переменную). Обращение к свойствам происходит при указании имени виждета и, через точку, названия свойства. Примеры применения свойств:

name.text:=’значение’; // полю name присвоен заголовок «значение».

name.FieldWidth:=10; // полю name присвоена ширина 10 пикселей.

a:=name.Value; // в переменную а передано значение, введённое в поле name.

- RealField– однострочное поле, в которое пользователь может вписать вещественное число. Работа с данным типом вилжетов аналогична предыдущему, только при задании вместо IntegerField указывают тип RealField.
- Field –однострочное поле для ввода строковых значений. Работа с ним аналогично предыдущим виджетам типа поле, однако вместо IntegerField или RealField указывают тип Field. Кроме того, свойство text позволяет обратится к введённой в поле информации (считать значение в переменную соответствующего типа), а свойство Value для данного типа поля не применяется.
- TextBox–многострочное поле для ввода строковых переменных. Создание поля происходит при помощи команд:

name: TextBox;

name:= **new**TextBox;

У данноговиджета существует ряд свойств: Width – отвечает за ширину поля, text – отвечает за заголовок поля, Height – отвечает за высоту поля, Text – позволяет обратится к значению поля (с помощью данной процедуры можно передать значение, введённое в поле в переменную, любо наоборот, передать в поле некоторые значения переменных, текст, используя его для вывода информации). Обращение к свойствам, их изменение происходит при указании имени виждетаи, через точку, названия свойства (пример см. выше). Кроме того, для данного класса существует ряд функций: name.Undo – отменяет последнее изменение, name.Redo – отменяет отмену последнего изменения, name.Cut – вырезает выделенное, name.Copy – копирует выделенное, name.Paste – вставляет значение из буфера обмена в местоположение курсора , name.AddLine('t') – добавляет текст в новой строке:

name.AddLine('text '+ b); //вывод в поле текста и переменной

Общим свойством всех типов полей и виджетаTextBox, является то, что их можно использовать для вывода информации, полученной при работе программы, для этого свойству value или text присваивают соответствующие значение переменной или константы и выводят его в определённое место на экране.

- textLabel –метка, она позволяет выводит текст на форму окна, без использования полей. Создание метки происходит при помощи следующих команд:

name:TextLabel;// вvar

name:= **new** TextLabel ('text');// в begin-end

В скобках указывают переменные иликонстанты, которые необходимо вывести.

- ComboBox–список, т. е. поле, при нажатии на которое выпадает список

вариантов, которыми это поле может быть заполнено. Создание виджета происходит следующими командами:

```
tupes:ComboBox;// bvar
modeficCor:= new ComboBox ();// в begin-end
```

У списка существует ряд свойств: width – отвечает за ширину, count – отвечает за количество элементов. Обращение к свойствам, их изменение происходит при указании имени виждета и, через точку, названия свойства (пример см. выше).Для данного виджета существуютфункции: name.Items.Add ('variant ') – добавляет в список name элемент со значением variant, name.Items.Clear – удаляет все элементы списка. Пример использования функций добавления элемента:

```
name.Items.Add ('variant1 ');
name.Items.Add ('variant2 ');//в выпадающем списке name появляются пункты variant1, variant2.
```

Создание окна и компоновка элементов на нём. Первое, что необходимо сделать при создании оконного приложения – подключить модуль FormsABC, это делается так: в начале программы пишем:usesFormsABC. Это действие подключает модуль для работы с формами, библиотеки команд и функций, создаёт главную форму, т. е. окно, которое называется MainForm. Главная форма является основной, т. к. запускается первой, на неё помещаются все элементы, поля, кнопки и т. п. Работать будем именно с этой формой. При открытии программы пользователь выполняет требования программы, заполняет форму и запускает работу вычислительного блока программы, который обрабатывает введённую информацию и выводит результат.

После того как создана главная форма, можно изменять её название, делается это командой MainForm.Title := 'название'. Команда MainForm.SetSize(x,y) позволяет изменить размер формы, а команда MainForm.CenterOnScreen помещает форму в центр экрана.

