



[ПРОГРАММИСТ]

Программирование и алгоритмизация.
Для молодых и активных людей..



Компилятор
домашнего
приготовления



Исследование
протокола PS/2
для мышки



Работа с
MySQL в C++

С днем программиста!



Разработчики -
интерфейс -
пользователи



Программы
для самых
маленьких



И многое
другое...

Издаётся с марта 2010. Выходит ежемесячно
№6, сентябрь 2010 г.

Редакция:

Выпускающий редактор
Сергей Бадло

Литературный редактор
Utkin

Редакторы
JTG, Василий Мединцев,
Алексей Шульга, Егор Горохов

Редактор-корректор
Yan Liplavskiy, Ксения Павлова

Редактор новостей
Антон Бердников

Дизайн и верстка:
Егор Горохов, Сергей Бадло

Авторский состав:
psycho-coder, Владимир Дегтярь,
Евгений Амосов, Дарья Устюгова
Виталий Белик, Александр Демьяненко,
Сергей Бадло

Официальный сайт журнала:
www.procoder.info

Контакты:
Авторские статьи направляйте на
maindatacentr@gmail.com
Вопросы и предложения для редакции
reddatacentr@gmail.com
Вопросы и предложения администратору
info@kotoff.info

Информационная поддержка:
Международная Академия Информатизации
(МАИН) РК www.academy.kz
Журнал «Радиолобитель»
www.radioliga.com
Клуб ПРОграммистов
www.programmersforum.ru
V.K. сайт...
www.kotoff.info
Free Legal Soft Group
www.flsoft.ru

Примечание:

Издание некоммерческое. Все материалы,
товарные знаки, торговые марки и логотипы,
упомянутые в журнале, принадлежат их
владельцам. Статьи, поступающие в редакцию,
рецензируются. Мнение авторов не всегда
совпадает с мнением редакции. Перепечатка
материалов журнала и использование их в
любой форме, в том числе в электронных СМИ,
возможны только с разрешения редакции.
Тираж неограничен. Формат А4, 53 стр.

Учредитель:
Клуб ПРОграммистов
www.programmersforum.ru

Обложка номера:
Дизайн Егора Горохова

ТЕМА НОМЕРА

Наши разработки с.0x02

НЕВЕРОЯТНО, НО ФАКТ

Любопытные факты с.0x03

ПЕРЕВОДНЫЕ МАТЕРИАЛЫ

Исследование протокола PS/2 для мышки с.0x06

ОТДЕЛ ТЕСТИРОВАНИЯ

Разработчики - интерфейс - пользователи. Часть 2 с.0x0E

ОБЩИЕ ВОПРОСЫ

0 правильном составлении ТЗ. Часть 1 с.0x14

2D ГРАФИКА

Программы для самых маленьких с.0x17

ЛАБОРАТОРИЯ

Компилятор домашнего приготовления с.0x1D

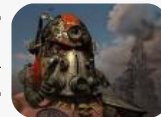
АРХИВ

Работа с MySQL в C++ с.0x2A

ЮМОР

Афоризмы - хохмы - загадки с.0x32

От редактора. Здравствуйте, уважаемые читатели журнала «ПРОграммист» от Клуба ПРОграммистов www.programmersforum.ru. С сентябрьского номера наша редакция работает в обновленном составе и со свежими силами продолжит вгрызаться в нелегкое и, в то же время, любимое издательское дело. Помогут нам в этом редакторы: Василий Мединцев и Ян Липлавский. Но, помощники требуются всегда. Не будем забывать, что наш проект некоммерческий и базируется на энтузиазме наших читателей, авторов и конечно же редакторского состава. Поэтому мы всегда рады новым предложениям, статьям и заметкам, которые вы можете прислать и высказать на редакторский ящик-копилку maindatacentr@gmail.com. Не стесняемся и будем активнее.



В этом выпуске...

Материал про юзабилити интерфейсов продолжит Александр Демьяненко в рубрике «Отдел тестирования».

И наконец-то стартует рубрика «Переводные материалы». Статья Евгения Амосова подробно расскажет нам про протокол работы наших хвостатых PS/2-помощников.

Что такое ТЗ? Как правильно составить ТЗ? Кто должен составлять ТЗ? Прямо голова идет кругом. Но не расстраивайтесь. На эти и другие вопросы постарается ответить Дарья Устюгова.

Для развития координации, цветового восприятия ребенку до 3-5 лет требуются простые и понятные игрушки. Одним из ярких примеров является раскраска и интерактивный алфавит. Не мне вам рассказывать, родители меня поймут, что увлечь малыша могут веселые и красочные картинки зверей, игрушек. Мало того, возможность почувствовать себя художником – это непередаваемое ощущение для ребенка. А поможет нам в этом ваш верный помощник – компьютер и Владимир Дегтярь в рубрике «2D графика». Оценивать же будет самый строгий и требовательный судья – ребенок, экзаменатор нашего труда.

Хорошо-ли вы умеете готовить? Нет, это не шутка :), вспомните про направленность нашего журнала. Вспомнили? Сегодня в рубрике «Лаборатория» шеф-повар, Виталий Белик, приготовит компилятор по своему домашнему рецепту. *Bon Appetit*, друзья!

В рубрике «Архив» мы решили поднять тему взаимодействия с MySQL в C++. В данной статье делится опытом наш форумчанин **psycho-coder**.

Рубрики журнала (плавающие)

- Новости программирования
- Отдел тестирования
- Общие вопросы (правовое использование)
- Алгоритмы
- Переводные материалы
- Разработка (проекты от этапа ТЗ до сдачи)
- Wi-Fi сети
- Лаборатория
- Архив (по материалам клуба и форума)
- Юмор (специфические хохмы программеров)

Общие требования к материалам

У нас нет категоричных требований к оформлению, но в связи с особенностями верстки (используется свободное ПО «SCRIBUS») и облегчения труда редакторов, есть некоторый желательный минимум:

- статья должна иметь выраженную структуру с разделами и содержать название статьи, сведения об авторах, экскурс или введение, информацию об используемых средствах разработки, теоретическую и/или практическую часть, заключение и ресурсы к статье;
- текст статьи без табуляции в формате MS Word, [VK WordPad](#) или обычным текстовым файлом, шрифт Arial 10;
- все рисунки, таблицы должны быть подписаны и иметь упоминание в тексте;
- рисунки к статье должны прилагаться **в виде отдельных файлов** в формате PNG, TIF;
- разделы статьи отделять двумя <ENTER>.

По присланным материалам автор получает рецензию и корректирует статью согласно замечаниям. Шаблон для написания статьи можно взять [тут](#).

С уважением, Редакция

Вот и настал час 0xFF! Мы знаем, что среди наших читателей есть не только программисты, но и биологи, радиоинженеры и радиолюбители, дизайнеры, физики и химики, профессионалы и просто обычные люди :) С этим стоит немножко поспорить. Не совсем обычные. Ведь практически сейчас все, что нас окружает, то чем мы пользуемся ежедневно и ежечасно, так или иначе связано с программированием. Пожалуй, ни одна профессия не пустила так глубоко свои корни во все сферы нашей жизни, как программист. Поэтому, с днем Программиста всех Вас, читающих нас и пишущих нам, поздравляет коллектив Редакции журнала «ПРОграммист». Но, это не все новости на сегодня. Итак...



Сергей Бадло

by **raxp** <http://raxp.radioliga.com>

Для «айтишников» в украинском Налоговом кодексе предусмотрены привилегии, сообщил в своем блоге глава Государственного комитета по вопросам регуляторной политики и предпринимательства Михаил Бродский.

В новой редакции проекта единый налог составит 1500 гривен www.search.ligazakon.ua/l_doc2.nsf/link1/JF50T00I.html плюс отчисления в ПФ (всего 1820 грн.). Но, только для компаний с годовым оборотом менее 300 тыс. грн. При превышении оборота компания будет переводиться на общую систему налогообложения. Эта цифра записана и проработана с МВФ, в том числе и Тигипко... Интересно то, что упрощенная система налогообложения не распространяется на субъекты хозяйствования – юридических лиц и физических лиц-предпринимателей, которые осуществляют:

- оптовую и розничную торговлю через сеть Интернет
- деятельность по предоставлению услуг Интернет-провайдеров
- деятельность в сфере информатизации
- деятельность по доступу в сеть Интернет

Что, в конечном итоге, может привести к уходу части данного бизнеса в подполье.

Концепт мобильного радара представлен исследовательским центром Nokia, позволяющего измерять скорость и направление приближения объекта, как и традиционные радары. Для своей работы аппарат использует электромагнитные волны, что позволяет реализовать различные виды зондирования.



В ядре Linux устранена крайне опасная уязвимость, расположенная в связке «ядро - графический X-сервер». За счет использования определенных команд и запросов, через X-сервер можно было получить администраторский доступ к ОС. Сообщается, что над устранением уязвимости работал лично Линус Торвалдс. Сам Торвалдс в рассылке охарактеризовал проблему как «крайне опасную». По его данным, проблема была устранена в последней версии ядра, а также в последних патчах для предыдущих версий ядер Linux.



Уязвимость затрагивает все ядра Linux, начиная с версии 2.6.0, причем она характерна для 32 и 64-битных систем на базе процессоров x86.

Во Вьетнаме запретили интернет-кафе в непосредственной близости от школ и других учебных заведений. Интернет-кафе, расположенные ближе, чем в 200 метрах от учебных заведений, должны быть закрыты.

Кроме того, для пунктов коллективного доступа в интернет вводится еще одно ограничение – они не должны работать с 11 вечера до 6 утра. Таким

образом, власти страны пытаются бороться с популярностью онлайн-игр среди подростков. Руководство страны считает, что из-за онлайн-игр молодые люди бросают учебу и пропускают школу. В целом интернет-кафе должны установить у себя программное обеспечение, которое фильтрует интернет-контент, прежде всего, порно. Это третье требование является обязательным для всех интернет-кафе страны.

Хранить отработанное ядерное топливо* в стеклянных блоках предложили химики из Института геохимии и аналитической химии РАН под руководством академика Бориса Мясоедова.

Суть новой технологии – в применении сверхкритической двуокиси углерода. При определенном давлении и температуре этот газ переходит в особое состояние, промежуточное между жидким и газообразным, в результате чего становится мощным растворителем. В частности, в нем растворяют специальное вещество, способное извлечь из отработанного топлива весь уран. После того как весь уран перейдет в такой сверхкритический раствор, его пропускают через воду. Уран в виде соли в ней остается, а чистая углекислота улетает. Второй этап – разделение остатка, то есть смеси плутония с продуктами деления урана. Пока новая технология еще не отработана, предлагается хранить концентрированную смесь в небольших стеклянных блоках.

Первые нетбуки на базе двухъядерных мобильных процессоров Atom N550 начала продавать корпорация Intel. До конца текущего

года такие системы анонсируют Acer, Asustek, Fujitsu, Lenovo, LG, Samsung, Toshiba, MSI и другие вендоры.

SSD память уменьшила и ускорила компания SanDisk. Технология получила название iSSD, так как позволяет поместить модуль памяти и контроллер в очень маленький чип, который к тому же быстрее, чем обычная NAND-память. Пиковая скорость чтения такого чипа (от 4 до 64 ГБ) составляет 160 Мб/сек, записи – 100 Мб/сек,

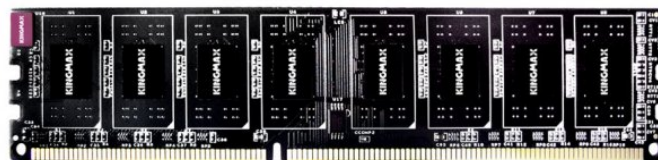
что быстрее многих жестких дисков. Память использует интерфейс SATA, но при этом она может крепиться и непосредственно к печатной плате.

Новые модули памяти HERCULES DDR3 с невидимыми радиаторами

представила компания KINGMAX.

Эффективность предложенной технологии Nano Thermal намного превосходит эффективность традиционных радиаторов: повышает теплоотдачу на 9.5%. Пропускная же способность новых двухканальных модулей DDR3 2200

МГц позволяет достигать скорости передачи данных 17.6 Гб/с. Отсутствие радиаторов делает модули меньше и легче, что также способствует лучшей циркуляции воздуха внутри корпуса. С другой стороны, отсутствие необходимости в применении радиаторов позволяет снизить цену на память.



*** Справка.**

Отработанное ядерное топливо – кладёз ценных элементов. И действительно, ядерная энергетика – единственный процесс, когда при сгорании топлива образуется другое топливо. Это плутоний, который даст энергию в реакторе на быстрых нейтронах. Помимо него есть там и уран. Следующий компонент – цезий, радиоактивный источник для многочисленных приборов. Трансурановые элементы отсутствуют в земной коре, и отработанное топливо – их единственный источник.

Во всем мире используют так называемый ПУРЭКС- процесс. Не потому, что он так хорош, а потому, что лучше ничего пока не придумали. Старые ТВЭЛ-ы растворяют в азотной кислоте. Это уже опасно: ведь азотная кислота – сильнейший окислитель. Перед ней трудно устоять даже специальному оборудованию. Из получившегося раствора компонентов бывших ТВЭЛов и продуктов растворения оборудования в азотной кислоте как раз и вылавливают первым делом уран и плутоний. Это делают специальным реагентом, трибутилфосфатом. В результате получаются два радиоактивных раствора. В одном – почти весь уран и плутоний. В другом – все что осталось: немного урана и плутония, осколочные элементы и продукты коррозии. И растворов этих тысячи тонн.

До недавнего времени растворы заливали в специальные емкости (в России – бетонные, в США – из нержавеющей стали) и закапывали в землю. Понятно, что ни тот, ни другой материал не в состоянии выдержать такую агрессивную среду в течение тех тысяч, а то и миллионов лет, за которые радиоактивность сравняется с фоном.



Шпионское приложение для Android маскируется под «Змейку» предупреждает антивирусная компания Symantec. Бесплатное



приложение, под названием Tapsnake, выдает себя за Android-версию знаменитой игры. О чем разработчики не сообщают, так это о том, что каждые 15 минут программа отправляет GPS-координаты жертвы на специальный сервер, на котором их могут просмотреть пользователи другой, на этот раз платной, программы, известной как GPS Spy. Та, в свою очередь, накладывает данные GPS на карты Google, что позволяет точно определить координаты отслеживаемого объекта.

Впрочем, желающим проследить за пользователями Tapsnake придется получить к их смартфонам физический доступ, поскольку данные аккаунта пересылки нужно вводить вручную. Кроме того, Android уведомляет пользователей про действия в системе. Тем не менее, появление такого приложения – это еще один призыв к бдительности при использовании стороннего, в особенности бесплатного, ПО на мобильных устройствах.



Необычную конструкцию Wi-Fi антенны соорудил один из активистов опен-Wi-Fi движения. Наряду с обычным диполем-антенной, в качестве фокусирующего зеркала была применена маска со сгоревшего кинескопа...

Управление перспективных исследовательских проектов (DARPA)

Пентагона выделило средства на разработку процессора, оперирующего вероятностями, а не единицами и нулями. Ожидается, что вероятностный чип более эффективен в фильтрации спама, обнаружении мошеннических схем при анализе финансовых транзакций, а также во многих сложных алгоритмах, включая применяемые в системах вооружения. Одним из важнейших достоинств вероятностного чипа является его способность к коррекции ошибок. В настоящее время именно большое число ошибок сдерживает дальнейшее увеличение объемов флеш-памяти. Так, современные чипы справляются с одним неверным битом на тысячу, считанных из памяти.

Специализированный чип LEC, разработанный Lyrisc, позволит исправлять ошибки даже в том случае,

если ошибка будет происходить в десять раз чаще.

При этом, чип занимает в 30 раз меньше места. Серийное производство LEC начнется в течение года.



Sony предлагает одножильный вариант интерфейса обмена данными. В интерфейсе используется одна линия медного проводника (Single Wire Interface Technology). Принцип действия изобретения** основан на методе временного разделения (Time Division Duplex) и временного мультиплексирования (Time Division Multiplexing) каналов. Передача информации может производиться в обоих направлениях, по этому же проводнику передается напряжение питания. Максимальная длина проводника – 60 см. Новая разработка Sony позволит повысить надежность при сокращении каналов передачи.

**** Комментарий редакции.**

...стоит обратить ваше внимание, что так называемое изобретение от Sony напоминает ситуацию с интерфейсом 1-Wire, когда для исключения судебных исков, этот интерфейс был переименован в TWI компанией Atmel. Поэтому, для разработчиков, данное нововведение никак не новость, а уже давно используемый интерфейс.

В этой статье я попытаюсь объяснить аспекты, связанные с интерфейсом PS/2 для периферийного устройства типа мышь, включая физический, электрический интерфейс, протокол низкого уровня, режимы работы, команды и расширения...



Евгений Амосов

by **Gambit_OZ** gambitoz@mail.ru

Мышь воспринимает свое перемещение в рабочей плоскости (обычно - на участке поверхности стола) и передает эту информацию компьютеру. Программа, работающая на компьютере, в ответ на перемещение мыши производит на экране действие, отвечающее направлению и расстоянию этого перемещения.

В дополнение к детектору перемещения, мышь имеет от одной до трех и более кнопок, а также дополнительные элементы управления (колеса прокрутки, потенциометры, джойстики, трекболы, клавиши и т.п.), действие которых обычно связывается с текущим положением курсора (или составляющих специфического интерфейса).

Фактически все эти устройства общаются с компьютером при помощи двух интерфейсов: универсальной последовательной шины (USB) или интерфейса PS/2.

Интерфейс PS/2 расшифровывается как «Personal System/2». Широкую популярность он приобрел благодаря использованию для взаимодействия с манипулятором типа мышь в компьютерах Apple Macintosh и позднее в ОС Windows для IBM PC в конце 80-х. Однако сегодня быстро завоевавший популярность



Рис. 1. Обозначение контактов PS/2

интерфейс USB в конечном итоге практически заменил PS/2 полностью. Тем не менее, протокол PS/2 представляет собой интересную платформу для различных экспериментов научного



характера. Например, можно использовать мышку с разъемом PS/2 для управления каким-либо передвижным устройством - роботом или сделать свой манипулятор, который при помощи датчиков - гироскопов, закрепленных на пальцах руки, будет эмулировать передвижение курсора на экране, а щелчком пальцев открывать папки и т.д. Интерфейс мыши PS/2 использует двунаправленный последовательный протокол, по которому передаются данные о движении и состоянии кнопок вспомогательному диспетчеру устройств компьютера (далее по тексту - хост). Диспетчер, в свою очередь, посылает команды для мыши, чтобы установить частоту обмена, разрешение, ее перезагрузку, выключение и т.д. Напряжение питания на линии передачи 5В/ ~100 мА. На рисунке 1 показано обозначение контактов разъема PS/2, на рисунке 2 показано возможное подключение к своему устройству:

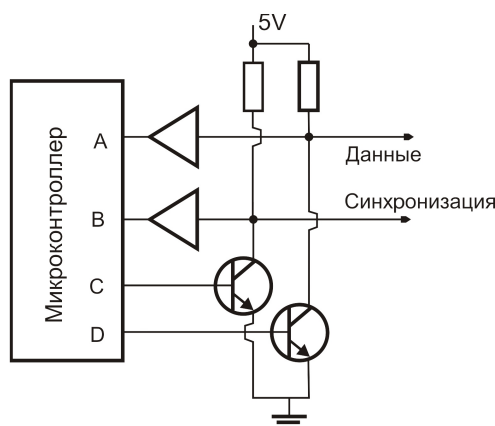


Рис. 2. Подключение мыши к своему устройству

Входы, Разрешение, и Масштабирование

Стандартный интерфейс мыши PS/2 поддерживает следующие входы: перемещение по координате X (право/лево), перемещение по координате Y (вверх/вниз), левая кнопка, средняя кнопка, и правая кнопка. Мышь периодически читает эти входы, обновляет связанные с ними счетчики и устанавливает флаги, чтобы в дальнейшем отправить хосту текущее состояния кнопок и перемещение. Есть много манипуляторов PS/2, которые имеют дополнительные входы и могут сообщить о данных по-другому, чем описано в этом документе. Одним из популярных расширений - является Microsoft Intellimouse, которое включает в себя поддержку стандартных входов, а также колеса прокрутки и двух дополнительных кнопок.

У стандартной мыши есть два 9-ти разрядных счетчика с флагами переполнения, которые отслеживают перемещение: по координате X и по координате Y. Их содержание, а также состояние трех кнопок мыши, отправляется хосту в виде 3-байтового пакета данных. Счетчики перемещения определяют смещение мыши относительно ее предыдущей позиции, когда было совершено перемещение или когда пришла команда «Resend» (0xFE).

Когда мышь читает свои входы, она записывает текущее состояние своих кнопок и постепенно увеличивает/уменьшает счетчики перемещения согласно значению перемещения, которое произошло относительно последнего значения. Если любой из счетчиков переполнится, то будет установлен соответствующий флаг переполнения. Параметр, который определяет величину увеличения/уменьшения счетчиков

перемещения, является разрешением. Разрешение по умолчанию равно 4-м значениям на миллиметр, и хост может изменить это значение, используя команду «Set Resolution» (0xE8).

Существует параметр, который не влияет на счетчики перемещения, но влияет на значения, которые снимаются со счетчиков. Этот параметр называется - масштабирование. По умолчанию, мышь использует масштабирование 1:1. Однако, хост может выбрать масштабирование 2:1, используя команду «Set Scaling 2:1» (0xE7). Если будет включено масштабирование 2:1, то мышь применит следующий алгоритм к счетчикам перемещения прежде, чем отправить их содержание хосту (см. таблицу 1).

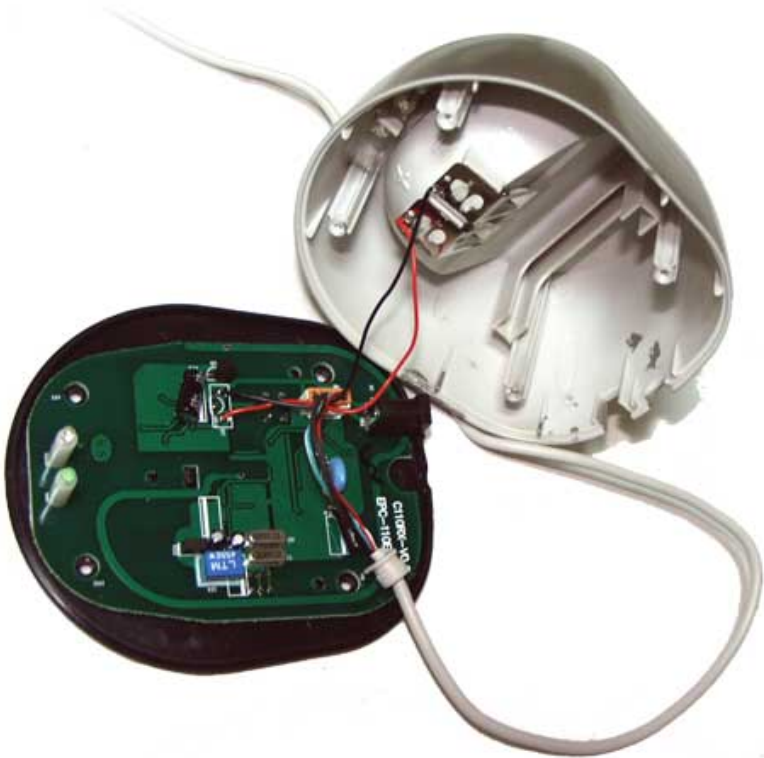
2:1 масштабирование применяется только к автоматически отсылаемым данным в потоковом режиме. Это не влияет на данные, которые отправляются в ответ на команду «Read Data» (0xEB).

Табл. 1. Алгоритм 2:1 масштабирования

Состояние счетчика движения	Отсылаемое значение
0	0
1	1
2	1
3	3
4	6
5	9
N > 5	2 * N

Пакет данных о перемещениях

Стандартная мышь PS/2 посылает информацию о



движении/кнопках хосту, используя следующий 3-байтовый пакет (см. таблицу 2). Значения перемещения – это два 9-разрядных целых числа, где старший значащий бит появляется битом «X/Y знак» в 1-м байте из передаваемого пакета данных. Их значение представляет смещение мыши относительно своей предыдущей позиции, определенных текущим разрешением. Диапазон отсылаемых значений лежит в пределах от -255 до +255. Если этот диапазон будет превышен, то установится соответствующий бит переполнения.

Табл. 2. Пакет данных о перемещениях

Байт	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
1	Y переполнение	X переполнение	Y знак бит	X знак бит	всегда 1	Средняя кнопка	правая кнопка	левая кнопка
2	X перемещение							
3	Y перемещение							

Режимы работы

Создание отчетов данных обработано согласно режиму, в котором работает мышь. Есть четыре режима работы:

- сброс (Reset) – начальный режим, в котором мышь выполняет инициализацию и самодиагностику;
- поток (Stream) – рабочий режим по умолчанию, в котором мышь отправляет пакеты данных о перемещениях, когда происходит перемещение или изменяются состояния кнопок;
- удаленный (Remote) – хост должен упорядочить пакеты данных о перемещениях;
- обратный (Wrap) – диагностический режим, в котором мышь отправляет каждый полученный пакет назад к хосту.

