**SQL**

1. Конвертация строки в hex-строку:

SELECT sys.fn\_varbintohexstr(CONVERT(VARBINARY(MAX), 'ABCD'))

1. Контекстный поиск по всем скриптам в базе:

SELECT O.type, O.name

FROM sys.sql\_modules M

JOIN sys.objects O ON O.object\_id = M.object\_id

WHERE definition LIKE '%test%'

ORDER BY O.type, O.name

1. Тип данных значения поля SQL\_VARIANT:

SQL\_VARIANT\_PROPERTY(note\_Value, 'BaseType')

1. Пауза:

WAITFOR DELAY '00:00:10'

1. Запуск при старте SQL-сервера без агента:

Создать процедуру в базе master. Включить ее для запуска при старте:

EXEC sp\_procoption '<procedure name>', 'STARTUP', 'ON'

1. Статистика запроса

SET STATISTICS TIME ON

1. Объективный замер времени выполнения запроса:

CHECKPOINT 1

WAITFOR DELAY '00:00:05' -- ВНИМАНИЕ! Это 5 секунд к общему времени.

DBCC DropCleanBuffers

DECLARE @D DATETIME

SELECT @D = GETDATE()

/\* запрос \*/

SELECT DATEDIFF(ms, @D, GETDATE()) Time

1. Получение результатов незавершенной транзакции

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

1. Номера некоторых событий профайлера после выброса в таблицу:

42 = SP:Starting

43 = SP:Completed

44 = SP:StmtStarting

45 = SP:StmtCompleted

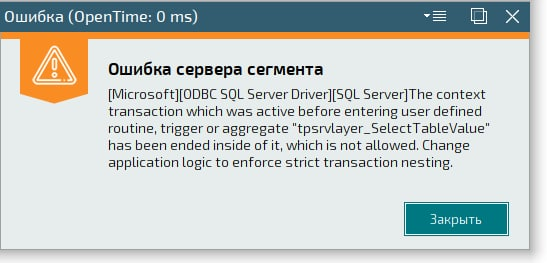
1. Наш СП. Отключение версионного кэша (нужно, когда в запросе нет обращений к рабочим таблицам):

EXEC tpsrv\_DisableCache

1. Ошибка при считывании табличного параметра с помощью tpsrv\_SelectTableValue

INSERT INTO #Divisions (dvsn\_ID)

EXEC tpsrv\_SelectTableValue 'Divisions', ''



Означает что структура временной таблицы для считывания не совпадает со структурой таблицы в параметре.

1. Сброс кэша, в том числе версионного:

EXEC tpsrv\_Event '#ClearCache'

**SQLite**

1. IS NULL всех перечисленных полей:

COALESCE(alcit\_FullName, alcit\_ShortName) IS NULL

**Delphi**

1. Расположение Live Templates, которые работают в редакторе кода (Ctrl+J):

C:\Program Files (x86)\Embarcadero\RAD Studio\7.0\ObjRepos\en\Code\_Templates\Delphi

1. Включение по дефолту «Allow side effects and function calls»:

Tools – Options – Debugger Options – Allow side effects and function calls in new watches.

1. Выключение файлов хистори:

Tools – Options – Editor Options – Create backup files.

1. Текущий/основной потоки: GetCurrentThreadId = MainThreadID
2. Добавление (и удаление) типа к набору перечислимых типов (сложение наборов перечислимых типов):

MySetOf := MySetOf + [soElement];

Include(MySetOf, soElement);

MySetOf := MySetOf - [soElement];

Exclude(MySetOf, soElement);

1. Тип, раскрывающий параметр открытого массива констант: TVarRec.
2. Изменение размерности вариант-массива VarArrayRedim (Variant-а с типом varArray).
3. К указателю PByte можно обращаться как к массиву:

**procedure** Do(Value: RawByteString);

**var**

P: PByte;

**begin**

P := PByte(Value);

**if** (P[0] = 0) and (P[1] = 0) **then**

...

1. Посмотреть область памяти по указателю:

F7 -> PByte(P)^,16mx

1. Чей метод в переменной:

TObject(TMethod(FExternalCallback).Data).ClassName

1. Перенос открытого массива в динамический без цикла:

TShortIntArray = array of ShortInt;

function GetStateArray(const Source: array of ShortInt): TShortIntArray;

var

L: Integer;

begin

L := Length(Source);

SetLength(Result, L);

if L > 0 then Move((@Source[0])^, Result[0], SizeOf(Source));

end;

1. В случае ошибки чтения свойства при загрузке из ресурса:

Classes.ReadError($40CF38)

Classes.PropValueError

Classes.TReader.FindMethod(???, ???)

