Министерство образования и науки Республики Казахстан

Павлодарский государственный университет им. С. Торайгырова

Кафедра Вычислительная техника и программирование

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам

дисциплина «Инструментальные средства разработки программ»

специальность 050704 «Вычислительная техника и программирование»

Павлодар

УТВЕРЖДАЮ
Декан ФФМиИТ
_____С. К. Тлеукенов
“___”_____200_г.

Составитель: ст. преподаватель _____ м.т.т. Варакута Алёна Александровна
(подпись)

Кафедра «Вычислительная техника и программирование»

Методические указания к лабораторным работам

дисциплина «Инструментальные средства разработки программ»

специальность 050704 «Вычислительная техника и программирование»

Рекомендовано на заседании кафедры
«____» _____200_ г., Протокол №___

Заведующий кафедрой _____О.Г. Потапенко
(подпись)

Одобрено методическим советом факультета ФМиИТ
«____» _____200_ г., Протокол №___

Председатель МС _____ А. З. Даутова
(подпись)

Техническое оснащение лабораторных работ

Лабораторные и практические работы выполняются в компьютерных классах. Работы выполняется индивидуально студентом за отдельным компьютером.

Программное обеспечение, необходимое для выполнения лабораторных и практических работ – Delphi, C++, Visual Basic, NC и FAR, Visio.

Требования по безопасности и охране труда при выполнении лабораторных работ

Во время выполнения лабораторных и практических работ студенты должны соблюдать технику безопасности и правила поведения в компьютерном классе. Инструктаж по технике безопасности студенты проходят на первом практическом занятии в компьютерном классе. Отметка о прохождении инструктажа имеется в журнале по технике безопасности класса.

Лабораторная работа №1

Тема - FAR. Разработка и компиляция плагинов. Подключение плагинов к FAR

Теоретические сведения

Принцип построения и подключения динамических модулей (plug-in и их использование).

Для начала откроем директорию в которой установлен FAR и поищем то, что могло бы нам пригодится при разработке плагинов. Здесь мы обнаружим архив PlugDoc.rar и распакуем его в отдельный каталог. Теперь можно более детально разобраться с содержимым этого архива:

- Папка Examples – здесь находятся примеры уже имеющихся расширений (Временная панель, Архиватор, Просмотр сетевого окружения и т.п.)
- В папке Headers.c, как видно из названия хранятся заголовки и объявления функций необходимых для написания плагинов на C.
- Папка Headers.pas включает модули с теми же объявлениями для языка Pascal.

Из других полезных вещей, на которые Вам следует обратить внимание можно назвать файл справки plugins.hlp в котором достаточно подробно описаны все возможности по использованию функций FAR-a, к которым имеет доступ плагин.

И хотя для беглого ознакомления с техникой разработки плагинов для FAR-a нам будет достаточно заглянуть в исходник HelloWorld.pas расположенный в Examples\HelloWorld. Но его точно будет недостаточно для написания более менее полезных модулей.

Как уже говорилось в каталоге Expamples\HelloWorld есть исходник, простенького плагина с которого мы и начнем изучение расширений для FAR-a.

Итак скомпилировав файл HelloWorld.pas (а это можно сделать написав в командной сроке “dcc32 helloworld.pas”) мы получим файл библиотеки HelloWorld.dll. который вместе с дополнительными файлами (*.lng и *.hlf) поместить в отдельную папку каталога Plugins который размещается в директории FAR-a (к примеру C:\Program Files\Far\Plugins\Simple\). Далее, следуя рекомендациям из справки и дополнительных файлов из архива PlugDoc.rar, мы закрываем FAR и открываем его снова (дабы он обнаружил наш вновь созданный плагин). Хотя лично я просто открываю еще одну его копию. Теперь для просмотра результатов наших титанических усилий мы в заново открытом FAR-е жмем F11 и внимательно изучаем список подключенных модулей (рис.1). Наш плагин в этом списке выведен под названием “Здравствуй, Мир!”. При нажатии на него появляется следующее предупреждение.

Реализация этого модуля довольно проста. Если взглянуть на исходник HelloWorld.pas то там мы увидим:

В самом конце – секцию экспортируемых функций SetStartupInfo, GetPluginInfo, OpenPlugin;

Первая из этих функций вызывается один раз при загрузке плагина. Здесь необходимо как минимум сохранить передаваемую ей ссылку на внутренние функции FAR-а при помощу которых и будет создаваться и вызываться сообщение.

При вызове второй функции мы возвращаем ей информацию о нашем модуле (название плагина в списке, флаги его присутствия в списке при разных экранных ражимах - Viewer, Editor, FilePanel…).

Последняя функция отвечает за собственно отображения сообщения при запуске модуля из списка.

Остановимся на каждом пункте более подробно.

Процедура SetStartupInfo.

```
procedure SetStartupInfo(var psi: TPluginStartupInfo); stdcall;
begin
  Move(psi, FARAPI, SizeOf(FARAPI));
end;
```

Как уже говорилось она вызывается 1 раз при загрузке плагина и передает ему всю необходимую информацию в параметре psi. Параметр psi в свою очередь содержит список всех жизненно важных функций, большинство из которых отвечают за вызов тех или иных сервисных функций самого FAR-а. Так функции Menu, Message, Dialog, Control отвечают за вызов соответственно указателей на меню, вывод информационных сообщений, создание диалогов, получение информации об имеющихся в наличии панелях (Файловая, Информационная, Временная…).Другие же функции позволяют выводить текст, справку, открывать редактор и просмотрщик файлов. Кроме этих функций используется также и другой набор из т.н. Стандартных функций (FSF:PFarStandardFunctions) в который включены функции по работе со строками (сравнение, преобразование…), с буфером обмена, с таблицами перекодировок символов и т.п.