Режим сброса (Reset)

Мышь переходит в режим сброса при включении питания или в ответ на команду «Reset» (0xFF). После ввода этого режима мышь выполняет диагностическую самопроверку и устанавливает следующие значения по умолчанию:

- Частота дискретизации = 100 значениям в секунду;
- Разрешение = 4 значениям на 1мм;
- Масштабирование = 1:1;
- Создание отчетов* данных = отключено.

После режима самодиагностики мышь отправляет хосту результат проверки – 0xAA (успешно) или 0xFC (Ошибка).

Следующим значением после 0xAA или 0xFC, мышь отправляет свой ID = 0x00. Это значение указывает на то, что это мышь, а не клавиатура или иное устройство. Иногда в описаниях протокола указано, что хост не должен передавать данные, пока тот не получит ID устройства. Однако, некоторые БИОСы

отправляют команду «Reset» (0xFF) сразу после команды 0xAA, полученной после того, как включенное питание было сброшено.

Как только мышь отправила свой ID устройства хосту, она входит в потоковый режим.

Потоковый Режим (Stream)

В потоковом режиме мышь отправляет данные перемещения, когда обнаруживается перемещение или изменение в состоянии одной или более кнопок мыши. Максимальная частота, на которой можно сообщить об этих данных, известна как частота дискретизации. Этот параметр изменяется в пределах от 10 до 200 значений в секунду (по умолчанию 100 значений в секунду). Хост может изменить это значение, используя команду «Set Sample Rate» (0xF3).

Потоковый режим – рабочий режим по умолчанию, в который можно установить, используя команду «Set Stream Mode» (0xEA).

Удаленный Режим (Remote)

В удаленном режиме мышь читает свои входы и обновляет счетчики/флаги на текущей частоте дискретизации, но не отправляет автоматически

*** Комментарий автора.**
Отметим, что создание отчетов отключено по умолчанию. Мышь не будет фактически слать пакеты данных о перемещениях, пока она не получит команду «Enable Data Reporting» (0xF4).

пакеты данных, если произошло перемещение.

Вместо этого хост опрашивает мышь, используя команду «Read Data» (0xEB). После получения этой команды мышь отправляет единственный пакет данных о перемещениях и сбрасывает свои счетчики перемещения.

Мышь входит в удаленный режим при получении команды «Set Remote Mode» (0xF0). Удаленный режим используется редко.

Режим обратный (Wrap)

Это «эхо-режим», в котором каждый байт, полученный мышью, отправляется назад к хосту. Даже, если полученный байт будет соответствовать системной команде, то мышь не будет отвечать на ту команду, а только отправит этот байт назад хосту. Однако, есть два исключения: команды «Reset» (0xFF) и «Reset Wrap Mode» (0xEC). Мышь обрабатывает их как допустимые команды и не отправит их назад хосту. Обратный режим редко используется.

Расширения Intellimouse

Популярное расширение стандартной мыши PS/2 – Microsoft Intellimouse, включает поддержку в общей сложности пяти кнопок мыши и трех осей

После включения питания или сброса Microsoft Intellimouse работает точно так же, как стандартная мышь PS/2 (то есть, использует 3-байтовый пакет данных о перемещения, отвечает на все команды таким же образом, как на стандартную мышь PS/2, и сообщает о ID устройства 0x00).

Чтобы включить колесо прокрутки, хост отправляет следующую последовательность команд:

- 1. Установить частоту дискретизации = 200
- 2. Установить частоту дискретизации = 100
- 3. Установить частоту дискретизации = 80

Хост отправляет команду «Get device ID» (0xF2) и ожидает ответа. Если это стандартная мышь PS/2 (то есть, не IntelliMouse) то она ответит ID устройства 0x00. В этом случае хост распознает факт, что мышь действительно не имеет колесо прокрутки и будет продолжать обрабатывать ее как стандартную мышь PS/2.

Однако, если будет подключена Microsoft Intellimouse, то она ответит ID = 0x03. Это сообщит хосту, что у присоединенного манипулятора есть колесо прокрутки, и хост перейдет в режим ожидания 4-х-байтового пакет данных о перемещения(см. таблицу 3):

Табл. 3. 4-х байтовый пакет данных

Байт	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
1	Y переполнение	X переполнение	Y знак бит	X знак бит	всегда 1	Средняя кнопка	правая кнопка	левая кнопка
2	X перемещение							
3	Y перемещение							
4	Z перемещение							

перемещения (право/лево, вверх/вниз, и колесо прокрутки). Эти дополнительные функции требуют использования 4-байтового пакета данных перемещения, а не стандартного 3-байтового пакета. Так как стандартные драйверы мыши PS/2 не могут распознать этот пакетный формат, Intellimouse обязан работать точно как стандартная мышь PS/2. Таким образом, если Intellimouse будет использоваться на компьютере, который поддерживает только стандартную мышь PS/2, то она (мышка) будет все еще функционировать, за исключением колеса прокрутки и 4-й и 5-й кнопки.

«Z перемещение» – дополнительное значение, которое представляет собой перемещение колеса прокрутки, начиная с последнего отчета данных.

Допустимые значения находятся в диапазоне от -8 до +7. Это означает, что число фактически представлено как четыре младшие, значащие бита; старшие четыре бита – только как расширение знака.

Чтобы включить колесо прокрутки, а также 4-ю и 5-ю кнопки, хост отправляет следующую последовательность команд:

- установить частоту дискретизации = 200
- установить частоту дискретизации = 200
- установить частоту дискретизации = 80

Хост шлет команду «Get device ID» (0xF2) и ожидает ответа. Microsoft Intellimouse отправляет ID устройства = 0x04, затем использует следующий 4-х байтовый пакет данных перемещения (см. таблицу 4):

Табл. 4. Пакет данных для активации колеса прокрутки и дополнительных кнопок

Байт	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
1	Y переполнение	X переполнение	Y знак бит	X знак бит	всегда 1	Средняя кнопка	правая кнопка	левая кнопка
2	X перемещение							
3	Y перемещение							
4	0	0	5-я кнопка	4-я кнопка	Z переме- щение			

«4-я кнопка» = 1, если нажата 4-я кнопка мыши, и «5-я кнопка» = 1, если нажата 5-я кнопка мыши. «Z перемещение» представляет собой значение движения, которое произошло, начиная с последнего отчета данных. Действительные значения изменяются в пределах от -8 до +7.

Набор команд

Следующий набор команд является стандартом для мыши с интерфейсом PS/2:

- Если мышь находится в потоковом режиме, то хост должен отключить создание отчетов данных (команда 0xF5) прежде, чем отправить какие-то команды.
- 0xFF (Reset) – мышь отвечает командой (0xFA) и входит в режим сброса.
- 0xFE (Resend) – хост отправляет эту команду всякий раз, когда он получает ошибочные данные от мыши. Мышь отвечает, посылая последний пакет, который она отправила хосту. Если мышь отвечает на команду «Resend» другим недопустимым пакетом, хост может или дать другую команду «Resend» или сформировать команду «Error» (0xFC) и произвести сброс питания с мыши. Эта команда не буферизована, а значит «Resend» никогда не будет отправляться в ответ на команду «Resend».
- 0xF6 (Set Defaults) – мышь отвечает командой (0xFA) и устанавливает следующие значения:

частота дискретизации = 100, разрешение = 4 значения на 1 мм, масштабирование = 1:1, создание отчетов данных = отключено. Мышь сбрасывает все свои счетчики перемещения и входит в потоковый режим.

- 0xF5 (Disable Data Reporting) – мышь отвечает командой (0xFA), отключается создание отчетов данных и сбрасываются счетчики перемещения.
- 0xF4 (Enable Data Reporting) – мышь отвечает командой (0xFA), включается режим создания отчетов данных и сбрасываются счетчики перемещения. Эта команда может быть получена, в момент когда мышь находится в удаленном режиме, но выполнится, только если мышь переключится в потоковый режим.
- 0xF3 (Set Sample Rate) – мышь отвечает командой (0xFA) и ожидает один байт из хоста. Мышь сохраняет этот байт как новую частоту дискретизации. После получения частоты дискретизации мышь снова отвечает командой (0xFA) и сбрасывает свои счетчики перемещения. Допустимые частоты дискретизации: 10, 20, 40, 60, 80, 100, и 200 значений в секунду.
- 0xF2 (Get Device ID) – мышь отвечает командой (0xFA), сопровождая его ID устройства (0x00 для стандартной мыши PS/2). Мышь должна также сбросить свои счетчики перемещения.
- 0xF0 (Set Remote Mode) – мышь отвечает командой (0xFA), сбрасывает свои счетчики перемещения и входит удаленный режим.
- 0xEE (Set Wrap Mode) – мышь отвечает командой (0xFA), сбрасывает свои счетчики перемещения и входит в эхо режим.
- 0xEC (Reset Wrap Mode) – мышь отвечает командой (0xFA), сбрасывает свои счетчики перемещения и входит в режим, в котором она была до эхо-режима (в потоковый или удаленный).

- 0xEB (Read Data) – мышь отвечает командой (0xFA) и отправляет пакет данных перемещения. Это единственный способ считать данные в удаленном режиме. После того, как пакет данных был успешно отправлен, мышь сбрасывает свои счетчики перемещения.
- 0xEA (Set Stream Mode) – мышь отвечает командой (0xFA), сбрасывает свои счетчики перемещения и входит в потоковый режим.
- 0xE9 (Status Request) – мышь отвечает командой (0xFA), затем передается следующий 3-байтовый пакет состояния (см. таблицу 5). Правая, Средняя, Левая кнопка = 1, если соответствующая кнопка нажата; 0, если кнопка не нажата. Масштабирование = 1, если масштабирование 2:1 и 0, если масштабирование 1:1 (см. команды 0xE7 и 0xE6). Включение = 1, если включено создание отчетов данных и 0, если создание отчетов данных отключено (см. команды 0xF5 и 0xF4). Режим = 1, если удаленный режим включен и 0, если включен потоковый режим (см. команды 0xF0 и 0xEA).
- 0xE8 (Set Resolution) – мышь отвечает командой (0xFA), читает один байт из хоста и снова отвечает, командой (0xFA), затем сбрасывает свои счетчики перемещения. Побайтовое чтение от хоста определяет разрешение следующим образом (см. табл. 6).
- 0xE7 (Set Scaling 2:1) – мышь отвечает командой (0xFA), затем включает 2:1 масштабирование.
- 0xE6 (Set Scaling 1:1) – мышь отвечает командой (0xFA), затем включает 1:1 масштабирование.

Инициализация

Мышь PS/2 обычно обнаруживается или инициализируется, только когда компьютер загружается. Таким образом, мышь не может использоваться в горячем подключении. В результате, приходится перезапускать свой компьютер всякий раз, когда устанавливается или удаляется мышь PS/2.

Начальное обнаружение мыши PS/2 происходит во время включения компьютера. Если мышь будет обнаружена, то BIOS позволит операционной системе конфигурировать или использовать мышь. Иначе, устанавливается запрет на передачу по шине мыши. Если загружать компьютер с присоединенной мышью, отключить мышь и повторно ее включить в то время, когда загружается Windows, то ОС в состоянии обнаружить мышь и дать с ней работать.

Рассмотрим передачу данных между компьютером и стандартной мышью PS/2 во время процесса начальной загрузки (см. листинг):

Включение питания:

Мышь: AA Тестовый режим

Мышь: 00 Идентификатор Мыши

Хост: FF Команда сброса

Мышь: FA Подтверждение

Мышь: AA Тестовый режим

Мышь: 00 Идентификатор Мыши

Хост: FF Команда сброса

Табл. 5. Пакет состояния

Байт	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
1	всегда 0	Режим	Включе- ние	Масшта- биров.	всегда 0	левая кнопка	средняя кнопка	правая кнопка
2	Разрешение							
3	Частота опроса							

Табл. 6. Расшифровка данных принятых с Хоста

Байт считанный с хоста	Разрешение
0	1 значение/мм
1	2 значения/мм
2	4 значения/мм
3	8 значений/мм

Мышь: FA Подтверждение

Мышь: AA Тестовый режим

Мышь: 00 Идентификатор Мыши

Хост: FF Команда сброса

Мышь: FA Подтверждение

Мышь: AA Тестовый режим

Мышь: 00 Идентификатор Мыши

Хост: F3 Установить частоту дискретизации

Мышь: FA Подтверждение

Хост: C8 число 200

Мышь: FA Подтверждение

Хост: F3 Установить частоту дискретизации

Мышь: FA Подтверждение

Хост: 64 число 100

Мышь: FA Подтверждение

Хост: F3 Установить частоту дискретизации

Мышь: FA Подтверждение

Хост: F2 Прочитать тип устройства

Мышь: FA Подтверждение

Мышь: 00 Идентификатор Мыши

Хост: F3 Установить частоту дискретизации

Мышь: FA Подтверждение

Хост: 0A число 10

Мышь: FA Подтверждение

Хост: F2 Прочитать тип устройства

Мышь: FA Подтверждение

Мышь: 00 Идентификатор Мыши

Хост: E8 Установить разрешение

Мышь: FA Подтверждение

Хост: 03 8 значений/мм

Мышь: FA Подтверждение

Хост: E6 Установить масштабирование 1:1

Мышь: FA Подтверждение

Хост: F3 Установить частоту дискретизации

Мышь: FA Подтверждение

Хост: 28 число 40

Мышь: FA Подтверждение

Хост: F4 Включить

Мышь: FA Подтверждение

Инициализация завершена...

Если нажата левая кнопка:

Мышь: 09 1 1 00001001; Бит0 - состояние левой кнопки мыши; Бит3 всегда = 1

Мышь: 00 1 1 Нет X-перемещений

Мышь: 00 1 1 Нет Y-перемещений

... и отпущена левая кнопка мыши:

Мышь: 08 0 1 00001000 Бит0 - состояние левой кнопки мыши; Бит3 всегда = 1

Мышь: 00 1 1 Нет X-перемещений

Мышь: 00 1 1 Нет Y-перемещений

Теперь рассмотрим обмен данными между компьютером и мышью Intellimouse (при загрузке компьютера):

Включение питания:

Мышь: AA Тестовый режим

Мышь: 00 Идентификатор Мыши

Хост: FF Команда сброса

Мышь: FA Подтверждение

Мышь: AA Тестовый режим

Мышь: 00 Идентификатор Мыши

Хост: FF Команда сброса

Мышь: FA Подтверждение

Мышь: AA Тестовый режим

Мышь: 00 Идентификатор Мыши

Хост: FF Команда сброса

Мышь: FA Подтверждение

Мышь: AA Тестовый режим

Мышь: 00 Идентификатор Мыши

Хост: F3 Установить частоту дискретизации

Мышь: FA Подтверждение

Хост: C8 число 200

Мышь: FA Подтверждение

Хост: F3 Установить частоту дискретизации

Мышь: FA Подтверждение

Хост: 64 число 100

Мышь: FA Подтверждение

Хост: F3 Установить частоту дискретизации

Мышь: FA Подтверждение

Хост: 50 число 80

Мышь: FA Подтверждение

Хост: F2 Прочитать тип устройства

Мышь: FA Подтверждение

Мышь: 03 Идентификатор Мыши

Хост: E8 Установить разрешение

Мышь: FA Подтверждение

Хост: 03 8 значений/мм

Мышь: FA Подтверждение

Хост: E6 Установить масштабирование 1:1

Dev: FA Подтверждение

Хост: F3 Установить частоту дискретизации

Мышь: FA Подтверждение

Хост: 28 число 40

Мышь: FA Подтверждение

```
Хост: F4 Включить устройство
Мышь: FA Подтверждение

Если нажата левая кнопка мыши:
Мышь: 09 00001001 Бит0 - состояние левой кнопки мыши; Бит3
всегда = 1
Мышь: 00 Нет X-перемещений
Мышь: 00 Нет Y-перемещений
Мышь: 00 Нет Z-перемещений

... и отпущена левая кнопка мыши:
Мышь: 08 00001000
Мышь: 00 Нет X-перемещений
Мышь: 00 Нет Y-перемещений
Мышь: 00 Нет Z-перемещений
```

После установки драйвера IntelliMouse Microsoft поддержкой 4-х и 5-х кнопок, была найдена следующая последовательность:

```
... тоже самое, что и раньше
Хост: F3 Установить частоту дискретизации
Мышь: FA Подтверждение
Хост: C8 число 200
Мышь: FA Подтверждение
Хост: F3 Установить частоту дискретизации
Мышь: FA Подтверждение
Хост: 64 число 100
Мышь: FA Подтверждение
Хост: F3 Установить частоту дискретизации
Мышь: FA Подтверждение
Хост: 50 число 80
Мышь: FA Подтверждение
Хост: F2 Прочитать тип устройства
Мышь: FA Подтверждение
Мышь: 03 Идентификатор Мыши
Хост: F3 Установить частоту дискретизации
Мышь: FA Подтверждение
Хост: C8 число 200
Мышь: FA Подтверждение
Хост: F3 Установить частоту дискретизации
Мышь: FA Подтверждение
Хост: C8 число 200
Мышь: FA Подтверждение
Хост: F3 Установить частоту дискретизации
...
```

Заключение

Удалось выяснить принцип работы мышки – протокол обмена с компьютером. Благодаря полученной информации представилась возможность использовать мышь не только по своему прямому назначению, но и для своих поделок...

Самый простой пример – это прикрепление мышки под днище самодельного робота. А так как мы знаем протокол обмена, то становится возможным, например, сделать дешевый импровизированный «GPS навигатор» и ваш робот будет всегда знать, где он находится (а заодно шарики мышки будут использоваться как дополнительные колеса).

Другой пример. Подключаем мышку к тому же самому роботу и управляем им: двинул мышь вперед – робот поехал вперед, кликнул мышкой – робот выстрелил импровизированной ракетой.

Ресурсы

- Ресурс википедии http://ru.wikipedia.org/wiki/Компьютерная_мышь
- Мышь, которая всегда под рукой. Веб-архив журнала Наука и жизнь <http://www.nkj.ru/archive/articles/8810>
- Принцип работы компьютерной мыши http://www.molodinfo.n-vartovsk.ru/insite/mouse/mouse_tech.htm
- The PS2 protocol http://pcbheaven.com/wikispages/The_PS2_protocol
- Подключение PC AT клавиатуры к AVR <http://radioded.ru/content/view/53/49>
- Описание протокола PS/2 для мыши и клавиатуры <http://marsohod.org/index.php/ourblog/11-blog/57-ps2proto>
- Adam Chapweske. The PS/2 Mouse/Keyboard Protocol <http://www.computer-engineering.org/ps2protocol>
- Adam Chapweske. The PS/2 Mouse Interface <http://www.computer-engineering.org/ps2mouse>
- Обсуждение PC/Mouse <http://www.microchip.ru/phorum/read.php?f=2&i=6286&t=6286>
- AVR313: Interfacing the PC AT Keyboard. Rev. 1235B-AVR-05/02 http://www.atmel.com/dyn/resources/prod_documents/DOC1235.PDF

По своей работе я оказался в роли разработчика программы, работающей с базой данных. Самое основное в этом процессе – быстрый ввод информации в программу. Форма для ввода данных была создана еще до меня, и пользователи не один год работали с нею. Однажды, по стечению обстоятельств, мне пришлось сменить профиль и перейти из роли разработчика, в роль пользователя.



Продолжение. Начало цикла смотрите в предыдущем номере журнала...

Александр Демьяненко
by Grenles GRENLES@yandex.ru

Лишние мелкие движения

Одно дело, когда ты пишешь код и в течение нескольких минут его тестируешь, а другое дело, когда ты работаешь с ним. Повторюсь и поставлю акцент на слово «работать». Эти процессы весьма разнятся. Когда пишешь код – ты решаешь задачи реализации. Когда работаешь с исполняемым кодом – ты решаешь задачи эксплуатации. И вот в эксплуатации, как всегда и бывает, возникает множество вопросов и нюансов. Те самые нюансы и мелочи, которые так не любят доделывать разработчики*, считая этот процесс бесконечным и не нужным.

Число исписанных страниц с замечаниями в моем блокноте каждый день росло. Не скажу, что я исправил все ошибки и неточности. До сих пор мой список еще не исчерпан и тому есть разные причины. Однако, эту неудобную ошибку, на мой взгляд, я исправил. Вполне допускаю, что найдутся пользователи, которых устраивал старый вариант реализации ввода данных, но я их мнение проигнорировал, посчитав новый вариант лучшим.

Итак, вернусь к собственному примеру (см. рисунок 1). Обычный процесс ввода информации идет так: ввод даты, лицевого счета, данных и дальше все по кругу. Где же неудобство? На первый взгляд, его тут вовсе нет. Секрет кроется

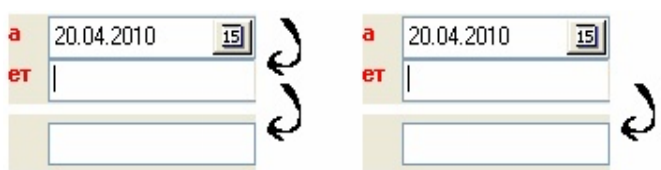


Рис. 1. Часть формы ввода информации в базу данных

в исходных данных. Дело в том, что дата изменяется только от листа к листу с данными, в пределах одного листа дата неизменна. Вот и выходит, что операция ввода даты в пределах одного листа – лишняя. Если на листе 30 записей, то надо выполнить, условно говоря, 3х30 действий. Выкинув циклический ввод даты, число действий можно сократить до 2х20+1 (первоначальный ввод даты).

Есть два решения этой проблемы. Первый – каждый раз оставлять дату неизменной и, при получении фокуса ввода в поле даты, пользователь должен нажимать какую-либо клавишу, чтобы пропустить ввод даты. Это чуть лучше, но не избавляет от лишних действий. Второй способ – единожды дать пользователю ввести дату и в последующие разы автоматически переключать фокус ввода только между полями лицевого счета и данных. При необходимости сменить дату пользователь может воспользоваться или мышью, или специальным сочетанием клавиш.

Кажется, это все мелочи. Но наша жизнь как раз и складывается из разных мелочей, а если их предусмотреть и учесть, то пользователю будет удобно и приятно, да и ваш статус в их глазах заметно возрастет.

Проблема безусловной интерпретации данных [1...5]

Всем известен Microsoft Excel, входящий в состав Microsoft Office. Ни для кого не является секретом то, что эта программа стала де-факто основным средством ведения учета и документооборота в различных компаниях и

*** Комментарий автора.**
Самое главное умение хорошего разработчика – уметь признать и исправить собственные ошибки. Впрочем, это утверждение справедливо в целом и для человека, как личности.

фирмах. Укажу на одно неудобство, заложенное в эту программу, которое меня раздражает тем, что никто не удосужился спросить: «а надо ли мне это?». Не буду стопроцентно утверждать, но та же проблема есть и в Open Office, который является бесплатной альтернативой платному продукту от Microsoft.

У компании Microsoft есть одно решение в части интерпретации вводимых данных. Думаю, многие сталкивались с ситуацией, когда при вводе данных в ячейку вида «9/12» или «9-12», программа Excel автоматически переводит эти данные в формат даты (см. рисунок 2). Раздражает отсутствие вариантов: перевод происходит автоматически (о том, как это исключить, напишу ниже). За меня кто-то решил, что надо поступать только так, а не иначе. Честно скажу, я пытался найти опцию в настройках как-то разрешающую это действие, и не нашел.

В зависимости от настроек предпочтений, результат отображения даты может быть различным, как это и показано на рисунке 2. Для случаев 1, 2, 3 явно видно, что программа интерпретирует

данные, как дату. При высоком темпе работы, усталости оператора, большом объеме данных эту «ошибку» можно запросто пропустить. В дальнейшем она может вызвать совершенно непредсказуемые неприятности при обработке данных, например, потеря точности, если предположить, что данные мог вводить один человек, а использовать их - другой. У второго человека может не быть информации, что «9 декабря» - это не дата, а «9/12». В случае 4 на рисунке 2 (скажу сразу, я сделал его искусственно, но между тем, такой вариант возможен), кажется, все в том же виде, как и введено, и такую ситуацию тем более можно пропустить.

На самом же деле «9.12» - это не число, а дата «9.12.2010», просто в этом формате на экране год

не отображается. Допустим, что пользователь заметил ошибку и решил ее исправить. Пользователь у нас все-таки человек умный и не один год работает за компьютером, то есть не «чайник» и не совсем уж глуп - он некий «среднячок», имеющий навыки и опыт. Поэтому он решил сделать обратную операцию - «вернуть формат». Что он увидит, думаю, опытные пользователи и программисты догадаются: вовсе не то, что хотел увидеть в результате (см. рисунок 3).

