Современные технологии программирования

И.А. Вилков
Научный руководитель – доцент, канд. техн. наук. Е.П. Догадина
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23
Тел.: (49234)77272
e-mail: ivvilkov@yandex.ru

Определение временной и емкостной сложности алгоритмов при равномерном весовом критерии

В работе исследуется такая проблема, как емкостная сложность алгоритмов, характеризующая длины описания алгоритмов.

Сложность алгоритма может быть различной и при фиксированном размере входа, но различных входных данных. Примером может послужить поиск контрарной пары двух дизъюнктов фиксированной длины, контрарные литералы могут оказаться первыми или последними в этих дизъюнктах. Следовательно, время поиска контрарной пары в этих случаях окажется различным, поэтому различают сложность в худшем и в среднем случае.

Сложность в худшем случае – это максимальная при данном размере входа сложность.

Средняя сложность – это средняя по всем входам данного размера сложность.

Емкостная сложность в худшем случае – это функция $S(n)$ – равная максимальной (по всем входам размера n) из сумм емкостей всех ячеек (элементов) памяти к которым было обращение.

Для точного определения емкостной сложности требуется знать объем памяти используемый каждой переменной. Так как это значение зависит от конкретной вычислительной машины, в теории сложности алгоритмов принято оценивать порядок сложности.

При определении порядка емкостной сложности в качестве единицы необходимо выбрать некоторый элементарный в данном классе алгоритмов объект, таким объектом в задаче логического вывода на основе метода резолюций является литерал.

При оценке сложности могут использоваться различные критерии:

1. Равномерный весовой критерий предполагает, что каждая переменная в алгоритме занимает одну ячейку памяти независимо от принимаемых ей значений.

2. Логарифмический весовой критерий учитывает ограниченность размера реальной ячейки памяти.

Например, необходимо определить временную и емкостную сложность алгоритма сортировки массива по возрастанию методом «пузырька» при равномерном весовом критерии. В соответствии с данным методом для всех элементов массива, начиная со второго, последовательно выполняется «всплывание», при котором текущий элемент, если он меньше предыдущего, меняется с ним местами. Обозначим i – текущий номер элемента при просмотре массива в глубину, j – номер текущего элемента при «всплывании» (процедура elem_up), $A(j)$ – j-ый элемент массива, n - число элементов в массиве.

Определим временную сложность при равномерном весовом критерии. Очевидно, что размером входа для данного алгоритма следует считать число n элементов в массиве, а шагом алгоритма – тело цикла в процедуре «всплывание». В теле основной программы имеем $(n-1)$ проходов по циклу. В процедуре elem_up ("всплывание" элемента) – в худшем случае выполняется i проходов по циклу. Тогда, общее число шагов в алгоритме равно:

$$\sum_{i=2}^{n} i = (n+2)*(n-1)/2 = (n^2+n-2)/2$$

Например, при $n=4$ осуществляется три входа в основной цикл $(n-1)$ и 6 операций в подпрограмме $(n+2)$.

Таким образом, временная сложность алгоритма при равномерном весовом критерии есть $O(n^2)$.

Емкостная сложность при равномерном весовом критерии, очевидно, определяется размером n массива, поэтому есть $O(n)$.

Литература

1. Архангельский Язык Pascal и основы програмирования в Delphi. Учебное пособие - М.: ООО «Бином-Пресс», 2004 – 496с.

2. Хомоненко А., Гофман В., Никифоров В. Delphi 7. – СПб.: БХВ- Петербург, 2004. – 1200 с.

С.В. Изъюров
Научный руководитель – старший преподаватель
кафедры электроники и вычислительной техники
Д.В. Бейлекчи
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23
e-mail: evt@mivlgu.ru
e-mail: kaf-eivt@yandex.ru

Учебно-отладочный стенд для изучения применения беспроводной технологии Bluetooth в микроконтроллерных системах

В докладе рассматриваются результаты исследования методов применения беспроводной технологии Bluetooth микроконтроллерных системах и разработка учебно-отладочного стенда для изучения данной технологии.

В настоящее время технология Bluetooth активно применяется в компьютерных технологиях для беспроводного управления и передачи данных. Она обеспечивает обмен информацией между персональными компьютерами, мобильными устройствами, а также периферийными устройствами, такими как мышки, клавиатуры, джойстики, наушники, гарнитуры. В промышленности данная технология может применяться для дистанционного управления различными объектами, а также дистанционного мониторинга среды.

Для обеспечения беспроводной связи используется свободный от лицензирования диапазон 2,4-2,4835 ГГц. В Bluetooth применяется метод расширения спектра со скачкообразной перестройкой частоты, что обеспечивает устойчивость к широкополосным помехам, а также позволяет исключить конфликты с другими беспроводными технологиями, которые используют такой же диапазон частот, например, WiFi. Кроме того, данный метод является также составной частью системы защиты конфиденциальности передаваемой информации: перестройка происходит по псевдослучайному алгоритму и определяется отдельно для каждого соединения. Недостатком применения данного диапазона частот является то, что дальность связи сильно зависит от преград и помех.

В компьютерной техники наиболее распространены передатчики Bluetooth класса 2, которые работают на дальностях связи 10-30 метров (класс 2). При применении данной технологии в промышленности, часто необходимо большее расстояние связи. Это обеспечивается несколько более дорогими передатчиками класса 1, которые обеспечивают связь в радиусе 100-200 метров. Существуют также передатчики класса 0, обеспечивающие связь на расстоянии 1 км.

Для изучения применения технологии Bluetooth для беспроводного управления и передачи данных в микроконтроллерных системах, актуальна задача разработки учебного стенда.

Реализация технологии Bluetooth возможна с использованием микросхем приемопередатчиков с контроллером физического уровня (микросхемы CSR, STM, Toshiba) или с использованием микромодулей реализующих логический уровень передачи Bluetooth-модулей (Infineon, Rayson, BlueGiga).

В настоящее время для большинства разработок применяются микромодули, так как они, имея миниатюрный размер около 1 на 3 см, содержат все необходимые интерфейсы связи для управления модулем и передачи данных, такие как USB, UART, аудио-PCM. Кроме того, они содержат радиочастотную часть печатной платы, к которой предъявляются повышенные требования на конфигурацию проводников. Таким образом, готовая реализация этой части платы упрощает разработку всей микроконтроллерной системы.

В результате исследования возможностей элементной базы различных производителей было определенно, что Bluetooth-модули финской фирмы BlueGiga наиболее подходят для реализации проектируемого стенда, так как они имеют встроенный цифровой аудио-интерфейс, хорошо документированную систему команд конфигурации и управления, а также встроенную чип-антенну с возможностью подключения при необходимости внешней антенны для

увеличения дальности связи. Для данных модулей существуют отладочные платы, которые содержат кроме модуля также микросхему аудио-кодека, который обеспечивает кодирование и декодирование аналогового аудио-сигнала, микросхемы микрофонного усилителя и выходного аудио-усилителя для подключения гарнитуры.

На рисунке 1 приведена общая структурная схема учебно-отладочного стенда для изучения применения беспроводной технологии Bluetooth представленного в виде системы из двух модулей: стационарного и удаленного.

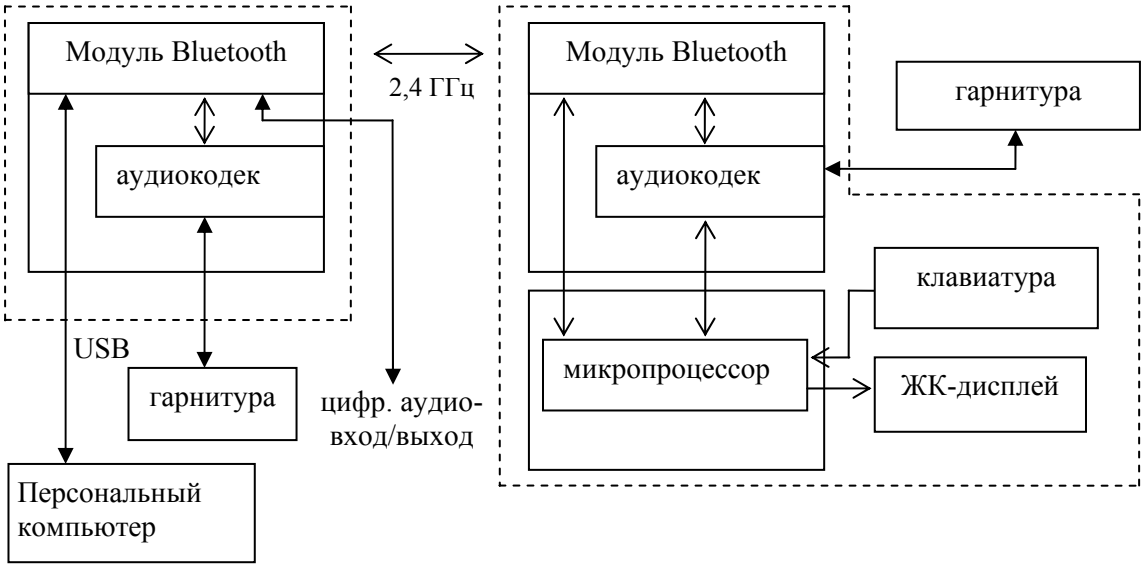

Рис. 1. Структурная схема учебно-отладочного стенда

Стационарный модуль подключается к персональному компьютеру через интерфейс USB и обеспечивает связь удаленного модуля с персональным компьютером. С помощью персонального компьютера выполняется конфигурирование модуля, а также осуществляется передача управляющих команд на удаленный модуль и данных отображаемых на ЖК-дисплее.