Процедура GetPluginInfo. Здесь происходит “заполнение анкеты” нашего плагина (т.е. структуры TPluginInfo) путем указания флага и названия нашего модуля. Кроме этих параметров можно также заполнить параметры:

- DiskMenuStrings: PPCharArr – указатель на список с названиями пунктов, которые будут добавлены в меню дисков (Alt+F1,Alt+F2).
- DiskMenuNumbers: PintArr – указатель на список с клавишами быстрого вызова пункта из меню дисков. Если этот параметр равен NULL, то FAR автоматически назначит горячие клавиши для каждого пункта.
- DiskMenuStringsNumber: integer – количество строк, которые будут добавляться в меню дисков.
- PluginConfigStrings: PPCharArr – список пунктов, которые будут добавлены в меню конфигурации подключаемых модулей.
- PluginConfigStringsNumber: integer – количество этих самых пунктов.
- CommandPrefix: Pchar – префикс команд, используется для перехвата стандартной реакции FAR на командную строку с различными префиксами типа http://www.site.com, ftp://ftp.site.com и т.д.

Функция OpenPlugin вызывается каждый раз при запуске плагина тем или иным образом. Здесь размещается код, который собственно и отвечает за отображение и деятельность нашего модуля.

Здесь в первых строках идет заполнение массива строк из которых будет состоять наше сообщение. Значения этих строк берутся из стандартного языкового файла (т.е. файла с расширением lng расположенного в той же директории, что и модуль и соответствующего используемому в FAR языку). Далее происходить вызов стандартной функции Message которая и отображает окно сообщения. Присвоение результату функции значения INVALID_HANDLE_VALUE необходимо для того, чтоб потом не возится с освобождение ресурсов, которые выделяет FAR под создание новой панели для нашего плагина. В других

же случаях необходимо будет реализовать еще и процедуру ClosePlugin, которая теоретически принимает дескриптор созданный в функции OpenPlugin и освобождать соответствующие ему ресурсы.

Задание

Разработать плагин. Подключить к FAR.

Контрольные вопросы:

1.Функциональные возможности NC и FAR.
2.Различия между NC и FAR.
3.Виды плагинов. Примеры.
4.Назначение плагинов.
5.Построение плагинов.

Рекомендуемый список литературы:

1.Кенин А.М., Печенкина Н.С. IBM PC для пользователей или как научиться работать на компьютере: Научно-популярное издание/Екатеринбург: Издательство «АРД ЛТД», 1997.

2.Фленов М. Е. Программирование в Delphi глазами хакера. — СПб.: БХВ-Петербург, 2003.

Лабораторная работа №2

Тема - Архиваторы. ARJ, PKZIP.

Теоретические сведения

Общие принципы архивации. Классификация методов

Следующей большой темой является архивация данных. Как Вам известно, подавляющее большинство современных форматов записи данных содержат их в виде, удобном для быстрого манипулирования, для удобного прочтения пользователями. При этом данные занимают объем больший, чем это действительно требуется для их хранения. Алгоритмы, которые устраняют избыточность записи данных, называются алгоритмами сжатия данных, или алгоритмами архивации. В настоящее время существует огромное множество программ для сжатия данных, основанных на нескольких основных способах.

Зачем же нужна архивация в криптографии? Дело в том, что в современном криптоанализе, то есть науке о противостоянии криптографии, с очевидностью доказано, что вероятность взлома криптосхемы при наличии корреляции между блоками входной информации значительно выше, чем при отсутствии таковой. А алгоритмы сжатия данных по определению и имеют своей основной задачей устранение избыточности, то есть корреляций между данными во входном тексте.

Все алгоритмы сжатия данных качественно делятся на 1) алгоритмы сжатия без потерь, при использовании которых данные на приемной восстанавливаются без малейших изменений, и 2) алгоритмы сжатия с потерями, которые удаляют из потока данных информацию, незначительно влияющую на суть данных, либо вообще невоспринимаемую человеком (такие алгоритмы сейчас разработаны только для аудио- и видео- изображений). В криптосистемах, естественно, используется только первая группа алгоритмов.

Существует два основных метода архивации без потерь:

- алгоритм Хаффмана (англ. Huffman), ориентированный на сжатие последовательностей байт, не связанных между собой,
- алгоритм Лемпеля-Зива (англ. Lempel, Ziv), ориентированный на сжатие любых видов текстов, то есть использующий факт неоднократного повторения "слов" – последовательностей байт.

Практически все популярные программы архивации без потерь (ARJ, RAR, ZIP и т.п.) используют объединение этих двух методов – алгоритм LZH.

Алгоритм Хаффмана

Алгоритм основан на том факте, что некоторые символы из стандартного 256-символьного набора в произвольном тексте могут встречаться чаще среднего периода повтора, а другие, соответственно, – реже. Следовательно, если для записи распространенных символов использовать короткие последовательности бит, длиной меньше 8, а для записи редких символов – длинные, то суммарный объем файла уменьшится.

Хаффман предложил очень простой алгоритм определения того, какой символ необходимо кодировать каким кодом для получения файла с длиной, очень близкой к его энтропии (то есть информационной насыщенности). Пусть у нас имеется список всех символов, встречающихся в исходном тексте, причем известно количество появлений каждого символа в нем. Выпишем их вертикально в ряд в виде ячеек будущего графа по правому краю листа (рис. 1а). Выберем два символа с наименьшим количеством повторений в тексте (если три или большее число символов имеют одинаковые значения, выбираем любые два из них). Проведем от них линии влево к новой вершине графа и запишем в нее значение, равное сумме частот повторения каждого из объединяемых символов (рис.2б). Отныне не будем принимать во внимание при поиске наименьших частот повторения два объединенных узла (для этого сотрем числа в этих двух вершинах), но будем рассматривать новую вершину как полноценную ячейку с частотой появления, равной сумме частот появления двух соединившихся вершин. Будем повторять операцию объединения вершин до тех пор, пока не придем к одной вершине с числом (рис.2в и 2г). Для проверки: очевидно, что в ней будет записана длина кодируемого файла. Теперь расставим на двух ребрах графа, исходящих из каждой вершины, биты 0 и 1 произвольно – например, на каждом верхнем ребре 0, а на каждом нижнем – 1. Теперь для определения кода каждой конкретной буквы необходимо просто пройти от вершины дерева до нее, выписывая нули и единицы по маршруту следования. Для рисунка 4.5 символ "А" получает код "000", символ "Б" – код "01", символ "К" – код "001", а символ "О" – код "1".