Как видно, пользователь получил странное число «40521», а не «9/12» как бы ему того ни хотелось. Почему? Программисты знают ответ: а потому, что кто-то за нас при разработке программы решил, что это число дней, прошедшее с даты «01.01.1900». С точки зрения такой логики, программа сделала все верно и сделала из даты - число. Но только тут кроется еще одно умолчание, которое и ошибкой не назовешь, но в данном случае - это ошибка, так

как ожидания пользователя естественным образом обманывает.

Самое интересное во всей этой ситуации с превращениями, «9/12»

и не число, и не дата, а ТЕКСТ (!). В данном случае есть как минимум два выхода из этой ситуации. Либо задавать пользователю вопрос: «Интерпретировать данный текст как дату?», либо где-то в настройках программы сделать «флажок», разрешающий эту ситуацию.

Ручное решение данной проблемы таково - это сделать ячейки, куда предполагается ввод данных подобного вида заранее «текстового» формата. Как известно, «текстовый» формат Excel никак не интерпретирует. Сложность может возникнуть в том случае, когда заранее неизвестно, в какую ячейку могут быть введены данные подобного вида. Если

	40521	

Рис. 3. Итог попытки «вернуть формат назад»

9/12	9/12	9-12	9-12
↓	↓	↓	↓
09.дек	дек.10	9 дек	9.12
1	2	3	4

Рис. 2. Интерпретация данных MS Excel до и после ввода

сделать все ячейки «тестом», то это плохо для данных формата «числовой», так как ни одна математическая операция их не воспримет как число, а выдаст ошибку.

Казалось бы, одно удобство, а столько сложностей! Не открою никакого секрета и в этот раз, если скажу, что большинство программ в качестве промежуточного формата для обмена данными использует формат файла MS Excel. Разработчики, зная такую особенность этой программы, заранее делают все ячейки текстового вида (правда это делают грамотные и уважающие пользователя разработчики, некоторые забывают) и уже потом выгружают данные в файл. Точность данных сохраняется, но удобства не прибавляется. И, к тому же, возникает задача преобразовать формат ячейки «текстовый» в формат ячейки «числовой» для данных, являющихся числами, чтобы дать возможность пользователю выполнять математические операции над числами. В итоге безусловное «удобство» оказывается источником множества лишних действий.

Интерпретация как таковая

Обобщу проблему интерпретации, но перед этим скажу, как она касается интерфейса. Известно, что человек не менее 80% информации воспринимает с помощью зрения. Таким образом, мы сначала видим что-то, а потом неосознанно запускается процесс сопоставления увиденного с чем-то ранее известным. В случае, когда подсознательно сопоставить не получается, запускается процесс осознанного сопоставления, или, иначе говоря - логической интерпретации данных. Значит то, что нам покажут и что мы увидим весьма важно, а это как раз и есть интерфейс.

Возьмем текст «14-12-10». Что это такое? Мне кажется, большинство ответит: «это дата 14 декабря 2010 года». А почему? Например, потому, что выше я вел речь о дате и в нашей кратковременной памяти остался контекст, в котором ключевым словом была «дата». А так как другого контекста нет, то мозг его автоматически распространил на новые увиденные данные.

Итак, воспринимаемая информация сильно зависит о того, в контексте какой другой информации мы ее узнаем. Но ведь эти цифры могут являться номером телефона, кодом для сейфа, частью ряда убывающей арифметической последовательности. Интерпретаций можно придумать множество.

Вернемся к обсуждению примера с датой. Я немного подкорректирую цифры, чтобы было «удобно» поиграть ими: заменяю первую «1» на «0». Итак, имеем дату «04-12-10». Применительно к компьютерной технике, задам другой вопрос: «Что это за дата?». На первый взгляд это «4 декабря 2010». Но ведь может быть и «10 декабря 2004 года». Ответ зависит от того, как в программе заложена интерпретация входных данных. Есть несколько нотаций даты, при этом «число-месяц-год» могут меняться местами как угодно. Мало того, если поглядеть представления, какие нам предлагают различные программы, то задача интерпретации данных без контекста вообще не решается. Приведу всего лишь два формата, в котором все представлено двумя числами:

1) ДД.ММ.ГГ

2) ГГ.ДД.ММ

Может быть «день-месяц-год», а может быть «год-день-месяц». Хотя в большинстве случаев принята нотация «день-месяц-год», никто не застрахован от подвохов. Пользователь привык к тому, к чему его когда-то «приучили», и не обязательно вы. Вас тоже к чему-то когда-то кто-то приучил, да еще так, что некоторые вещи вы считаете безусловными и сами собой разумеющимися.

Как дополнительную информацию, полностью процитирую источник [6]**. Приручая пользователя к чему-либо, подумайте, что он об этом думает.

Стандартные сообщения об ошибках. Это...

Прошли те времена, когда каждый придумывал свои форматы, методы сопряжения, способы обмена и решал вопросы совместимости своих данных с данными других коллег. Все эти сложности решила стандартизация.

Стандартизация – это способ согласованного взаимодействия разработчиков между собой. В большинстве случаев это очень хорошо. Но, если задуматься, это же одновременно и источник различных проблем. Почти всегда и все отступают и отклоняются от стандартов в угоду различных причин. И, как часто это бывает, о том, что было отступление от стандарта либо умалчивают, либо забывают сообщить (что, по сути, одно и то же).

Долго думал, какой же пример привести, чтобы он был понятным и наглядным. Были соображения сравнить две версии MS Office 2003 и MS Office 2007 в плане значительном смены интерфейса и появления новых элементов. В определенной степени этот пример подходит под заданную тему «стандарты», но все же не наглядно. А кто сказал, что отображение элементов интерфейса в одной программе есть стандарт и он неизменяем много лет подряд. Пожалуй, также подумали разработчики MS Office и кардинально изменили интерфейс своих приложений.

Были мысли поговорить о цветовых схемах различных программ, но это тоже не в тему. Во-первых, если разобраться, то практическая любой уважающий себя разработчик дает возможность пользователю сменить «скин» («оболочку», внешний вид) своей программы, в том числе шрифты и цвета отображаемых элементов. Собственно говоря, идея была раскритиковать программы, где это сделать невозможно, а подбор и сочетание цветов неприятны при использовании программы, но таковых сейчас очень мало или почти уже нет. Даже на некоторых «продвинутых» сайтах и то есть возможность смены оболочки.

В итоге решил написать вот на что. Как известно, что предугадать и исправить все возможные

ошибки, нереально. В современных языках программирования есть различные средства, для выхода из ошибочных ситуаций, которые не были предусмотрены разработчиком. Одно из них – обработка исключений. Это хорошее средство, шаг навстречу пользователю. Даже если разработчик забыл где-то вставить отработку возможной ошибки, за него это сделает операционная система. Но, в данном случае, речь немного о другом. Точнее сказать, взгляд под другим углом зрения. Обработчик ошибочной ситуации выводит на экран сообщение об ошибке, в котором указывается какая-то информация об ошибке. То есть, отработка ошибочной ситуации вывод информации – это уже стандарт. Сложность в другом: каждый делает это по-своему.

**** Комментарий автора.**

...приведу реальный пример, как рекламный бюджет выкидывается псу под хвост (нам этот случай рассказывали на семинаре московские рекламисты). Крупная компания-производитель зубной пасты решила выйти на рынок в арабские страны и дала задание агентству – найти самый эффективный рекламный ход из уже запущенных, чтобы затраты были по минимуму. Агентство выдало грамотное обоснование – по всей Европе стоят рекламные щиты, на которых нет ни одного слова. значит, ничего переделывать не надо, а надо просто заказать их побольше и поставить в арабских странах. Компания идею одобрила. На эти цели были выделены порядка миллиона долларов.

Что из себя представлял рекламный щит? Всего три фотографии: на первой молодой парень улыбается желтыми зубами, на второй – чистит зубы, на третьей – улыбка белоснежная. И все это на фоне рекламируемой зубной пасты. Установили щиты в арабских странах. Продаж – ноль! Заказали исследование компании по мониторингу – почему нет продаж. Ответ был быстр – рекламные щиты не возымели своего действия потому, что в арабских странах читают справа-налево. Думаю, смысл всего сказанного достаточно прозрачен.

Возьмем, например, сеть Интернет. Все пользователи рано или поздно сталкиваются с сообщением об ошибке 404, которое появляется в том случае, когда запрашиваемая страница не найдена. Это логично, когда нет запрашиваемого источника – надо выводить об этом информацию. В Интернет есть даже целый сайт, посвященный

тому, что на нем собраны все возможные варианты ответов для пользователя с сообщением об этой ошибке. Пока все верно и правильно.

Теперь рассмотрим, по сути, даже не важно какую, программу, в которой может возникнуть любая ошибка. Как известно, в исключительной ситуации пользователю будет выдано сообщение об ошибке. В лучшем случае, будет указан номер ошибки и поясняющий текст, в худшем – просто номер ошибки, а там – догадывайся сам, как хочешь.

Пока все вписывается в стандарт – пользователь

информируется. Ошибка в этом случае совершенно другая, и заключается она как раз в том, что пользователь информируется об ошибке. Это стандартно и признак хорошего тона – информировать об ошибке, но, давайте перейдем на сторону пользователя. Мне сообщили об ошибке, написали ее номер и текст. И что мне делать дальше? Закрыть сообщение и аварийно завершить программу. При повторном выполнении тех же действий снова случится та же ошибка. А мне, как пользователю, необходимо выполнить некоторые действия с этой программой, а разработчик не дает мне такой возможности.

С моей, пользовательской, точки зрения, сообщение об ошибке означает то, что мне в красивой форме говорится о том, что я глуп. А если подумать еще чуть-чуть, мне это ни о чем не говорит, кроме того, что разработчик не захотел позаботиться обо мне. Какое мне дело, что произошло деление на ноль или переполнение стека, или страница не найдена? Мне нужен конечный результат, которого я хочу достичь, используя предлагаемые разработчиком средства, а сообщения об ошибке мне только мешают и прекращают мой путь к достижению цели.

Если задуматься над этой задачей, то решение может быть таким. Конечно же, есть ошибки, которые просто неизбежны, фатальны, и ничего с ними нельзя сделать. Отказ оборудования, отсутствие страницы на сайте, сбой локальной сети и прочее – эти причины предвидеть сложно. Но нужно постараться сделать так, чтобы у программного продукта было предусмотрено свое решение этих проблем, помимо стандартного сообщения об ошибке.

При отсутствии страницы на сайте можно предложить пользователю перейти на другие страницы, или предложить другой способ поиска нужной информации. А серьезно, то на сайтах не должно быть устаревших и отсутствующих страниц.

Если это программа, то, например, обрыв сетевого соединения не должен вызывать потерю данных, особенно, если идет закачка из сети большого по объему файла. Кстати, до сих пор

этой болезнью болеет Internet Explorer: при обрыве соединения или таймауте (времени, в течение которого источник не отвечает), при повторной попытке файл начинает закачиваться сначала, а не с места обрыва. Наверное, в том числе и это стало одной из причин появления программ-загрузчиков информации из сети. В FireFox такой проблемы нет, браузер при обрыве соединения предлагает докачать по возможности файл. То есть, разработчик должен предусмотреть не просто вывод сообщение об ошибке, а реакцию, действие в результате ошибочной ситуации.

Кстати, еще одна смешная ошибка, которую выдает операционная система: «...программа выполнила недопустимую операцию и будет закрыта». Если вдуматься, абсолютно бессмысленное сообщение. Во-первых, я даже и не предполагал, что программа выполняет какую-то операцию, тем более кем-то и зачем-то недопустимую. Не буду рассуждать долго на эту тему***, так как мне могут возразить, что если разрешить дальнейшее выполнение этой программы, то это приведет к потере информационной безопасности и вообще краху системы в целом.

В итоге, у меня возник вопрос: «...почему во многих программах, кроме операционной системы не предусмотрен запуск в безопасном режиме, который гарантированно и железно работает?». А вместо этого, выдается стандартное сообщение об ошибке. По крайней

*** Комментарий автора.

В данном случае просто приведу конкретный пример. Internet Explorer, вдруг, перестал запускаться и постоянно, через некоторое время после запуска, операционная система закрывала программу с сообщением о страшной ошибке с недопустимой операцией. Некоторое время я ломал голову - в чем дело. Даже по глупости переустанавливал программу. Как ни странно - не помогло. Потом я решил сделать вот что. Программа при запуске показывает рамку окна запускаемой формы с пустой внутренней областью, и только потом появляется все внутреннее наполнение окна. В момент, пока это наполнение не появилось, я, нажав правую кнопку мышки, вызвал контекстное меню и стал отключать установленные дополнительные панели.

Не спору, что есть другие способы решения этой проблемы. Я выбрал именно такой, пожалуй, средний пользователь сделал бы также. В итоге, отключив какую-то панель, я увидел, что приложение замечательно запустилось и заработало без ошибок, как и раньше.

мере, это можно решить так: фиксировать число запусков и закрытий самым первым действием при запуске. Если это число не совпадает определенное число раз (не исключен случай принудительного закрытия программы из диспетчера задач), то запускать программу в «безопасном режиме» и пытаться искать причину «поломки».

Итак, думаю понятно, что просто сообщить о собственно ошибке пользователю – это ничего не сделать. Гораздо лучше сделать и ничего об этом не говорить, а молча про себя улыбнуться и подумать: «Какой я молодец!».

Заключение

Разговор об интерфейсе я только начал. И только этой одной статьей не ограничусь, тем более, что я больше не сказал, чем сказал, а объем получился большим. Что сказать и написать у меня еще найдется. Конечно же, огромную помощь в написании статьи я получаю из книг и из различных источников, в том числе и Интернет. По возможности, я указываю оригинальные источники, тем не менее стараюсь пересказывать своими словами ту информацию, которую узнал, чтобы это не было глупым плагиатом. До встречи в следующих статьях...

С уважением, Демьяненко Александр.

Ресурсы

- Об оформлении программной документации <http://www.raai.org/about/persons/karpov/pages/ofdoc/ofdoc.html>
- ГОСТ 19.102-77. Стадии разработки <http://www.nist.ru/hr/doc/gost/19102-77.htm>
- Студия Артемия Лебедева. Создание интерфейса навигатора «Штурман» <http://www.artlebedev.ru/everything/shturmann/process>
- Этот мерзкий, неудобный, противоестественный оконный интерфейс <http://epikoiros.narod.ru/public/antiwind.htm>
- Обзор эргономических проблем и недостатков пользовательского интерфейса ПО бухгалтерского учета на примере 1С: Предприятие 7.5 <http://www.usability.ru/>

[Articles/1.htm](#)

- Ссылка на отдельное сообщение форума <http://forums.drom.ru/1067823597-post12.html>

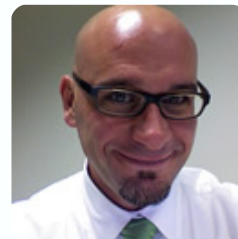
АНОНС НОМЕРА: Конференция по юзабилити User Experience Russia 2010 пройдет в октябре в Москве.



Сообщество UXRussia при поддержке европейской ассоциации юзабилити UPA Europe приглашает 6 – 8 октября в Москве на четвертую ежегодную конференцию User Experience Russia. В этом году конференция посвящена вопросам юзабилити, проектированию и взаимодействию с пользователем услуг в интернете.

В рамках конференции 6 октября состоятся мастер-классы от международных гуру юзабилити – Эрика Райса (FatDUX Group) и Ронни Баттиста (UPA International).

Ронни Баттиста представит мастер-класс «Мастерская UX: говори и показывай» (UX Show and Tell). Это новый для России формат, в рамках которого участники могут рассказать про свои интересные проекты, наработки, проблемы, или гениальные мысли, а также про все, что может показаться интересным другим. Выступление каждого обсуждается всеми присутствующими и самим Ронни Баттиста. UX Show and Tell — это место, где можно попрактиковаться, критиковать и отражать критику.



Эрик Райс является одним из самых влиятельных фигур в области проектирования и информационной архитектуры. Его выступления вдохновляют многих и многих UX-специалистов, а признанные эксперты отмечают его яркие и уникальные доклады. Действительно, работы Эрика Райса являются поистине неоценимым вкладом в мировое развитие юзабилити. Эрик проведет мастер-класс «Пишем для Web» ("Writing for the Web") и поделится опытом создания контента для сайтов, который легко и удобно находить, просматривать и читать. Подобные вещи создают понимание, вызывают доверие и увеличивают показатели конверсии. Почему Нобелевская премия по литературе не вручается дизайнеру суперобложки. Потому что контент остается единственным главным элементом.

Основная часть конференции пройдет 7 и 8 октября. Всего в программе конференции ожидается 40 докладов ведущих экспертов области, а также множество подарков и сюрпризов.

<http://userexperience.ru>

Рано или поздно в жизни программиста появляется заказчик с неким проектом. Разработчик берется за этот проект, но каждая последующая встреча с заказчиком заканчивается появлением новых функциональных и нефункциональных требований, сроки сдачи остаются прежними, бюджет тоже, все наваливается снежным комом и наступает коллапс. Для того, чтобы такая скверная ситуация не наступила, с заказчиком нужно договориться еще на берегу. Но нельзя полагаться на какие-то словесные договоренности и т.п.: мы должны максимально себя обезопасить. Разрабатывая программу, мы преследуем определенную цель, ставим перед собой определенные задачи. Успех нашего дела во многом зависит от того, насколько наша цель будет четкой и ясной. Знакомясь с выполненной работой, заказчик, естественно, будет анализировать выполнили ли мы поставленные задачи, поэтому все они должны быть четко сформулированы, описаны и согласованы с ним. Вот тут нам на помощь и приходит техническое задание (далее ТЗ).



Дарья Устюгова

by Sparky ustugova90@mail.ru

В цикле статей, основываясь на своем опыте и на опыте других, я попробую рассказать, что такое ТЗ, зачем оно нам нужно, из чего состоит, как его написать и оформить, какие подводные камни можно обойти, написав грамотное ТЗ. Сегодня, я постараюсь объяснить, зачем ТЗ используется, в чем его плюсы, кто его должен писать и приведу примерное содержание ТЗ.

Техническое задание. Что это и зачем оно нам [1...5]?

Начнем мы с определения: техническое задание (ТЗ) – это точная формулировка задачи. Можно даже сказать, что ТЗ всему голова. Многие слышали о таком понятии, как жизненный цикл программного обеспечения (далее ПО). Для тех, кто не слышал, а быть может просто забыл, напомним: жизненный цикл ПО, системы процедур, правил и инструментальных средств, используемых для разработки и поддержания работоспособности программной системы. Жизненный цикл включает в себя этапы создания программы. Существует множество версий: по Глассу, по ГОСТ, ЕСПД, по Майерсу. Все эти версии имеют общие черты и различия. Нас в них будут интересовать лишь 2 этапа. В некоторых версиях, например в ГОСТ и ЕСПД, эти этапы объединены в один. Это идея и ТЗ: иногда вместо ТЗ можно встретить требования, но это одно и то же.

У заказчика появляется идея создать программу

для решения какой-то бизнес-задачи, он приходит к нам. Мы «достаем» из него то, что он в итоге хочет. Как правильно это сделать – отдельный разговор, я бы даже сказала – искусство. После того как мы поняли, что заказчик хочет, мы приступаем к созданию ТЗ. Почему именно мы будем использовать ТЗ? Как я уже сказала, первым кирпичиком нашей успешно выполненной работы является точная формулировка задачи. Используя ТЗ, мы как раз формулируем то, что нужно заказчику, в довольно наглядной и формальной форме. Кроме того, мы получаем некий план наших действий. Согласитесь, когда есть план, работать проще. Еще раз повторюсь: избегайте словесных договоренностей, тетрадок с записанными функциональными и нефункциональными требованиями. Запомните, нам с вами нужна официальная бумага. Как говорится: «без бумажки ты ...». ТЗ можно не использовать, когда мы пишем программу для себя, но когда мы пишем программу для кого-то и за определенное вознаграждение, то ТЗ нам жизненно необходимо.

Для кого ТЗ?

Многие из читателей – студенты. Я, кстати, тоже. Для них пара строк отдельно. Всем нам предстоит писать курсовые, дипломные и прочие работы, и чтобы ваш руководитель знал, что именно в итоге вы должны сделать, сразу же представьте ему ТЗ. Поверьте, он будет доволен, заодно подскажет вам, что в нем нужно исправить и т.д. Благодаря этому получите хороший опыт. Лучше, чтобы на вас побурчал руководитель, чем дать возможность наживаться

на вас реальному заказчику.

Нужно отметить, что помимо ТЗ существуют и другие документы, применяемые в объектном подходе при проектировке систем. В частности, на практике я встречалась с технологий RUP*. В ней для формулировки задачи используется не один документ, а целый набор (Vision, Search, UseCases и т.д.). Почему же я использую ТЗ, а не спецификации RUP? Потому, как ТЗ более известно в нашей стране, писать его намного проще. Особенно, если вы непрофессиональный проектировщик и аналитик. Спецификации RUP более подходят для крупных фирм, где есть аналитики, проектировщики и для фирм, работающих на иностранных заказчиков. Если же вы работаете в одиночку или маленьким коллективом, ТЗ подходит вам больше. Для того, чтобы использовать RUP, нужно хорошо знать объектный подход к проектированию и UML*.

Итак, вы решили, что будете писать ТЗ. А тут заказчик приносит нам уже готовое ТЗ. Вы тут же начинаете радоваться, думая: «Ура, минус одна ненужная бумажка! Пусть сам тратит время и пишет ее». И тут вы совершаем ошибку. В чем же она заключается? Однажды я сама встретила с таким заказчиком и получила от него ТЗ. Так как фирма была небольшая, людей обученных писать ТЗ там не было. Я получила документ совершенно не похожий на ТЗ, там не было ничего близкого к ТЗ. Это был документ, состоящий из перечня полей, которые должна содержать БД: ни сроков начала и сдачи программы, ни функциональных требований...

Пришлось переписывать и доказывать заказчику, что он не прав и это не ТЗ. Поэтому, если в роли заказчика – маленькая фирма, сразу же договоритесь с ним, что ТЗ вы берете на себя и потом с ним согласуете. Но есть более крупные заказчики, которые все-таки составляют ТЗ

грамотно и тут нельзя расслабляться, так как любой заказчик хочет получить хорошую программу за минимальные деньги и в короткие сроки. И если вы внимательно не проанализируете все ТЗ, не решите спорные моменты, то вас загонят в угол и вы не будете рады новой работе. Поэтому мой вам совет: пишите ТЗ сами. Даже, если заказчик придет с готовым, внимательно прочтите его и не беритесь за работу, пока не решите все спорные моменты.

Сейчас немного пофантазируем: допустим, что у вас есть своя фирма по созданию ПО. Советую вам не пожалеть денег и нанять человека, который может грамотно и быстро писать ТЗ: он сэкономит вам время, деньги и нервы.

Вот мы подошли к еще одному плюсу ТЗ. ТЗ – документ официальный, на нем должны стоять ваша подпись, подпись заказчика и дата, а это очень важно! Если у заказчика к вам будут какие-то претензии (допустим он заявил, что вы реализовали не все необходимые функции), то вы сможете ему предъявить ТЗ с описанием всех функций и доказать, что он не прав. А если, не дай Бог, он пойдет в суд, то ТЗ будет отличным аргументом в вашу пользу.

Но плюс в нашу сторону может обернуться и минусом: подписывая ТЗ, мы возлагаем на себя определенные обязательства. И если функциональность, сроки сдачи, качество программы не будут совпадать с описанным в документе, нам придется нести за это ответственность. Так что для заказчика ТЗ – это гарантия качества и отражение того, что он хочет видеть в нашей программе.

Как же написать ТЗ?

Мы с вами не первые и не последние, все придумано до нас. Кроме того, все уже стандартизировано: ГОСТ 19.201-78 (техническое задание, требования к содержанию и оформлению). Да, согласна, ГОСТы читать неинтересно, но нужно себя к этому приучить, зачастую именно с этого и нужно начинать. Если у вас после чтения ГОСТа вопрос «как же писать» сохранился, не расстраивайтесь, у меня было тоже самое. ГОСТ не дает ответ на вопрос

* Комментарий редакции.

RUP (Rational Unified Process) – методология разработки программного обеспечения, созданная компанией Rational Software
http://ru.wikipedia.org/wiki/Rational_Unified_Process.

UML (англ. Unified Modeling Language – унифицированный язык моделирования) – язык графического описания для объектного моделирования в области разработки программного обеспечения
<http://ru.wikipedia.org/wiki/UML>.

как писать, он лишь отвечает на вопрос, как оформить ТЗ и что должно туда входить. Мне, для того чтобы ответить на этот вопрос, пришлось прочитать довольно много вспомогательной литературы, искать примеры ТЗ и т.д.

Я слышала еще и о программах, которые позволяют автоматизировать процесс написания ТЗ. Но я ими не пользовалась: так как лучше помучаться самому, особенно если пишешь впервые. Потому как, используя программы для автоматизации, можно чего-нибудь не учесть, вручную все же надежнее, да и получаешь огромный опыт.