К стационарному модулю, также подключается гарнитура, обеспечивающая ввод/вывод голосового сигнала через удаленный модуль.

Удаленный модуль представляет два блока: блок беспроводной связи и блок управляющего микроконтроллера.

Блок беспроводной связи управляется микроконтроллером и обеспечивает ввод/вывод голосового сигнала при помощи гарнитуры, беспроводную прием/передачу аудио-сигнала, беспроводную передачу данных принимаемых от микроконтроллера и передачу микроконтроллеру данных, принимаемых по беспроводному каналу.

Блок управляющего микроконтроллера обеспечивает управление блоком беспроводной связи, ЖК-дисплеем и клавиатурой. В качестве управляющего микропроцессора применен микроконтроллер семейства AVR фирмы Atmel. Программное обеспечение управляющего микропроцессора разрабатывается на языке С и обеспечивает конфигурирование Bluetooth-модулей, обработку обнаружения доступных устройств в радиусе работы передатчика, управление сеансами беспроводной связи, управление системой кодирования/декодирования аудио-сигнала, обработку клавиатуры, индикацию состоянии устройства и отображение принятых данных на ЖК-жисплее.

Учебно-отладочный стенд позволяет изучать применение технологии Bluetooth следующими способами:

- конфигурирование и управление Bluetooth модулем при помощи программы-терминала на персональном компьютере;
- разработку программ управления и передачи данных для персонального компьютера;
- разработку программ управления и передачи данных для микроконтроллера.

Таким образом, в результате работы была исследована технология Bluetooth, элементная база необходимая для реализации данной технологии в микроконтроллерной системе и разработан учебно-отладочный стенд для изучения применения технологии Bluetooth в микроконтроллерных системах.

Я.В. Коновалов,
А.Ю. Сутягин
Научный руководитель –
ассистент кафедры электроники и вычислительной техники
А.А. Колпаков
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23
Тел.: (49234)77272

Методы и технологии решения задач общего назначения с использованием потоковых графических процессоров

Параллельные вычисления на сегодняшний день являются одной из наиболее актуальных и приоритетных тем для исследования. Во всех ведущих IT компаниях (Microsoft, Intel, AMD, NVIDIA, Google, и т.д.) уже давно имеются подразделения и лаборатории, занимающиеся проектированием и разработкой высокопроизводительных систем основанных на алгоритмах параллельной обработки данных. Таким пристальным вниманием, параллельные вычисления обязаны стремительным ростом объемов данных, нуждающихся в обработке. Спектр исследуемых задач крайне широк. Сюда входит анализ и обработка изображений, симуляция физических процессов, прогнозирование различных процессов, анализ спутниковых данных, финансовые расчеты, электромагнитные расчеты, нейронные сети и многое другое. Часть этих задач может быть решена за счет использования распределенных вычислений в облаке (Cloud Computing), другая часть может быть решена за приемлемое время лишь, будучи запущена на суперкомпьютере обладающим производительностью в сотни терафлопов (TeraFLoating point Operations Per Second, TFLOPS). Но также существует ряд задач, которые по тем или иным должны выполняться локально, на клиентской машине. До недавнего времени эта задача являлась почти что не решаемой. Центральный процессор, являющийся основным вычислительным устройством любого современного компьютера просто не способен решать эти задачи за приемлемое время.

На сегодняшний день развитие центральных процессоров практически остановилось. Дальнейший рост вычислительной производительности CPU сопряжен с рядом сложностей технологического характера. Так, к примеру, увеличение частот универсального процессора уперлось в физическое ограничение на размер чипа и высокое энергопотребление. Тем не менее, роста производительности можно добиться за счет размещения нескольких ядер в одном чипе. Большинство современных настольных систем оснащается двуядерными процессорами, что позволяет пользователю комфортно работать с подавляющим большинством прикладных пакетов. В серверных конфигурациях, где уровни нагрузок на систему существенно выше, как правило, используются более производительные чипы, оснащенные четырьмя или даже восьмью ядрами.

Графические процессоры изначально нацелены на решение узкого круга задач, связанного с компьютерной обработкой графических данных. В связи с этим архитектуры GPU и CPU существенно отличаются друг от друга. Так, к примеру, в видеочипах от NVIDIA основной блок представляет собою мультипроцессор с восьмьюдесятью ядрами, и несколькими тысячами регистров. Графические процессоры от NVIDIA так же оснащены несколькими видами памяти: локальная, разделяемая общая, константная, а так же глобальная память, доступная всем мультипроцессорам на чипе.

В настоящий момент большая часть персональных компьютеров оснащается отдельными видеокартами либо встроенными видеочипами. Они могут стать решением проблемы обработки трудоемких вычислительных задач. Графические процессоры изначально проектируются таким образом, чтобы обрабатывать огромные объемы данных. На сегодняшний день они способны обеспечить производительность в сотни и даже тысячи миллиардов операций над вещественными числами в секунду (GigaFLoating point Operations Per Second, GFLOPS).

Целью данного доклада является изучение проблемы использования графических

процессоров для решения задач общего назначения, анализ существующих решений, а так же составление ряда алгоритмов позволяющих упросить процесс составления программ для GPU (Graphical Processing Unit).

Литература

[1] А.Ю. Сморкалов, М.Н. Морозов. Поддержка дискретных вейвлет-преобразований для компрессии в системе обработки изображений на потоковых графических процессорах. Материалы всероссийской научно-практической конференции "Информационные технологии в профессиональной деятельности и научной работе". - Йошкар-Ола: МарГТУ, 2010.- С. 91-96.

[2] Боярченков А.С., Поташников С.И. Использование графических процессоров и технологии CUDA для задач молекулярной динамики. Т. 10. 2009.

[3] Евстигнеев Н.М. Интегрирование уравнения Пуассона с использованием графического процессора технологии CUDA // Вычислительные методы и программирование. – 2009. – Т. 10.

[4]Боресков А.В. Иерархия памяти CUDA [Электронный ресурс] / А.В. Боресков, А.А. Харламов // Российское сообщество разработчиков CUDA. – 2009.

[5]Воеводин В. Графический вызов суперкомпьютерам / В. Воеводин, А. Адинец // Открытые системы. – 2008. – № 4. – С. 35 – 41.

[6] Черняк Л. Многоядерные процессоры и грядущая параллельная революция / Л. Черняк // Открытые системы. – 2007. – № 4. – С. 34 – 42.

Е.А. Лаптева
Научный руководитель – доцент, канд. техн. наук А.А. Белов
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23
Тел.: (49234)77272
e-mail: kaf-eivt@yandex.ru

Программирование. Языки программирования. История развития программного обеспечения

При помощи вычислений компьютер способен обрабатывать информацию по определённому алгоритму. Любая задача для компьютера является последовательностью вычислений.

Способность машины к выполнению определённого изменяемого набора инструкций (программы) без необходимости физической переконфигурации является фундаментальной особенностью компьютеров. Дальнейшее развитие эта особенность получила, когда машины приобрели способность динамически управлять процессом выполнения программы. Это позволяет компьютерам самостоятельно изменять порядок выполнения инструкций программы в зависимости от состояния данных. Первую реально работающую программируемую вычислительную машину сконструировал немецкий ученый и изобретатель Конрад Цузе в 1941 году

Программирование — в обычном понимании, это процесс создания компьютерных программ. Программирование позволяет настроить компьютер или иное программируемое логическое устройство на те или иные действия.

Единственный язык, напрямую выполняемый ЭВМ — это *машинный язык (Assembler)*. Изначально все программы писались в машинном коде, но сейчас этого практически уже не делается. Вместо этого программисты пишут текст исходный код на том или ином языке программирования, затем используя компилятор или интерпретатор транслируют его, в один или несколько этапов в машинный код.

Язык программирования ПАСКАЛЬ (PASCAL) получил свое название в честь великого французского математика и физика Блеза Паскаля, который в 1642 г. изобрел счетную машину для арифметических операций. История создания языка начинается с 1965 г., когда Международная федерация по обработке информации предложила нескольким специалистам в области информатики принять участие в разработке нового языка программирования – преемника АЛГОЛа-60. Среди них был швейцарский ученый, работавший в то время доцентом на факультете информатики Стэнфордского университета Никалаус Вирт. В результате этой работы в конце 1968 г. профессор Вирт и его сотрудники из Швейцарского федерального института технологии в Цюрихе разработали первую версию ПАСКАЛЯ, а спустя два года – первый вариант компилятора. В 1971 г. Н. Вирт выпустил описание своего языка. Создавая ПАСКАЛЬ, Н. Вирт преследовал две цели: во-первых, разработать язык, пригодный для обучения программированию как систематической дисциплине; во-вторых, реализация языка должна быть эффективной и надежной на существующих вычислительных машинах.