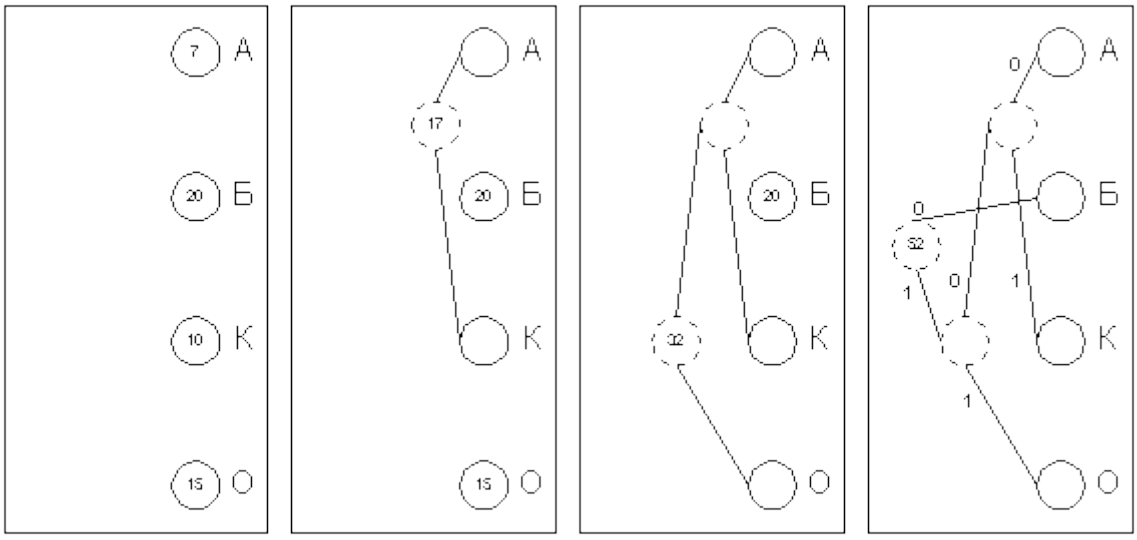

Рис.1.

В теории кодирования информации показывается, что код Хаффмана является префиксным, то есть код никакого символа не является началом кода какого-либо другого символа. Проверьте это на нашем примере. А из этого следует, что код Хаффмана однозначно восстановим получателем, даже если не сообщается длина кода каждого переданного символа. Получателю пересылают только дерево Хаффмана в компактном виде, а затем входная последовательность кодов символов декодируется им самостоятельно без какой-либо дополнительной информации. Например, при приеме "0100010100001" им сначала отделяется первый символ "Б" : "01-00010100001", затем снова начиная с вершины дерева – "А" "01-000-10100001", затем аналогично декодируется вся запись "01-000-1-01-000-01" "БАОБАБ".

Алгоритм Лемпеля-Зива

Классический алгоритм Лемпеля-Зива – LZ77, названный так по году своего опубликования, предельно прост. Он формулируется следующим образом : "если в прошедшем ранее выходном потоке уже встречалась подобная последовательность байт, причем запись о ее длине и смещении от текущей позиции короче чем сама эта последовательность, то в выходной файл записывается ссылка (смещение, длина), а не сама последовательность". Так фраза "КОЛОКОЛ_ОКОЛО_КОЛОКОЛЬНИ" закодируется как "КОЛО(-4,3)_(-5,4)О_(-14,7)ЬНИ".

Распространенный метод сжатия RLE (англ. Run Length Encoding), который заключается в записи вместо последовательности одинаковых символов одного символа и их количества, является подклассом данного алгоритма. Рассмотрим, например, последовательность "АААААААА". С помощью алгоритма RLE она будет закодирована как "(А,7)", в то же время ее можно достаточно хорошо сжать и с помощью алгоритма LZ77 : "А(-1,6)". Действительно, степень сжатия именно такой последовательности им хуже (примерно на 30-40%), но сам по себе алгоритм LZ77 более универсален, и может намного лучше обрабатывать последовательности вообще несжимаемые методом RLE.

Задание

Разработать программу, реализующую один из предложенных алгоритмов шифрования. Протестировать. Произвести анализ.

Контрольные вопросы:

1.Назначение архиваторов.
2.Параметры архиваторов.
3.Ключи архиваторов.
4.Опции архиваторов.
5.Алгоритмы архивации.

Рекомендуемый список литературы:

1. Жельников В. Криптография от папируса до компьютера. – М.: Dore Print, 1999.

Лабораторная работа №3

Тема - - Visio - построение графических моделей программ.

Теоретические сведения

Office Visio 2007 позволяет специалистам из сферы информационных технологий и бизнеса наглядно представлять, анализировать сложную информацию и обмениваться данными систем и процессов. Используя профессионально построенные диаграммы Office Visio 2007, можно лучше понять функционирование систем и процессов, быстрее разобраться в сложных данных и использовать эти знания для принятия более качественных решений по ведению бизнеса.

Программа Office Visio 2007 доступна в двух отдельных выпусках: Office Visio Professional 2007 и Office Visio Standard 2007, который имеет ту же базовую функциональность, что и Office Visio Professional, однако содержит поднабор собственных возможностей и шаблонов.

