Содержание

Благодарности

На протяжении шести лет, пока создавалась и шлифовалась эта книга, я получал очень доброжелательную поддержку со стороны моих друзей-коллег, которые читали главы по мере их написания и дали целый ряд важных рекомендаций, касающихся содержания и ясности изложения книги. Гаврику А.П. я благодарен также за то, что он расширил опыт применения пособия в учебном процессе Харьковского национального университета радиоэлектроники. Неоценимые советы и пожелания дал зубр программирования математик Милованов Ю.Б., его замечания, аккуратно записанные на полях, я не только полностью учёл, но отдельные его советы по стилю изложения навсегда взял на вооружение при написании научных работ. Особую признательность и благодарность за ценные предложения и замечания выражаю Дорохову В.Л., добровольно взявшему на себя столь большой труд по рецензированию и редактированию первоначального наброска книги. Все названные мои друзья, как и я, – научные сотрудники, поэтому их мнение весьма существенно, ведь книга в первую очередь направлена на удовлетворение программистских запросов научных сотрудников и инженеров. Испытываю определённую неловкость за возложенную на них нагрузку, несмотря на их заверения о том, что книга доставила им удовольствие. Спасибо вам, друзья.

Предисловие

Все вопросы программирования в этом пособии рассматриваются с *самого начала* и *исчерпывающе полно*. Это делает его полезным даже для тех, кто причисляет себя к программистам с определённым опытом. Однако в первую очередь пособие предназначено для научных сотрудников, инженеров и студентов технических высших и средних учебных заведений, которым в ходе основной деятельности требуется уметь самостоятельно разрабатывать программы.

Место C++ Builder *среди других языков и сред программирования*

C++ Builder – это одна из самых современных и эффективных сред программирования. Что собой представляет среда программирования и язык программирования? Для ответа на эти и другие близкие вопросы вначале кратко рассмотрим историю развития программирования.

В конце 40-х годов прошлого столетия, когда появились первые ЭВМ (электронно-вычислительные машины), использовалось программирование в адресах: последовательность команд процессора в машинных кодах. Машинный код – это совокупность операций, которые может выполнять ЭВМ. Машинный код иными словами это язык машины. ЭВМ (а позже и персональный компьютер) воспринимает все команды *только* в двоичной системе счисления. При программировании в кодах команды (они значительно отличались на разных ЭВМ) для удобства записывали в восьмеричной системе счисления (получалась более короткая вереница цифр команды). В конечном итоге программа представляла собой набор двоичных цифр. Для выполнения, например, какого-то арифметического действия над двумя числами требовалось обратиться к ячейкам (адресам), где хранятся эти числа и применить к ним код операции (специально выделенный номер), а затем результат операции занести в ячейку по определённому адресу (номеру). В целом этот кошмар, по-другому такой способ программирования назвать трудно, продолжался 5 – 10 лет. Хотя и в середине 70-х ещё выпускались учебники по упомянутому языку программирования, однако многие разработчики программ (но далеко не все) уже переходили на языки, более удобные для человека.

В 50-е годы разработан язык *Assembler* (сборщик), в котором номера команд (операций) процессора заменили словами (или частью слов) английского языка. Этот язык программирования является языком *низкого* уровня, его применение существенно упростило разработку программ (написание, проверку и отладку). У каждого процессора имелся *свой* Ассемблер с разным числом команд. *Assembler*

используется и в настоящее время для написания очень быстродействующих и компактных программ. Однако разрабатывать на нём большие программы практически невозможно.

Создателю программы более удобно весь ход решения задачи записать на своём *человеческом* языке, а не в машинных кодах (пусть даже эти коды и заменены мнемоническими, легко запоминающимися обрывками слов). Языки программирования, где используются команды в виде слов, называются языками *высокого* уровня. Поскольку ЭВМ изначально создана для проведения *научных* вычислений, то для удобства «перевода» формул на язык машины первым (1954–57 г.г.) был разработан *Fortran* (*FORmula TRANslator*). Прежде всего, он предназначался научным сотрудникам.