Си (C++) — стандартизированный процедурный язык программирования, разработанный в начале 1970-х годов сотрудниками Bell Labs Кеном Томпсоном и Денисом Ритчи. Си был создан для использования в операционной системе UNIX. С тех пор он был портирован на многие другие операционные системы и стал одним из самых используемых языков программирования. Си ценят за его эффективность. Он является самым популярным языком для создания системного программного обеспечения. Язык программирования Си отличается минимализмом. Си часто

называют языком *среднего уровня* или даже *низкого уровня*, учитывая то, как близко он работает к реальным устройствам. Однако, в строгой классификации, он является языком высокого уровня.

Basic - это один из самых старых языков программирования. Создателями его были Джон Джорж Кемени и Том Куртз, работавшие в Дортмундском колледже в 1964 году. Новый язык быстро завоевал популярность благодаря своей простоте в изучении, особенно среди начинающих. В середине 80-х годов был реализован QBasic. Это полностью компилируемый язык, с нормальными структурными конструкциями, пользовательскими типами данных. По тем временам это был большой шаг вперед Basic стало возможным использовать наравне с Pascal или C. С появлением Windows и моды на визуальные средства разработки изменился и Basic. Его новая версия, названая Visual Basic, была отлично приспособлена для написания несложных программ с развитым пользовательским интерфейсом.

Java — объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems. Приложения Java обычно компилируются в специальный байт-код, поэтому они могут работать на любой виртуальной Java-машине независимо от компьютерной архитектуры. Дата официального выпуска — 23 мая 1995 года. Java — так называют не только сам язык, но и платформу для создания и исполнения приложений на основе данного языка.

Современные языки программирования старше Интернета, Windows и персонального компьютера минимум на десятилетие. При этом новые языки не переставали регулярно появляться, однако ни один из них не задержался в практике программирования, хотя приносимые ими новые идеи дополняли уже известные языки (как это произошло с объектно-ориентированным программированием). Другой важной особенностью языкотворчества последних десятилетий можно считать прекращение попыток создания “универсального” языка программирования, призванного объединить в себе все последние достижения в области разработки языков. Наконец, появление персонального компьютера и ОС с графическим интерфейсом (прежде всего MacOS и Windows) переместило внимание разработчиков программного обеспечения из сферы языков программирования в другие области средств разработки ПО, такие, как визуальное или объектно-ориентированное программирование, сетевые протоколы или модели баз данных. Программист сегодня использует в качестве инструмента не столько язык, сколько конкретную систему программирования (например, Delphi), а какой язык является для нее базовым, не так уж важно.

Е.А. Лаптева
Научный руководитель – старший преподаватель
кафедры электроники и вычислительной техники
Д.В. Бейлекчи
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23
e-mail: evt@mivlgu.ru
e-mail: kaf-eivt@yandex.ru

Delphi Prism – средство разработки приложений на языке Oxygene для платформы .NET

Платформа .NET решает многие проблемы программирования. К их числу относятся проблемы, связанные с развертыванием приложений, управлением версиями, утечкой памяти, а также проблемы безопасности. Платформа .NET позволяет разрабатывать мощные, независимые от языка программирования, настольные приложения и масштабируемые (расширяемые) Web-службы, построенные на базе новой мощной полнофункциональной библиотеки классов .NET Framework. Платформа .NET содержит общеязыковую среду выполнения (Common Language Runtime - CLR). Общеязыковая среда выполнения CLR поддерживает управляемое выполнение, которое характеризуется рядом преимуществ. Совместно с общей системой типов (Common Type System - CTS) общеязыковая среда выполнения CLR поддерживает возможность взаимодействия языков платформы .NET. [1]

Delphi Prism – средство программирования, обеспечивающее гибкость разработки приложений на Delphi-подобном языке для платформ .NET и приложений для Windows, Linux и Mac OS. Delphi Prism сочетает в себе простой в освоении синтаксис, основанный на языке Object Pascal, и возможности, которые недоступны в других средствах для .NET.

Программирование в данной среде обладает рядом следующих особенностей:

- уменьшение времени разработки с помощью полного решения для .NET. С помощью Delphi Prism можно разрабатывать приложения для платформы .NET. Это решение включает поддержку новых версий Visual Studio, графический интерфейс WinForms, технологию создания веб-приложений и веб-сервисов ASP.NET, графическую (презентационную) подсистему Windows Presentation Foundation (WPF) во время разработки и выполнения, благодаря чему разработчики получают полный доступ ко всем функциональным возможностям языка и среды выполнения .NET для быстрого создания приложений;

- расширение возможностей благодаря мощному полнофункциональному языку разработки. Решение Delphi Prism предоставляет полностью читаемый, усовершенствованный язык для Microsoft .NET Framework. Разработчики могут создавать управляемые приложения с помощью современного языка Oxygene («кислород») – языка программирования, созданного на основе языка Object Pascal и разработанного для стандарта CLI (спецификация общеязыковой инфраструктуры). Наиболее известными реализациями стандарта CLI являются Microsoft .NET Framework и Mono. Спецификация CLI определяет, в частности, архитектуру исполнительной системы .NET - CLR и сервисы, предоставляемые CLR выполняемым программам, классы, синтаксис и мнемонику общего промежуточного языка;

- возможность работы со всеми источниками данных. Delphi Prism предоставляет разработчикам среду доступа к базам данных - dbExpress, которая не зависит от базы данных. Среда dbExpress обеспечивает разработчикам доступ сразу к нескольким серверам баз данных;

- гибкость благодаря поддержке Windows, Mac и Linux. Разработчики могут компилировать код для различных платформ Windows, Linux и Mac. Компилятор Delphi Prism - это собственное приложение .NET, которое может быть перемещено в другие реализации CLR, например, Mono. Delphi Prism будет работать на альтернативных платформах CLR, поддерживая связи между ними. Эта межплатформенная функциональность помогает экономить ресурсы, сохраняя время создания кода. [3]

Первоначально привязанный к Delphi.NET, после приобретения компанией Embarcadero, Oxygene не обладает полной обратной совместимостью с Delphi. Так, например, в его состав не входят библиотеки классов, совместимые с VCL, что практически исключает перенос на платформу .NET унаследованных приложений, созданных с помощью Delphi и использовавших библиотеку классов VCL.

Язык Oxygene обладает следующими преимуществами по сравнению с Object Pascal:

-интеграция с CLI;

- определение переменных непосредственно внутри блока кода;

- обобщённое программирование при помощи шаблонов – вид программирования, заключающийся в таком описании данных и алгоритмов, которое можно применять к различным типам данных, не меняя само это описание;

- анонимные типы – нововведение, позволяющее типам данных инкапсулировать набор свойств в едином объекте без необходимости предварительного явного указания типа;

- улучшенная поддержка событий с несколькими обработчиками (multicast events);

- инициализаторы переменных внутри кода;

- итераторы – объекты, позволяющие программисту перебирать все элементы коллекции без учёта особенностей её реализации; инициализаторы переменных внутри кода. [2]

Delphi Prism представлен в нескольких редакциях:

- Delphi Prism Professional представляет собой среду разработки с подключением к локальной базе данных с помощью dbExpress представляет собой интегрированную среду разработки на основе Microsoft Visual Studio;

- Delphi Prism Enterprise включает функциональность редакции Professional и средства для создания многоуровневой базы данных, а также средства создания web-приложений.

Обе редакции поддерживают процессорные архитектуры x86 и x64.

Таким образом, Delphi Prism является универсальной средой разработки приложений для платформы .NET, поддерживающей современный Delphi-подобный язык программирования Oxygene. К тому же Delpi Prism поддерживает наиболее распространенные операционные системы и компьютерные архитектуры.

Литература

1. Основы технологии .NET: [Электронный ресурс]. Доступ: http://cpu.h17.ru/net/2/

2. Embarcadero Prism XE2. Разработка для .NET и Mono: [Электронный ресурс]. Доступ: http://www.embarcadero.com/products/prism

3. Особенности работы с .NET: [Электронный ресурс]. Доступ: http://www.webdelphi.ru/2011/06/delphi-prism-net-framework-4/

А.А. Моряков
Научный руководитель –
заведующий кафедрой электроники и вычислительной техники,
профессор, канд. техн. наук Ю.А. Кропотов
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23
Тел.:(49234) 7-72-73
e-mail: evt@mivlgu.ru
e-mail: kaf-eivt@yandex.ru

Облачная операционная система

В настоящее время всё больше развиваются облачная операционная система (Internet ОС) - это клиент-серверное программное обеспечение, функционирующего обычно в среде современного Веб - браузера. тетевая «облачная» технология – большая часть информации пользователя хранится на удалённых серверах, доступ пользователя к данным осуществляется по средствам глобальных и локальных сетей. Все данные для работы пользователь может получить, используя глобальную компьютерную сеть Internet. Пользователь может работать со своими данными, на каком либо электронно-цифровом устройстве, обеспечивающем доступ в Internet через специальный Веб - браузера который может быть установлен на устройстве, даже как основная система. Также возможно Веб - браузер хранить на флэш-накопителе, и загружаться с него на любом другом устройстве. Возможности у Internet ОС сравнимы с обычными операционными системами: например пакеты видеомонтажа, текстовые редакторы, научные пакеты приложений, возможно использовать все сервисы которые представляет та или иная ОС. Internet ОС постоянно сохраняет ваши данные, которые меняются в реальном времени, поэтому потери нужных данных минимальны, сами серверы хранят, данные пользователей в нескольких экземплярах. Такой вид операционных систем в некоторой степени обеспечивает наибольшее быстродействие всей системы, ограничивающееся лишь скорость соединения с Internet. Веб приложения на Internet ОС имеют социальную функциональность. Это означает, что за одни приложением могут работать несколько пользователей, пример текстовый процессор - с текстом могут работать одновременно несколько человек, это может увеличить эффективность разработки каких либо проектов. Такие Веб-системы имеют все возможности традиционных операционных систем: работа с файловыми системами, интеграция с периферийными устройствами. Таким образом облачные операционные системы обеспечивают полную интеграцию с аппаратным средствами пользователей.