Легкое получение наглядного представления процессов, систем и данных

С помощью Office Visio 2007 можно наглядно документировать, разрабатывать и оценивать состояние бизнес-процессов и систем, пользуясь широким набором доступных диаграмм, в том числе блок-схемами деловых процессов, сетевыми диаграммами, диаграммами рабочего потока, моделями базы данных и диаграммами программного обеспечения. Чтобы сделать диаграммы еще более полезными и наглядными, можно связать их в Office Visio Professional 2007 с исходными данными.

Более простое отображение в графическом виде процессов, систем и комплексных данных благодаря новым и усовершенствованным возможностям Office Visio 2007:

Шаблоны позволяют быстро приступить к работе. В Office Visio 2007 можно легко создавать диаграммы, используя стандартные фигуры Microsoft SmartShapes и мощные возможности поиска нужной фигуры на компьютере или в Интернете. Office Visio 2007 содержит множество средств, которые позволяют специалистам из сферы информационных технологий и бизнеса создавать различные диаграммы в соответствии со своими потребностями.

Быстрый доступ к часто используемым шаблонам. Доступ к часто используемым шаблонам обеспечивается в новом режиме «Последние шаблоны» в новом окне «Приступая к работе», которое появляется при запуске Office Visio 2007.

Начало работы с примерными диаграммами. В Office Visio Professional 2007 можно с легкостью найти новые примерные диаграммы, открыв новое окно «Приступая к работе» и воспользовавшись категорией «Примеры». Примерные диаграммы, которые интегрированы с данными, позволят специалистам почерпнуть идеи по созданию собственных диаграмм, понять, как данные образуют контекст различных типов диаграмм, а также определить, какой шаблон нужно использовать.

Соединение фигур без рисования соединителей. Новая функция автосоединения в Office Visio 2007 позволяет соединять фигуры, равномерно располагать их и выравнивать одним щелчком мыши. При перемещении соединенных фигур они остаются соединенными, а соединители автоматически изменяют путь между фигурами.

Простое подключение данных к диаграммам и привязка данных к фигурам. Новая функция связывания данных в Office Visio Professional 2007 автоматически подключает диаграммы к одному или нескольким источникам данных, например к электронным таблицам Microsoft Office Excel 2007 или базам данных Microsoft Office Access 2007. Интуитивно понятные новые методы привязки данных к фигурам позволяют экономить время, заполняя значениями данных каждое свойство фигуры (также известны как данные фигур). Новый мастер автоматически обновляемых связей привязывает все фигуры диаграммы к строкам данных из подключенных источников данных.

Привлекательное оформление данных в диаграммах. Новая функция «Графические символы данных» в Office Visio Professional 2007 содержит множество параметров форматирования данных, с помощью которых можно создать привлекательное оформление данных, связанных с фигурами. Можно одним щелчком кнопки мыши отобразить поля данных в виде выносок рядом с фигурой, поместить поля в квадраты ниже фигуры или разместить поля данных непосредственно над фигурой или рядом с ней.

Быстрое обновление данных в диаграммах. Новая функция обновления данных в Office Visio Professional 2007 позволяет автоматически обновлять все данные в диаграммах, так что делать это вручную больше не нужно. Если возникают конфликты данных, они легко разрешаются с помощью области задач «Разрешение конфликтов», включенной в Office Visio Professional 2007.

Анализ сложной информации для быстрого понимания смысла данных

Office Visio Professional 2007 позволяет визуально исследовать сложную информацию для выявления основных тенденций и исключений, а также оценки смысла данных. Анализ, детализация и разные представления деловых данных позволяют лучше понять бизнес-процессы. Office Visio 2007 помогает легко выявлять ключевые проблемы, отслеживать тенденции и помечать исключения с помощью богатой библиотеки значков и флагов.

В Office Visio Professional 2007 можно выполнять анализ комплексных деловых данных с использованием новых и усовершенствованных возможностей:

Наглядное представление деловых данных. Используя сводные диаграммы, можно анализировать деловые данные, представленные в графическом виде вместо привычного статического текста и таблиц. Для более качественной оценки проблем можно создать разные представления одних и тех же данных.

Выявление проблем, отслеживание тенденций и пометка исключений. Пользователь может быстро выделить ключевые проблемы, тенденции и исключения, а также отобразить ход развития проекта. Новая функция «Графические символы данных» упрощает условное форматирование с помощью привлекательных интуитивно понятных фигур, таких как флаги и элементы данных, вывод которых будет зависеть от заданных пользователем условий.

Наглядное отображение данных проекта. Office Visio 2007 — необходимое средство для наглядного представления сложных данных проектов. Программа позволяет непосредственно из Microsoft Office Project и Microsoft Office SharePoint Server создавать отчеты, в которых отслеживаются задачи, владельцы, роли и должности по проектам, а также создавать графическое отображение комплексных структур принадлежности проекта. Отчеты автоматически обновляются при изменении данных проекта.

Эффективный обмен данными для принятия более качественных решений

Диаграммы Office Visio 2007 помогают доводить до аудитории значение данных более эффективно, чем с помощью только слов и чисел. Профессионально оформленные диаграммы Office Visio 2007 доступны даже тем пользователям, у которых нет Visio.

С помощью новых и усовершенствованных возможностей Office Visio 2007 можно повысить эффективность обмена данными, разнообразить способы их передачи и расширить доступную аудиторию:

Эффективное взаимодействие с помощью новых типов фигур и диаграмм. Теперь в Office Visio 2007 стало легче найти нужный шаблон с помощью упрощенных категорий диаграмм. Новые шаблоны Office Visio Professional 2007, такие как ITIL (Information Technology Infrastructure Library) и потоковое сопоставление значений, расширяют диапазон доступных диаграмм. Теперь можно создавать более динамичные рабочие потоки с новыми объемными фигурами рабочего потока.