Для профессиональных программистов созданы языки: *Algol* (1958–1960 г.г.), *Cobol* (1959–1960 г.), *C* (1972 г.), *C++* (1983 г.), *Ada* (1983–1995 г.г.), *C++ANSI* (1988 г.), *Object Pascal* (1988 г.) и другие. Если *Cobol* в основном ориентирован на разработку программ обработки финансовых операций, то *C* изначально задуман для системного программирования, написания компиляторов и операционных систем. Операционная система *UNIX* и большинство программ для неё созданы на *C*. Язык *C++* является результатом совершенствования и развития *C*, он обогащён технологией объектного программирования (см. ниже).

Для уровня школьников порождён легко усваиваемый *Basic* (1964 г.), а для обучения профессиональному программированию (в основном для студентов) разработан – строгий *Turbo Pascal* (1971–1990 гг.). Полагается, что *Basic* наиболее пригоден для самого первого знакомства с программированием. Однако наш опыт преподавания свидетельствует о том, что первым языком программирования, безусловно, должен быть *Turbo Pascal* или *C++*, но никак не *Basic* (имеет более трёхсот разновидностей), приучающий разработчиков приложений к *плохим* стилю и приёмам программирования.

С течением времени в ходе своего развития все языки программирования высокого уровня заимствуют друг у друга всё лучшее. Поэтому современные версии, например *Basic* и *Turbo Pascal*, по своим возможностям *вполне* могут использоваться для решения большинства научных и инженерных задач. Но вплотную подойти к богатейшим возможностям *C++* ещё *ни одному* языку не удалось.

Когда число строк кода программы более 10–100 тыс., то применяется специальная технология ООП – объектно-ориентированное программирование (без её использования *быстрая коллективная* работа над приложением невозможна). Эта технология подробно рассматривается в нашем пособии. Языки *C++*, *C++ANSI*, *Object Pascal* ориентированы на ООП-технологию, а *Turbo Pascal* лишь поддерживает её.

Для неспециалистов кажется *невероятным* тот факт, что, применяя языки программирования высокого уровня, приблизительно 90–95% времени, затрачиваемого на разработку программы, уходит на создание интерфейса (средств управления: окон ввода и вывода информации, пунктов меню, командных кнопок и др.). И только 5 – 10% времени посвящается решению собственно основной задачи. Для успешной борьбы с такой несправедливостью фирмы-разработчики про-

граммного обеспечения предложили специальные среды быстрой разработки приложений или *RAD*-среды (*Rapid Application Development*). Такими средами являются:

- *Borland C++ Builder* (версия 5.0–выпущена в 1998 г., версия 6.0–в 2000 г., версия 2006– в 2005 г., версия 2007– в 2007 г., версия 2009– в 2008 г. (выпускается фирмой *Code gear*));
- *Borland Delphi* (версия 1.0– выпущена в 1995 г., версия 5.0–в 1999 г., версия 6.0– в 2001 г., версия 7.0–в 2002 г., версия 2005–в 2005 г., версия 2007– в 2007 г., версия 2009–в 2008 г. (выпускается фирмой *Code gear*));
- *Microsoft Visual Basic* (версия 1.0 выпущена–в 1991 г, версия 6.0–в 1998 г);
- *Microsoft Visual C++* (версия 1.0– выпущена в 1992 г., версия 6.0–в 1998 г).

При помощи большого набора специального инструментария эти среды берут на себя заботы по разработке всех удобств, облегчающих использование программ-приложений: полей ввода и вывода информации, наглядности форм её представления и многое, многое другое. При этом упомянутые среды генерируют соответствующие коды программ автоматически и практически мгновенно. В основу *RAD*-среды *Delphi* положен *Object Pascal*, а *RAD*-сред *Visual C++* и *C++ Builder* – язык *C++*. Отличия *C++ Builder* 2007 от *C++ Builder* 6 небольшие: появилась ещё одна стандартная библиотека и несколько визуальных компонентов, усовершенствована *Интегрированная Среда Разработки* (ИСР). В *C++ Builder* 2009 улучшена комфортность работы с базами данных, добавлены библиотека и визуальные *Internet*-компоненты, на 10% ускорен запуск *RAD*-среды. Кроме того, в *C++ Builder* 2009 повышена совместимость с кодом на *Delphi* и появилась возможность применять компоненты *Delphi*. *Builder* 2009 можно установить в любой из операционных систем: *Windows* 2000, *Windows XP* или *Windows Vista*. Вместе с тем, ошибки компилятора, обнаруженные нами в *Builder* 5, *Builder* 6, в *Builder* 2009 не устранены (подробнее см. в п. 14.10.2).