А теперь о самом ценном на сегодня. Написав уже не одно ТЗ, просмотрев множество примеров, я сформировала свой подход к содержанию** ТЗ, основывающееся на ГОСТ-е.

Вместо эпилога

Именно на нем я и буду основываться в следующих статьях. В ближайшем номере я расскажу вам, как правильно написать первые пять пунктов и на какие подводные камни можно наткнуться при их написании.

Примеры реальных ТЗ, методических инструкций по этапам ТЗ приложены в виде ресурсов в теме «Журнал клуба программистов. Шестой выпуск» или непосредственно в архиве с журналом [5].

Продолжение следует...

Источники

- ГОСТ 19.201-78. Техническое задание, требования к содержанию и оформлению <http://www.nist.ru/hr/doc/gost/19201-78.htm>
- ГОСТ 19.106-78. Требования к программным документам, печатным способом <http://www.nist.ru/hr/doc/gost/19106-78.htm>
- Наталья Дубова. В круге разработки <http://citforum.ru/SE/project/circle>
- Единая система программной документации (ЕСПД) <http://www.philosoft.ru/espd.zhtml>
- Примеры ТЗ и инструкций <http://procoder.info>

** Содержание ТЗ:

1. Введение
 - 1.1 Наименование программы
 - 1.2 Краткая характеристика области применения программы
 - 1.3 Сроки исполнения работ
2. Основания для разработки
 - 2.1 Заказчик
 - 2.2 Исполнитель
 - 2.3 Основание для разработки
3. Назначение разработки
 - 3.1 Общая концепция системы
 - 3.2 Описание функциональности системы
4. Требования к программе
 - 4.1 Требования к информационным структурам и методам решения
 - 4.2 Требования к функциональным характеристикам
 - 4.3 Требования к надежности
 - 4.3.1 Требования к обеспечению надёжного функционирования системы
 - 4.3.2 Типы отказов при работе системой
 - 4.3.3 Время восстановления после отказа
 - 4.3.4 Допустимые потери данных при отказе
 - 4.3.5 Важная информация, которая должна быть защищена от разрушения
 - 4.3.6 Отказы из-за некорректных действий пользователей системы
 - 4.4 Условия эксплуатации
 - 4.4.1 Климатические условия эксплуатации
 - 4.4.2 Требования к квалификации и численности персонала
 - 4.5 Требования к составу и параметрам технических средств
 - 4.5.1 Требования к серверному аппаратному обеспечению
 - 4.5.2 Требования к клиентскому аппаратному обеспечению
 - 4.5.3 Требования к сетевому аппаратному обеспечению
 - 4.6 Требования к информационной и программной совместимости
 - 4.6.1 Требования к исходным кодам и языкам программирования
 - 4.6.2 Требования к программным средствам, используемым программой
 - 4.6.3 Требования к защите информации и программы
 - 4.7 Маркировка и упаковка
 - 4.8 Транспортировка и хранение
 - 4.9 Специальные требования
5. Требования к программной документации
6. Техничко-экономические показатели
7. Стадии и этапы разработки
 - 7.1 Стадии разработки
 - 7.2 Этапы разработки
 - 7.3 Содержание работ по этапам
8. Порядок контроля и приемки
 - 8.1 Виды испытаний
 - 8.2 Общие требования к приемке работ
9. Порядок корректировки Технического задания

Компьютер в настоящее время для большинства уже перестал быть чем-то диковинным. Он используется и для развлечений, и для работы, а среди его пользователей школьники и студенты, любители и профессионалы, люди разных возрастов и профессий. Идея использовать компьютер, а точнее компьютерные программы для обучения и развития, естественно, родилась и реализуется уже давно. И вот, видя, с каким интересом мой маленький внук (а было ему тогда всего лишь 3.5 годика) наблюдает, как я работаю за компьютером, и как он старается тоже подвигать мышкой или понажимать клавиши, я понял: пора использовать его интерес для развития. Первым делом, конечно, поискал в Интернете программы, обучающие ребенка счету, рисованию и раскраске простеньких рисунков, знакомству с азбукой. Увы, результат оказался плачевным. Большинство программ, найденных мной в Интернете, или платные, или настолько сложные для ребенка, что и взрослому не всегда удастся разобраться, как ею пользоваться. Так и появилась идея самому создать развивающие программы для маленьких с учетом интересов и способностей самого ребенка. Вскоре появились красочные программы «Раскраска» и «Азбука», очень понравившиеся малышу. Яркие рисунки с любимыми игрушками и персонажами сказок и мультфильмов, простота управления программой – все это привлекает ребенка, прививая ему первые навыки координации движений, правильного выбора действий.



Владимир Дегтярь

by DeKot degvv@mail.ru

Целью данной статьи является помощь начинающим программистам в создании полноценных программ для маленьких детей. Вы можете просто скачать из приложения к журналу файлы *.exe (не забыв про ресурсы программы – папку Image с рисунками) и предложить поиграть своим маленьким пользователям.

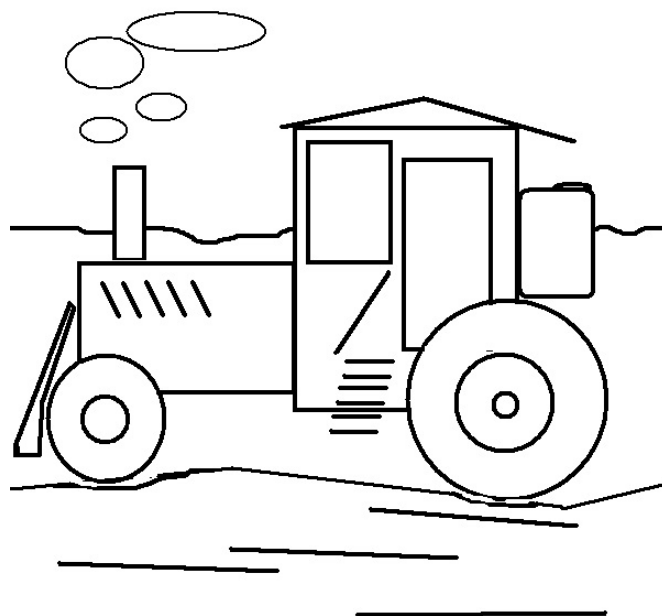
В данной статье рассмотрим создание двух небольших программ, которые наверняка понравятся малышам. Думаю, что тем, кто хоть немного знаком с компьютером (тем более с программированием), не составит большого труда заменить или добавить рисунки, более подходящие для конкретного ребенка.

Программа «Раскраска»

Программа* представляет из себя, в прямом смысле, обычную раскраску, где ребенок выбирает понравившийся рисунок и раскрашивает его выбранными им же цветами.

* Комментарий автора.

Сразу отмечу, что данная программа обсуждалась на форуме <http://programmersforum.ru/showthread.php?t=80497&highlight=%D0%E0%F1%EA%F0%E0%F1%EA%E0>. Форумчанин **гахр** устранил некоторые ошибки и добавил ряд красивых рисунков, а **mutabor** заменил стандартный курсор на симпатичную кисточку.



Прежде чем приступить к самому процессу программирования, определимся со структурой программы. А именно, что же мы хотим получить от программы:

1. Все действия будем производить на форме, точнее на канве формы. Следовательно, будем использовать в программе свойства и методы работы с канвой формы – вывод рисунков на форму (методы [Draw](#), [StretchDraw](#)), заливку области канвы цветом (свойства кисти [Brush.Color](#), метод заливки [FloodFill](#) и т. п.).

2. Нам понадобятся рисунки для раскраски. При этом каждый рисунок будет иметь два вида. Один

– это полноцветный рисунок, используемый для выбора и для подсказки (чтобы ребенок видел, каким цветом закрашивать фрагменты рисунка). Второй – непосредственно сам выбранный рисунок для раскраски в виде контуров фрагментов рисунка. Все контуры должны образовывать замкнутую область.

3. К выбору рисунков следует подходить индивидуально для каждого ребенка. Кому-то нравятся машинки, самолеты, кораблики, кому-то – персонажи сказок и мультфильмов. Самым маленьким вполне достаточно простейших рисунков – солнышка, шарика, звездочки и т. п. Множество рисунков можно найти в Интернете. Это и готовые рисунки для раскрасок, и просто красочные картинки с различными персонажами.

4. Еще нам потребуется набор палитры цветов. Естественно, что применение, скажем, стандартного компонента выбора палитры цветов вряд ли будет интересно для ребенка. Куда привлекательней, если он будет выбирать краску из разноцветных ведерочек.

Все действия в программе производим при помощи мыши – курсором, а затем кнопкой мыши выбираем или меняем рисунок, цвет заливки, закрашиваемую область рисунка. Вместо стандартного системного курсора будем использовать самодельный в виде кисточки.

Вот, в принципе, и все основные пункты структуры программы. Итак, начнем постепенно создавать саму программу...

Предварительно следует подобрать рисунки для раскраски. Для получения из цветного рисунка контурного для раскрашивания можно применить простой способ. Откройте цветной рисунок в графическом редакторе Paint, а затем сохраните его как черно-белый рисунок. При необходимости можно залить области рисунка белым цветом и подправить контурные линии,

**** Комментарий автора.**

Казалось бы, детской ручке тяжело манипулировать мышкой (даже если подобрать мышку самых маленьких размеров). Однако мой внук быстро сообразил, что удобно передвигать мышку и управлять курсором одной рукой, а второй нажимать кнопку мыши в нужном месте.

чтобы получить замкнутые области. Все рисунки храним в папке Image в файлах формата bmp, которую расположим в папке с проектом. Для выбора рисунка выводим на форму одновременно 12 рисунков. При смене рисунков выводим следующую группу из 12 рисунков. При этом совсем необязательно, чтобы размеры рисунков были одинаковыми. При выводе рисунков на форму будем подгонять размер рисунка до необходимого. Файлам цветных рисунков, для удобства обработки в коде программы, присвоим цифровые имена. Для первой группы цветных рисунков – [1.bmp... 12.bmp](#), для второй – [21.bmp... 32.bmp](#), для следующей – [41.bmp... 52.bmp](#) и т. д.

Файлы черно-белых рисунков для раскрашивания соответственно имеют имена [p1.bmp... p12.bmp](#), [p21.bmp... p32.bmp](#), [p41.bmp... p52.bmp](#). Здесь же находятся еще два вспомогательных файла – рисунок пока пустого ведерка для краски [col.bmp](#) и изображение курсора в виде кисточки [brush.bmp](#).

Окно программы состоит из трех частей (см. рисунок 1):

- Область выбора рисунка. Здесь расположены 12 цветных рисунков (размером 100x100 пикселей) для выбора и кнопка смены рисунков (в виде зеленого треугольника).
- Область выбора цвета – 15 разноцветных ведерок с красками.
- Область выбранного рисунка для раскраски. Здесь выводится выбранный рисунок (размером 600x600 пикселей).

Определимся, какие процедуры нам понадобятся для реализации программы. Так как все манипуляции в программе осуществляются при помощи мыши, будем использовать стандартные обработчики [FormMouseDown\(\)](#) и [FormMouseMove\(\)](#), а также процедуру обработчика событий [FormClick\(\)](#). Вывод всех рисунков, а также курсора в виде кисточки будем производить в стандартных процедурах для окна формы [FormShow\(\)](#) и [FormPaint\(\)](#). Еще две стандартные процедуры: [FormCreate\(\)](#) используется для инициализации приложения после запуска, [SetBrushColor](#) – для окрашивания кисточки-курсора в выбранный цвет.

Две процедуры мы создадим сами. Одна из них – процедура выбора рисунка `SelPicture` для считывания цветных рисунков из файлов папки `Image`. Данная процедура вызывается при запуске (инициализации) приложения и каждый раз при смене рисунков выбора (когда нажимается треугольная кнопка смены рисунков). Вторая – процедура выбора области окна программы `SelRegion`. В зависимости от того, в какой области окна мы нажмем кнопку мыши, происходит одно из четырех событий:

1. Выбор рисунка для закрашивания.
2. Смена блока из 12 рисунков для выбора.
3. Выбор цвета для закрашивания.
4. Сама процедура закрашивания фрагментов выбранного рисунка.

В качестве параметра в процедуру `SelRegion` при нажатии кнопки мыши передаются координаты курсора мыши. Определимся с необходимыми нам переменными...

Так как все необходимые рисунки находятся в

виде файлов, то будем по необходимости загружать их в объекты типа `Tbitmap`: `BufCol` – изображения ведерка с краской, `MyBrush` – курсор-кисточка, `Buffer` – буфер для загрузки всех изображений, кроме курсора, и `BufWin` – буфер окна программы, из которого все изображения выводятся на форму.

Значения цветов будем хранить в одномерном массиве `mas_col`. Также нам понадобятся координаты курсора в момент нажатия кнопки мыши (`xm` и `ym`) и еще одна переменная типа `byte` – `page`. По ней будем определять имена файлов цветных рисунков при смене блоков рисунков (по 12 штук). Все это переменные глобального типа (см. код ниже):

```
var
  Form1: TForm1;

  BufCol: TBitmap; // буфер загрузки ведерка с краской
  MyBrush: TBitmap; // буфер загрузки курсора-кисточки
  Buffer: TBitmap; // буфер загрузки всех компонентов,
                  // кроме кисточки
```

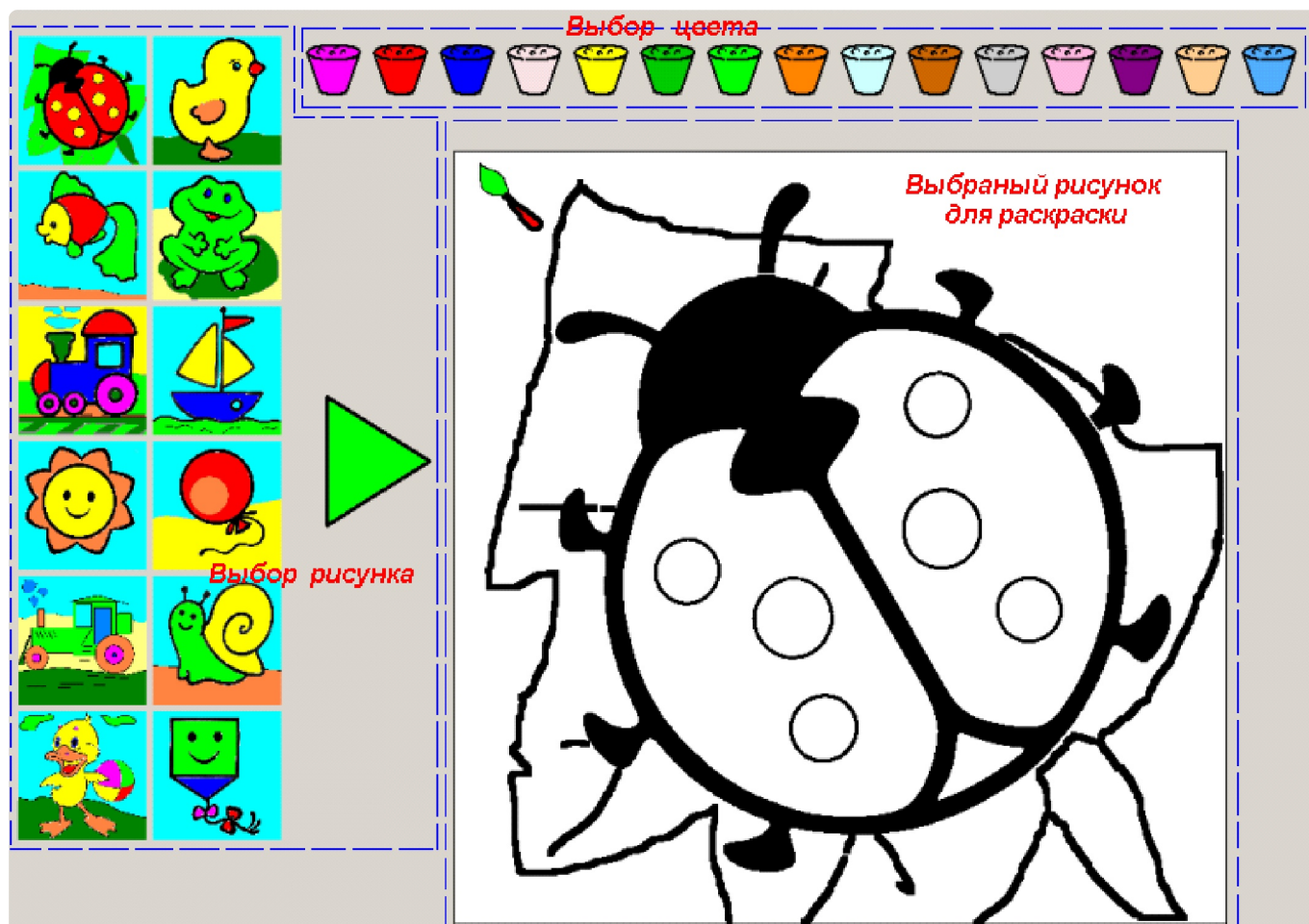


Рис. 1. Окно программы «Раскраска»

```

BufWin: TBitmap; // общий буфер вывода окна программы
                // на форм
xm,ym: integer; // координаты курсора мыши
mas_col: array[0..14] of integer; // массив палитры цветов
page: byte;      // страница блока из 12 рисунков

```

Необходимые локальные переменные будем рассматривать по ходу описания кода приложения. Сначала изучим процедуры выборов рисунка и области окна приложения.

1. Выбор рисунков *SelPicture*

```

procedure SelPicture; // выбор рисунка
var i: byte;          // переменная цикла
    BufPic: TBitmap; // буфер загрузки цветных рисунков
    x,y: integer;     // координаты рисунков на канве буфера
begin
    // создаем буфер
    BufPic:= TBitmap.Create;

    // координаты первого из 12 рисунков
    x:= 10; y:= 20;

    // цикл загрузки рисунков
    for i:= 1 to 12 do
    begin
        BufPic.LoadFromFile('Image\' + IntToStr(i+page) + '.bmp');
        // вывод на канву Buffer
        Buffer.Canvas.StretchDraw(bounds(x,y,100,100),BufPic);
        x:= x + 105;
        if x > 115 then begin x:= 10; y:= y + 105; end;
    end;
    BufPic.Free // освобождаем память
end;

```

В буфер *BufPic* типа *TBitmap* в цикле загружаем из файлов 12 цветных рисунков и помещаем на канву основного буфера *Buffer*, приводя к размеру 100x100 пикселей каждый. Имя файла определяется значением переменной цикла и значением страницы блоков из 12 рисунков: *(i + page).bmp*.

Значение переменной *page* при запуске приложения равно нулю. При нажатии кнопки смены рисунков значение *page* увеличивается на 20 и при значениях > 40 опять принимает значение, равное 0 (организовано в процедуре выбора области окна приложения).

2. Выбор области окна приложения

Процедура *SelRegion* вызывается по событию *OnClick()* на форме, если мы нажимаем кнопку мыши в области выбора рисунка или в области выбора цвета (см. рисунок 1). В зависимости от координат курсора в момент нажатия кнопки мыши происходит следующее:

- а) «Клик» по кнопке выбора рисунков – изменяется значение переменной *page* (0 - 20 - 40 - 0 -) и вызывается процедура *SelPicture* для загрузки других 12 рисунков;
- б) «Клик» по одному из цветных рисунков – определяется имя файла рисунка для раскраски по локальной переменной *num*, загружаем файл в буфер рисунка *BufPic*, а затем переносим в основной буфер *Buffer* (размер рисунка 600x600 пикселей) в область раскраски.
- в) «Клик» по любому из цветных ведерок – происходит выбор цвета, и цвет передается на канву основного буфера *Buffer* и на канву формы.

Теперь рассмотрим работу программы. При запуске приложения в процедуре *FormCreate* происходит инициализация:

создаются объекты типа *TBitmap* для загрузки графических изображений; рисунки ведерка и курсора-кисточки загружаются из файлов в соответствующие буферы *BufCol* и *MyBrush*; заполняем массив *mas_col* значениями цветов.

В процедуре *FormShow()* присваиваются атрибуты основным буферам *Buffer* и *BufWin* – размеры, равные размерам клиентской области формы, и цвет канвы. Здесь же выводим на канву буфера *Buffer* 15 цветных ведерок для выбора цвета, цветные рисунки выбора, затем рисуем кнопку смены рисунков и прямоугольник области рисунка для раскрашивания.

В процедуре *FormPaint()* считываем координаты курсора (*GetCursorPos(P: TPoint)*) в пределах клиентской области формы, содержимое буфера *Buffer* копируем в буфер окна *BufWin* и в него же добавляем изображение курсора-кисточки. Затем содержимое *BufWin* выводим на форму. Далее управление программой осуществляет пользователь с помощью мыши. При перемещении мыши обрабатывается событие

`FormMouseMove()`, в буфер окна `BufWin` копируется содержание основного буфера `Buffer` и накладывается новое положение курсора-кисточки, а затем все это выводится на форму.

При нажатии кнопки мыши вызываются два обработчика события – `FormMouseDown()`, в котором переменным `xm` и `ym` присваиваются значения координат курсора, и `FormClick()`, в котором, если курсор находится в области окна для раскрашивания рисунка, происходит заливка замкнутой области, где находится курсор, выбранным ранее цветом (метод `FloodFill`). При нахождении курсора в любой другой области окна и нажатии кнопки мыши вызывается процедура `SelRegion`, работу которой мы уже рассмотрели выше. При этом координаты мыши передаются в процедуру `SelRegion (xm,ym)` в качестве параметра.

И последняя процедура `SetBrushColor(C: Tcolor)` устанавливает для курсора-кисточки выбранный цвет. Более подробно весь код приложения можно посмотреть в приложении в файле `<Unit1.pas>`, который содержит подробные комментарии кода***.

Программа «Азбука»

Данная программа знакомит ребенка с буквами алфавита, помогая быстрее их запоминать и составлять простые слова. Чтобы понять смысл программы, рассмотрим окно приложения (см. рисунок 2). Оно разбито на четыре части:

1. Область выбора рисунка (1) – здесь выводятся два рисунка или фотографии знакомых ребенку игрушек или персонажей сказок, мультфильмов, которые можно выбрать, «кликнув» по изображению. Также здесь находятся кнопки (в виде треугольников) смены рисунков или фотографий. Также можно взять фотографии родителей или домашних питомцев.

2. Область выбранного рисунка с подписью (2), которая повторяется в поле вывода слова (3) по мере того, как ребенок будет правильно выбирать буквы из алфавита (4) в верхней части окна программы. В программе используются не все буквы алфавита, а только наиболее употребляемые. Кроме того, можно заменить русский алфавит на любой другой национальный (украинский, казахский и т. д.).



Рис. 2. Окно программы «Азбука»

*** Комментарий автора.

Считаю, что нет необходимости добавлять в программу какие-либо «красивости» (разве, что добавить звуковое оформление), так как маленькому ребенку в первую очередь интересны предложенные красочные рисунки и простота управления программой.

Суть программы в том, что после выбора изображения ребенку**** нужно повторить подпись к выбранному рисунку, нажимая на соответствующие буквы в поле алфавита. При правильном выборе букв слово-подпись повторяется в поле вывода. В качестве подсказки очередная буква подписи к рисунку мигает.

Аналогично предыдущей программе сначала опишем ее структуру и особенности:

- Все действия программы производим на канве формы.
- В папке Image проекта храним набор рисунков и фотографий, которые по мере необходимости выводим на форму.
- Все манипуляции в программе осуществляют-ся мышью.

Процедуры для реализации программы: аналогично предыдущей программе будем использовать стандартные обработчики `FormMouseDown()` для получения координат курсора мыши, а также процедуру обработчика событий `FormClick()`, в которой обрабатываем все манипуляции мышкой в окне формы. Стандартная процедура `FormCreate()` используется для инициализации приложения после запуска, а в процедуре `FormPaint()` выводим все необходимые изображения на форму. В обработчике таймера `Timer1.Timer()` организуем подсказку в виде мигания очередной буквы в слове-подписи.

Кроме этих стандартных процедур, напишем еще две. `SelAlfavit` – процедура выбора буквы в поле алфавита. В качестве параметров в процедуру передаются координаты курсора, и по ним из алфавита выбирается буква. Если выбранная буква совпадает с очередной буквой подписи (для подсказки ребенку она мигает), то выбранная буква выводится в поле вывода слова. В процедуре `SelFoto` производится выбор рисунка или фото и вывод его изображения на форму.

Заключение

Программа настолько проста, что нет смысла подробно рассматривать каждую процедуру.

Весь код программы с подробными комментариями смотрите в файле `<Unit1.pas>` в виде ресурсов в теме «Журнал клуба программистов. Шестой выпуск» или непосредственно в архиве с журналом.