Разработка профессиональных диаграмм. Новая функция Office Visio 2007 «Тема» позволяет одним щелчком кнопки мыши задать цвета и эффекты для всей диаграммы. В Office Visio 2007 используются те же цветовые темы, что и в Microsoft Office PowerPoint 2007, поэтому можно создавать профессионально оформленные диаграммы Visio, а затем использовать их в презентациях PowerPoint.

Расширение потенциальной аудитории. Сохранение диаграмм Visio в формате PDF или XPS делает их более мобильными и позволяет адресовать их более широкой аудитории. Диаграммы Visio можно просматривать как вложения сообщений в Microsoft Office Outlook 2007.

|Примечание| В программах системы Microsoft Office 2007 файл можно сохранить в формате PDF или XPS только после установки соответствующего расширения. Дополнительные сведения см. в статье Установка и использование расширения PDF или XPS.

Диаграммы, доступные каждому. Диаграммы можно сохранить как веб-страницы с элементами управления перемещением, средством просмотра данных фигур, отчетами, разными форматами изображения, а также параметрами таблицы стилей. Если затем разместить диаграммы в локальной или глобальной сети, они будут доступны каждому пользователю, который использует средство просмотра Visio с Windows Internet Explorer.

Совместная работа с диаграммами Visio. Общая рабочая область поддерживает взаимодействие со службами Microsoft Windows SharePoint Services. Диаграммы Visio, сохраненные на веб-узлах Windows SharePoint Services, могут быть открыты с веб-узла непосредственно в Office Visio 2007 и даже извлечены и возвращены из Office Visio 2007. Если диаграмма открыта с веб-узла Windows SharePoint Services, в Office Visio 2007 открывается область задач «Общая рабочая область», которая содержит все данные рабочей области, включая другие файлы, участников, задачи и связи.

Диаграммы с рукописными примечаниями. Интегрированная поддержка рукописного ввода в Office Visio 2007 позволяет естественным образом создавать и вносить примечания в существующие изображения и добавлять эскизы с помощью пера на планшетном

компьютере. Улучшения рукописного ввода и поддержка более высокого разрешения на планшетном компьютере обеспечивают истинную универсальность создаваемых решений.

Рецензирование диаграмм Visio. Функция отслеживания разметки позволяет нескольким разработчикам работать с одной диаграммой Visio. Эта функция используется при рецензировании диаграммы для обеспечения лучшего взаимодействия ее рецензентов. При этом каждый из рецензентов, вносящих изменения в исходный файл, в том числе лицо, принимающее окончательное решение по изменениям, видит изменения, внесенные другими рецензентами.

Задание

Работа с диаграммами. Построить блок-схему программы, разработанной в предыдущей лабораторной работе..

Контрольные вопросы:

1.Преимущества Visio при построении блок-схем.
2.Функциональные возможности Visio при построении блок-схем.
3.Функциональные возможности Visio при построении схемы сети.

Рекомендуемый список литературы:

1. Microsoft Office Visio 2003. Шаг за шагом. Практ. пособ. «СП Эком»,2006. – 352с.
2. ТрофимовС.А. CASE-технологии: практическаяработа в Rational Rose. Изд. 2-е. – М.: Бином-Пресс, 2002 г.

Лабораторная работа №4

Тема - Мастер (Wizards). Install Shield.

Теоретические сведения

Мастера (Wizards) позволяют облегчать и ускорять работу, за счет того, что выполняют самые необходимые действия или предлагают типичный набор свойств, компонентов для чего – либо. Delphi содержит различные мастера – мастер приложений, мастер диалогов, мастер баз данных и прочие.

Мастера представляют собой набор диалоговых окон, в каждом из которых представляются различные аспекты того или иного вопроса, мастером которых данный мастер и является. Методика использования мастеров сводится к тому, чтобы в каждом диалоговом окне выбирать необходимые опции, отмечать различные компоненты, которые должны будут появиться в результате. На основе выбранных параметров мастера генерируют заготовку, в которой выполнена за программиста большая часть рутинной работы, в результате чего программист может сосредоточиться непосредственно на программировании необходимых ему функций.

Контрольные вопросы:

1.Функциональные возможности мастера.
2.Примеры.
3.Назначение.
4.Что представляют собой мастера?
5.Методика использования мастеров.

Рекомендуемый список литературы:

1. Рейсдорф К., Хендерсон К. Borland C++bulider. Освой самостоятельно. - М.: Бином – М., 1998. – 702с.

2. http://delphin.xost.ru/?set=content&mc=11
3. http://articles.org.ru/cfaq/index.php?qid=969&catid=34

Лабораторная работа №5

Тема - Компиляция приложения. Виды компиляции.

Теоретические сведения

Options/Compiler

Code generation

Optimization - Допускает оптимизацию компилятора. Соответствует {$O}.

Stack frames - Вынуждают компилятор генерировать стековый фрейм на всех процедурах и функциях.{$U}.

Pentium-safe FDIV - Генерирует код Delphi, который обнаруживает дефектную команду раздела с плавающей запятой.

Record field alignment - Выравнивает элементы в структурах к указанному числу байтов {$A}.

Runtime errors

Range checking - Проверяет, что массивы и строки не выходят за пределы диапазона памяти{$R}.

I/O checking – Проверяет ошибки ввода/вывода после каждого запросы ввода/вывода {$I}.

Overflow checking Проверяет переполнение для целочисленных операций{$Q}.

Syntax options

Strict var-strings – Устанавливает проверку ошибок параметров строки {$V}.

Complete boolean eval – Оценивает каждую часть выражения в логических членах, независимо от того, оценивается ли результат операнда как ложь{$B}.

Extended syntax - Дает возможность вызывать функцию как процедуру и игнорировать ее реузльтат. Также включает поддержку PChar{X}.