Традиционными операционными системами можно пользоваться в частых случаях на одном устройстве, в случае с Internet ОС все ваши данные, настройки пользователя, персонализация хранится на одном удалённом устройстве доступ, к которым можно получить с любого устройства из любой точки земного шара.

Вариант аппаратной части пользователя и облачной операционной системы, должен выглядеть следующим образом. Устройство обеспечивающее доступ к удалённому серверу, должно иметь компактные размеры, оно должно представлять что-то вроде планшета, аппаратная часть должна соответствовать современным требованиям: способность обрабатывать в реальном времени современные Web технологии – JavaScript, Flash, HTML5, обработка 3D графики, широкополосный доступ в интернет или VPN соединение, некоторые современные порты ввода/вывода, hdmi, usb, и т.д. для обеспечения подключения дополнительных устройств. Для обеспечения безопасности на устройство, нужно обеспечить невозможность установки дополнительного программного обеспечения без цифровой подписи или если оно не пройдёт проверку на удалённой операционной системе. Нужно обеспечить совместную обработку информации устройством и облачной операционной системы, для наибольшего быстродействия.

Литература

1. Максимов Н.В., Партыка Т.Л., Попов И.И. Архитектура ЭВМ и вычислительных систем; Учебник. – М.: ФОРУМ: ИНФРА-М, 2006.
2. http://ru.wikipedia.org/wiki/облачные_вычисления

А.А. Моряков
Научный руководитель – доцент, канд. техн. наук Е.П. Догадина
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23
Тел.:(49234) 7-72-73
e-mail: kaf-eivt@yandex.ru

Реализация алгоритма игры «Жизнь» как модели биологической эволюции

В работе исследовано и разработано одно из средств моделирования в ряде поколений на основе алгоритма игры «жизнь». Игра́ «Жизнь» - клеточный автомат, который является моделью биологической эволюции. Место действия этой игры - «вселенная» - которая представляет собой поверхность или плоскость, размеченная на клетки. Каждая клетка на этой поверхности может находиться в двух состояниях: быть "живой" или быть "мёртвой" (пустой). Клетка имеет восемь соседей (окружающих клеток).

Распределение живых клеток в начале игры называется первым поколением. Каждое следующее поколение рассчитывается на основе предыдущего по таким правилам:

Правило рождения: пустая (мёртвая) клетка, рядом с которой ровно три живые клетки, оживает;

Правило выживания: если у живой клетки есть две или три живые соседки, то эта клетка продолжает жить;

Правило гибели: если соседей меньше двух или больше трёх клетка умирает от «одиночества» или от «перенаселённости».

Игра прекращается, если на поле не останется ни одной "живой" клетки, или если при очередном шаге ни одна из клеток не меняет своего состояния (складывается стабильная конфигурация).

Эти простые правила приводят к огромному разнообразию форм, которые могут возникнуть в игре.

Рис.1. Модели популяций

В работу добавлен новый вид клеток, «хищники» которые поедают, зелёные клетки «жертв», и живут по следующим правилам:

Правило рождения: если клетка «хищник» «поедает» зелёную клетку «жертву», то она даёт новое «потомство» - клетку красного цвета, а сама «стареет» и превращается в клетку бордового цвета.

Правило жизни: для жизни клетки «хищники» должны «питаться», если клетка «поела» то она даёт потомство и превращается в «старую» клетку и может жить 100мс, «родившиеся» клетки живут 500мс.

Правило гибели: если клетка не «питается» то она погибает через 100мс или через 500мс, в зависимости от того «старая» это клетка или «молодая».

Клетки «хищники» генерируются случайно в малом количестве, клетки жертвы,

образуются случайно в большем количестве, также рассмотрен вопрос принудительного заполнения плоскости особями «жертв» с учётом мнения пользователя.

Для наглядного поведения клеток, была разработана программа на языке программирования Object Pascal, без которой сложно понять процесс «рождения» и «гибели» клеток.

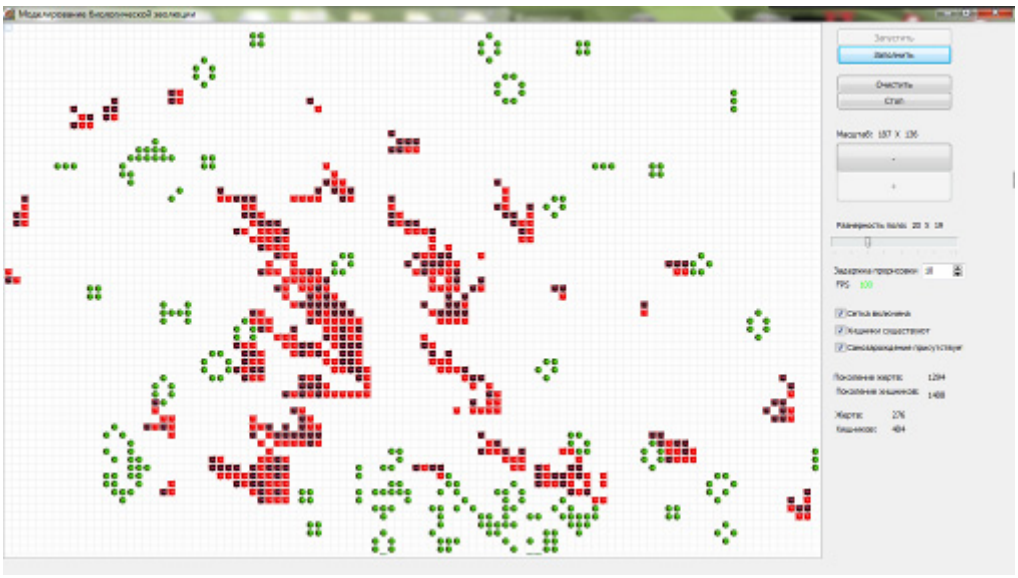

Рис.2. Моделирование биологической эволюции

В программе присутствует функции масштабирования поверхности с клетками, регулирования генерируемой площади особей, регулировка задержки изменения состояния клеток, отключение сетки, учитывается количество поколений и конкретное количество хищников и жертв. Существует возможность произвольного зарождения клеток «жертв», выбор развития «жизни» как с клетками «хищниками» так и без них.

Замечена тенденция образования клетками «хищников» групп клеток.

Таким образом, можно изучить поведение простейших организмов двух видов в одной среде обитания. В процессе жизни «хищников» присутствует элементы соперничества, то есть вероятность выживания клеток которые находятся в нутрии скоплений меньше, чем вероятность клеток, которые находятся по краям скоплений.

Литература

1.Уэзерелл Ч. «Этюды для программистов» 1982.

2. Книга Т. Тоффоли и Н. Марглоуса «Машины клеточных автоматов» 1987.

А.А. Осокин
Научный руководитель – старший преподаватель
кафедры электроники и вычислительной техники
Д.В. Бейлекчи
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23
e-mail: evt@mivlgu.ru
e-mail: kaf-eivt@yandex.ru

Разработка графического интерфейса пользователя в Delphi XE2 2011 при помощи встроенной библиотеки компонент FireMonkey

FireMonkey – это новая прикладная платформа для создания интерфейса визуальных приложений в Delphi и C++ Builder. FireMonkey использует собственные графические библиотеки для того, что бы абстрагироваться от основной ОС. Она обеспечивает построение интерфейса пользователя с графикой высокого качества HD (HighDefinition) и 3D-графикой, стили интерфейса, поддерживает значительный набор графических элементов интерфейса, а также новую модель привязки данных для бизнес-приложений.

FireMonkey как платформа для разработки приложений основана на модели, которая одновременно предлагает абстрактные интерфейсы пользователя и «нативные» решения:

- «абстрактная» прикладная платформа. Ранее библиотеки пользовательского интерфейса, обрабатывали соответствующие элементы операционной системы. В VCL Delphi, например, "TButton" компонент был оболочкой Windows элемента управления Button (кнопка). Вместо этого, в FireMonkey появилось абстрактное понятие кнопки, к которой могут применяться различные стили, для того чтобы она выглядела обычной встроенной кнопкой под различными платформами, или использовала полностью настраиваемый стиль пользовательского интерфейса.

- «нативная» платформа. В отличие от других библиотек, которые абстрагируют пользовательский интерфейс, FireMonkey привязана непосредственно к встроенной графической библиотеке, такой как OpenGL или DirectX, предлагая наилучшее решение с точки зрения использования графического процессора(GPU) на целевом компьютере. Кроме того, код приложения будет являться непосредственно машинным кодом используемого центрального процессора. Это означает, что есть возможность использовать встроенный программный интерфейс операционной системы, на данной платформе. Также нет необходимости разворачивать или устанавливать дополнительные библиотеки на целевом компьютере, так как библиотека обеспечивает построение отдельного исполняемого файла, который включает в себя как код разработчика, так и все необходимые библиотеки. По сравнению с большинством других аналогичных решений, такой подход во многом упрощает развертывание приложения.