**** Комментарий редакции.

Как известно, готовность ребенка к школе определяется комплексом его общей интеллектуальной, психологической и физической подготовки. Причем, психологическая – одна из самых трудных, так как она не возникает сама по себе и требует особого внимания родителей. В чем она проявляется? Все просто как дважды-два и через эти проблемы каждый раз проходят и родители и преподаватели и сам ребенок. Это и отсутствие сосредоточенности ребенка на предмете и личной инициативы, частое отвлечение внимания. Быть готовым к школе – это не только уметь читать, писать и считать. Это прежде всего, готовность всему этому учиться.

Для детей дошкольного возраста очень важна усидчивость и способность доводить дело до конечной цели. В этом плане хорошо помогают разнообразные настольные игры, вроде конструктора или лепки поделок из пластилина или глины, игры на счет, используя аппликации, палочки и тому подобные. Что примечательно, эти же игры помогают развивать и мелкую моторику пальцев.

Сегодня, в свете огромного скачка вперед в информационных технологиях, компьютер есть практически в каждой семье. И нельзя отрицать тот факт, что использование его на школьных занятиях практикуется уже повсеместно. Так почему-бы не познакомить ребенка с чудом двадцатого века раньше и дать возможность заранее адаптироваться к школьным трудностям. Также не будем забывать про акселерацию детей.

Конечно же, все это должно быть в форме игры и обязательно под присмотром старших. Для развития мелкой моторики малыша не нужно дорогих игрушек и особого подхода. Достаточно поиграть в:

- 1) пальчиковый театр;
- 2) электронную раскраску или говорящий алфавит.

Именно, о них и шла речь в представленном материале. Мы не предлагаем панацею, а лишь показываем, что такая область как программирование полезна и в детском творчестве. Примечательно,



что этот номер вышел как раз в начале учебного года. Надеемся, статья нашего постоянного автора Владимира Дегтяря поможет вам в игровой форме занять ваших малышей, а вам самим подарить несколько свободных минут. Удачи!

Так-с... С чего бы начать? Пожалуй, начну с того, откуда мне пришла в голову идея разработать собственный компилятор. Однажды мне на глаза попала книга, где описывались примеры проектирования в AutoCAD на встроенном в него языке AutoLISP. Я захотел с ними разобраться, но прежде меня заинтересовал сам ЛИСП. «Неплохо-бы поближе познакомиться с ним», – подумал я и начал доставать литературу и среду разработки. С литературой все оказалось просто – по ЛИСПу ее море в Интернете. Достаточно зайти на портал [1]. Дело оставалось за малым – найти хорошую среду программирования, и вот тут-то начались трудности. Компиляторов под ЛИСП тоже немало, но все они мне оказались малопонятны. Ни один пример из Вики, по разным причинам, не отработал нормально в скаченных мною компиляторах. Собственно, серьезно я с ними не разбирался, но, увы, во многих банально не нашел как скомпилировать EXE файл. Самое интересное, что компиляторы эти были собраны разными людьми в практически домашних условиях...



Виталий Белик

by **Stilet** www.programmersforum.ru

И мне пришла в голову мысль: «...а почему бы не попробовать самому написать свой компилятор или, основываясь на каком-либо диалекте какого-либо языка написать свой собственный язык программирования?». Как идея, а? К тому-же, я на форумах часто видел темы, где слезно жаловались на тиранов-преподавателей, поставивших задачу написания курсовой – компилятора или эвалюатора (программы вычисляющей введенное в виде строки выражение). Мне стало еще интереснее: «А чего будет стоить написать простому, неискушенному книгами Вирта или Страуструпа, студенту такую программу?». Появился мотив.

In the beginning

Итак, начнем. Прежде всего, нужно поставить задачу хотя бы на первом этапе. Задача будет банальная: доказать самому себе, что написание компилятора не такой уж сложный и страшный процесс. И что мы*, хитрые и смекалистые, способны родить в муках собственного творчества шедевр, который, возможно,

* Замечание автора.

Часто, в процессе создания, возникает иллюзия «понятности». Это состояние, в котором разработчик настолько проникается идеей продукта и его тонкостями, что ему начинает казаться, что и другим также все очевидно и понятно в функционале создаваемого продукта, как и ему самому. На деле же все выходит наоборот – даже знающего и опытного пользователя всегда требуется обучать и отвечать на такие вопросы, которые с точки зрения разработчика находятся в ряду «элементарных».

полюбится массам. Да и вообще: приятно писать программы на собственном языке, не так ли?

Что-ж, цель поставлена. Теперь самое время определиться со следующими пунктами:

1. Под какую платформу будет компилировать код программа?
2. На каком языке будет код, переводимый в машинный язык?
3. На чем будем писать сам компилятор?

Первый пункт достаточно важен, ибо широкое разнообразие операционных систем (даже три монстра Windows, Linux и MacOS) уже путают все карты. Их исполняемые файлы по-разному устроены, так что нам, простым смертным, придется выбрать из этой «кагалы» одну операционную систему и, соответственно, ее формат исполняемых файлов. Я предлагаю начать с Windows, просто потому, что мне нравится эта операционная система более других. Это не значит, что я терпеть не могу Линукс, просто я его не очень хорошо знаю, а такие начинания лучше делать по максимуму, зная систему для которой проектируешь.

Два остальных пункта уже не так важны. В конце концов, можно придумать свой собственный диалект языка. Я предлагаю взять один из старейших языков программирования – LISP. Из всех языков, что я знаю, он мне кажется более простым по синтаксису, более атомарным, ибо в нем каждая операция берется в скобочки, таким образом, к нему проще написать анализатор. С выбором, на чем писать, еще проще: писать нужно на том языке, который лучше всего знаешь. Мне ближе паскалевидные языки, я

хорошо знаю Delphi, поэтому в своей разработке я избираю именно его, хотя никто не мешает сделать то же самое на Си. Оба языка прекрасно подходят для написания такого рода программ. Я не беру в расчет Ассемблер потому, что его диалект приближен к машинному языку, а не к человеческому.

To shopping

Выяснив платформу, для которой будем писать компилятор (я имею в виду Win32), подберем все необходимое для комфортной работы. Давайте составим список, что же нам пригодится в наших изысканиях.

Для начала, нам просто крайне необходимо выяснить, как же все-таки компиляторы генерируют исполняемые EXE-файлы под Windows. Для этого стоит почитать немного об устройстве этих «экзешек», как их часто называют за глаза, покопаться в их кишках. В этом могут помочь современные отладчики и дизассемблеры, способные показать из чего состоит «экзешка». Я знаю два, на мой взгляд, лучших инструмента: OllyDebugger (он же «Оля») и The Interactive Disassembler (в простонародье зовущийся IDA).

Оба инструмента можно достать на их официальных сайтах <http://www.ollydbg.de> и <http://www.hex-rays.com/idapro>. Они помогут нам заглянуть в святая святых – храм, почитаемый загрузчиком исполнимых файлов, и посмотреть каков интерьер этого храма, дабы загрузчик наших экзешек чувствовал в нем себя так же комфортно как «ковбой в подгузниках Хаггис».

Также нам понадобится какая-нибудь «экзешка» в качестве жертвы, которую мы будем препарировать этими скальпелями-дизассемблерами. Здесь все сложнее. Дело в том, что благородные компиляторы имеют дурную привычку пихать в экзешник, помимо необходимого для работы кода, всякую всячину, зачастую излишнюю. Это конечно не мусор, но без него вполне можно обойтись, а вот для нашего исследования внутренностей «экзешек» он может стать серьезной помехой. Мы ведь не Ричарды Столлманы и искусством реверсинга в совершенстве не владеем. Поэтому нам лучше

было бы найти такую программу, которая содержала бы в себе как можно меньше откомпилированного кода, дабы не отвлекаться на него. В этом нам может помочь компилятор Ассемблера для Windows. Я знаю два неплохих компилятора: Macro Assembler (он же MASM) и Flat Assembler (он же FASM). Я лично предпочитаю второй – у него меньше мороки при компилировании программы, есть собственный редактор, в отличие от MASM компиляция проходит нажатием одной-единственной кнопки. Для MASM разработаны среды проектирования, например MASM Builder. Это достаточно неплохой визуальный инструмент, где на форму можно кидать компоненты по типу Delphi или Visual Studio, но, увы, не лишенный багов. Поэтому воспользуемся FASM. Скачать его можно везде, это свободно распространяемый инструмент. Ну и, конечно, не забудем о среде, на которой и будет написан наш компилятор. Я уже сказал, что это будет Delphi. Если хотите конкретнее – Delphi 6.

The Theory and Researching

Прежде чем приступить к написанию компилятора, неплохо бы узнать, что это за формат «экзешка» такой. Согласно [2], Windows использует некий PE формат. Это расширение ранее применявшегося в MS-DOS, так называемого MZ формата [3]. Сам чистый MZ формат простой и незатейливый – это 32 байта (в минимальном виде, если верить FASM'у. Турбо Паскаль может побольше запросить), где содержится описание для DOS загрузчика. В Windows его решили оставить, видимо для совместимости со старыми программами.

Вообще, если честно, размер DOS заголовка может варьироваться в зависимости от того, что после этих 28 байт напишет компилятор. Это может быть самая разнообразная информация, например для операционок, которые не смогли бы использовать скомпилированный DOS или Windows «экзешник», представленный в качестве машинного кода, который прерываниями BIOS выводит на экран надпись типа: «Эта программа не может быть запущена...».

Кстати, сегодняшние компиляторы тоже поступают так же.

Давайте посмотрим на это чудо техники, воспользовавшись простенькой программой, написанной на чистом Ассемблере FASM (см. рисунок 1):

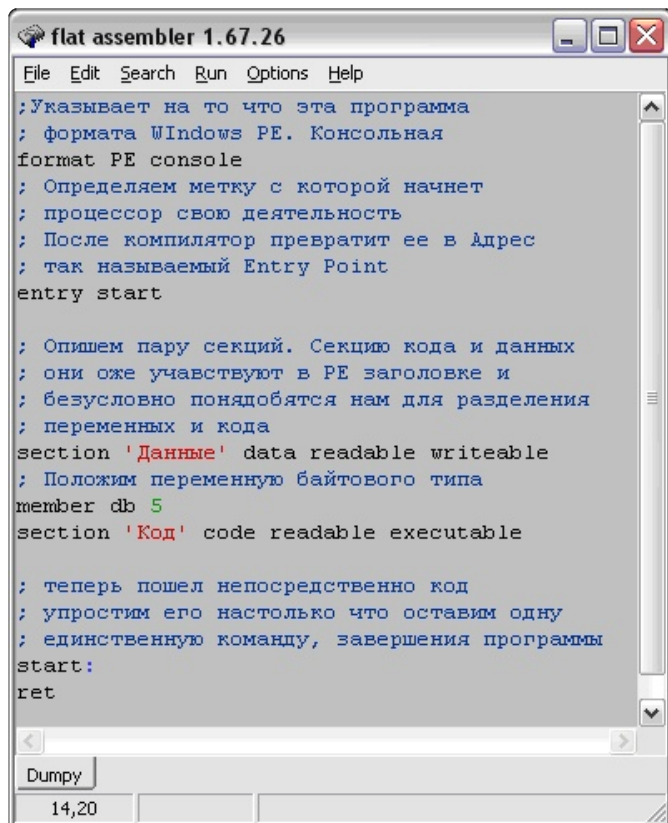


Рис. 1. Исходник для препарирования

Сохраним файл под неким именем, например Dumpy. Нажмем F9 или выберем в меню пункт RUN. В той же папке будет создан EXE файл. Это и будет наша жертва, которую мы будем препарировать.

Теперь ничто не мешает нам посмотреть: «...из чего-же, из-чего же сделаны наши девочки?».

Запустим OllyDebugger. Откроем в Оле наш экзешник. Поскольку фактически кода в нем нет, то нас будет интересовать его устройство, его структура. В меню View есть пункт Memory, после выбора которого «Оля» любезно покажет структуру загруженного файла (см. рисунок 2). Это не только сам файл, но и все, что было загружено и применено вместе с ним, библиотеки, ресурсы программы, и библиотек, стек и прочее, разбитое на блоки, называемые «секциями». Из всего этого, нас будут интересовать три секции, владельцем которых Dumpy – это непосредственно содержимое загруженного файла.

Собственно, эти секции были описаны нами в исходнике, я не зря назвал их по-русски (дело все в том, что операционной системе все равно, как называются секции, главное – их имена должны укладываться точь-в-точь в 8 байт. Это придется учесть обязательно).

Заглянем в первую секцию PE Header. Сразу же можем увидеть (см. рисунок 3), что умная «Оля» подсказывает нам, какие поля** у этой структуры. Последнее поле, которое для нас важно, находится по смещению 3Ch от начала файла (см. рисунок 4).

Это поле – точка начала PE заголовка. В Windows, в отличие от MS-DOS, MZ заголовков заканчивается именно на отметке 40**, что соответствует 64 байтам. При написании компилятора будем соблюдать это правило неукоснительно.

Что ж, перейдем непосредственно к PE заголовку (см. рисунок 5). Как показывает «Оля», нам нужно перейти на 80-й байт. Да, чуть не забыл. Все числа адресации указываются в 16-тиричной системе счисления. Для этого после чисел ставится латинская буква «H». «Оля» не показывает ее, принимая эту систему по умолчанию для адресации. Это нужно учесть, чтоб не запутаться в исследованиях. Фактически, 80h – это 128-й байт.

Вот она, святая обитель характеристик «экзешника». Именно этой информацией пользуется загрузчик Windows, чтобы расположить файл в памяти и выделить ему необходимую память для своих нужд. Вообще, считается, что этот формат хорошо описан в литературе. Достаточно выйти через Википедию по ссылкам в ее статьях [4] или банально забить в поисковик фразу вроде «ФОРМАТ ИСПОЛНЯЕМЫХ ФАЙЛОВ Portable Executables (PE)», как сразу же можно найти кучу описаний. Поэтому я поясню только основные его поля,

**** Комментарий автора.**

Сразу хочу оговориться, не все из этих полей нам важны. Тем паче, что сам Windows использует из них от силы 2-3 поля. Прежде всего, это DOS EXE Signature – здесь (читайте в Википедии по ссылке выше) помещаются две буквы MZ – инициалы создателя MS-DOS, и поле DOS_PartPag. В нем указывается размер MZ заголовка в байтах, после которых помещается уже PE заголовок.

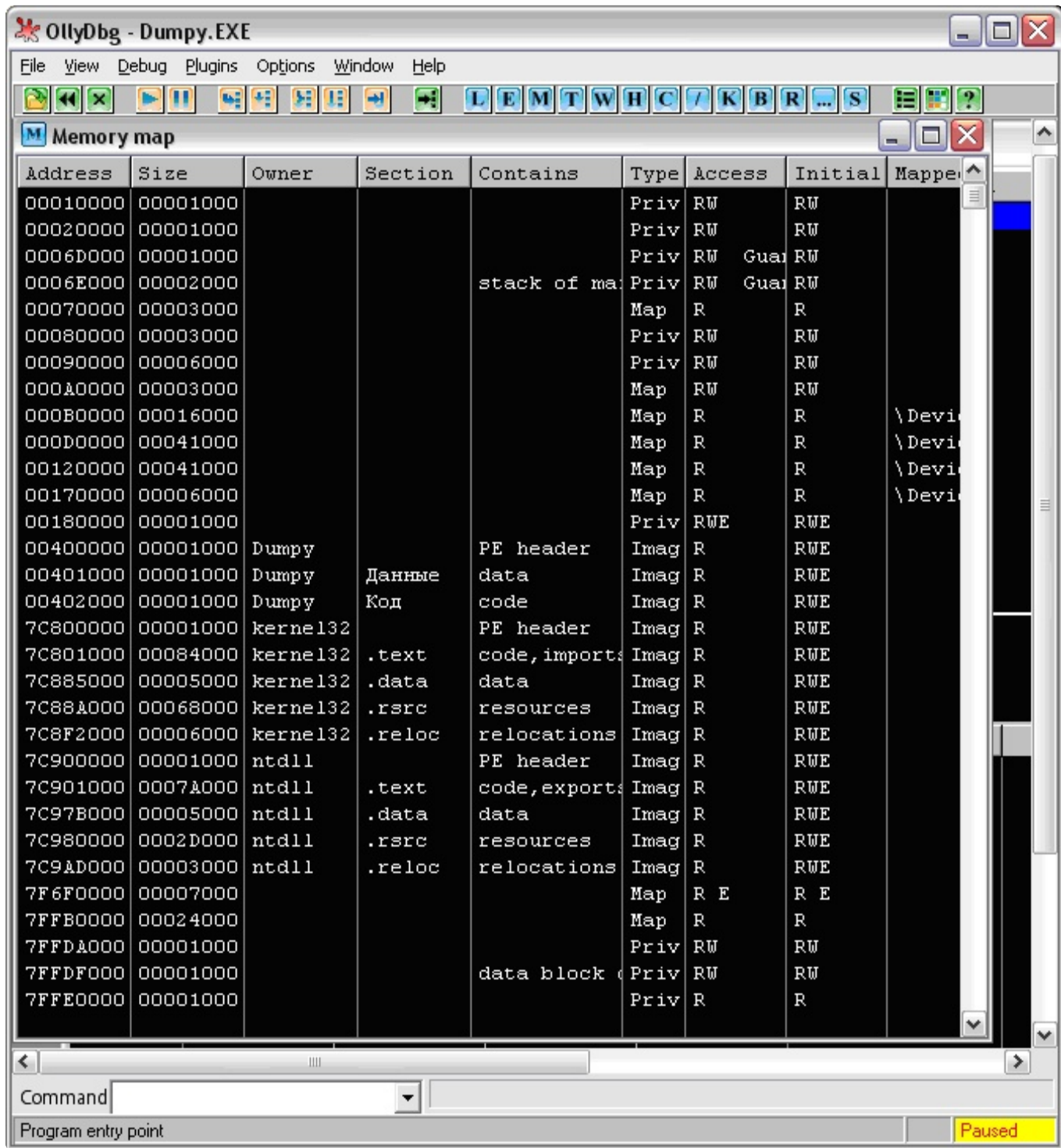


Рис. 2. Карта памяти Dumpy

*** Комментарий автора.

Обратите внимание! Далее, с 40-го смещения «Оля» показывает какую-то белиберду. Эта белиберда - есть атавизм DOS, и представляет из себя выше оговоренную информацию, с сообщением о том, что данная программа может быть запущенна только под DOS - Windows. Эдакий перехватчик ошибок. Как показывает практика, этот мусор можно без сожаления выкинуть. Наш компилятор не будет генерировать его, сразу переходя к PE заголовку.

которые нам понадобятся непосредственно для

написания компилятора...

Прежде всего, это PE Signature - 4-х байтовое поле. В разной литературе оно воспринимается по-разному. Иногда к ней приплюсовывают еще поле Machine, оговариваясь, чтобы выровнять до 8 байт. Мы же, как любители исследовать, доверимся «Оле с Идой», и будем разбирать поля непосредственно по их подсказкам. Это поле содержит две латинские буквы верхнего

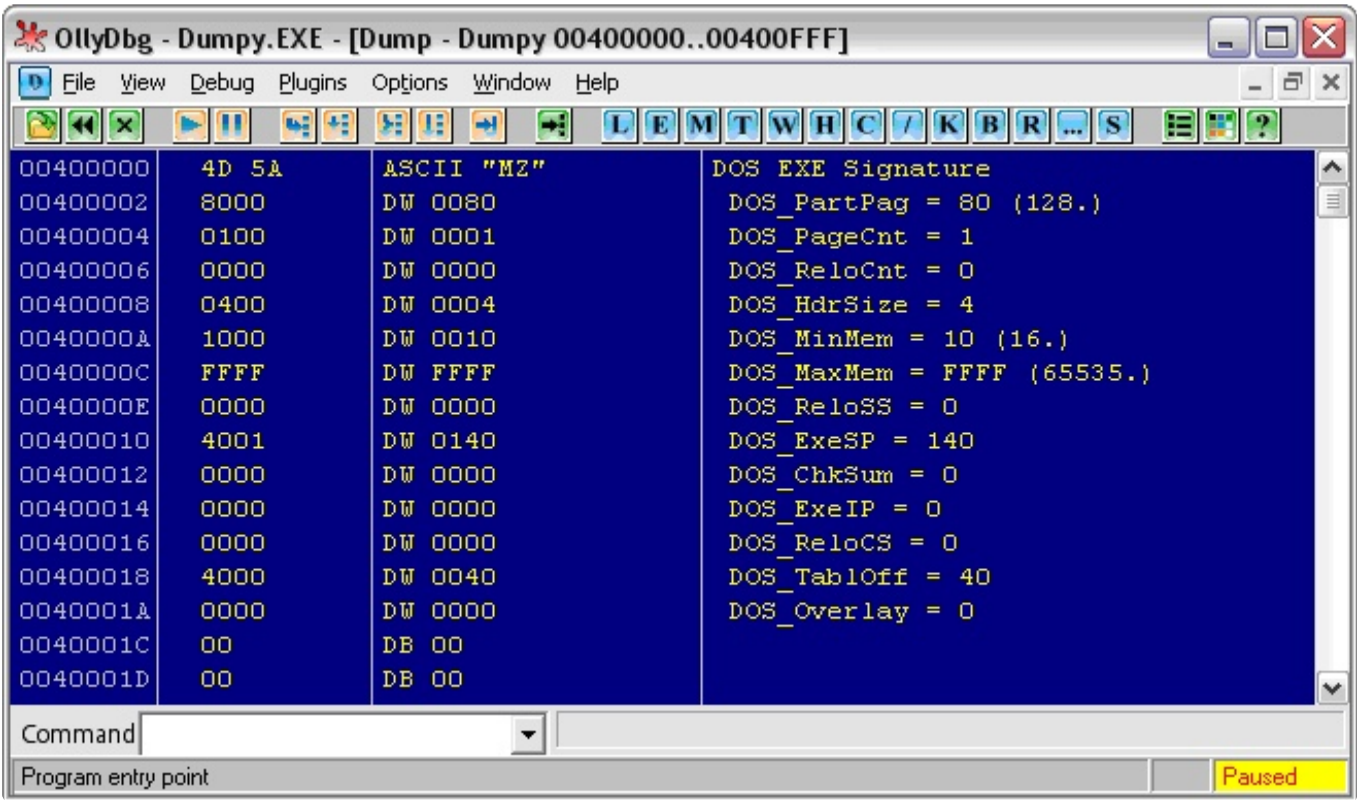


Рис. 3. MZ заголовок

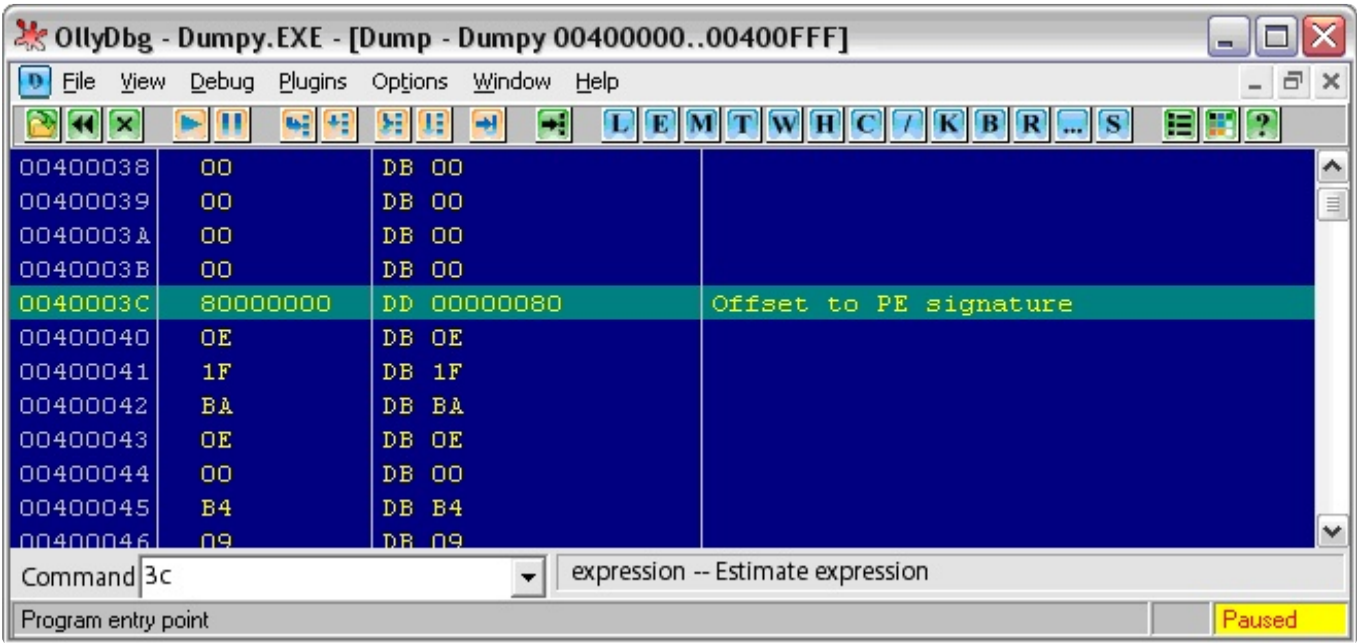


Рис. 4. Смещение на PE заголовок

регистра «PE», как бы намекая на то, что это Portable Executable формат.