Современный мир немыслим без *Internet* (Всемирная Глобальная Сеть *Word Wide Web* была создана в 1989 г.), поэтому в настоящее время все новые среды программирования позволяют осуществлять взаимодействие приложений во всемирной сети (однако не все такие приложения являются платформенно-независимыми). Для создания *Web*-страниц используются специальные языки *разметки*, позволяющие управлять форматированием и размещением элементов страниц. Наибольшее распространение получил язык *HTML* (*HyperText Markup Language* – язык гипертекстовой разметки, предложен в 1989 г.).

Язык *HTML* даёт возможность создавать документы лишь со *статическими* текстами, таблицами и изображениями. Для разработки *Web*-страниц, на которых ведётся диалог с клиентом, выполняется анимация изображений, реализуется контекстно-зависимый текстовый и графический ввод-вывод, вначале стали использоваться языки высокого уровня *Java* (разработан в 1995 г.) и *JavaScript* (разработан в 1995–1996 г.г.). Созданные с их помощью приложения являются *независимыми* как от типа используемого компьютера, так и от операционной системы. Это позволяет применять такие приложения на *всех HTTP*-серверах и *всех*

браузерах клиента. Следует отметить, что первый браузер появился в 1991 г. (с 1994 г. браузеры начала выпускать фирма *Netscape*). Язык *Java* – объектно-ориентированный язык, созданные с его помощью приложения работают на *различных* платформах (межплатформенный язык), он унаследовал синтаксис *C* и *C++*. *JavaScript* – очень похож на *Java*, но существенно проще его. *JScript* – это *JavaScript* в реализации фирмы *Microsoft*.

Фирмы-разработчики программного обеспечения, непрерывно конкурируя друг с другом, создают всё новые и новые продукты (с интервалом 1,5 – 2 года). В их приложениях не только реализуются *новые* пути решения актуальных задач, но и даётся возможность осуществлять упомянутые решения с привлечением *уже существующих* языков программирования, которые давно стали надёжным инструментарием многих разработчиков программ. Такая политика, конечно, весьма способствует продажам продуктов, повышает их востребованность.

В частности, фирма *Microsoft* предложила среду *Visual Studio.Net*, как конкурента среды программирования *Borland C++ Builder*. *Visual Studio.Net* выпускалась в различных версиях: версия 6.0 – в 2000 г., версия 2002 или 7.0 – в 2002 г., версия 2005 или 8.0 – в 2005 г., версия 2008 или 9.0 – в 2007 г. Среда программирования *Visual Studio.Net* является средством разработки *Windows*-приложений (*Windows Application*) в операционных системах *Windows*-95, 98, *NT*, *Vista* (только в *Visual Studio.Net* 2008), *Internet*-приложений (*ASP.Net*) и консольных *Dos*-приложений (*Consol Application*). Следует заметить, что консольные приложения можно разрабатывать всеми средами программирования. Основой обсуждаемой среды является *новый* компонентно-ориентированный язык программирования *C#* (читается – *си шарп* или *си диез*, создан в 1999–2000 г.г.), он *специально* разработан для технологии «*.Net*» (читается – *дот нет*). Этот самый новый язык разработки программ обеспечивает *совместную* работу компонентов, написанных с использованием *различных* языков программирования. В среде *Visual Studio.Net* можно создавать приложения при помощи следующих языков программирования: *Visual Basic* 6.0, *Visual C++*, *C#*, *JavaScript* (*JavaScript* имелся только в версии 6.0). Синтаксис *C#* похож на синтаксис *C++* и *Java*.

Платформа или оболочка «*.Net*» – это *многоязыковая* среда, она *открыта* для *свободного* включения *новых* языков (компиляторов и инструментальных средств), созданных не только *Microsoft*, но и другими фирмами. Программы, написанные с использованием технологии «*.Net*», могут функционировать на других компьютерах, *только* в том случае, когда на них установлена исполняющая (операционная) среда «*.Net Framework*» (запущен файл *dotnet.exe*), являющаяся оболочкой, надстройкой над существующими операционными системами и языками программирования Необходимо заметить, что среда «*.Net Framework*» распространяется фирмой *Microsoft* бесплатно. Код, порождённый в среде разработки «*.Net*», носит название *управляемого* кода (*manage code*) в отличие от обычного *неуправляемого* кода (*unmanage code*). Программа с управляемым кодом *непрерывно* контролируется средой «*.Net*»: операционная среда *не позволяет* программе выполнить действия, нарушающие концепцию безопасности «*.Net*» (что повышает надёжность функционирования приложений). В настоящее время, по

прошествию около 9 лет с момента появления «*.Net*», только *небольшая* часть программистов (около 25%) использует эту новую весьма оригинальную технологию.