Delphi был первым инструментом, представившим RAD-модель («быстрая разработка приложений»), основанную на собственных компонентах с внутренней объектно-ориентированной архитектурой. FireMonkey расширяет эту модель, позволяя разработчикам очень быстро создавать приложения, контролировать изменения кода, использовать визуальный дизайнер, подключаться к базам данных и другим источникам данных.

Так как библиотеки Delphi содержат DBExpress, котрые реализуют независимый от конкретной БД механизм доступа к данным, и архитектуру DataSnap для многозвенного доступа, визуальные элементы FireMonkey управления могут также использовать эти технологии. Кроме того элементы интерфейса FireMonkey могут использовать новую архитектуру LiveBindings для отображения данных из различных источников, в том числе таблиц базы данных.

Для построения приложений с красивым и современным интерфейсом, нужно иметь возможность расширять пользовательский интерфейс и выходить за рамки ограничений стандартного интерфейса пользователя ОС. FireMonkey обеспечивает не только HD-качество, но и полную поддержку 3D для пользовательских интерфейсов. Также существует возможность объединения обеих моделей (HD и 3D) в рамках одного приложения.

С учетом объектно-ориентированной и компонентной архитектуры FireMonkey позволяет легко настраивать компоненты, а так же позволяет сторонним производителям создавать пользовательские компоненты для расширения библиотеки. В отличие от большинства других решений, вместе с двоичными файлами библиотеки предоставляется исходный код библиотеки, для изучения внутренние приемов программирования и отладки приложений, используя трассировку в библиотеке классов.

Библиотека FireMonkey использует существующие библиотеки различных платформ для графической обработки:

- для HD-приложений под Windows используется Direct2D (или GDI + в случае отсутствия Direct2D);
- для 3D-приложений под Windows используется Direct3D;
- для HD-приложений под Mac OS используется Quartz;
- для 3D-приложений под Mac OS используется OpenGL.

Графический процессор используется для отображения 3D-объектов, элементов управления и фильтров/эффектов.

FireMonkey может быть использована в самых разных приложениях. Данная библиотека, наряду с существующими в Delphi RTL- и VCL-компонентами, может быть использована для создания:

- автономных приложений и утилит с расширенными графическими возможностями, которые можно разворачивать на нескольких платформах;
- высококачественных 3D графических приложений, оптимизированных под GPU;
- клиент / серверных бизнес-приложений с развитым пользовательским интерфейсом.

Таким образом, FireMonkey предлагает максимальное использование центрального и графического процессоров для реализации интерфейса в действительно кроссплатформенной и открытой среде, с компилируемыми «родными» приложениями, без необходимости установки виртуальных машин, трансляторов или сред выполнения на компьютерах пользователей и прочих устройствах. В настоящий момент она уже работает под Windows, Mac OS, частично поддерживает iOS на iPad/iPhone и, возможно, довольно скоро заработает на других операционных системах.

Литература

1. Описание RAD Studio XE2: [Электронный ресурс]. Доступ: http://delphi2010.ru
2. Форум программистов. Обсуждение библиотеки FireMonkey: [Электронный ресурс]. Доступ: http://delphixe.com.ua/articles/14-cantu1
3. Описание библиотеки FireMonkey 2011: [Электронный ресурс]. Доступ: http://ru.wikipedia.org/wiki/FireMonkey

К.А. Панина
Научный руководитель – доцент, канд. техн. наук А.А. Белов
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23
e-mail: evt@mivlgu.ru
e-mail: kaf-eivt@yandex.ru

Языки программирования

Язык программирования – формальная знаковая система, предназначенная для записи программ. Программа обычно представляет собой некоторый алгоритм в форме, понятной для исполнителя. Язык программирования определяет набор лексических, синтаксических и семантических правил, используемых при составлении компьютерной программы.

Язык программирования предназначен для написания компьютерных программ, которые применяются для передачи компьютеру инструкций по выполнению того или иного вычислительного процесса и организации управления отдельными устройствами.

Языки программирования могут быть разделены на компилируемые и интерпретируемые.

Программа на компилируемом языке при помощи специальной программы компилятора преобразуется в набор инструкций для данного типа процессора и далее записывается в исполняемый файл, который может быть запущен на выполнение как отдельная программа.

Интерпретируемые языки обладают некоторыми специфическими дополнительными возможностями, кроме того, программы на них можно запускать сразу же после изменения, что облегчает разработку.

Системы объектно-ориентированного программирования содержат программу-транслятор и позволяют работать в режиме как интерпретатора, так и компилятора. На этапе разработки и отладки проекта используется режим интерпретатора, а для получения готовой программы – режим компилятора.

Всю историю компьютерной индустрии и компьютерных наук с определенной точки зрения можно представить как историю развития языков программирования. Меняются времена, усложняются задачи, то, что раньше требовало человеко-лет, нынче энтузиасты делают на коленке за несколько недель; накоплена огромная масса типовых решений, типовых библиотек и типовых программистов. А создание, развитие и изменение языков программирования идет полным ходом.

Программы на машинном языке – очень длинные последовательности единиц и нулей, являлись машинно-зависимыми, т.е. для каждой ЭВМ необходимо было составлять свою программу.

Язык Ассемблера (начало 50-ых годов XX в.) – это символическое представление машинного языка. Он облегчает процесс программирования по сравнению с программированием в машинных кодах.

Программисту не обязательно употреблять настоящие адреса ячеек памяти с размещенными в них данными, участвующими в операции, и вычисляемые результаты, а также адреса тех команд, к которым программа не обращается.

Некоторые задачи, например, обмен с нестандартными устройствами обработки данных сложных структур невозможно решить с помощью языков программирования высокого уровня. Это под силу ассемблеру.

Первые языки программирования высокого уровня

С середины 50-ых гг. XX в. начали создавать первые языки программирования

высокого уровня (high-level language). Эти языки были машинно независимыми (не привязаны к опред. типу ЭВМ). Но для каждого языка были разработаны собственные компиляторы.

Примеры таких языков: FORTRAN (FORmula TRANslator; 1954) предназначен для научных и технических расчетов; COBOL (1959) был предназначен в основном для коммерческих приложений (обрабатывал большие объемы нечисловых данных) – Common Business-Oriented Language); язык BASIC (Beginner's All Purpose Instuction Code – универсальный язык символьных инструкций для начинающих) (1964 г.)

Бейсик – это продукт Новой Англии. Созданный в 1964г., как язык обучения программированию. Бейсик является общепринятым акронимом от"Beginner's All-purpose Symbolic Insruction Code" (BASIC) - Многоцелевой Символический Обучающий Код для Начинающих".

Вскоре как обучаемые, так и авторы программ обнаружили, что Бейсик может делать практически все то, что делает скучный неуклюжий Фортран. А так как Бейсику было легко обучиться и легко с ним работать, программы на нем писались обычно быстрее, чем на Фортране. Бейсик был также доступен на персональных компьютерах, обычно он встроен в ПЗУ. Так Бейсик завоевал популярность. Интересно, что спустя 20 лет после изобретения Бейсика, он и сегодня самый простой для освоения из десятков языков общецелевого программирования, имеющихся в распоряжении любителей программирования. Более того, он прекрасно справляется с работой.

С начала 80-ых г. XX в. начали создаваться языки программирования, которые позволили перейти к структурному программированию (использование операторов ветвления, выбора, цикла и практически отказ от частого использования операторов перехода (goto). К этим языкам относятся: язык Pascal (назван его создателем Никлаусом Виртом в честь великого физика Блеза Паскаля; 1970); язык Си, позволяющий быстро и эффективно создавать программный код (1971)

Язык программирования Pascal был создан Никлаусом Виртом, и назван в честь французского философа и математика XVIIв. Блеза Паскаля.

Языки объектно-ориентированного программирования (90-ые г. XX в.).

В основу этих языков положены программные объекты, которые объединяют данные и методы их обработки. В этих языках сохранялся алгоритмический стиль программирования. Для них были разработаны интегрированные среды программирования, позволяющие визуально конструировать графический интерфейс приложений: язык C++ (1983) – продолжение алгоритм. языка Си, язык Object Pascal (1989) был создан на основе языка Pascal. После создания среды программирования – Delphi (1995), язык Visual Basic(1991) был создан корпорацией Microsoft на основе языка Qbasic (1975) для разработки приложений с графическим интерфейсом в среде ОС Windows.

Языки программирования на платформе .NET.

Интегрированная среда программирования Visual Studio .Net, разработанная корпорацией Microsoft, позволяет создавать приложения на различных языках объектно-ориентированного программирования, в том числе: на языке Visual Basic .Net (на основе Visual Basic) - 2003 г.; на языке Visual C# (C-шарп) – на основе языков C++ и J – 2003 г.;на языке Visual J# (J-шарп) – на основе Java и JavaScript – 2003 г..

К. А. Панина,
А.А. Вирабян
Научный руководитель – старший преподаватель
кафедры электроники и вычислительной техники
Д.В. Бейлекчи
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23
e-mail: evt@mivlgu.ru
e-mail: kaf-eivt@yandex.ru

Исследование возможностей кроссплатформенной среды разработки MonoDevelop для платформы .NET

В данном докладе рассматриваются результаты исследования возможностей кроссплатформенной среды разработки MonoDevelop для платформы .NET.