Следующее за ним поле указывает, для какого семейства процессоров пригоден данный код.

Всего их, как показывает литература, 7 видов:

- 0000h __unknown

- 014Ch __80386
- 014Dh __80486
- 014Eh __80586
- 0162h __MIPS Mark I (R2000, R3000)
- 0163h __MIPS Mark II (R6000)
- 0166h __MIPS Mark III (R4000)

Думаю, нам стоит выбрать из всего этого второй вид – 80386. Кстати, наблюдательные личности

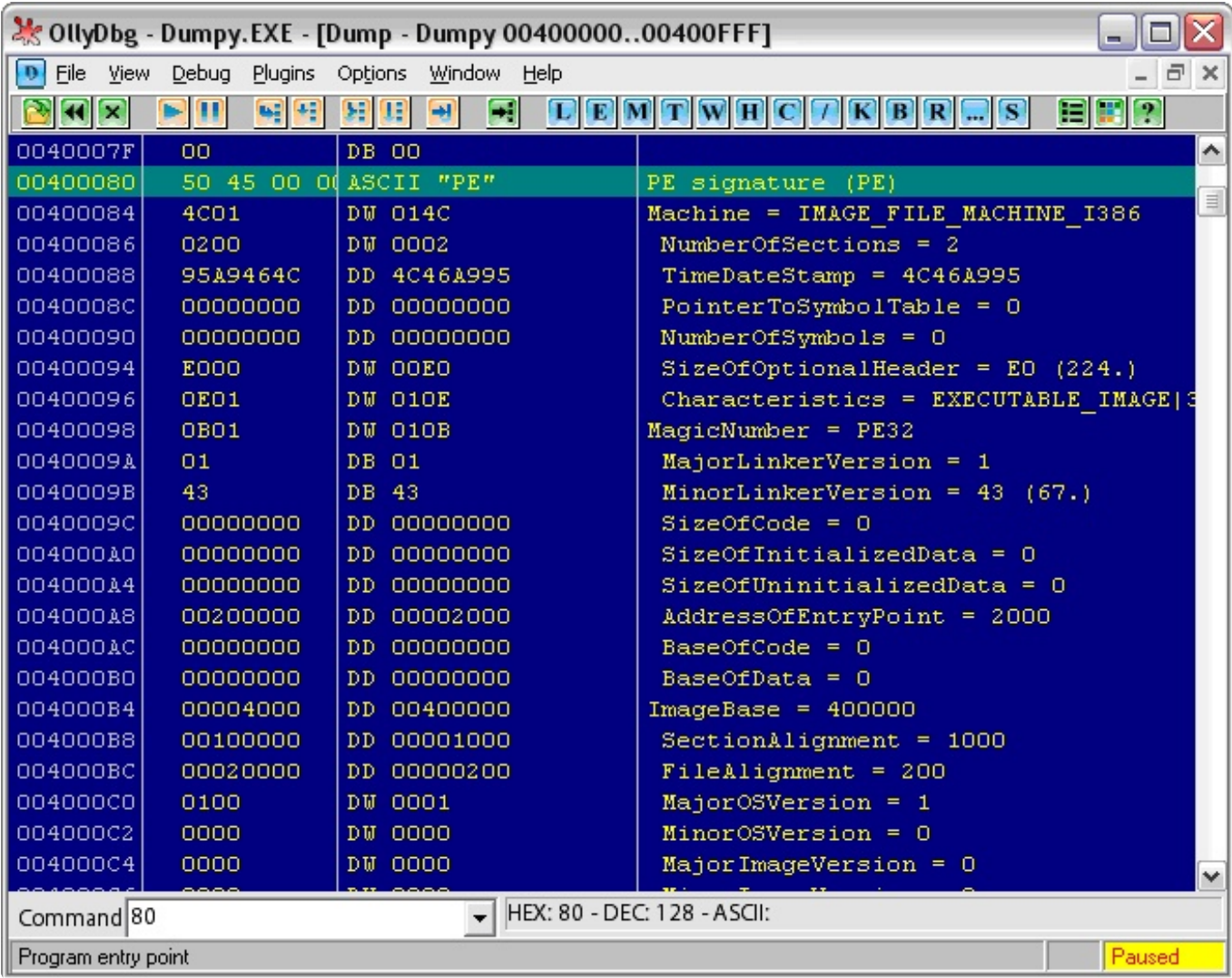


Рис. 5. Начало PE заголовка

могли заметить, что в компиляторах ассемблера есть директива, указывающая, какое семейство процессора использовать. Вот например, как в MASM (см. рисунок 6).

.386 как раз и говорит о том, что в этом поле будет стоять значение 014Ch****.

Следующее важное для нас поле - NumberOfSections. Это количество секций без учета PE секции. Имеются ввиду только те секции, которые принадлежат файлу (в карте памяти их владелец Dumpy). В нашем случае, это «Данные» и «код». Следующее поле, хоть оно и не столь важно, но я его опишу. Это

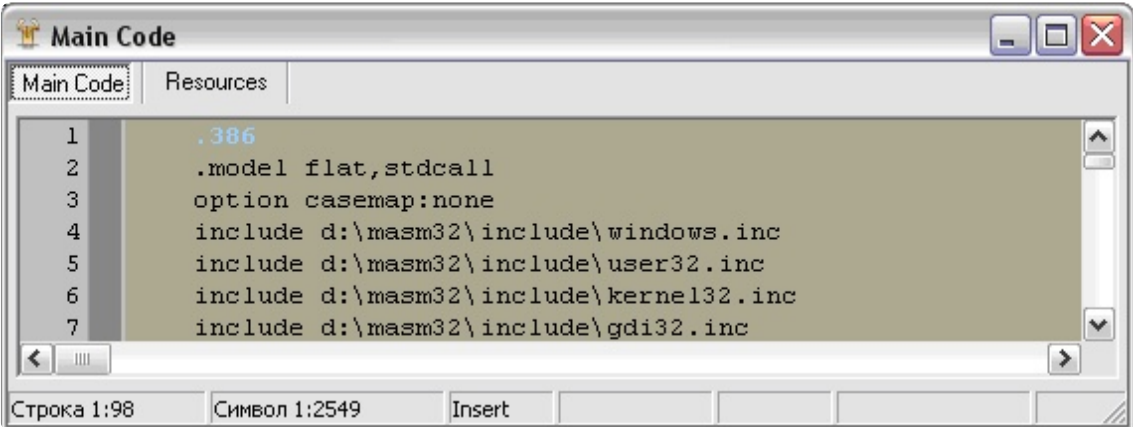


Рис. 6. Указание семейства процессоров в MASM

TimeDateStamp - поле, где хранится дата и время компиляции.

Вообще, я его проигнорирую, не суть важно сейчас, когда был скомпилирован файл. Впрочем, если

**** Комментарий автора.

Обратите внимание на одну небольшую, но очень важную особенность – байты в файле непосредственно идут как бы в перевернутом виде. Вместо 14C, в файл нужно писать байты в обратном порядке, начиная с младшего. т.е. получится 4C01 (0 здесь дополняет до байта. Это для человеческого глаза сделано, иначе все 16-ричные редакторы показывали бы нестройные 4C1. (Согласитесь, трудно было понять какие две цифры из этого числа, к какому байту относятся). Эту особенность обязательно придется учесть. Для простоты вполне нелишним было бы написать пару функций, которые число превращают в такую вот перевернутую последовательность байт, что мы в дальнейшем и сделаем.

кому захочется помещать туда время, то флаг в руки.

Сразу хочу предупредить, что меня как исследователя не интересовали поля с нулевыми значениями. На данном этапе, они действительно неважны, поэтому в компиляторе их и нужно будет занулить.

Следующее важное поле – `SizeOfOptionalHeader`. Оно содержит число, указывающее, сколько байт осталось до начала описания секций. В принципе, нас будет устраивать число 0Eh (224 байта).

Далее идет поле – характеристики экзешника. Мы и его будем считать константным:

Characteristics

```
(EXECUTABLE_IMAGE|32BIT_MACHINE|LINE_NUMS_STRIPPED|LOCAL_SYMS_STRIPPED)
```

И равно оно 010Eh. На этом поле заканчивается, так называемый, «файловый заголовок» и начинается «Опциональный».

Следующее поле – `MagicNumber`. Это тоже константа. Так называемое, магическое число. Если честно, я не очень понял для чего оно служит. В разных источниках это поле преподносится по-разному, но все хороши ссылаются на знаменитый дизассемблер HIEW, в котором якобы впервые появилось описание этого поля именно в таком виде. Примем на веру.

Следующие два поля, хоть и не нулевые, но нам малоинтересны. Это `MajorLinkerVersion` и `MinorLinkerVersion`. Это два байта версии компилятора. Угадайте, что я туда поставил?

Следующее важное поле – `AddressOfEntryPoint`. Важность этого поля в том, что оно указывает на адрес, с которого начинается первая команда, с которой процессор начнет выполнение. Дело в том, что на этапе компиляции значение этого поля не сразу известно. Ее формула достаточно проста. Сначала указывается адрес первой секции плюс ее размер. К ней плюсятся размеры остальных секций до секции, считаемой секцией кода. Например, в нашей жертве это выглядит так (см. рисунок 7). Здесь секция кода вторая по счету, значит, Адрес точки входа равен размеру секции «Данные» плюс ее начало и равно 2000. К этому еще пририсовывается базовый адрес, в который загрузчик «сажает» файл, но он в вычислении для нашего компилятора не участвует. Поэтому в жертве точка входа имеет значение равное 2000.

Следующее поле – `ImageBase`. Это поле я приму как константу, хотя и не оговаривается ее однозначное значение. Это значение указывает адрес, с которого загрузчик поместит файл в память. Оно должно нацело делиться на 64000. В общем, не обязательно указывать именно 400000h, можно и другой адрес. Уже не помню, где я слышал, что загрузчик может на свое усмотрение поменять это число, если вдруг в тот участок памяти нельзя будет загружать. Но не будем это проверять, а примем на веру как константу равную 400000h.

Следующая важная константа – `SectionAlignment`. Это значение говорит о размере секций после загрузки. Принцип прост – каждая секция (имеется ввиду ее реализация) дополняется загрузчиком пустыми байтами до числа указанного в этом поле. Это так называемое выравнивание секций. Тут уж хороший компилятор должен думать самостоятельно, какой размер секций ему взять, чтобы все переменные (или сам код), которые в коде используются, поместились без проблем. Согласно спецификации, это число должно быть степенью двойки в пределах от 200h (512 байт) до 10000h (64 000 байт).

В принципе, пока-что для простенького компилятора можно принять это значение как константу. Нас вполне устроит среднее значение

1000h (4096 байт - не правда-ли расточительный мусор? На этом весь Windows построен - живет на широкую ногу, память экономить не умеет).

Далее следует поле - FileAlignment. Это тоже хитрое поле. Оно содержит значение, сколько байт нужно дописать в конец каждой секции в сам файл, т.е. выравнивание секции, но уже в файле. Это значение тоже должно быть степенью двойки в пределах от 200h (512 байт) до 10000h (64 000 байт). Неплохо-бы рассчитывать функцией это поле в зависимости от размеров

данных в секции.

Следующие поля: MajorSubsystemVersion и MinorSubsystemVersion, примем на веру, как константы. 3h и Ah соответственно. Это версия операционной системы, под которую рассчитывается эта компиляция*****.

***** Комментарий автора.
Я не проверял на других ОС у меня WinXP. В принципе можно не поленился, и попробовать пооткрывать Олей разные программы, рассчитанные на другие версии Windows.

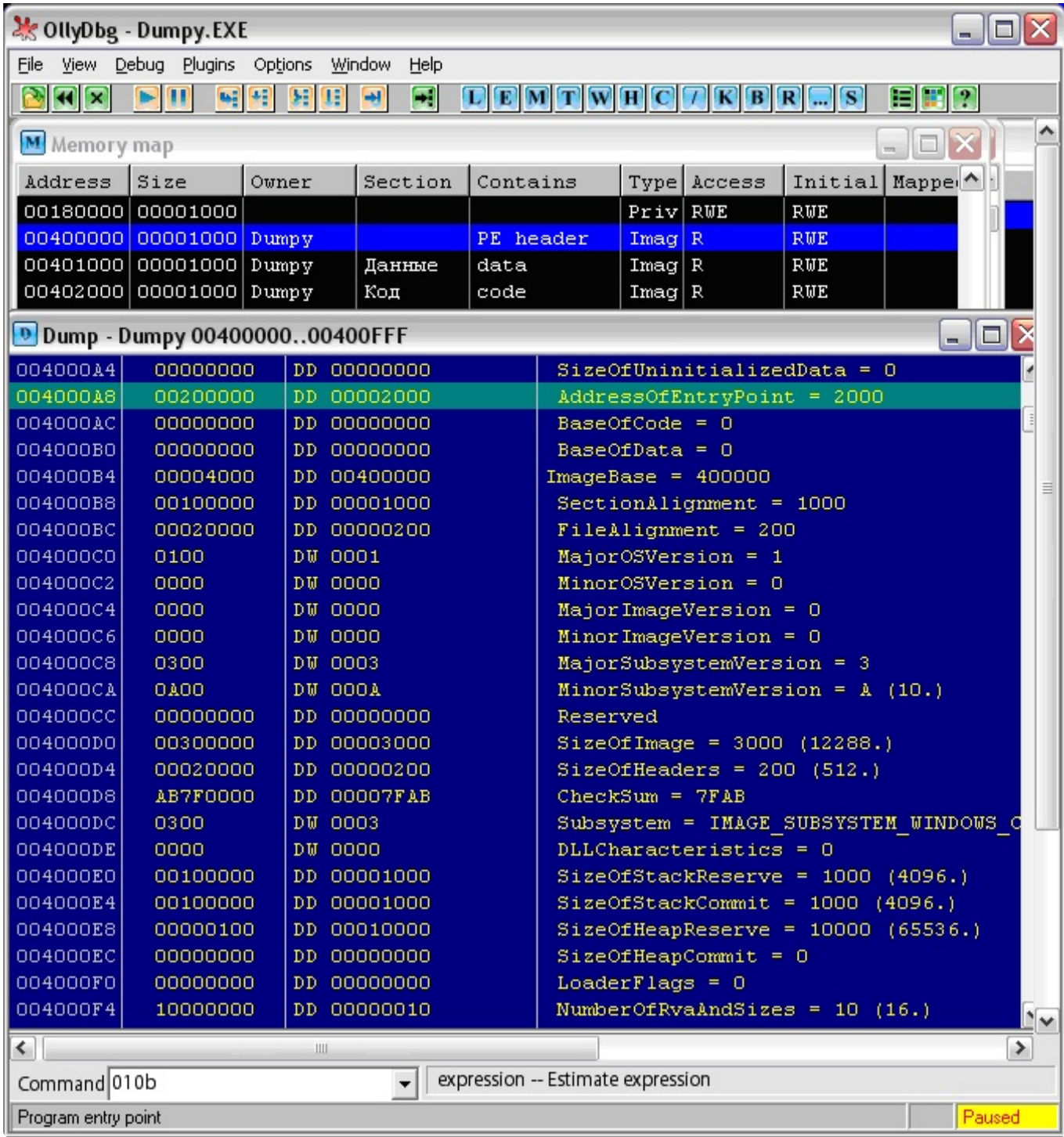


Рис. 7. Расчет точки входа

Далее, из значимых следует – `SizeOfImage`. Это размер всего заголовка, включая размер описания всех секций. Фактически, это сумма PE заголовка плюс его выравнивание и плюс сумма всех секций, учитывая их выравнивание. Ее тоже придется рассчитывать.

Следующее поле – `SizeOfHeaders` (размер файла минус суммарный размер описания всех секций в файле). В нашем случае, это $1536 - 512 * 2 = 200h$ (512 байт). Однако, PE тоже выровнен! Это поле тоже нужно будет рассчитывать.

Далее следует не менее коварное поле – `Checksum`. Это CRC сумма файла. Ужас... Мы еще файл не создали, а нам уже нужно ее посчитать (опять таки вспоминается Микрософт злым громким словом). Впрочем, и тут можно вывернуться. В Win API предусмотрена функция расчета CRC для области данных в памяти, проще говоря, массива байт – `ChecksumMappedFile`. Можно ей скормить наш эмбрион файла. Причем веселье в том, что эта операция должна быть самой последней до непосредственно записи в файл. Однако, как показывает практика, Windows глубоко наплевать на это поле, так что мы вполне можем не морочить***** голову с этим расчетом.

Следующее поле – `Subsystem`. Может иметь следующие значения*****:

1. `IMAGE_SUBSYSTEM_WINDOWS_CUI=3` – Это говорит что наш откомпилированный «экзешник» является консольной программой.
2. `IMAGE_SUBSYSTEM_WINDOWS_GUI=4` – Это говорит, что «экзешник» может создавать окна, и оперировать сообщениями.

Вот собственно и все важные для нас параметры. Остальные можно оставить константно, как показывает «Оля»:

```
DLLCharacteristics = 0
SizeOfStackReserve = 1000h (4096)
SizeOfStackCommit = 1000h (4096)
SizeOfHeapReserve = 10000h (65536)
NumberOfRvaAndSizes = 10h (16)
```

И остаток забить нулями (посмотрите в «Оле» до начала секций какие там еще параметры). О них можно почитать подробнее по ссылкам, которые

***** Комментарий автора.

Согласитесь, держать в файле поле, которое никому не нужно, да еще и напрягать нас лишним расчетом – это глупо, но, увы, в этом изюминка политики Микрософта. Складывается впечатление, что программисты, писавшие Windows, никак не согласовывали между собой стратегию. Спонтанно писали. Импровизировали.

я привел. После идет описание секций. Каждое описание занимает 32 байта. Давайте взглянем на них на рисунке 8. В начале секции идет ее имя (8 байт), после этого поле – `VirtualSize`, описывает (я процитирую из уроков Iczeliona) RVA секции: «PE-загрузчик использует значение в этом поле, когда мэмпирует секцию в память. То есть, если значение в этом поле равняется $1000h$ и файл загружен в $400000h$, секция будет загружена в $401000h$ ».

Однако «Оля» почему-то показывает для обеих секций одно и то же значение. Что это? Я не понял, почему так. Пока оставим это как данное. Вдруг в будущем разберемся.

Далее следует `VirtualAddress`, который указывает, с какого адреса плюс `ImageBase` будет начинаться в памяти секция – это важное поле, именно оно станет для нашего компилятора базой для расчета адреса к переменной. Собственно, адрес этот напрямую зависит от размера секции. Следующий параметр `PointerToRawData` – это смещение на начало секции в скомпилированном файле. Как я понял, этот параметр компиляторы любят подводить под `FileAlignment`. И последнее – поле `Characteristics`. Сюда прописывается доступ к секции. В нашем случае, для секции кода оно будет равным `60000020 = CODE|EXECUTE|READ`, а для секции данных `C0000040 = INITIALIZED_DATA |READ|WRITE`.

Вот и все. Закончилось описание заголовка. Далее он выравнивается нулями до 4095 байт (с этим числом связан один прикол). В файле мы его будем дополнять до `FileAlignment` (в нашем случае до $200h$).

Hello world. Hey! Is there anybody out there?

Вот мы и прошли по кишкам нашей жертвы – «экзешника». Напоследок попробуем на скорую руку закрутить простейший компилятор для DOS

***** Комментарий автора.

Для справки, кто хорошо знает Delphi, директивы компилятора {APPTYPE GUI} и {APPTYPE CONSOLE}. Именно эти параметры он и выставляет.

системы без PE заголовка. Для этого подойдут инструменты, которые так (почему-то) любят преподавать до сих пор.

Я говорю о Классическом Паскале. Итак, предположим, «злой преподаватель» поставил задачу: написать компилятор программы вывода на экран некой строки, которую мы опишем для компилятора, введя ее ручками (см. листинг 1):

var header, commands, s: string;

e, i: integer;

f: file;

begin

{Это MZ заголовок}

header:= #4D\$5A\$3E\$00\$01\$00\$00\$00\$02\$00\$00\$01\$FF\$FF\$02\$00\$00\$00;

header:= header+\$10\$00\$00\$00\$00\$00\$00\$00\$1C\$00\$00\$00\$00\$00\$00\$00\$00\$00;

writeln('give me wellcome :');readln(s);

{Поскольку у нас все в одном сегменте, и код и данные, лежащие

ЛИСТИНГ 1

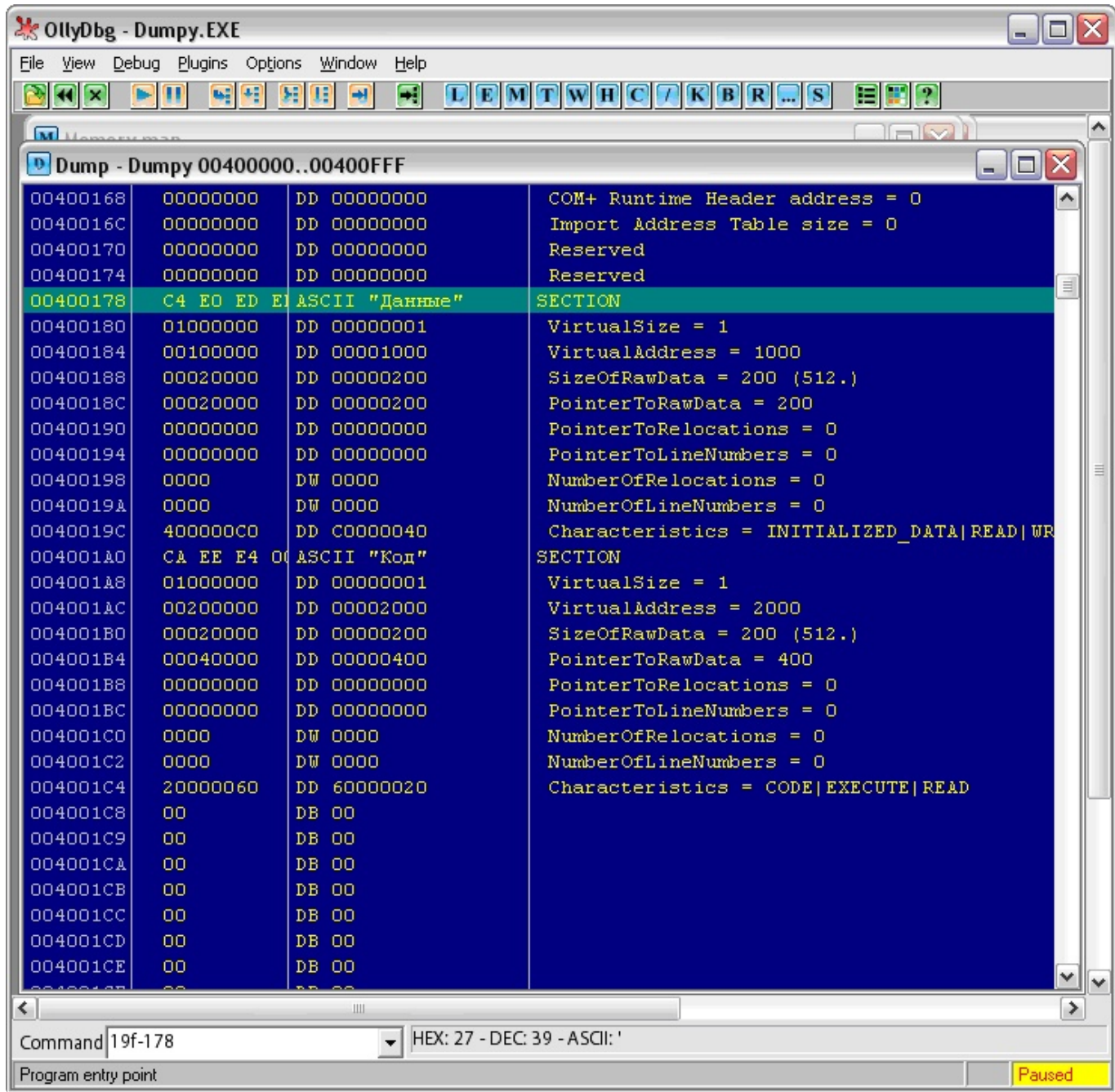


Рис. 8. Описание секций

непосредственно в конце кода нужно чтоб регистр, содержащий базу данных указывал на код. Предположим мы будем считать что и сам код представляет из себя данные. Для этого поместим в стек адрес сегмента кода }

```
{*****}

Commands:=$0E;           { push    cs}

{и внесем из стека этот адрес в регистр сегмента данных}

Commands:=Commands+$1F;   { pop     ds}

Commands:=Commands+$B4$09; { mov  ah, 9 - Вызовем функцию
                           вывода строки на экран}

{Передадим в регистр DX адрес на строку. Поскольку пока что
строка у нас не определена, передадим туда нули, а позже
подкорректируем это место}

Commands:=Commands+$BA$00$00; { mov    dx, }

{Запомним место, которое нужно будет скорректировать. Этим
приемом я буд пользоваться, чтоб расставить адреса в коде,
который обращается к переменным}

e:=length(commands)-1;

{Выведем на экран строку}

Commands:=Commands+$CD$21; { int 21h; DOS - PRINT STRING}

{подождем пока пользователь не нажмет любую клавишу}

Commands:=Commands+$B4$01; { mov  ah, 1}

Commands:=Commands+$CD$21; { int 21h; DOS - KEYBOARD INPUT}

{После чего корректно завершим программу средствами DOS}

Commands:=Commands+$B4$4C; { mov  ah, 4Ch}

Commands:=Commands+$CD$21; {int 21h; DOS-2+-
                           QUIT WITH EXIT CODE}

Commands:=Commands+$C3;    {retn}

{*****}

{Теперь будем править адреса, обращающиеся к переменной}

Поскольку само значение переменной у нас после всего кода (и
переменная одна. Мы получим длину уже имеющегося кода - это и
будет смещение на начало переменной}

i:=length(commands);

{В запомненное место, куда нужно править запишем в обратном
порядке это смещение}

commands[e]:=chr(lo(i));
commands[e+1]:=chr(hi(i));
```

{Учтем что в DOS есть маленький аттавизм - строки там должны завершаться символом \$. По крайней мере для этой функции. }

```
commands:=commands+s+'$';
```

{не забудем дописать вначале заголовок}

```
commands:=header+commands;
```

{Теперь скорректируем поле DOS_PartPag. Для DOS программ оно указывает на общий размер файла. Чесно говоря я не знаю зачем это было нужно авторам, возможно когда они изобретали это еще не было возможности получать размер файла из FAT. Опять таки запишем в обратном порядке}

```
i:=length(commands);
```

```
commands[3]:=chr(lo(i));
```

```
commands[4]:=chr(hi(i));
```

{Ну и кульминация этого апофигея - запись скомпилированного массива байт в файл. Все заметили что я воспользовался типом String - он в паскалевских языках был изначально развит наиудобнейшим образом}

```
Assign(f,'File.exe');rewrite(f);
```

```
BlockWrite(f,commands[1],length(commands), e);
```

```
Close(f);
```

```
end.
```

Не удивляет, что программа получилась небольшой? Почему-то преподаватели, дающие такое задание, уверены, что студент «облажается» или завалится по полной. Однозначно, такие преподаватели сами не смогли-бы написать компилятор. А студенты смогут, ибо, как видим, самая большая сложность - это найти нужные машинные коды, для решения задачи. А уж скомпилировать их в код, подкорректировать заголовок и расставить адреса переменных - задача второстепенной сложности. В изучении ассемблерных команд поможет любая книга по ассемблеру. Например, книга Абея «Ассемблер для IBM PC». Еще неплохая книга есть у Питера Нортон, где он приводит список функций DOS и BIOS.