В среде разработки *Borland Developer Studio* 2006 (она выпущена в 2006 г, составной частью в неё входит и *C++Builder* 2006) помимо традиционных приложений *Windows* можно создавать «*.Net*»-программы при помощи языков *Delphi Language* (так стал называться *Object Pascal*) и *C#*. Заметим, всё, что можно написать на *C#*, можно запрограммировать и на *Delphi Language* для «*.Net*» с не меньшей эффективностью [15]. Среда разработки *Borland Developer Studio* 2006 (далее – *BDS*) предназначена, прежде всего, для корпоративных пользователей.

Для повышения продаж (путём существенного уменьшения цены) фирма *Borland* в 2006 г. «разрезала» *BDS* на составные части. Каждая часть включает только *один* язык и платформу. В результате появилась линейка продуктов серии *Turbo*: *Turbo Delphi* для *Win*32, *Turbo Delphi* для «*.Net*», *Turbo C++* для *Win*32 (вариант *C++Builder*), *Turbo C#* для «*.Net*». Эти продукты предназначены для *небольших* групп: студентов, программистов-одиночек и индивидуальных пользователей. Каждый из семейства *Turbo* выпускается в двух редакциях: бесплатная *Explorer* (для студентов и непрофессионалов) и *Professional* (для профессионалов). Бесплатная лицензионная версия имеет не все возможности версии для профессионалов, однако её *разрешается* использовать для разработки и *коммерческих* приложений.

Наш обзор не претендует на полноту, в нём упомянуты лишь *наиболее* популярные среды разработки программ *ведущих* фирм. Поэтому отметим ещё и фирму *Sun Microsystems*, выпустившую в 2007 г. новую версию среды разработки для языков *C*, *C++* и *Fortran* под операционные системы *Solaris* и *Linux*.

Будущее покажет, насколько перспективной окажется «*.Net*»-технология, будет ли ей сопутствовать удача, или она погибнет (будет забыта) в результате субъективизма, инертности программистов или появления ещё более совершенной технологии.

Мода правит бал не только на подиуме, она распространяет свои неподдающиеся логическому анализу чары даже на языки программирования. К настоящему времени создано около 3 000 языков, самые распространённые из них не обязательно являются лучшими языками программирования. Начинающему выбрать «самый-самый» не представляется возможным. Даже умудрённые опытом специалисты не могут это сделать, ведь требуется в совершенстве знать каждый из них. История и фирмы-разработчики программного обеспечения, конечно, делают свой выбор, в результате популярным становится тот или иной язык. При этом следует отметить [19], что за рубежом *C++Builder* и *Delphi* такой известностью, как у нас (в постсоветском пространстве) не пользуются, там царствует *Visual Basic* (в котором даже нет пользовательских объектов...). Поэтому популярность имеет не только временную, но ещё и региональную составляющую.

Мы несколько отвлеклись от темы (место *C++Builder* среди других языков) для того, чтобы подготовить вас к следующему заключению. В настоящее время практически все задачи можно *успешно* решать при помощи целого *ряда* языков

программирования. Язык *C#* и технология «*.Net*» весьма перспективны, однако на подавляющем числе компьютеров не установлена оболочка «*.Net Framework*», что несколько препятствует их распространению. Вместе с тем далеко не всем нужны *Internet*-программы, *Windows*-приложения пока ещё *никто* не отменял, а старые платформенно-независимые механизмы основываются на *си*-подобных языках (*Java, JavaScript*). Поэтому выбор для изучения *C++Builder* позволяет *уже сейчас* вооружить вас, дорогие читатели, мощнейшим инструментом разработки *Windows*-приложений, создаёт прочные основы для быстрого изучения (если вам это понадобится) любого из языков: *C#*, *Java, JavaScript*. Доверьтесь нашему опыту и примите во внимание рекомендацию: начинать обучение *C++* и визуальную технологию разработки приложений значительно комфортнее и проще на основе *C++Builder* 6 (или 5), но не *C++Builder* 2009. Последним продуктом и его последующими популяциями (они, без сомнения, появятся спустя 2 – 3 года) вы сможете овладеть *самостоятельно* на основе знаний, полученных из настоящего пособия.