Classes.TReader.FindMethodInstance($6083B50, 'DataChange')

1. Если в аргумент Format функции SysUtils.Format нужно передать строку содержащую символы % не как начало параметра для вставки аргумента, нужно экранировать проценты. На выход функция вернет разэкранированные символы % в этих местах.
2. ShowWindow(Handle,SW\_HIDE); // Скрыть форму  
   ShowWindow(Application.Handle,SW\_HI DE); // Скрыть кнопку TaskBar'а
3. Список внешних функций EXE или DLL: из командной строки tdump. Входит в состав Delphi.
4. Удаленная отладка
   1. В опциях проекта Delphi Compiler – Linking выставить Debug information = True, Include remote debug symbils = True, Map File = Detailed
   2. Скомпилировать и положить на удаленный компьютер скомпилированный экзешник и рядом с ним файлы .map и .rsm
   3. Скопировать на удаленный компьютер папку [\\tsd-fs\Work\Transfer\Vasilyev\remote debugging](../../..//tsd-fs/Work/Transfer/Vasilyev/remote%20debugging)
   4. Запустить там Register\_DLL\_on\_Remote\_Comp.bat и Run rmtdbg140.bat
   5. Запустить экзешник на удаленном компьютере
   6. В среде вызвать Run – Attach to Process
   7. Ввести имя удаленного компьютера в Remote machne и нажать Refresh
   8. Найти процесс в списке и нажать Attach
   9. Подождать, пока процесс остановится в окне CPU, продолжить выполнение, точки работают

То же самое с отладкой DLL, можно так же ее скомпилировать и подключиться к хост-приложению на удаленном компьютере.

1. Компиляция наших дизайнтайм пакетов:
   1. Delphi – Components – Install packages… – удалить все наши пакеты
   2. Удалить все bpl и dcp из Документы\RAD Studio\7.0
   3. Открыть группу \_VCLComponents\TillypadComponents.groupproj
   4. Выделять двойным щелчком каждый пакет и инсталировать
   5. То же самое с группой FastReport.groupproj
2. Наши константы результата диалогов:

bcOk = 1

bcCancel = 2

bcYes = 6

bcNo = 7

1. Прозрачный лэйбл без границы: TfmAbout.SetupLabel.
2. Директива компиляции NOFREETYPE – обычные шрифты
3. Настройка группированного грида, не забывать в ИО: TssGrid.Options.DataManager = dmtGroupedList. Иначе в группах будут колонки.
4. Не компилируется вызов функции ShortCut: E2010 Incompatible types: 'TssShiftState' and 'Set'. Нужно добавить в uses uCommonTypes.
5. Если дизайнтайм-пакет с компонентами устанавливается, но при следующей загрузке среды как баран выламывается с текстом «Не найден указанный модуль», это значит, что он использует в своем requires другой дизайн-тайм-пакет с установленными визуальными компонентами. И при загрузке использующий пакет почему-то хочет видеть используемый в общем каталоге (по умолчанию это $(BDSCOMMONDIR)\Bpl) независимо от того, что используемый пакет установлен из другого местаи вообще-то сам по себе загружается оттуда успешно в правильной очередности. Добавлять путь в переменные среды неправильно, потому что это обяжет всех разработчиков делать это же самое. Но самое удивительное, что это еще и бесполезно, потому что использующий пакет так же продолжает выламываться. Поэтому проще на этапе отладки пакетов настроить билд-ивэнт, копирующий каждый пакет в каталог (BDSCOMMONDIR)\Bpl.
6. Ошибка компиляции F2051 Unit ... was compiled with a different version of ... происходит из-за неявной циклической ссылки между модулями. Поскольку в верхнем **uses** циклической ссылки нет, билд проекта проходит успешно, но просто компиляция – нет. Видимо при составлении списка изменившихся модулей компилятор запутывается в этих ссылках и не компилирует какой-то из них, вследствие чего потом уже возникает ошибка с указанным текстом. От этого помогает размещение части классов из использующих друг друга модулей в третий модуль.

Например, есть сложный класс TWorker и его помощники в модуле uWorker. И есть облегчающий его класс-фасад TFasade, который мы расположили в отдельном модуле uCommon специально, чтобы разгрузить в том числе и основной модуль тоже. uCommon использует uWorker, но и uWorker может использовать uCommon в случае, если ему нужны какие-нибудь функции оттуда. Вообще-то это стилистически неверно. Фасад должен использоваться только снаружи и нужно избавить uWorker от ссылки на модуль с ним. Но такая взаимосвязь может и случайно возникнуть в процессе разработки чего-то сложного, а на разведение классов по модулям может просто не быть времени, когда это обнаружится.