Впрочем, можно и банальнее. Наберите в поисковике фразу «команды ассемблера описание». Первейшая же ссылка выведет нас на что-нибудь вроде **[5]** или **[6]**, где описаны команды Ассемблера. Например, если преподаватель задал задачку написать

компилятор сложения двух чисел, то наши действия будут следующими:

1. Выясняем, какая команда складывает числа. Для этого заглянем в книгу того же Абеля, где дается такой пример:

сложение содержимого

```
ADD AX,25      ;Прибавить 25
ADD AX,25H     ;Прибавить 37
```

2. Значит, нам нужна команда ADD. Теперь определимся: нам же нужно сложить две переменные, а это ячейки памяти. Эта команда не умеет складывать сразу из переменной в переменную, для нее нужно сначала слагаемое поместить в регистр (AX для этого лучше подходит). А уж потом суммировать в него. Для помещения из памяти в регистр, согласно тому же Абелью, нужна команда `mov [адрес], ax`

3. Таким образом, инструкции будут выглядеть так:

```
mov [Адрес первой переменной], ax
add [Адрес второй переменной], ax
```

4. Теперь нужно определиться с кодами этих команд. В комплекте с MASM идет хелп, где описаны команды и их опкоды (машинные коды, операционные коды).

Вот, например, как выглядит опкод команды MOV из переменной:

Видим (см. рисунок 9), что его опкод A1 (тут тоже любят 16-тиричность). Таким образом, выяснив все коды, можно написать компилятор что-то вроде этого (см. листинг 2):

ЛИСТИНГ 2

```
{ mov [Из памяти] в AX}
Commands:= Commands+#A1#$00#$00;

{Запомним позицию для корректировки переменной a}
{ $03 - Это опкод команды ADD $06 - Это номер регистра AX}
aPos:= Length(Commands)-1;
Commands:= Commands+#03#$06#$00#$00;

{Запомним позицию для корректировки переменной b}
bPos:= Length(Commands)-1;
{ mov из AX в переменку b}
Commands:= Commands+#A3#$00#$00;

{Запомним позицию для корректировки для переменной b}
b2Pos:= Length(Commands)-1;
```

А далее, в конце скорректируем эти позиции (см. листинг 3).

Запустим компилятор, он скомпилирует «экзешник», который посмотрим в «Иде» (см. рисунок 10).

Все верно, в регистр пошло значение одной переменной, сложилось со второй и во вторую же записалось. Это эквивалентно паскалевскому:

b:=b+a;

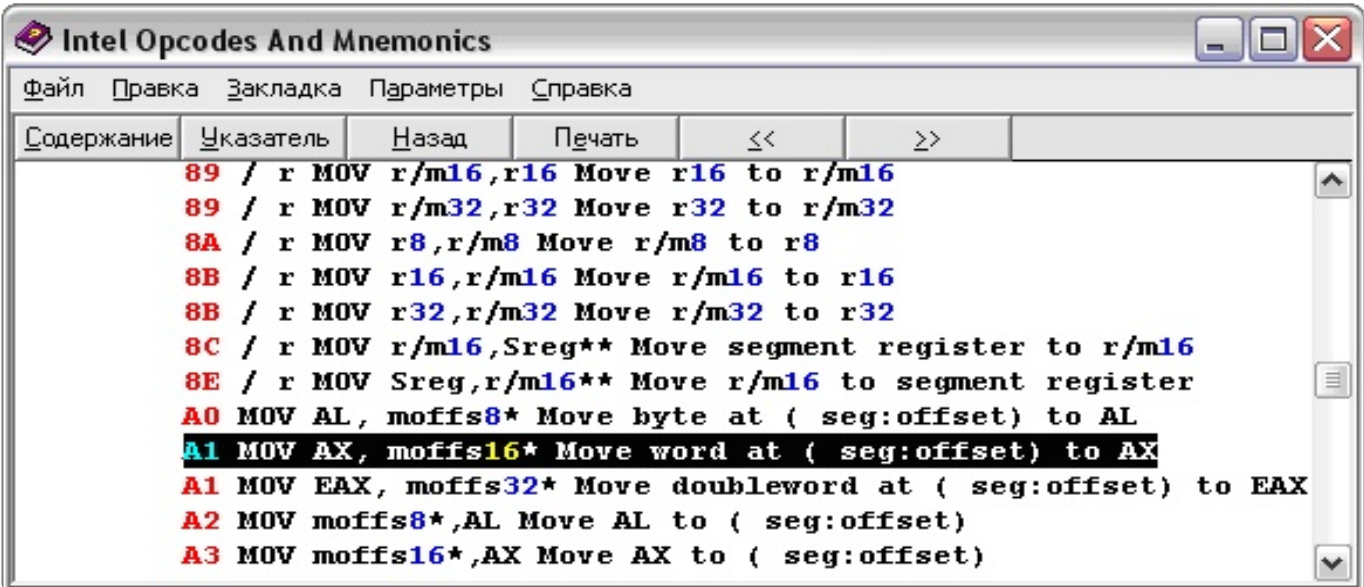


Рис. 9. Опкоды MOV

ЛИСТИНГ 3

```
{Это переменка а, ее значение}

commands:= commands+#$01#$00;

i:= length(commands);

{Не забудем что адреса в перевернутом виде}

commands[aPos]:= chr(lo(i));

{Поэтому сначала запишем младший байт}

commands[aPos+1]:= chr(hi(i));

{Это переменка б, ее значение}

commands:= commands+#$02#$00;

i:= length(commands);

{Поскольку переменка б фигурирует в коде}

commands[bPos]:= chr(lo(i));

{дважды придется дважды корректировать ее}

commands[bPos+1]:= chr(hi(i));

commands[b2Pos]:= chr(lo(i));

commands[b2Pos+1]:= chr(hi(i));
```

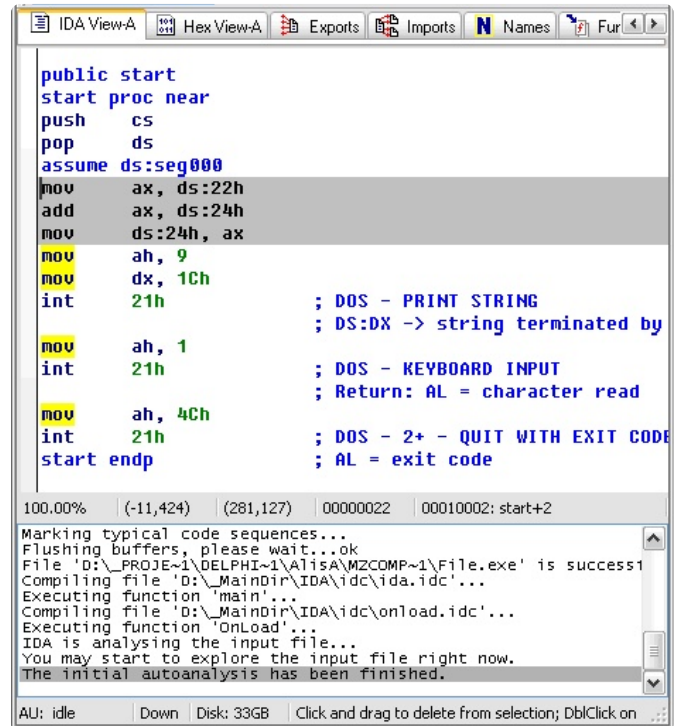


Рис. 10. Реверсинг нашего кода

Post Scriptum

Ну как, студенты, воспряли духом? Теперь понятен смысл, куда двигаться в случае столкновения с такими задачами? Это вряд ли потянет на курсовую, но вполне подойдет для простенькой контрольной.

А вот следующая задача – написать транслятор, уже действительно тянет даже на хороший диплом. Так что пока переварим все то, что выше, а в следующий раз попробуем приготовить основное ядро компилятора, дабы потом уже делать упор на сам код программы.

Весь упомянутый код с подробными комментариями смотрите в виде ресурсов в теме «Журнал клуба программистов. Шестой выпуск» или непосредственно в архиве с журналом.

***** Комментарий автора.

Обратите внимание: значения переменных мы заранее проинициализировали. При желании можно сделать, чтоб компилятор спросил их инициальное значение, ну и собственно подставить в нужное место:

```
commands:=commands+#$01#$00; {Вот сюда вместо этих циферок}
```

The Чтиво

- Ресурс википедии по языку LISP <http://ru.wikipedia.org/wiki/Lisp>
- Ресурс википедии по формату PE <http://ru.wikipedia.org/wiki/PE>
- Ресурс википедии по формату MZ [http://ru.wikipedia.org/wiki/MZ_\(формат\)](http://ru.wikipedia.org/wiki/MZ_(формат))
- Micheal J. O'Leary (Microsoft). The portable executable format <http://www.nikse.dk/petxt.html>
- Описание простейших команд Ассемблера <http://students.uni-vologda.ac.ru/pages/it10/2.php>
- Питер Абель. Ассемблер и программирование для IBM PC. – British Columbia Institute of Technology



Иногда, бывают случаи, когда нужно получить доступ к СУБД MySQL. Обычно, для этого используют сторонние клиенты или модули для разработчиков, такие как: connector.NET, MySQL++ или phpMyAdmin, как самый распространенный в вебе. В данной статье будет рассмотрен краткий вариант соединения и взаимодействия с СУБД MySQL из программы, написанной на Си или Си++. Также будет рассмотрен минимум необходимого и более или менее подробное описание всех действий, происходящих в статье. В заключении, напишем небольшой клиент к БД «Библиотека», которая по составу будет минимальной. Это необходимо для общего понимания принципа «как писать клиенты для СУБД», так как принцип везде один, разница лишь в СУБД. Все примеры будут написаны на MySQL API [1...8]. При желании, можно написать свою «обертку» для MySQL API и использовать ее для более быстрой разработки Вашего клиента...



by **psycho-coder**

www.programmersforum.ru

Немного теории

```
// Дескриптор соединения.
// Структура, содержащая HANDLE для 1 подключения к серверу
MYSQL mysql;

// Дескриптор результирующей таблицы
MYSQL_RES *res;

// Массив полей текущей строки
MYSQL_ROW row;

// Структура, которая содержит всю информацию,
// касающуюся отдельного поля таблицы
MYSQL_FIELD *field;
```

Функции, которые нам понадобятся:

```
MYSQL *mysql_init(MYSQL *mysql); // Функция инициализации
```

Где соответственно host – компьютер, на котором запущена СУБД MySQL, user – имя юзера для подключения, passwd – пароль, db – название предполагаемой для использования базы данных, port – порт, unix_socket – сокет или pipe-канал, который необходимо использовать.

```
MYSQL *mysql_real_connect(MYSQL *mysql, const char *host,
                           const char *user, const char *passwd,
                           const char *db, unsigned int port,
                           const char *unix_socket, unsigned int
                           client_flag)
```

client_flag может принимать несколько значений:

- **CLIENT_COMPRESS** – используется сжатие;
- **CLIENT_FOUND_ROWS** – возвращать число найденных строк;
- **CLIENT_IGNORE_SPACE** – делает все имена функций зарезервированными словами;
- **CLIENT_INTERACTIVE** – разрешает interactive_timeout секунд бездействовать (вместо wait_timeout) перед закрытием подключения;
- **CLIENT_NO_SCHEMA** – запрещает синтаксис вида “db_name.tbl_name.col_name” (имя_базы_данных.имя_таблицы.имя_колонки). Используется для ODBC;
- **CLIENT_ODBC** – устанавливает то, что это клиент ODBC;
- **CLIENT_SSL** – используется защищенный протокол SSL.

Мы флагами пользоваться не будем. Функция, выполняющая запрос и возвращающая строку с описанием ошибки:

```
int mysql_query(MYSQL *mysql, const char *query);

char *mysql_error(MYSQL *mysql);

// функция, которая получает все строки результата запроса
// и хранит их в буфере-клиенте
MYSQL_RES * mysql_store_result(MYSQL *mysql);

// Получает количество строк в результате запроса
my_ulonglong mysql_num_rows(MYSQL_RES *res);

// Получает количество полей (столбцов) в результате запроса
unsigned int mysql_num_fields(MYSQL_RES *res);

// Заполняет массив полей для текущей строки
MYSQL_ROW mysql_fetch_row(MYSQL_RES *result);

// Заполняет структуру для текущего поля (fieldnr)
MYSQL_FIELD *mysql_fetch_field_direct(MYSQL_RES *res,
                                       unsigned int fieldnr);
```

Начальная «настройка»

Для работы в Builder необходимо конвертировать `libmysql.lib`. Для этого нужно открыть консоль и набрать следующее:

```
C:\>"C:\Program Files\Borland\CBuilder6\Bin\coff2omf.exe"
-lib:st "C:\Program Files\Borland\CBuilder6\Lib\libmysql.lib"
"C:\Program Files\Borland\CBuilder6\Lib\libmysql_.lib"
```

Здесь:

"C:\ProgramFiles\Borland\CBuilder6\Lib\libmysql.lib" – оригинальная библиотека,
 "C:\ProgramFiles\Borland\CBuilder6\Lib\libmysql_.lib" – конвертированная библиотека.

У каждого пути будут свои. Также, в папке с программой (или в "C:\ProgramFiles\Borland\CBuilder6\Lib\") должны быть `libmysql.lib`, а для VS `libmysql.dll`. Заголовочные файлы можно бросить в папку с программой или в "C:\Program Files\Borland\CBuilder6\Include\". Для VS "C:\Program Files\Microsoft Visual Studio X.0\VC\include". Где X – версия VS.

В среде MS VC++ можно использовать библиотеку без конвертации, т.е. `libmysql.lib`. Все заголовочные файлы могут быть в папке с программой, но тогда нужно подключать их локально.

Есть замечания для VC++ WinForms. Так как типы `String^` и `char[]` несовместимы, то для конвертирования из `String^` в `char[]` можно использовать следующие функции (взято из MySQL++):

```
private: String^ ToUCS2(const char* utf8)
{
    try
    {
        return gcnew String(utf8, 0, strlen(utf8),
                           System::Text::Encoding::Default);
    }
    catch(...)
    {
        return "";
    }
}
```

```
private: Void ToUTF8(char* pcOut, int nOutLen, String^ sIn)
{
    try
    {
        array^ bytes = System::Text::Encoding::Default
            ->GetBytes(sIn);

        nOutLen = Math::Min(nOutLen - 1, bytes->Length);
        System::Runtime::InteropServices::Marshal::Copy(bytes, 0,
            IntPtr(pcOut), nOutLen);

        pcOut[nOutLen] = '\0';
    }
    catch (...)
    {
        pcOut[nOutLen] = '\0';
    }
}
```

Пример использования

```
const int buf = 512;
char host[buf];

// Перевод из String^ в char[]
ToUTF8(host, buf, hostText->Text);

// Перевод из char* в String^
String ^tmp = ToUCS2(mysql_error(&mysql));
```

Вот все необходимое для работы **[1]**:

- `libmysql_lib.rar`
- `libmysql.lib.rar`
- `LibMySQL.dll.rar`
- `include.rar`

Вывод в консоль

Рассмотрим вывод таблиц в консоли. Интерфейс, правда, не супер, но для практики, думаю, хватит. В коде есть комментарии:

```
#define __LCC__ // Объявляем директиву без которой программа
               // не может работать. Можно конечно подключить
               // windows.h, но это будет не красиво

#pragma comment(lib, "libmysql_.lib") // подключаем библиотеку
#include // Заголовочный файл с описанием функций

void mysql(const char query[])
```



```

{
    MYSQL mysql;           // Дескриптор соединения
    MYSQL_ROW row;         // Массив полей текущей строки
    MYSQL_RES *res;        // Дескриптор результ.таблицы
    char host[] = "localhost"; // хост
    char user[] = "admin";  // пользователь
    char passwd[] = "admin"; // пароль
    char db[] = "library";  // название базы данных
    int port = 0;           // порт. Если порт у сервера
                           // MySQL не по умолчанию (3306),
                           // то нужно указывать конкретный
                           // номер порта

    mysql_init(&mysql);     // Инициализация
    mysql_real_connect(&mysql,
                      host,
                      user,
                      passwd,
                      db,
                      port,
                      NULL, 0); // соединение

    // запрос. Если ошибок нет, то продолжаем работу
    if (mysql_query(&mysql, query) > 0)
    {
        // Если была ошибка, ...

        printf("%s", mysql_error(&mysql)); // ... выведем ее и
        return;                          // завершим работу
    }

    res = mysql_store_result(&mysql); // Берем результат,
    int num_fields = mysql_num_fields(res); // количество полей
    int num_rows = mysql_num_rows(res); // количество строк
    for (int i = 0; i < num_fields; i++) // названия полей
    { // Получение названия текущего поля

        field = mysql_fetch_field_direct(res, i);
        printf("| %s |", field->name);
    }
    printf("\n");

    for (int i = 0; i < num_rows; i++) // Вывод таблицы
    {
        row = mysql_fetch_row(res); // получаем строку
        for (int l = 0; l < num_fields; l++)
            printf("| %s |", row[l]); // Выводим поля
        printf("\n");
    }

    printf("Count records = %d", num_rows); // Вывод инфо о
                                           // кол-ве записей

```

```

mysql_free_result(res); // Очищаем результаты

mysql_close(&mysql);     // Закрываем соединение
}

int main()
{
    mysql("SELECT * FROM t_mid_author"); // Запрос

    getch();              // ждем нажатие клавиши

    return 0;
}

```

Графический интерфейс

Мутим простейший интерфейс (см. рисунок 1). Кидаем на форму: TLabel (в свойство Caption пишем хост, порт и т.д.), TEdit (названия TEdit'ов: hostText, userText, passText, dbText и portText), TButton (в свойства Caption пишем «Пошел!» и «Заккрыть»), Tmemo, TStringGrid.

Подключаем заголовочные файлы, библиотеку и объявим одну константу:

```

#define __LCC__
#include

#pragma comment(lib, "libmysql.lib") // Для Builder 6
#pragma comment(lib, "libmysql.lib") // Для MS VC++

// Для других сред программирования не пробовал

const int buf = 512;

```

Обработчик кнопки «Заккрыть», думаю, понятен. А в обработчике кнопки «Пошел!» пишем следующее:

```

/* Проверим, что все данные введены и сам запрос (Memo1) */
if (hostText->Text.IsEmpty() || userText->Text.IsEmpty() ||
    passText->Text.IsEmpty() || dbText->Text.IsEmpty() ||
    portText->Text.IsEmpty() || Memo1->Text.IsEmpty())
{
    MessageBox(this->Handle, "Не все поля заполнены!",
               "Ошибка!",
               MB_OK | MB_ICONERROR);

    return;
}

// Тут Вам все должно быть знакомо
MYSQL mysql;

```

```

MYSQL_ROW row;
MYSQL_RES *res;
MYSQL_FIELD *field;

/* Объявляем массивы для работы */

char host[buf];
char user[buf];
char passwd[buf];
char db[buf];
char query[buf];
int port = portText->Text.ToInt();
int num_fields = 0;
int num_rows = 0;

/* Инициализируем имя хоста, пользователя, пароль и БД */

strcpy(host, hostText->Text.c_str());
strcpy(user, userText->Text.c_str());
strcpy(db, dbText->Text.c_str());
strcpy(passwd, passText->Text.c_str());
strcpy(query, Memo1->Text.c_str()); /**/

```

```

mysql_init(&mysql);

if (!mysql_real_connect(&mysql,
                        host,
                        user,
                        passwd,
                        db,
                        port,
                        NULL, 0))

{ /* Пробуем подключиться, если ошибка, то сообщим о ней */
    MessageBox(this->Handle, mysql_error(&mysql), "Error!",
               MB_OK | MB_ICONERROR);
    return;
}

if (mysql_query(&mysql, query) > 0)
{ /* Пробуем выполнить запрос, если не верен, то ошибка */
    MessageBox(this->Handle, mysql_error(&mysql), "Error!",
               MB_OK | MB_ICONERROR);
    return;
}

```

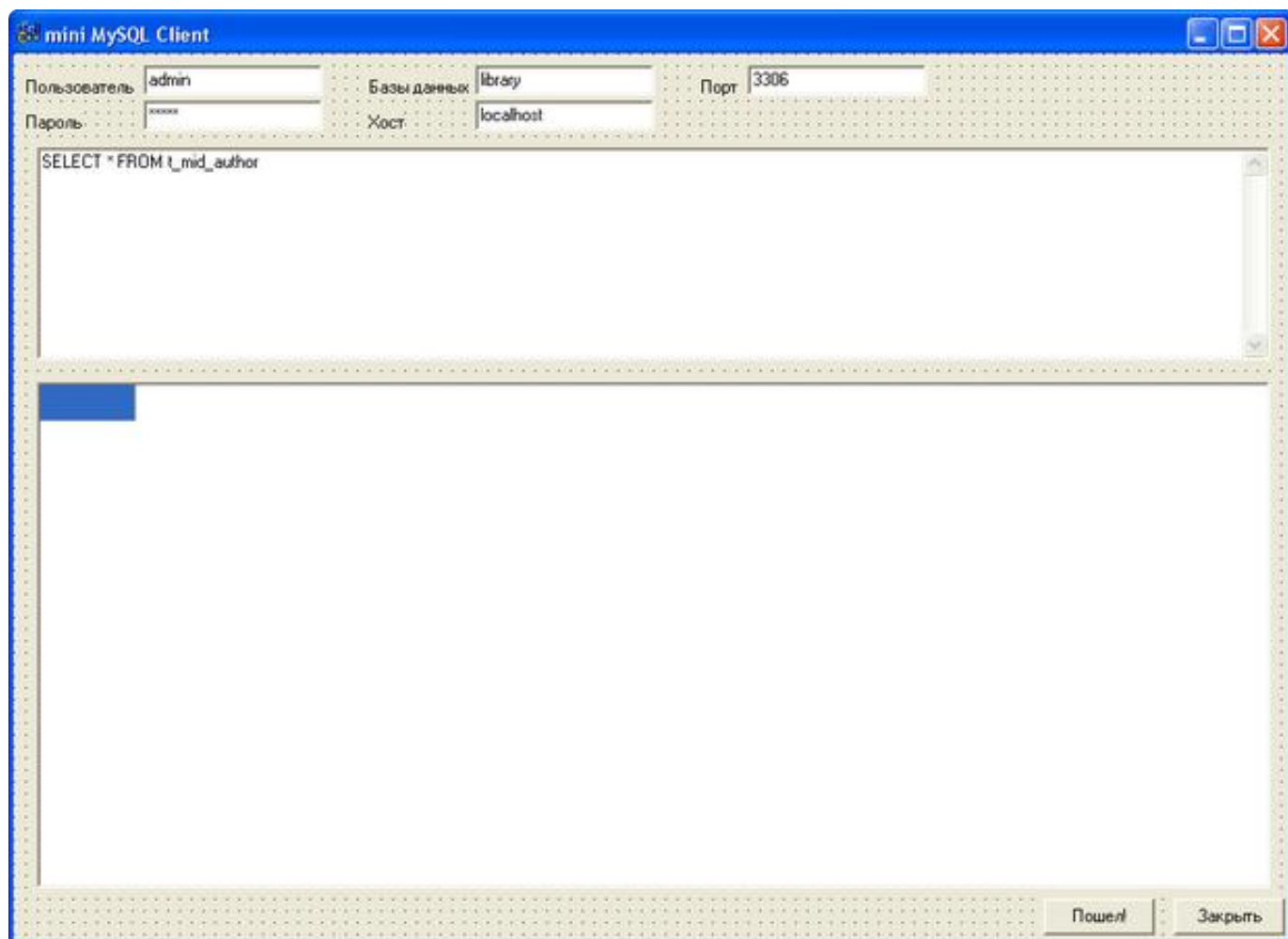


Рис. 1. Создание тестовой формы

```

}

// Получаем результат
res = mysql_store_result(&mysql);

/* Устанавливаем кол-во строк в таблице и сохр.кол-во строк */
StringGrid1->RowCount = num_rows = mysql_num_rows(res);
/* Устанавливаем кол-во полей и сохр.это кол-во столбцов */
StringGrid1->ColCount = num_fields = mysql_num_fields(res);
StringGrid1->FixedRows = 1;          // Фикс.первую строку
for (int i = 0; i < num_fields; i++) // Вывод названий полей
{
    field = mysql_fetch_field_direct(res, i);
    // В первую строку, которую мы зафиксировали
    StringGrid1->Cells[i][0] = field->name;
}
for (int i = 1; i < num_rows; i++) // Вывод рез-а запроса
{
    row = mysql_fetch_row(res);      // Получаем строку и
                                     // вывод по ячейкам
    for (int l = 0; l < Cells[l][i] = row[l];
}

mysql_free_result(res);              // Освобождаем память
mysql_close(&mysql);                 // Закрываем соединение

```

Вот и все. Пишем запрос – и «Пошел!».