Не для рекламы, а в качестве констатации факта заметим, что на *C++* написаны и операционная система *Windows* и многие известные прикладные программы. При этом ещё раз подчеркиваем, *C++ специально* разрабатывался для *профессиональных* программистов (системных программистов и разработчиков программного обеспечения). Поэтому для новичков в программировании он не всегда сразу понятен. Среди программистов бытует шутка: «язык *C++* был придуман только для того, чтобы программистами не становились *случайные* люди». Однако, делающие в программировании первые шаги, могут не огорчаться: среда программирования *C++ Builder* практически ликвидировала все сложности *C++*, сделала процесс разработки программ ясным и увлекательным. Да, да – очень увлекательным. Заверяем читателей: вы овладеете *основными* приёмами программирования в среде *C++ Builder*, научитесь совершенствовать ваши навыки для дальнейшего расширения набора средств по программированию задач *достаточной* сложности.

Что собой представляет пособие

Это пособие – самоучитель. Изложение ведётся в виде беседы автора с читателем, практически *все* возможности языка иллюстрируются краткими примерами, закрепляются вопросами для самоконтроля и задачами для программирования. ООП освещается *вначале* лишь в общих чертах, в объёме, необходимом для управления средой *Builder*. После овладения основными вопросами синтаксиса и семантики *C++Builder* подробно излагается технология *объектного* конструирования программ.

Цель пособия – изучить *основные* возможности языка *C++*, реализованного средами *C++Builder* 6.0 и *C++Builder* 5.0. Всё изложение ведётся на базе *C++Builder* 6.0, а в тех случаях, когда эти операции выполняются по-другому в *C++Builder* 5.0, даются подробные разъяснения (но таких ситуаций менее полдесятка). Цель учебника: дать читателю такой объём глубоко осознанных знаний,

который позволил бы ему с применением справочной системы *C++Builder* или справочников по *C++*[8] и *C++Builder* 6.0 [1–3, 20–21] самостоятельно решать и круг вопросов, не затронутых в пособии.

Материал самоучителя излагается так, чтобы читатель не робел перед разработкой программ. Для этого изучаемые вопросы, задачи и алгоритмы расположены по нарастанию их сложности. При этом круг тем ограничен теми, которые *наиболее* важны для программирования *подавляющего большинства* задач, возникающих при обработке и анализе данных.

Настоящее пособие не только иллюстрирует *C++Builder* примерами и задачами, подобранными по нарастанию их сложности. Прежде всего, оно является *подробным последовательным* учебником. В ходе повествования не только указываются пути решения вопросов или смысл того или иного ограничения, но и подробно разъясняется, почему эти пути должны быть именно такими, а не другими, чем обусловлено ограничение. Это способствует как пониманию материала, так и его запоминанию.

В учебнике достаточно вопросов и заданий, с которыми можно справиться, не имея перед собой компьютера, даже в транспорте. Это было сделано не для того, чтобы увеличить набор литературы для путешественников или жителей спальных районов крупных городов. Дело в том, что для глубокого понимания вопросов очень полезно «прокручивать» все шаги компьютера у себя в голове, следует уметь *читать* тексты программ. Однако каждую порцию прочитанного и, на первый взгляд, хорошо понятого материала *необходимо* закреплять его практической реализацией на компьютере. Иначе научиться программировать *невозможно*.

Основные рекомендации для читателей

Рекомендуется *последовательно* изучать урок за уроком, *отвечать* на вопросы для самоконтроля и *выполнять* задания. При этом эффективность усвоения *существенно* повысится, если полностью отлаженные программы *записывать* на отдельных листах бумаги. Именно записывать, а не распечатывать при помощи принтера, поскольку в первом случае знания надолго фиксируются в вашей памяти. Этот банк задач удобно использовать при разработке новых программ, когда требуется быстро вспомнить уже применяемый ранее алгоритм или забытую конструкцию языка.