Как решение проблемы на данном примере можно применить следующее. Вытащить все классы из модуля uWorker кроме самого TWorker в отдельный модуль, например, uClasses и скомпилироваться. Проблема исчезнет на время. Но полагаться на такое решение не стоит и лучше как можно скорее провести рефакторинг классов. Некоторые программисты вообще не пользуются ссылками в разделе **implementation**, поскольку среда не контролирует цикличность там. Если это не заменяется складыванием всех кассов некоего функционала в один модуль, это может быть решением. Но всегда нужно помнить и желательно выработать привычку взаимного использования классов и модулей по дереву внутри каждого функционала. Есть итоговые классы, содержащие наружную часть, они обращаются к другим, функционально внутренним классам, но те уже в свою очередь к наружным не обращаются даже через несколько звеньев в цепочке. То есть, ветви имеют основание в стволе или других ветвях, но не врастают обратно.

1. Доступ к private:

TssKeyboardLayoutButton = **class** (TssDesignComponent)

**private**

FNumber: Integer;

FFunctionType: Integer;

FImageIndex: Integer;

FImages: TCustomImageList;

TKLButtonCracker = **class**(TssDesignComponent)

**private**

F1: Integer;

F2: Integer;

FImageIndex: Integer;

FImages: TCustomImageList;

TKLButtonCracker(Control).FImages := ImageList;

TKLButtonCracker(Control).FImageIndex := ImageIndex;

1. Передача процедуры из стека в параметр функции:

**type**

TSomeProc = procedure (Arg: **String**);

**procedure** DoAction(CallBack: TSomeProc);

**begin**

...

**end**;

**procedure** DoWrok;

**procedure** \_CallBack*(*\_Arg: **String**);

**begin**

...

**end**;

**begin**

**for** i := 0 **to** Count – 1 **do**

DoAction(@CallBack);

**end**;

1. var и out. Очистка out параметра процедуры производится компилятором только для строк и интерфейсов. Для всех других типов – нет!
2. NewGUID
   1. TGUIDHelper.NewGUID
   2. SysUtils.CreateGUID
   3. OleCheck(CoCreateGuid(Guid));
3. Обнуление записи: FillChar(\_EvaluateResult, SizeOf(TEvaluateResult), 0);
4. Об использовании интерфейсануление просрана статья :( А там как раз было упоминание антипаттерна «Проверка типа вместо интерфейса».
5. Все простые типы данных

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Знак** | **Размер** | **Диапазон** |
| Byte (UInt8) | Нет | 1 (8) | 0..255 |
| ShortInt (Int8) | Да | 1 (8) | - 128..127 |
| Word (UInt16) | Нет | 2 (8) | 0..65 535 |
| SmallInt (Int16) | Да | 2 (16) | - 32 768..32 767 |
| Cardinal (UInt32) | Нет | 4 (32) | 0..4 294 967 295 |
| Integer (Int32) | Да | 4 (32) | - 2 147 483 648..2 147 483 647 |
| UInt64 | Нет | 8 (64) | 0..184 46 744 073 709 551 615 (- 263..263 - 1) |
| Int64 | Да | 8 (64) | - 9 223 372 036 854 775 808..9 223 372 036 854 775 807 (0..264 - 1) |
|  |  |  |  |
| Single |  | 4 | 1.18e-38..3.40e+38 |
| Real48 |  | 6 | 2.94e-39..1.70e+38 |
| Double (Real) |  | 8 | 2.23e-308..1.79e+308 |
| Extended |  | Win32: 10  Win64: 8  macOS: 16  Linux: 16 | 64-bit Intel Linux, 64-bit Intel macOS, 32-bit Intel iOS Simulator:  3.37e-4932..1.18e+4932  64-bit Intel Windows, ARM platforms (32-bit Android, 64-bit Android, 64-bit iOS, 64-bit macOS):  2.23e-308..1.79e+308 |
| Comp |  | 8 | -9223372036854775808..9223372036854775807 |
| Currency |  | 8 | -922337203685477.5808..922337203685477.5807 |
|  |  |  |  |

1. Артибуты свойства объекта типа **record** могут назначаться только, если в записи свойство с методом записи. Иначе не скомпилируется.
2. Перегенерация исключения в **raise**. Тут есть грабли, *raise E* это повторная генерация уже созданного объекта. Если надо сгенерировать новое исключение, есть высокая вероятность случайно написать так:

**try**

...