Typed @ operator – Контролирует тип указателя, возвращаемого оператором @ {$T}.

Open parameters – Позволяет открывать параметры строки в объявлениях процедур и функций{$P}. Открытые параметры эффективнее и безопаснее.

Huge strings – Если этот параметр включен, то string будет обозначать AnsiString, иначе ShortString{$H}.

Assignable typed constants – Допускает применение типизированных констант {$J}.

Debugging Effect

Debug information – Помещает информацию об отладке в .dcu or .dpu файл {$D}.

Local symbols – Генерирует информацию о локальных символах {$L}..

Reference info/Definitions only – Генерирует символьные ссылки на информацию используемуюCode Browser, Code Explorer, и Project Browser. {$Y}. Если выставлены обе галочки({$YD}), компилятор записывает информацию о том, где идентификаторы определены. Если выставлено только Reference Info ({$Y+}), }), компилятор записывает информацию о том, где каждый идентификатор определен и используется. Эти опции не имеют эффекта, если выставлены Debug Information и Local Symbols.

Assertions – Разрешает/запрещает проверку утверждений{$C}. В отличие от исключений, утверждения могут быть удалены для финального builda.

Use Debug DCUs - позволяет отлаживать исходный код системных библиотек.

Настройка среды Delphi 7. Tools|Environment options

Environment Options

Пункт меню Tools => Environment Options...

Preferences (Настройка)
Autosave options - Автосохранение
Editor files - Редактируемые файлы
Project desktop - Рабочая область
Desktop contents - Содержание рабочей области
Desktop only - Только рабочая область
Desktop and symbols - Рабочая область и символы
Docking - Парковка
Auto drag drocking - Автопарковка
Pressing the Control key while dragging will Not clock windows - Нажмите Ctrl для переноса без парковки
Shared repository - Общая база
Directory - Каталог
Compiling and running - Компиляция и выполнение
Show compiler progress - Показывать состояние
Warn on package rebuild - Предупреждать о пакетах
Minimize on run - При выполнении минимизировать (сворачивать)
Hide designers on run - При выполнении скрывать дизайнеры

Designer (Дизайнер)
Grid options - Опции сетки
Display grid - Показывать сетку
Snap to grid - Привязать к сетке
Grid size - Размер
Module creation options - Опции создания модулей
New form as text - Форма в виде текста
Auto create forms & data modules - Автоматически создавать формы и модули данных
Options - Опции
Show component captions - Показывать заголовки
Show designer hints - Показывать подсказки дизайнера
Show extended control hints - Показывать расширенные подсказки

Object Inspector (Инспектор объектов)
Speed settings - Схема
Colors - Цвета
Options - Опции
Show instance list - Показывать список объектов
Show classname in instance list - Показывать имя класса в списке объектов
Show status bar - Показывать панель статуса
Render background grid - Горизонтальная сетка
Integral height (when not docked) - Выравнивать высоту
Show read only properties - Свойства только для чтения
References - Ссылки
Expand inline - Расширять
Show on events page - Показывать среди событий

Palette (Палитра)

Library (Библиотеки)
Directories - Каталоги
Library path - Путь к библиотекам
BPL output directory - Каталог для BPL

DCP output directory - Каталог для DCP
Browsing path - Каталог обзора

Explorer - Обозреватель
Explorer options - Опции обозревателя
Automatically show Explorer - Показывать автоматически
Highlight incomplete class items - Выделять незавершенные элементы классов
Show declaration syntax - Показывать синтаксические объявления
Explorer sorting - Сортировка
Alphabetical - По алфавиту
Source - По исходному коду
Class completion option - Заполнение класса
Finish incomplete properties - Завершать свойства
Initial browser view - Начальный вид
Classes - Классы
Units - Модули
Globals - Глобальные
Browser scope - Область обзора
Project symbols only - Только проект
All symbols - Все символы
Explorer categories - Категории

Type Library (Типы библиотек)
SafeCall function mapping - Функции SafeCall
All v-table interfaces - Все интерфейсы v-table
Only dual interfaces - Только двойные интерфейсы
Do not map - Не использовать
Language - Язык
Pascal - Паскаль
IDL - IDL
Ignore special CoClass Flags when importing - Игнорировать спец. флаги CoClass при импорте
Predefined - Встроенный
Restricted - Ограниченный
Hidden - Скрытый
Can Create - Создаваемый
Display updates before refreshing - Показывать перед обновлением

Environment Variables (Переменные среды)
System variables - Системные переменные
Variable - Переменная
Value - Значение
Add Override - Изменить
User overrides - Измененные пользователем

Delphi Direct
Automatically poll network - Автоматическая проверка
Polling Interval (in days) - Проверять каждые (в днях)
Last Poll - Последняя проверка
Automatically show Delphi Direct on refresh - Автоматически показывать Delphi при обновлении

Internet (Интернет)

Internet File Types - Фалй интернета
Description - Опписание
Extensions - Расширения
Script Debugging - Отладка скрипта
Enable Debugging - Включить
HTML File Extension - Расширения HTML
Sample Image File - Пример изображения

Контрольные вопросы:

1. Code generation.
2. Runtime errors.
3. Syntax options.
4. Debugging Effect.
5. Настройка среды Delphi 7. Tools|Environment options.
6. Delphi Direct.

Рекомендуемый список литературы:

1. Рейсдорф К., Хендерсон К. Borland C++bulider. Освой самостоятельно. - М.: Бином – М., 1998. – 702с.
2. http://delphin.xost.ru/?set=content&mc=11
3. http://articles.org.ru/cfaq/index.php?qid=969&catid=34

Лабораторная работа №6

Тема - Debugger. Отладка критических ошибок.

Теоретические сведения

Сеанс отладки начинается с запуска программы под управлением отладчика. Когда нажимается кнопка Run оперативной панели, программа автоматически запускается с использованием отладчика.