Графический интерфейс. Специальный проект

Теперь напишем «специальный» клиент для базы данных «Библиотека». Для начала создадим базу данных:

```

# create.sql
# Создаем базу данных
CREATE database lib;
# Переключаемся на нее
use lib;
# Добавляем пользователя admin с паролем admin и связываем его
с базой library
GRANT ALL ON lib.* TO 'admin'@'%' IDENTIFIED BY 'admin';
# Создаем таблицу книги
CREATE TABLE IF NOT EXISTS t_books
(
    ID INT(6) UNSIGNED NOT NULL AUTO_INCREMENT,
    Title CHAR(150) NOT NULL,
    FIO CHAR(150) NOT NULL,
    PRIMARY KEY(ID),
    KEY(Title)
);

```

Интерфейс формы у меня получился такой (см. рисунок 2).

Кинем на форму следующие компоненты: TLabel и Tedit (по 5 штук), ListBox (список книг), GroupBox и в нем 2 TLabel и 2 Tedit (название книги и автор), несколько Tbutton (их желательно назвать как у меня («Удалить, Изменить, Добавить, Подключиться, Закреть»), BitBtn (кнопка обновления данных), TTimer (для проверки соединения).

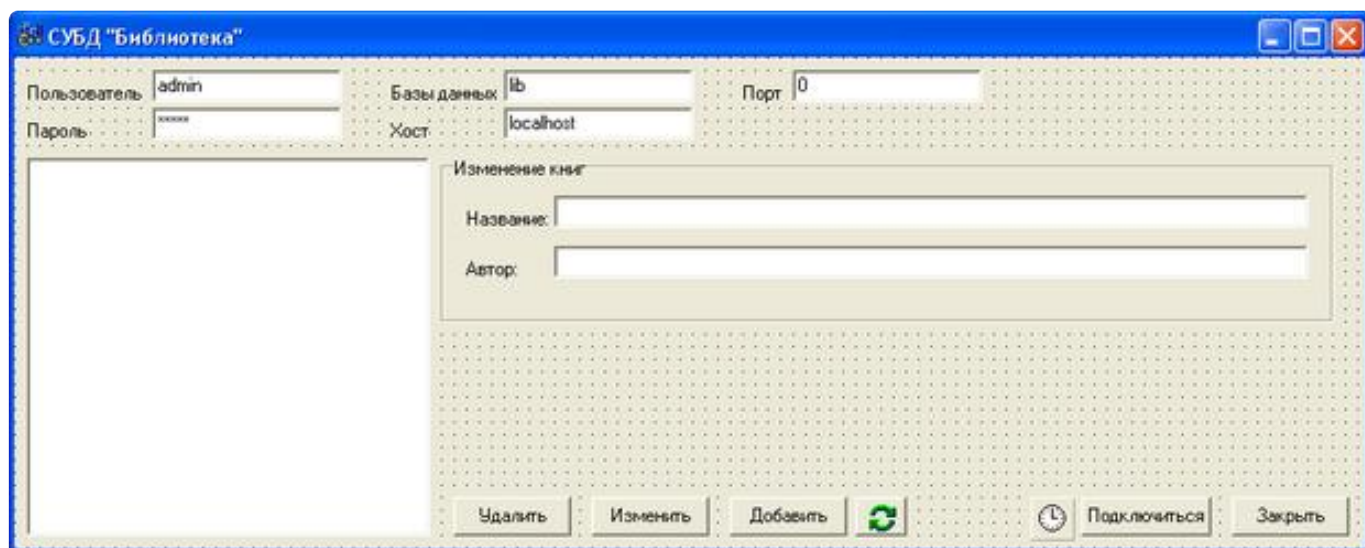


Рис. 2. Создание специального проекта

Все подробности на скрине. Объявляем необходимые переменные и подключаем все что нужно:

```
#define __LCC__

#pragma comment(lib, "libmysql.lib")

#include

const int buf = 512;    // Буфер

bool connected = false; // Есть соединение или нет

int *arrIDs = NULL;    // Массив идентификаторов

int ID = 0;            // Идентификатор текущей книги

MYSQL mysql;          // Дескриптор соединения

MYSQL_RES *res;        // Структура результатов

MYSQL_ROW row;         // Массив строк результата

/* Объявляем массивы для работы */

char host[buf];
char user[buf];
char passwd[buf];
char db[buf];
int port = 0;

/* Настройки таймера*/

Enabled := false;      // Выключен
Interval := 5000;      // 5 секунд

/* Код обработчика таймера*/

if (mysql_ping(&mysql) > 0)    // Если соединение разорвано...
{
    connected = false;        // ставим флаг дисконнекта
    if (arrIDs) delete []arrIDs; // Очищаем массив
    Timer1->Enabled = false;    // ... выключаем таймер
    MessageBox(this->Handle, "Соединение с сервером потеряно",
        "Ошибка!",
        MB_OK | MB_ICONERROR); // Выведем сообщение лоб этом
    Button5->Click();          // пуск повторного соединение
}

}
```

Далее обработчики кнопок. Код кнопки «Подключиться»:

```
if (connected) return; // Если уже соединены, то выходим

/* Проверим, что все данные были введены */

if (hostText->Text.IsEmpty() || userText->Text.IsEmpty() ||
    passText->Text.IsEmpty() || dbText->Text.IsEmpty() ||
```

```
portText->Text.IsEmpty())
{
    MessageBox(this->Handle, "Не все поля заполнены!", "Ошибка!",
        MB_OK | MB_ICONERROR);

    return;
}

/* Инициализируем имя хоста, пользователя, пароль, порт, БД */
strcpy(host, hostText->Text.c_str());
strcpy(user, userText->Text.c_str());
strcpy(db, dbText->Text.c_str());
strcpy(passwd, passText->Text.c_str());
port = portText->Text.ToInt();
mysql_init(&mysql); // Инициализация дескриптора

if (!mysql_real_connect(&mysql,
                        host,
                        user,
                        passwd,
                        db,
                        port,
                        NULL, 0))
{ /* Попробуем подключиться, если ошибка, то сообщим об этом */
    MessageBox(this->Handle, mysql_error(&mysql), "Error!",
        MB_OK | MB_ICONERROR);

    return;
}

connected = true;        // Соединены
Timer1->Enabled = true; // Порверка соединения каждые 5 сек.
BitBtn1->Click();        // Обновить список
```

Код кнопки «Обновить»:

```
// Обновление списка книг

if (!connected) return; // Если соединения нет, то выходим

/* Так как, мы знаем что нам нужно, то и запрос будет
статическим. Получим названия всех книг и идентификаторов, потом
заполним ими ListBox. Сортируем в порядке возрастания по
названию книги */

if (mysql_query(&mysql, "SELECT ID, Title FROM t_books ORDER BY
    Title") > 0)
{ // Проверка на ошибки
    MessageBox(this->Handle, mysql_error(&mysql), "Ошибка!",
        MB_OK | MB_ICONERROR);
```



```

return;
}

ListBox1->Clear();           // Очистка списка
if (arrIDs) delete []arrIDs; // Очистка массива

// Заполняем структуру
res = mysql_store_result(&mysql);
// Получаем количество записей
int count = mysql_num_rows(res); // Получаем количество строк
arrIDs = new int[count];         // Инициализируем массив

for (int i = 0; i < count; i++)
{
    // Получаем строку
    row = mysql_fetch_row(res);
    // Заполняем массив
    arrIDs[i] = StrToInt(row[0]);
    // Добавляем в ListBox название книги
    ListBox1->Items->Add(row[1]);
}

mysql_free_result(res);      // Освобождаем ресурсы

```

Специальный проект. Продолжение

Код кнопки «Добавить»:

```

// Добавление книг
// Если соединения нет или поля ввода пустые, то выходим
if (!connected && (bookText->Text.IsEmpty() || authorText
    ->Text.IsEmpty())) return;
// Формируем запрос на добавление книги
AnsiString tmp = "INSERT INTO t_books (ID, Title, FIO) VALUES
(NULL, '\" + bookText->Text + \"', '\" + authorText->Text + \"')";

char query[buf];           // Переменная для запроса
strcpy(query, tmp.c_str()); // Ковертируем в нужный формат

// Запрос. Если ошибки есть, то выводим их и выходим
if (mysql_query(&mysql, query) > 0)
{
    MessageBox(this->Handle, mysql_error(&mysql), "Ошибка!",
        MB_OK | MB_ICONERROR);
    return;
}

```

```

// Вывод сообщения о том, что все хорошо
MessageBox(this->Handle, "Книга добавлена!", "", MB_OK |
    MB_ICONINFORMATION);

bookText->Clear();
authorText->Clear();
BitBtn1->Click(); // Обновим данные

```

Код кнопки «Изменить»:

```

// Изменение книг
// Если соединения нет и книга не выбрана, то выходим
if (!connected && ID < 1) return;

AnsiString tmp = "UPDATE t_books SET Title = '\" + bookText
    ->Text + \"', FIO = '\" + authorText->Text + \"'
    WHERE ID = " + IntToStr(ID);

char query[buf];
strcpy(query, tmp.c_str());

if (mysql_query(&mysql, query) > 0)
{
    MessageBox(this->Handle, mysql_error(&mysql), "Ошибка!",
        MB_OK | MB_ICONERROR);

    return;
}

MessageBox(this->Handle, "Информация о книге обновлена", "",
    MB_OK | MB_ICONINFORMATION);

```

Код кнопки «Удалить»:

```

// Удаление книг. Если соединения нет и нет выделенной книги
// для удаления, то выходим
if (!connected && ID < 1) return;

// Подтверждение удаления
if (MessageBox(this->Handle, "Удалить?", "Удаление",
    MB_YESNO | MB_ICONQUESTION) != 6) return;
// Формируем запрос на удаление
AnsiString tmp = "DELETE FROM t_books WHERE ID = " +
    IntToStr(ID);

char query[buf];
// Приводим его
strcpy(query, tmp.c_str());
// Выполняем
if (mysql_query(&mysql, query) > 0)
{

```

```

        MessageBox(this->Handle, mysql_error(&mysql), "Ошибка!",
            MB_OK | MB_ICONERROR);
        return;
    }

    MessageBox(this->Handle, "Книга удалена", "", MB_OK |
        MB_ICONINFORMATION);

    BitBtn1->Click();

```

Выбор книги из ListBox будет производиться по двойному щелчку:

```

// Выбор книги
// Если соединения нет, то выходим
if (!connected) return;

AnsiString tmp = "SELECT FIO FROM t_books WHERE ID = "
    + IntToStr(arrIDs[ListBox1->ItemIndex]);
char query[buf];
strcpy(query, tmp.c_str());
if (mysql_query(&mysql, query) > 0)
{
    MessageBox(this->Handle, mysql_error(&mysql), "Ошибка!",
        MB_OK | MB_ICONERROR);
    return;
}

// Получаем результаты
res = mysql_store_result(&mysql);
row = mysql_fetch_row(res);

// Получаем идентификатор книги
ID = arrIDs[ListBox1->ItemIndex];

// Выводим название книги
bookText->Text = ListBox1->Items->Strings[ListBox1-
    >ItemIndex];

// Выводим автора
authorText->Text = row[0];

mysql_free_result(res);

```

И по закрытии программы написать:

```

// Закрытие программы. Если соединены, то разрываем соединение
if (connected)
{
    mysql_close(&mysql);
    if (arrIDs) delete []arrIDs;
}

```

Заключение

На этом все*. Все упомянутые библиотеки, модули и тестовые проекты с подробными комментариями приложены в виде ресурсов непосредственно в архиве «Журнал клуба программистов. Шестой выпуск».

Ресурсы

- Все исходники статьи и скрипт SQL <http://programmersforum.ru/attachment.php?attachmentid=13935&d=1248258141>
- Тема на форуме <http://programmersforum.ru/showthread.php?t=59147>
- Официальный сайт MySQL <http://www.mysql.com>
- Документация по MySQL на английском <http://dev.mysql.com/doc>
- Мануал по MySQL на русском <http://www.php.ru/mysql>
- Видеокурсы MySQL от Евгения Попова <http://php-mysql-video.ru>
- Сообщество ЖЖ, посвященное вопросам MySQL http://community.livejournal.com/ru_mysql
- Сайт PHPMyAdmin - приложения для работы с базами данных MySQL <http://php-myadmin.ru>

* Полезности в MySQL.

1- чтобы разрешить доступ к MySql не только с localhost надо просто удалить строчку в файле "my.cnf" (конфиг MySQL):

`bind-address = ...`

...или поставить впереди строки "дизел" (#);

2- альтернативный вывод консольного клиента MySQL:

```
mysql> select * from category \G
```

```
***** 1. row *****
```

```
id: 1
```

```
name: Все товары
```

```
sort: 2
```

```
image:
```

```
***** 2. row *****
```

```
id: 2
```

```
name: Идеи для подарков
```

```
sort: 5
```

```
image: /upload/files/category/2.jpg
```

```
2 rows in set (0.00 sec)
```

3- Смена MySQL пароля для пользователя debian-sys-maint:

```
UPDATE mysql.user SET Password = PASSWORD( 'новый пароль' )
WHERE user = 'debian-sys-maint' AND host = 'localhost';
FLUSH PRIVILEGES;
```

Как бы выглядела учебная программа в ВУЗе, который бы готовил Настоящих Админов...

Первый курс:

Изучение принципиальных схем и принципов работы чайников, кофеварок, кофемолок, телевизоров, телефонов сотовых и обычных, теория и практика работы с паяльником, упражнения с паяльником в движении и в строю, практические ночные занятия с осциллографом, разборка-сборка компьютера любой конфигурации на зачетное время из любых доступных деталей (где украдешь недоступные, преподавателей не волнует). Обязательная физическая подготовка: бег по лестницам и пересеченной местности с полной выкладкой (системник, блок питания, монитор, ноут, 2 мышки, клавиша, 55 дисков, 100 м витой пары, коннекторы, инструмент обжимной и на всякий случай шанцевый), ползание по-пластунски в ограниченном пространстве.

Второй курс:

Dos, Windows, Unix, Linux, OS/2, FreeBSD изнутри и снаружи. Установка, настройка, создание серверов и рабочих станций, создание индивидуальных рабочих мест и глобальных офисных систем, единых бухгалтерий по всему миру и локальных подземных командных центров, автоматизация биржевых обменов и систем спутникового слежения, системы защиты информации и в случае несанкционированного доступа - автоматического адекватного ядерного возмездия. Изучение всех созданных и перспективных офисных приложений, имеющихся и возможных в будущем бухгалтерских и

банковских программ, решение прикладных задач по налогообложению, трудовому и уголовному законодательству. Компьютерный и экономический шпионаж и контршпионаж. Зачет по знанию портов при условии обязательного похмелья и трехсуточного бодрствования. Допуском на экзамены служит наличие сертификата от Гейтса и поздравление с днем рождения от Торвальдса.

Третий курс:

Программирование на всевозможных (созданных и перспективных) языках. Создание домашних страничек, сайтов и порталов, программ управления банками и атомными электростанциями, холодильниками и электробритами. Принимается экзамен в устной форме, экзаменуемый разговаривает на языке программирования (по выбору преподавателя). Для оживления экзамена рекомендуется использовать чертежи и наброски, выполненные в Авто-, Мат-, Архикадах, раскрашенные в кореле и фотошопе и желательно анимированные как минимум в трех разных 3D приложениях. В качестве практического задания - пантомима на тему объектно-ориентированного программирования.

Четвертый курс:

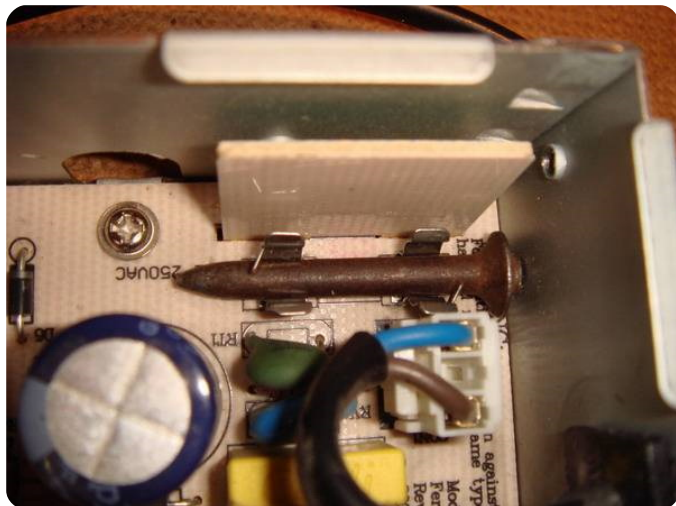
Теоретическая и прикладная психология (знать Карнеги, как общевоинской устав!), лингвистика общая и специальная (все виды сленга и арготизмов, феня, Албанский, эльфийский, орочий и бухгалтерский языки), теория и практика педагогики, методики обучения, возрастная



психология, психология власти, психология толпы, практический гипноз и измененные состояния сознания.

Пятый курс:

Магия, колдовство, шаманизм, астрология и эзотерика, углубленное изучение начертательной, рунической, ритуальной магии. Теология и некромантия. Навыки допроса и пыточного дела. Зачетный ритуал вокруг сервера и допрос мертвого немного и тупого пользователя. После окончания выдается диплом Сверх-Админа. Этот админ действительно сможет все, а что не учил - то сам поймет и сделает. Вопрос только один - сколько ему надо будет платить?



Несколько фраз для роботов, изучающих латинский язык:

- 1) Non in axungia sola vivet robotum / Не смазкой единой жив робот;
- 2) Dura Ohmi lex, sed lex / Суров закон Ома, но это закон;
- 3) Robotum proponit, sed Programmator disponit / Робот предполагает, а Программист располагает;
- 4) In nomine Patris et Filii et Vis Electricae Sanctae / Во имя Отца и Сына и Святого Электричества;
- 5) Robotum sum, humani aliquam-multo a me non alienum puto / Я робот, и многое из человеческого мне не чуждо.

```

program Елочка;

Begin

Лес.елочка.create();

While not Лес.елочка.dead do
Begin

лес.елочка.age := лес.елочка.age 1;

if зима or лето then
begin

лес.елочка.plain := true;
лес.елочка.color := cl_green;

end;

метель.filename:="'D:\Music\NSYNC - Bye Bye Bye.mp3'";
метель.open;
sleep(1000);
метель.play;
мороз.укутать(лес.елочка);
SendMessage(H, лес.елочка.handle, 0, 0);

if h.uit = wm_замерзла then
dead := true;

end;

лес.заяц.create;
лес.заяц.трусливый := true;
лес.заяц.color := cl_gray;

x := лес.елочка.left;
y := лес.елочка.top;

While x
Begin

лес.заяц.left := x;

inc(x);

лес.заяц.top := y round( abs(sin(x*pi/180)*10) );

end;

Winexec("'wolf3d.exe /evil'",0);

лес.елочка.cuttoclipboard;
Праздник.pastefromclipboard;
Праздник.елочка.skin.loadfromfile("'нарядная.jpg'");

For i:=1 to SizeOf(Праздник.елочка.радость) do
Праздник.Дети.Настроение Праздник.елочка.радость;

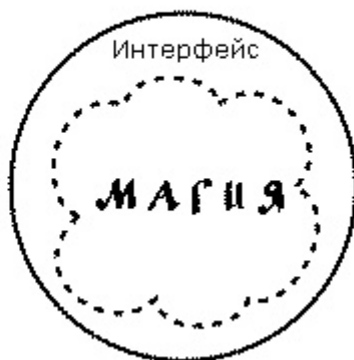
End.

```

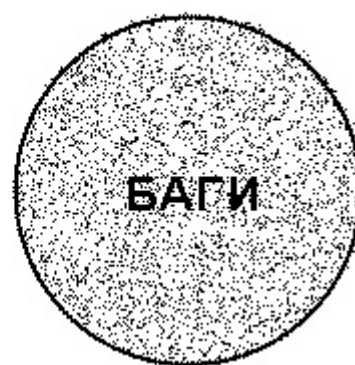

Ваша программа



Как ее видит пользователь



Как ее видит тестер



Теорема о Зарплатной плате утверждает, что инженеры и ученые никогда не смогут зарабатывать так же много, как бизнесмены, руководители и торговцы. Доказательство этой теоремы может быть продемонстрировано сведением к простому математическому уравнению. Уравнение основывается на двух постулатах:

Постулат 1. Знание – Сила (power)

Постулат 2. Время – Деньги.

Каждый инженер знает, что:

$\text{Сила (power)} = \text{Работа} / \text{Время}$

Так как:

$\text{Знание} = \text{сила(power)} \text{ и } \text{Время} = \text{Деньги}$

То получаем:

$\text{Знание} = \text{Работа} / \text{Деньги}$

Отсюда легко получить, что:

$\text{Деньги} = \text{Работа} / \text{Знание}$

Таким образом, если «Знание» стремится к нулю, «Деньги» стремятся к бесконечности, безотносительно к величине затраченной работы, даже если это величина мала. С другой стороны, когда «Знание» стремится к бесконечности, «Деньги» стремятся к нулю, даже если величина затраченной работы значительная.

Очевидное заключение: чем меньше вы знаете, тем больше денег зарабатываете.

Тот из вас, кто при прочтении этого текста испытывал трудности в понимании написанного, должен зарабатывать много денег!

```
A problem has been detected and windows has been shut down to prevent damage
to your computer.

DRIVER_IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen this stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced startup options, and then
select safe Mode.

Technical information:

*** STOP: 0x00000001 (0x00000000,0x00000002,0x00000000,0xf865a89)

***      gv3.sys - Address f865a89 base at f865000, DateStamp 3dd991eb

Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further
assistance. skvo-store.ru
```

...на одном из радиофорумов:

«Как подать объявление? Очень просто. Примерный текст объявления:

- Продаю трансивер свой, конструкция лучше японского – 50 у.е.
- Продаю антенну на все диапазоны. Титан. Позолоченная – 10 у.е.
- Продаю золотую тещу, почти даром, с прибабахами – 1 у.е.

И так далее. В конце текста укажите свои реквизиты и телефон. Уважайте труд админа!!!»