В ходе изучения уроков вы обнаружите, что уроки, пройденные ранее, несколько подзабыты. Это вполне нормально. Невозможно всё удержать в голове. Для борьбы с этим явлением вы создадите конспект решённых задач, есть, наконец, и настоящий учебник, который разумно перелистывать не только вперёд, но и назад.

Настоятельно советуем читать коды программ-решений задач для самостоятельного программирования. Это полезно делать, когда в вашей программе что-то не клеится, и даже когда вы *самостоятельно* получили правильное решение. При этом вы много раз удостоверитесь в том, что имеется много вариантов реше-

ний. Надеемся, что *ваши* программы будут оригинальнее тех, что приведены в пособии.

Если вы, дорогой читатель, не обладаете навыками работы в операционной среде *Windows,* в текстовом редакторе *Word* и в табличном процессоре *Excel*, то смело можете *отложить* чтение настоящего пособия. Оно вам окажется более полезным лишь после овладения, пусть даже минимальными, сведениями по этим продуктам. Мы ориентируемся на читателей, которые уже умеют набирать и редактировать текст в текстовом редакторе *Word,* знают, как в различных продуктах *Windows* осуществляется выделение блока, его помещение в буфер обмена, копирование, удаление и вставка. Методы перетаскивания (буксировки), изменения размеров выделенных объектов при помощи их маркеров и мыши (метод протягивания) также полагаются известными. Впрочем, всем этим элементарным премудростям можно научиться и в среде программирования. Однако овладеть упомянутыми офисными программами вам всё равно понадобится позже при *оформлении* результатов исследований, расчётов, нобелевских докладов и прочих материалов. Поэтому не рекомендуем откладывать эту работу в долгий ящик, ведь указанными продуктами обязаны владеть инженеры, научные сотрудники, журналисты, поэты и прочая писательская братия.

Если при изучении таких языков высокого уровня, как, например, *Fortran*, *C++, Turbo Pascal, QBasic* и др., знания английского языка лишь весьма желательны, однако далеко необязательны (30 – 40 ключевых слов языка можно легко запомнить всем), то ситуация с *C++ Builder* несколько иная. Для *эффективной* работы необходимо знать английский язык хотя бы в рамках программы средней школы. Имена всех идентификаторов и команд лучше запоминаются, «с лёта» понимается их назначение, если вам известны английские слова, положенные в основу упомянутых служебных слов; информацию о них значительно быстрее в этом случае найти (и понять) в англоязычной справочной системе. Для понимания различных сообщений *RAD*-среды, информации справочной системы необходимо уметь разбираться в технических текстах на английском языке, пусть даже со словарём.

К пособию прилагается диск с программами, приведенными во всех уроках (отдельно для *C++Builder* 5.0 и *C++Builder* 6.0). Наш совет: обращайтесь к ним только в исключительных случаях, программы значительно полезнее набирать и отлаживать *самостоятельно*.

В заключение желаем вам большого *интеллектуального наслаждения* в увлекательной работе по освоению *C++Builder*! Будем очень рады *любым* замечаниям и пожеланиям, касающимся данной книги.

Урок 1. Первая программа

На этом уроке вы познакомитесь с интегрированной средой разработки (*IDE – Integrated Development Environment*), а также основными определениями и этапами проектирования программ. Все эти вопросы иллюстрируются на приложении (синоним слова программа), назначение и внешний вид которого описаны ниже.

1.1. Внешний вид и назначение приложения *Умножитель*

Внешний вид приложения или его интерфейс показан на рис. 1.1. В области заголовка интерфейса находятся пиктограмма (небоскрёбы), название приложения (*Умножитель*) и стандартные граничные пиктограммы, характерные для большинства окон *Windows* (*Свернуть*, *Развернуть*, *Закрыть*). На интерфейсе имеется также два прямоугольных *поля ввода* данных (информации), над которыми расположены надписи *Цена* и *Количество*, а также командная кнопка с заголовком *Умножение*. С использованием этого приложения имеется возможность вычислить стоимость отобранного вами товара (например, в магазине). Для этого необходимо в упомянутые поля ввода занести данные о цене и количестве товара, после чего мышкой нажать командную кнопку *Умножение*. Произведение цены товара и его количества появится в *поле вывода* под надписью *Стоимость*.