Пункт	Сочетание клавиш	Описание
Toggle Breakpoint	F5	Установка или удаление контрольной точки в текущей строке редактора кода.
Run to Cursor	нет	Запуск программы (если необходимо) и ее выполнение до той строки в окне редактора кода, которая содержит курсор.
Inspect	Alt+F5	Вызов инспектора отладки для объекта, на который указывает курсор редактора кода.
Go To Address	нет	Задание определенного адреса в программе, с которого возобновится ее выполнение.
Evaluate/Modify	нет	Просмотр и/или модификация значений переменных во время выполнения программы.
Add Watch at Cursor	Ctrl+F5	Добавление в список объектов наблюдения переменной, на которую указывает курсор.

Пункты контекстного меню отладчика

Пункт	Сочетание клавиш	Описание
Run	F9	Компиляция программы (если необходимо) и ее выполнение под управлением отладчика IDE. То же, что и кнопка Run оперативной панели.
Parameters	нет	Ввод параметров командной строки для вашей программы.
Step Over	F8	Выполнение текущей строки исходного кода с остановкой на следующей строке.
Trace Into	F7	Трассировка функции, вызываемой в текущей точке выполнения.
Trace to Next Source Line	Shift+F7	Перемещение точки выполнения на следующую строку исходного кода.
Run to Cursor	F4	Запуск программы с остановкой на текущей строке исходного кода.
Show Execution Point	нет	Отображение точки выполнения программы в редакторе кода. При необходимости прокрутка текста в окне исходного кода. Работает только в том случае, если выполнение программы приостановлено.
Program Pause	нет	Приостановка выполнения программы, как только точка выполнения входит в исходный код программы.
Program Reset	Ctrl+F2	Прекращение выполнения программы и возврат
Inspect	нет	Отображение диалогового окна Inspect, где вы можете ввести имя инспектируемого объекта.
Evaluate/Modify	Ctrl+F7	Отображение диалогового окна Evaluate/Modify.
Add Watch	Ctrl+F5	Отображение диалогового окна Watch Properties.
Add Breakpoint	нет	Отображение диалогового окна Edit Breakpoint для добавления контрольной точки.

Отладочные пункты в меню Run

Контрольная точка (breakpoint) – это маркер, который указывает отладчику приостановить выполнение программы по достижению данной точки. Чтобы установить контрольную точку нужно либо сделать даблклик в пробельном поле напротив необходимой строки, либо нажать F5 либо выбрать Toggle Breakpoint в контекстном меню редактора кода. Контрольные точки должны быть установлены на строках с кодом. Они не действуют, если их установить на пустых строках, комментариях или объявлениях. Как только программа остановилась в контрольной точке, можно просмотреть переменные, стек вызовов, символы и перемещаться по исходному тексту. После этого можно продолжить выполнение программы щелкнув на кнопке Run. Просмотреть ваши контрольные точки, отредактировать их можно в списке контрольных точек (View|Debug Windows|Breakpoints). Контрольные точки могут быть простыми и условными. Условия задаются в списке контрольных точек (Properties|Condition). Контрольные точки могут быть с условными выражениями и с условиями на число проходов.

Наблюдение за переменными.

Осуществляется с помощью списка объектов наблюдения (View|Debug Windows| Watches). Он имеет единственную функцию – проверять значения переменных. В него можно добавить столько переменных, сколько нужно.

Также в Delphi есть инспектор отладки, с помощью которого удобно вести наблюдение за сложными структурами – записями, массивами, классами и т.п. Его можно вызвать Run| Inspects, либо нажав Alt+F5. Диалоговое окно Evaluate/Modify позволяет просматривать и изменять текущее значение переменной. В этом диалоговом окне можно проверить различные результаты изменения определенной переменной. Можно просмотреть стек вызовов (call stack), чтобы получить информацию о любых функциях, которые вызывала программа. Окно CPU Window позволяет просмотреть программу на уровне ассемблера. Все эти средства можно вызвать из меню View|Debug Windows.

Пошаговое выполнение программы – одно из самых мощных средств отладки. Оно выполняется с помощью нажатия клавиши F8.

Опции отладчика. Tools|Debugger Options.

General (Основные)

Map TD32 keystrokes on run - Комбинации TD32
Mark buffers read-only on run - Устанавливать только чтение
Inspectors stay on top - Оставлять окна сверху
Enable COM cross-process support - Поддержка COM кросс-процессов
Allow function calls in new watches - Вызов функций для новых наблюдений
Rearrange editor local menu on run - Изменять локальное меню
Debug spawned processes - Порожденные процессы
Inspector defaults - Параметры инспектора
Show inherited - Показывать наследование
Show fully qualified names - Показывать полное имя
Paths - Пути
Debug symbols search path - Символы для отладки
Debug DCU Path - Путь DCU для отладки
Integrated debugging - Встроенный отладчик

Event Log (Журнал событий)

Clear log on run - Очищать при запуске
Unlimited length - Длина не ограничена
Length - Длина
Display process info with event - Отображать информацию о процессе
Messages - Записывать сообщения
Breakpoint messages - об остановке
Process messages - от процесса
Thread messages - от потока
Output messages - от вывода
Window messages - от окна

Language Exceptions (Исключения языка)

Exception types to ignore - Игнорировать исключения
Stop on Delphi Exceptions - Останавливать Delphi на исключениях

OS Exceptions (Исключения ОС)

Exceptions - Исключения
Handled by - Обрабатывать

Debugger - Отладчиком
User program - использовать программу
On resume - Возвращать
Run handled - Обработанное
Run unhandled - Не обработанное

Контрольные вопросы:

1.Пункты контекстного меню отладчика.
2.Отладочные пункты в меню Run.
3.Контрольная точка. Наблюдение за переменными.
4.Опции отладчика. Tools|Debugger Options. General.
5.Опции отладчика. Tools|Debugger Options. Event Log.
6.Опции отладчика. Tools|Debugger Options. Language Exceptions.
7.Опции отладчика. Tools|Debugger Options. OS Exceptions.