MonoDevelop – свободная среда разработки, предназначенная для создания приложений C#, Java, Visual Basic .NET, CIL, C и C++. В 2012 также планируется поддержка языка Oxygene со стороны Embarcadero Technologies.

Изначально MonoDevelop был портом SharpDevelop на Mono/GTK+, но с того времени проект далеко ушёл от своего начального состояния. [1]

MonoDevelop преставляет собой интегри́рованную среду разработки (IDE) – систему программных средств, используемую программистами для разработки программного обеспечения (ПО), в первую очередь предназначеную для разработки на языке C# и других .NET языков. MonoDevelop позволяет разработчикам быстро создавать Web-приложения ASP.NET Web в Linux, Windows и Mac OS. MonoDevelop упрощает для разработчиков перенос .NET приложений, созданных с Visual Studio для Linux и поддерживает единый базовый код для всех платформ. [2]

Среда содердит мультиплатформенные библиотеки поддержки Linux, Windows и Mac OS и библиотеку GTK# (сокращение от GIMP ToolKit) – кроссплатформенную библиотеку элементов интерфейса, которая наряду с Qt является одной из двух наиболее популярных на сегодняшний день библиотек для X Window System [2]

В этой среде есть такие возможности, как подсветка синтаксиса, сворачивание кода, автодополнение кода, браузер классов, полностью настраиваемые схемы расположения окон, поддержка плагинов, встроенный отладчик, визуальный конструктор форм для GTK#, модульное тестирование, инструменты языковой локализации.

Для данной среды разработан специальный набор библиотек MonoTouch, позволяющий разрабатывать приложения для iPhone и iPod Touch на языке C#. [2]

Последнее дополнение среды MonoDevelop – это проект MonoMac, новый фреймворк, который позволяет строить приложения для Mac OSX с помощью Mono и поддерживает использование Xcode 4. Xcode – это интегрированная среда проектирования Apple на Mac OS для создания приложений для Mac, iPhone и iPad. Xcode включает инструмент анализа производительности для приложений MacOS и эмулятор iOS для тестирования приложений iOS [2]. То есть, с новой версией MonoDevelop теперь можно создавать приложения, использующие Apple iOS. Apple iOS (до 2010 года известная как iPhone OS) – мобильная операционная система, разработанная американской компанией Apple на основе Mac OS X первоначально для iPhone, а затем расширена для поддержки таких мобильных устройств, как Apple iPod

Touch, iPad и Apple TV.

Таким образом, среда разработки MonoDevelop удобна для разработки кроссплатформенных приложений, так как программы, разработанные для ОС Windows можно исполнять на Linux без поправок, а также достаточно просто переносить приложения на ОС MacOS для Mac и iOS для iPhone и iPod.

Литература

1. Сайт разработчиков MonoDevelop: [Электронный ресурс]. Доступ: http://monodevelop.com/

2. Описание MonoDevelop. Википедия: [Электронный ресурс]. Доступ: http://ru.wikipedia.org

3. Среда разработки Xcode: [Электронный ресурс]. Доступ: https://developer.apple.com/xcode/

С.В. Савинов
Научный руководитель – доцент, канд. техн. наук А.А. Белов
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23
e-mail: kaf-eivt@yandex.ru

Общие сведения о квантовой криптографии

Квантовая криптография представляет собой способ обеспечения защиты коммуникационных каналов, базирующийся на основных закономерностях квантовой физики. Если классический подход криптографической системы заключается в использовании математических способов для реализации защищенности информации, путём кодирования, то система квантовой криптографии базируется на основах квантовой физики, реализуя процесс передачи данных путем использования объектов квантовой физики. Для осуществления передачи информации на физическом уровне используются либо электроны, движущиеся в электрическом поле, либо фотоны, передающиеся с помощью коммуникационных каналов волоконно-оптической связи [1]. Под процессом перехвата информации в канале передачи подразумевается измерение свойств (параметров) объектов являющихся переносчиками данных. Принцип функционирования системы квантовой криптографии базируется на свойстве неопределённости поведения системы, а именно – отсутствие шанса единовременно определить координаты и импульс объекта; нет возможности произвести измерение одного параметра объекта передачи информации – фотона, не производя искажение другого параметра фотона. Данное утверждение является принципом неопределённости, которое было сформулировано в 1927 году Гейзенбергом [2].

С помощью свойств квантовой системы возможно можно реализовать систему передачи информации, в которой постоянно будет иметься возможность обнаружения процесса перехвата данных на любой стадии передачи. Это реализуется за счет того, что любая попытка произвести анализ связанных между собой параметров системы ведет к созданию нарушений в системе, а это в свою очередь приводит к искажению или разрушению исходных сигналов. Затем пользователи, передающие данные смогут по значению уровня шума (искажений) в коммуникационном канале, обнаружить перехват информации [3].

Стефан Вейснер в 1970 году сформулировал идею обеспечения защиты данных по средствам объектов квантовой физики – фотонов. Спустя десть лет инженеры фирмы IBM предложили произвести передачу секретного ключа с помощью фотонов. В 1989 году была реализована первая в мире квантово-криптографическая система, она была спроектирована в научно-исследовательском центре IBM инженерами Беннетом и Бассардом [4]. Реализованная схема представляла собой канал для передачи фотонов, на концах которого располагались передающий и принимающий аппараты. Обеспечение сохранности тайны информации передаваемой по каналу связи находится в зависимости от интенсивности вспышек света, с помощью которых осуществляется передача данных. Слабая интенсивность вспышек ведёт одновременно как и к обеспечению трудности перехвата сигнала, так и к увеличению количества ошибок у пользователя при проведении анализа поступивших по коммуникационному каналу данных. Высокая интенсивность вспышек повышает риск перехвата сигналов путём раздвоения начального пучка света на две составляющие, первая из которых направляется по исходному пути, а вторую анализирует перехватчик сигналов. Даже если произойдет перехват сигнала, описанным выше способом, пользователи,

производя анализ параметров генерации сигналов (вспышек света) и объём принятых и обработанных ошибок могут произвести оценку типа и количества перехваченной информации [5].

Почти все квантово-оптические системы квантовой криптографии являются сложными в процессе управления, кроме того возникает необходимость подстройки параметров передачи информации. Сейчас активно ведутся работы по усовершенствованию системы подключения к квантово-оптическому каналу, которую можно будет использовать по принципу «plug and play». Данная система основывается на применении зеркала Фарадея, с помощью которого удается избежать двойного лучепреломления, что приводит к отказу от периодической регулировки поляризации. Данный способ дает возможность передавать криптографические ключи по стандартным телекоммуникационным каналам передачи информации [6,7].

В настоящее время квантовые криптографические системы приближаются к повсеместному использованию. В качестве разработчиков технологий квантовой криптографии включает как крупные институты, так и небольшие компании.

Квантовая криптография является перспективной составной частью криптографии, благодаря передовым технологиям появляется возможность установления безопасности передаваемых данных на самый высокий уровень.

Литература

1. Ловцев Д.А. Защита информации // Информатика и образование, 2004;
2. Никитин Ф. Глобальная сеть нуждается в глобальной защите // Мир связи и информации Connect!, 2002.
3. Галатенко В. Информационная безопасность // Открытые системы, 2006.
4. Килин С. Я. «Квантовая информация / Успехи Физических Наук.» — 1999. — Т. 169. — С. 507—527
5. Галатенко В. Информационная безопасность - обзор основных положений // Открытые системы, 2006.
6. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования ГОСТ 28147–89, М., Госстандарт, 1989.
7. Воробьев, "Защита информации в персональных ЭВМ", изд. Мир, 1993 г.

С.В. Савинов
Научный руководитель – доцент, канд. техн. наук В.А. Ермолаев
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23
Тел.: (49234) 77-2-73
e-mail:electron@mivlgu.ru

Искусственные нейронные сети: вопросы применения и реализации

Искусственные нейронные сети представляют собой совокупность элементов, называемых искусственными нейронами, объединенных в единое целое множеством настраиваемых соединений. По своей архитектуре и принципам функционирования искусственные нейронные сети имеют сходство с биологическими нейронными сетями, откуда, собственно, и произошло их название [1, 2]. Однако полной аналогии между искусственными и биологическими нейронными сетями не существует [2]. Во-первых, по числу нейронов, которое в биологических сетях достигает 100 миллиардов. Во-вторых, по функциональным характеристикам отдельного нейрона. В биологических сетях каждая нервная клетка – это живой организм, способный к самоорганизации, саморегулированию, энергообеспечению и самообучению.

Количество нейронов, а значит и количество соединений в искусственных нейронных сетях неизмеримо меньше. Сами же нейроны являются простыми устройствами, выполняющими ограниченный набор элементарных функций. Статические нейроны – это элементы, обеспечивающие суммирование и безынерционное нелинейное преобразования взвешенной суммы входных сигналов. Динамические нейроны описываются либо системой дифференциальных уравнений невысокого порядка, либо, в дискретном случае, разностными уравнениями. Здесь следует отметить, что исследования в области искусственных нейронных сетей преследуют две различные цели [2]. Первая цель, чисто научная, – разобраться в механизмах функционирования нервной системы живых существ. Вторая, прагматическая, – разработать некоторое техническое устройство. При этом искусственная нейронная сеть может совершенно не походить на биологическую сеть.