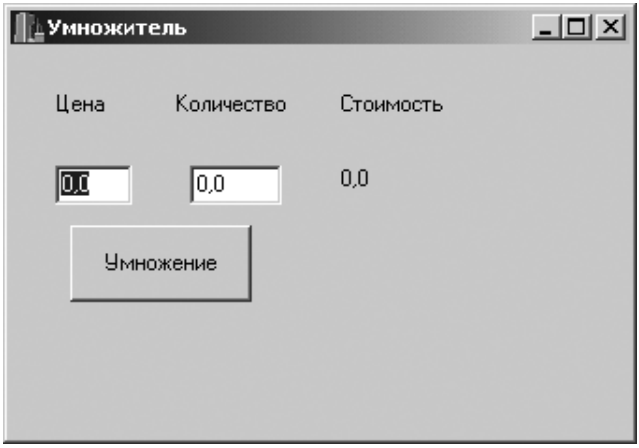

Рис. 1.1. Интерфейс программы *Умножитель*

В исходном состоянии в полях ввода и вывода установлены нулевые значения. Эта программа называется *Умножитель*, поскольку предназначена для умножения двух чисел.

Задача заключается в создании такого приложения. Она состоит из двух частей. Вначале с использованием средств визуального программирования построим интерфейс приложения, а затем разработаем программный код, обеспечивающий его функционирование. Назначение задачи состоит в том, чтобы на её основе сделать первые *глубоко осознанные* шаги в визуальном программировании и программировании на *C++*.

1.2. Знакомство с визуальным программированием

Когда вы первый раз увидите главное окно *IDE C++ Builder* (см. рис. 1.2), не пугайтесь! Все его вкладки, пиктограммы и меню *сразу* не понадобятся, с ними будем знакомиться *постепенно*. Далее для краткости изучаемая интегрированная среда разработки будет именоваться также *Builder*.

Интерфейс *IDE* состоит из четырёх макрокомпонентов (см. цифры на рис.1.2), их можно перетаскивать в любое место рабочего стола, изменять размеры. Познакомимся детально со всеми макрокомпонентами.

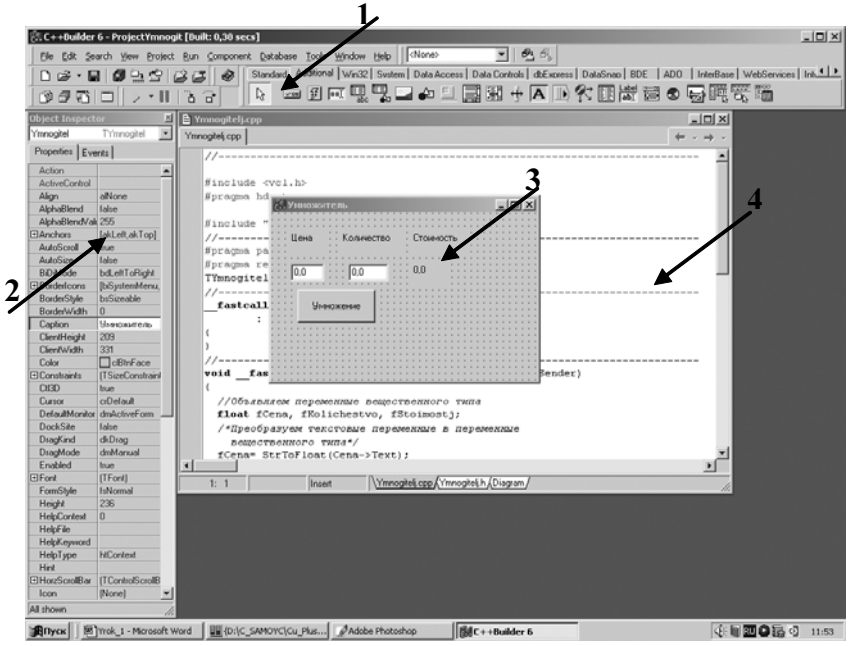

Рис. 1.2. Интерфейс *C++Builder*.
1– Панель Управления, 2– Инспектор Объектов,
3– Форма, Визуальный Проектировщик или Дизайнер Форм, 4– Редактор Кода