Рекомендуемый список литературы:

1. Рейсдорф К., Хендерсон К. Borland C++bulider. Освой самостоятельно. - М.: Бином – М., 1998. – 702с.
2. http://delphin.xost.ru/?set=content&mc=11
3. http://articles.org.ru/cfaq/index.php?qid=969&catid=34

Лабораторная работа №7

Тема - Tools. Способы настройки Delphi. Project options. Настройка проекта.

Теоретические сведения

Tools Options dialog box

Tools – Отображает инструментов установленных в данный момент в меню инструментов. Когда два или больше инструментов имеют конфликтующие ярлыки, над ними появляется красная звездочка.
Add – Нажмите, чтобы отобразить диалоговое окно свойств инструментов, где вы можете задать имя меню, путь, и параметры загрузки для программ.
Delete - нажмите, чтобы удалить выделенную программу из меню.
Edit - Нажмите, чтобы отобразить диалоговое окно свойств инструментов, где вы можете отредактировать имя меню, путь, и параметры загрузки для выделенных программ.
Up/down arrows - Используйте стрелки, чтобы перемещать программы в списке.
Close – Нажмите, чтобы закрыть меню и вернуться в Delphi.

Project|Options.

Forms

Раскрывающийся список main form определяет главную форму вашего приложения, которую delphi выводит первой и закрывает, когда приложение завершает работу. Два списка. auto-create forms и available forms, позволяют определить автоматически создаваемые формы.

Application

Поле title определяет название вашего приложения, выводимого в панели задач (например, delphi устанавливает его по имени текущего проекта). Если вы не введете названия, будет использоваться название, установленное по умолчанию (delphi32). Поле help file определяет файл справки, подключаемый к вашему приложению, что позволяет использовать систему контекстной справки.
Кнопка load icon позволяет установить пиктограмму приложения, используемую как пиктограмму по умолчанию в ярлыках и панели задач.

С помощью последней опции. target file extension, можно переопределить стандартное расширение создаваемого файла (dll — для динамически линкуемых библиотек, OCX — для элементов activex и т.п.). Однако это расширение можно не устанавливать, так как delphi весьма корректно работает по умолчанию.

Compiler

Вкладка компилятора содержит огромное количество переключателей, позволяющих устанавливать опции компилятора. Две особенно полезные опции. show hints и show warnings, помогут при отладке (при этом компилятор будет выдавать множество предупреждений, например об использовании неинициализированной переменной). Подробное описание смотрите в практической работе №5.

Linker

Опция map file полезна для тех программистов, которые интересуются технической информацией, например адресами сегментов, стартовым адресом программы и т.п. linker output определяет, что именно будет выдавать компилятор — delphi compiled unit (dcu) или объектные файлы (obj). Последние могут применяться, например, при разработке программ с использованием двух языков.

Опции EXE и dll позволяют создавать консольные приложения, описанные в разделе "Создание консольного приложения", и включать отладочную информацию, о которой подробно рассказывается в главе 2, "Тестирование и отладка". memory sizes определяет минимальный и максимальный размеры стека приложения и устанавливает предпочтительный базовый адрес для вашей dll. Если ваша программа часто использует глубокую рекурсию или большие локальные переменные, увеличьте размер стека. Базовый адрес лучше не изменять, если вы не совсем уверены в том, что собираетесь делать. EXE description представляет возможность внести в EXE или dll строку описания приложения.

Directories/conditionals

Установки output и unit output определяют, где компилятор размещает EXE или dll, а также скомпилированные модули. Если оставить опции незаполненными, создаваемые модули будут располагаться там же, где и исходные тексты, а выходные выполняемые файлы или dll — в папке проекта.

conditional defines определяют флаги, проверяемые в процессе компиляции и используемые, как правило, для включения или исключения блоков кода из проекта при компиляции. unit aliases существует для совместимости со старыми версиями delphi (в свое время модуль windows был разбит на два файла и для сборки одного модуля следовало присвоить один и тот же псевдоним windows обоим файлам).

Versioninfo

Вкладка versionlnfo дает возможность добавить к выполняемому модулю или dll информацию о версии— major version, minor version и file description. Действительно полезную возможность предоставляет auto-increment build number, заставляя delphi всякий раз увеличивать номер выпуска при компиляции программы. Раздел module attributes позволяет включать флаги, такие как debug build, в приложение. Выбор опций не влияет на процесс компиляции — они используются только в информативных целях.

Packages

Вкладка packages позволяет определить, какие пакеты доступны для использования при разработке приложения, и прилинковать их при создании результирующего файла. Группа design packages предоставляет список зарегистрированных пакетов, которые можно выбрать для использования в приложении.

Группа runtime packages дает возможность определить, какие пакеты компоновщик будет использовать при построении выходного файла. По умолчанию опция build with runtime packages отключена, а это означает, что все объекты из vcl будут скомпонованы с вашим приложением. Включение опции означает, что ваше приложение будет разделять с другими приложениями delphi одну копию пакетов.

Контрольные вопросы:

1. Tools.
2. Add.
3. Delete.
4. Edit.
5. Up/down arrows.
6. Close.
7. Forms.
8. Application.
9. Compiler.
10. Linker.
11. Directories/conditionals.
12. Versioninfo.
13. Packages.

Рекомендуемый список литературы:

1. Рейсдорф К., Хендерсон К. Borland C++bulider. Освой самостоятельно. - М.: Бином – М., 1998. – 702с.

2. http://delphin.xost.ru/?set=content&mc=11

3. http://articles.org.ru/cfaq/index.php?qid=969&catid=34