В рамках построения технического устройства возникает вопрос об обоснованности применения искусственной нейронной сети. Возможно, ответ на него заключен в главной особенности нейронных сетей, отличающей их от обычных средств, – это в обучающем принципе формирования законов их функционирования, не требующем разработки специальных алгоритмов. Это указывает на обоснованность применения искусственных нейронных сетей, по крайней мере, в задачах с трудно формализуемыми условиями функционирования. Естественно, что применение искусственных нейронных сетей предполагает решение вопросов выбора типа сети, ее архитектуры, числа слоев, алгоритмов обучения, типа нейронов и функций активации.

Все указанные вопросы должны решаться с учетом имеющейся элементной базы, с учетом обеспечиваемой ею степенью параллелизма обработки сигналов. На ранних стадиях развития техники нейронного моделирования использовались специальные микросхемы, имитирующие работу нейронов. В этом плане сегодня представляют интерес программируемые логические микросхемы, которые могут состоять из нескольких миллионов типовых компонентов, в том числе и аналоговых. При цифровой реализации использовались [2, 4] сети, состоящие из транспьютеров – универсальных процессоров с упрощенной системой команд, объединяемых в единое целое через коммуникационные порты, а также систолические матрицы – матрицы из процессоров с однонаправленной передачей информации.

При реализации искусственных нейронных сетей могут использоваться и цифровые сигнальные процессоры (ЦСП) общего назначения. К ним, в частности, относятся микросхемы платформы TMS 320C6XXX с 2,4 миллиардами операций в секунду. Эта скорость обеспечивается за счет одновременного выполнения 8 арифметических операций.

При программном моделировании нейронных сетей могут использоваться ускорительные платы. Данные платы, функционируя параллельно с процессором компьютера, забирают на себя основную вычислительную нагрузку, при этом процессор компьютера выполняет функции управления и обслуживания. Например, в НТЦ «Модуль» были созданы многопроцессорные ускорительные платы МЦ5.001 и МЦ5.002. МЦ5.001, содержащие 4 ЦСП типа TMS320C40 с тактовой частотой 50 МГц и пиковой производительностью 275 MIPS. ЦСП включает в себя локальную статическую память объемом 1 Мбайт. К двум микропроцессорам дополнительно подключены два блока динамической памяти объемом 16 Мбайт каждый. К одному из процессоров подключена также статическая память объемом 1 Мбайт. Ускорительная плата МЦ5.002 содержит 6 ЦСП типа TMS320C40. Коммуникационные порты ЦСП имеют пропускную способность до 20 Мбайт/с.

Первой отечественной микросхемой, позволяющей использовать ее при построении искусственных нейронных сетей, явился ЦСП NM 6403. Наличие у этих микросхем двух шин данных, каждая по 88 разрядов, обеспечивает возможность их объединения в матричную структуру и, соответственно, реализацию сложных систем параллельных вычислений, характерных для нейронных сетей.

Преимущества таких систем по сравнению с классическими компьютерными системами заключаются в быстродействии (за счет параллелизма вычислений) и повышенной надежности (при наличии соответствующих алгоритмов самообучения и самоорганизации).

Нейронный процессор обычно состоит из двух основных блоков: скалярного, выполняющего роль универсального вычислительного устройства, и векторного, ориентированного на выполнение векторно-матричных операций. Скалярное устройство обеспечивает интерфейсы с памятью и коммуникационными портами, позволяющими объединять процессоры в вычислительные сети различной конфигурации. Основное назначение скалярного устройства – подготовка данных для векторной части процессора.

Центральным звеном процессора является целочисленное векторное устройство, обладающее возможностями обработки данных различной разрядности. Оно оперирует n-разрядными словами. Таким образом, процессор рассчитан на высокопроизводительную обработку больших массивов целочисленных данных.

Нейронные сети находят применение при решении различных задач. Весьма часто они используются в задачах классификации и распознавания образов [5]. Нейронные системы позволяют классифицировать символы машинописного и рукописного текста, изображения и фрагменты звуковых сигналов. Число нейронов в выходном слое сети при решении этих задач обычно равняется числу классов. При этом каждому классу соответствует свой выход сети.

В задачах кластеризации и векторного кодирования обучение с помощью нейронных сетей осуществляется во многих случаях без учителя, что обеспечивает разбиение множества входных образов на классы при отсутствии априорной информации о них. К этому классу относятся задачи сжатия данных и построения ассоциативной памяти, выбор информации из которой осуществляется по содержанию, а не по адресу.

Задачи управления образуют еще одну значительную область применения

искусственных нейронных сетей [3]. Весьма широко их применение в роботизированных системах и во всевозможных моделях роботов. С помощью нейронных сетей решаются задачи управления объектами с неизвестными параметрами и решаются задачи идентификации систем. При управлении используются прямые и инверсные модели управляемых объектов, идентифицируемые в процессе их функционирования.

Еще одну область применения нейронных сетей образуют задачи аппроксимации и предсказания данных и временных рядов, например, природных процессов изменения погодных условий или загрязнения рек. С помощью нейронных сетей можно аппроксимировать многомодальную функцию плотности вероятностей наблюдаемых данных [5] и предсказать потребление какого-либо продукта.

Литература

1. Хайкин С. Нейронные сети: полный курс. – М.: Издательский дом «Вильямс», 2006. – 1104 с.

2. Кохонен Т. Самоорганизующиеся карты. – М.: Бином. Лаборатория знаний, 2008. – 655 с.

3. Терехов В.А., Ефимов Д.В., Тюкин И.Ю. Нейросетевые системы управления. М.: ИПРЖР, 2002. – 480с;

4. Рао С.К., Кайлат Регулярные итеративные алгоритмы и их реализация в процессорных матрицах // ТИИЭР, т.76, № 3, 1988, с. 58 – 69;

5. Bishop C.M. Pattern recording and machine learning. – New York: Springer, 2006. – 738 p.

Н.В. Семёнова
Научный руководитель – доцент, канд. техн. наук А.А. Быков
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23
e-mail: kaf-eivt@yandex.ru

Обзор семейства стандартов методологии моделирования сложных систем IDEF

Методология IDEF может использоваться не только при планировании бизнес-процессов, но и при проектировании сложных программных средств.

IDEF0-методология предназначена для функционального моделирования, согласно которому система представляется как совокупность взаимодействующих процессов. Такая чисто функциональная ориентация является принципиальной, функции системы анализируются независимо от объектов, которыми они оперируют. Это позволяет более четко смоделировать логику и взаимодействие процессов в программе.

Описание модели IDEF организовывается в виде иерархии упорядоченных и взаимосвязанных диаграмм, что отображает функциональную структуру объекта. Процесс моделирования начинается с представления работы как единого целого – одного функционального блока с интерфейсными дугами, эта диаграмма называется контекстной.

В настоящее время в целях проектирования сложных систем используются следующие частные частных методологии IDEF [1]:

- IDEF0 - методология функционального моделирования используется для создания функциональной модели, с помощью наглядного графического языка IDEF0 отображающая структуру, процессы и функции системы, в виде набора взаимосвязанных функций (функциональных блоков), а также потоки информации и материальных объектов, преобразуемые этими функциями. Как правило, моделирование средствами IDEF0 является первым этапом изучения любой системы;
- IDEF1X (IDEF1 Extended) - методология построения реляционных структур. IDEF1X относится к типу методологий «Сущность-взаимосвязь» (ER – Entity-Relationship) и, как правило, используется для моделирования реляционных баз данных, имеющих отношение к рассматриваемой системе;
- IDEF3 - методология моделирования процессов, происходящих в системе, предназначенная для создания сценариев и описания последовательности операций для каждого процесса. IDEF3 напрямую связана с методологией IDEF0: каждая функция (функциональный блок) может быть представлена средствами IDEF3 в виде отдельного процесса.

Кроме упомянутых методологий, традиционно используемых при проектировании, существует еще ряд методологий, которые так же рекомендуется использовать в задачах проектирования:

- IDEF4 - методология объектно-ориентированного проектирования и анализа систем. IDEF4 используется при создании модульного объектно-ориентированного кода, который можно повторно использовать, а также анализировать и оптимизировать. IDEF4 модель имеет две подмодели: классов и методов. При этом методология предоставляет пользователю графический язык для изображения классов, диаграмм наследования, таксономии методов.
- IDEF5 - методология определения онтологий (словарей) исследования сложных систем. С помощью словаря терминов и правил позволяет описать онтологию системы. В итоге могут быть сформированы достоверные утверждения о состоянии системы в некоторый момент времени, на основе которых делаются выводы о дальнейшем развитии системы и производится её оптимизация. Как схематический язык, IDEF5 ближе всего к IDEF1 и IDEF1X. Информация, отражающаяся в модели IDEF1 или IDEF1X, может быть выражена в IDEF5. Но для проектирования реляционных баз данных IDEF5 не подходит, так как не содержит хорошо продуманных, специализированных представлений IDEF1/1X.