Далее по схеме, описанной выше, на форму добавляют виджеты. При их добавлении важно помнить, что вставка первого виджета начинается в левый верхний угол, для переноса точки вставки существуют команды FlowBreak (y) и Space(x), которые делают отступ, соответственно на y пикселей по вертикали или на x пикселей по горизонтали.Виджетыдобавляют последовательно, сверху вниз, слева направо, регулируя расстояние между ними, а их размер регулируется вышеуказанными командами.

Для запуска работы расчётной части программы, вводят отдельную кнопку, либо привязывают начало выполнения расчёта к заполнению последнего поля, делают это с помощью события click, этому событию, при помощи символов «+=», присваивают соответствующую процедуру, которая и будет выполнять основные действия после заполнения формы. В этой процедуре происходит считывания значения полей в память, заполнения значений переменных, в эту процедуры подгружают подпрограммы, которые выполнят основной расчёт и выведут результат.

Пример простого калькулятора, где ввод/вывод данных осуществляются с использованием формы:

```
uses FormsABC; //обявляем модуль FormsABC
var
//задаём рабочие переменные и виджеты
a1,b1: real;
f1:FlowBreak;
f2:Space;
dec:Checkbox;
inc:Checkbox;
mul:Checkbox;
divid:Checkbox;
a:Realfield;
b:Realfield;
run:Button;
leb:TextLabel;
tb:TextBox;
procedureMyClick; //обраотчикклика
var
res: real;// переменная, где хранится ответ
begin
// присваеваем переменным значения, введённые в поля,
a1:=a.Value;
   b1:=b.Value;
ifdec.Checked=truethen
begin
res:= a1-b1;
tb.AddLine("+ res);
end else;
ifinc.Checked=true then
begin
res:= a.Value+b.Value;
tb.AddLine("+ res);
end else;
ifmul.Checked=true then
begin
res:= a.Value*b.Value;
tb.AddLine("+ res);
end else;
ifdivid.Checked=true then
begin
res:= a.Value/b.Value;
tb.AddLine("+ res);
end else;
end;
begin
MainForm.Title := 'Простой калбкулятор'; // задаём название формы
```

```
MainForm.SetSize(600,400); //задаём размер формы
MainForm.CenterOnScreen; // центрируем форму
//выводим заголовок с описанием заполнеия
leb := newTextLabel ('Введите два числа и поставьте маркер, напротив требуемого действия ');
f1 := newFlowBreak (30);//отступ по вертикали
a := newRealfield ('Число a');// вывод числа a
f2 := newSpace(25); // отступ по горизонтали
dec := new CheckBox ('-');
f2 := new Space(25);
b := new Realfield ('Число b');// выводчисла b
f2 := new Space(25);
leb := new TextLabel (' = ');
f2 := new Space(25);
tb := new TextBox;// выводокнасрезуультатом
tb.Width:= 80;
tb.Height:=20;
// выводвыборадействий
f1 := new FlowBreak ();
f2 := new Space(135);
inc := new CheckBox ('+');
f1 := new FlowBreak ();
f2 := new Space(135);
mul := new CheckBox ('*');
f1 := new FlowBreak ();
f2 := new Space(135);
divid := new CheckBox ('/');
f1 := new FlowBreak (10);
f2 := new Space(90);
run := new Button ('Рассчитать');
run.Click += MyClick;
end.
```

*Вывод.*Таким образом, PaskalABC позволяет создовать простые оконные приложения. Тут, в отличие от других языков, этот процесс является понятным, может быть реализован на базе элементарных знаний о структуре программы, тем не мение результат получится приемлимым по качеству, а при определённых условиях не будет уступать и профессионально созданным интерфейсам.

Литература

1. В. Г. Абрамов, Н. П. Трифонов, Г. Н. Трифонова, Введение в язык паскаль, главная редакция физико-математической литературы издательства "Наука, 1988г.
2. А. И. Гусева,Учимся программировать: Pascal 7.0, 2-е издание, Диалог-МИФИ, 1998г.