- IDEF9 - методологии моделирования требований. Этот метод исследования бизнес ограничений был разработан для облегчения обнаружения и анализа ограничений в условиях которых должно действовать программное средство. Обычно, при построении моделей описанию ограничений, оказывающих влияние на протекание процессов уделяется недостаточное внимание. Это приводит к тому, что реализация построенных моделей столкнется с непредвиденными трудностями, в результате чего их потенциал будет не реализован.

Обычно в качестве систем фигурируют сложные информационные системы с ориентацией на экономические и управленческие приложения. Под ограничением понимается отношение, которое должно соблюдаться. Ограничения делятся на контексты (группы родственных ограничений). Применение IDEF9 заключается в выполнении нескольких шагов:

1) сбор свидетельств (фактов, указывающих на наличие ограничения);
2) классификация — определение контекстов, объектов, отношений;
3) прогнозирование — выявление ограничений на основе свидетельств;
4) отбор значимых ограничений;
5) определение экспертов для тестирования результатов;
6) детализация и фильтрация ограничений.

Таким образом, в задачах проектирования сложных программных комплексов помимо традиционного использования методологий IDEF0, IDEF1X и IDEF3 рекомендуется использовать методологии IDEF4, IDEF5 и IDEF9. При этом следует учитывать область применения методологий. Рекомендуется привлекать специалистов по предметным областям проектируемых программных средств, которые могли проектировать архитектуру информационных систем не отвлекаясь на задачи программирования.

Литература

1. Кулябов Д.С., Королькова А.В. Введение в формальные методы описания бизнес-процессов. – М.: РУДН, 2008. – 173 с.: ил.

2. Проект IDEF.RU [Электронный ресурс]: www.idef.ru. Электрон. текстовые дан., [2012].- Режим доступа: http://www.idef.ru, свободный.-Загл. с экрана.

И.М. Ценилов
Научный руководитель – старший преподаватель
кафедры электроники и вычислительной техники
Д.В. Бейлекчи
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23
e-mail: evt@mivlgu.ru
e-mail: kaf-eivt@yandex.ru

Исследование средств разработки 64-разрядных приложений для ОС Windows

Прогресс не стоит на месте, и разработка 64-х битных приложений для персональных компьютеров становится все более важной в последние несколько лет. 64-х битные приложения постепенно вытесняют 32-х битные (как те в свое время вытеснили 16-битные) по ряду причин: они позволяют работать с большим количеством оперативной памяти, они совместимы с 64-х битными системами. Также для некоторых приложений увеличивается производительность из-за того, что 64 бита дает возможность обрабатывать данные больше 2 гигабайт в оперативной памяти без использования файла подкачки. В настоящее время большинство семейств операционных систем имеют 64-х битные версии. Первыми операционными системами для современных компьютеров были Windows XP Professional x64 Edition, вышедшая в 2005 году, и Mac OS X 10.3 вышедшая в 2007 году. Эти ОС вышли относительно недавно, хотя первая 64-х битная операционная система появилась ещё в 1985 году (UNICOS - первая 64-битная реализация операционной системы Unix[1]).

Ранее основным средством разработки 64-х битных приложений для ОС Windows являлась MS Visual Studio, которая начала поддержку 64-х битных приложений с выходом 64-разрадной ОС Windows. Уже версия Visual C++ 2005 поддерживала компиляцию как для x86-64 (AMD64 и Intel 64) архитектуры микропроцессора, так и для IA-64 (реализована в микропроцессорах Itanium и Itanium 2). В этой версии был 64-х битный компилятор, а так же были адаптированы под 64 бита стандартные библиотеки.

1 сентября 2011 года Embarcadero выпустила RAD Studio XE2, которая включает в себя Delphi XE2, а также C++Builder, Prism XE2 и RadPHP XE2. Одним из множества нововведений стал компилятор 64-х битных приложений для Windows (поддержка для Mac OS планируется в версии XE 3)[2]. Все технологии Windows в Delphi были переработаны для создания 64-х битных приложений, включая библиотеку виртуальных компонентов(VCL, Visual Component Library) и библиотеку времени исполнения (RTL, Run Time Library), а так же была добавлена пока не имеющая аналогов библиотека визуальных компонентов FireMonkey, которая тоже поддерживает разработку 64-х битных приложений. Стоит отметить, что Delphi позволяет разработчикам взять уже существующие 32-х битныеVCL приложения и создать на их основе 64-х битные приложения, просто выбрав 64 бита, как целевую платформу в меню управления проектом. Кроме того в Delphi XE 2 удобно разрабатывать 64-разрядные интернет-приложения и интернет-сервисы, например для Windows Server 2008 R2 и более новых версий, благодаря технологии DataSnap[3].

В настоящее время процесс перехода от 32-х битных приложений к 64-х битным протекает медленно. Перенос некоторых проектов получается очень трудоемким,

приходится исправлять значительные участки кода, а некоторые места и вовсе полностью переписывать. Но появление новой высокоэффективной среды разработки позволит частично упростить этот процесс, что приведет, в конечном счете, к фактически полному отказу от разработки 32-х битных приложений и переходу на 64-х битные приложения.

Литература

1. Статья, посвященная разработке 64-х битных приложений: [Электронный ресурс]. Доступ: http://www.viva64.com/ru/a/0063/#ID0EHC.

2. RAD Studio. Википедия: свободная энциклопедия: [Электронный ресурс]. Доступ: http://www.wikipedia.org/.

3. Официальный сайт Embarcadero Technologies: [Электронный ресурс]. Доступ: http://www.embarcadero.com/.

Д.О. Шмельков
Научный руководитель –
ассистент кафедры электроники и вычислительной техники
А.Н. Коноплев
Муромский институт Владимирского государственного университета
602264 г. Муром, Владимирской обл., ул. Орловская, д. 23
Тел.: (49234) 77-2-73
e-mail:electron@mivlgu.ru

Система охранной сигнализации с оповещением по канала связи GSM

Сегодня, когда мобильный телефон стал такой же обыкновенной вещью, как и наручные часы, многие производители наделяют его все большим объемом дополнительных функций и интегрированных устройств, превращая привычный телефон в нечто намного большее, нежели устройство для передачи голоса на расстояние. Разрабатываемое устройство, описанное ниже, позволит превратить сотовый телефон еще и в полнофункциональный пейджер охраной сигнализации, которая сможет в любое время суток и в любом месте сообщить о небезопасном прошествии на охраняемом объекте [1].

Предлагаемое устройство предназначено для охраны объектов, таких как: квартир, дачных домиков, гаражей, и других объектов. Собрано оно на микроконтроллере (PIC18F6525) и кроме подачи звукового и светового сигналов тревоги оповещает владельца охраняемого объекта по сети сотовой связи GSM. Предлагаемое устройство с помощью датчиков контролирует состояние охраняемого объекта и в случае несанкционированного проникновения или пожара включает световую и звуковую сигнализации, привлекающие общее внимание к объекту, а также осуществляет оповещение владельца по каналу GSM, формируя речевое сообщение (при необходимости SMS сообщение) о несанкционированном проникновении на охраняемый объект, а также о его состоянии в текущий момент[2].

Сигнализатор может быть дополнением к имеющейся системе охранной сигнализации автомобиля или работать самостоятельно, при этом никакой доработки или изменения программы не потребуется. Для вызова владельца объекта использован принцип быстрого набора телефонного номера, для чего необходимо на устройство запрограммировать необходимый номер, по которому будет произведен звонок. В состав системы охранной сигнализации входят: микроконтроллер (PIC18F6525), различные датчики (простые контактные, датчики движения и т.д.), кнопка постановки/ снятия с охраны объекта, звуковая (например, звуковая сирена) и световая (различные диоды) сигнализация, устройство передачи данных по каналу GSM (Simcom 300C). В соответствие с этим была разработана следующая структурная схема системы охранной сигнализации оповещение по каналу GSM (рис. 1)

Рис. 1. Структурная схема системы охранной сигнализации

В случае возникновения происшествия на охраняемом объекте датчики системы подают сигнал контроллеру, тот в свою очередь активирует устройство передачи данных по каналу GSM, входящий в состав охраной системы и вынуждает его связаться с телефоном владельца. В результате владелец получает на свой телефон сигнальный звонок о происшествии и SMS-сообщение с текстом, описывающим событие которое произошло. Если через 10-15 мин после получения подтверждения о принятом сообщении датчик останется сработавшим, т. е. никаких мер для изменения ситуации не предпринято, устройство повторит вызов по тому же алгоритму. После звонка владельцу устройство включит световую и звуковую сигнализацию, которая будет работать все время пока датчик не будет возвращен в первоначальное состояние, или не произойдет остановка системы от владельца.

Для данного устройства была выбрать и обосновать архитектуру разрабатываемого устройства, разработанная структурная схема, разработана схема электрическая принципиальная устройства, а также написано программное обеспечение на языке Си в среде разработке Mplab MCC18[3].

Литература

1. Кудряшов С.А. Охранная система с оповещением по сотовому телефону – Радио, 2005, №6, с. 42-45

2. Иванов А. В., Кленов С. И. Построение микропроцессорных систем на базе однокристальных микроЭВМ. М.: Изд-во МЭИ, 1992. 52 с.

3. MPLAB. Ide, simulator, editor users guide [Электронный ресурс].-Режим доступа: http://www.microchip.ru/files/software/mplabi/51025b.pdf -Имеется печатный аналог.