**except**

**on** E: Exception **do**

**raise** E.CreateFmt('Message');

**end**;

Это вызовет ошибку взаимодействия с памятью со стеком не из места создания объекта исключения, а при обработке *\_HandleFinallyInternal*, вызываемой компилятором неявно где-то в конце процедуры, вызвавшей метод с перегенерацией.

Либо, если перегенерированное таким образом исключение где-то будет еще раз отловлено с применением *on E: Exception*, то объект *E* уже будет ссылаться на произвольную память. И стек ошибки *Invalid pointer operation* укажет хотя бы на место попытки освобождения объекта *E* при выходе из внешнего блока *try except*. Немного поближе, но тоже не очень.

То есть, ошибку, допущенную при перегенерации придется искать по коду с собаками. Поэтому, дешевле будет просто запомнить, перегенерация исключения исполняется так:

**try**

...

**except**

**on** E: Exception **do**

**raise** ExceptClass(E.ClassType).CreateFmt('Message');

**end**;

1. Перечислимый тип с явным указанием индексов в цикле будет обрабатываться по неявному целому счетчику от нижнего до верхнего значения типа. В необъявленных диапазонах обращение к переменной-счетчику не выдаст никаких ошибок, хоть и мы видим его значение в отладчике как *(out of bound) n*. Поведение, мякго говоря, не дельфевое, даже в VCL не особо встречается этот способ. В case такие значения не перечислишь, не скомпилируется, но что толку? Вобщем, штука интересная, но почти бесполезная. Разве что использовать сдвиги при псевдонаследовании типов.

TCustomType = (stNone = 0, stBasic = 1, stCustom = 2);

TSpecificType = (stNone = 0, stBasic = 1, stCustom = 2, stWarm = $FF + 0, stSoft = $FF + 1, ssSharp = $FF + 2);

Но только, опять же, цикл по такому типу будет делать 258 шагов вместо 6-ти. И набор уже не объявишь, будет ошибка компиляции: Sets may have at most 256 elements. Вобщем, ни разу не пользовался и врядли придется.

1. Способ скрыть лишние элементы перечислимого типа из свойства:

**property** AllowedSelections: TDBGridEhAllowedSelections **read** FAllowedSelections **write** SetAllowedSelections **default** [gstRecordBookmarks..gstAll];

1. Уничтожение интерфейса после уничтожения исполнителя вызовет Access Violation. Если мы создали объект в процедуре и там же запросили интерфейс в переменную стека, необходимо присвоить **nil** в нее до уничтожения объекта. Иначе Access Violation произойдет при неявном вызове уничтожения интерфейса на выходе из процедуры.
2. Декларативная матричная функция для набора перечислимого типа (set of):

**type**

TMyType = (ttA, ttB, ttC, ttD);

TMyTypes = set of TMyType;

Matrix<T> = **class** **abstract**

**class** **function** **Get**(Value: T; **const** Map: **array** **of** **String**): **String**;

**end**;

**const**

SA\_Map: **array**[0..15] **of** **String** = (

{0000} '-',

{1000} 'A',

{0100} 'B',

{1100} 'AB',

{0010} 'C',

{1010} 'AC',

{0110} 'BC',

{1110} 'ABC',

{0001} 'D',

{1001} 'AD',

{0101} 'BD',

{1101} 'ABD',

{0011} 'CD',

{1011} 'ACD',

{0111} 'BCD',

{1111} 'ABCD'

);

**function** Get(Value: TMyTypes; **const** Map: **array** **of** **String**): **String**;

**begin**

Result := Map[Byte(Value)];

**end**;

Тестовый код:

**procedure** SomeProc;

**begin**

WriteOut(Matrix<TMyTypes>.Get([], SA\_Map));

WriteOut(Matrix<TMyTypes>.Get([ttA], SA\_Map));

WriteOut(Matrix<TMyTypes>.Get([ttB], SA\_Map));

WriteOut(Matrix<TMyTypes>.Get([ttC], SA\_Map));

WriteOut(Matrix<TMyTypes>.Get([ttD], SA\_Map));

WriteOut(Matrix<TMyTypes>.Get([ttA, ttB], SA\_Map));

WriteOut(Matrix<TMyTypes>.Get([ttA, ttC], SA\_Map));

WriteOut(Matrix<TMyTypes>.Get([ttA, ttD], SA\_Map));

WriteOut(Matrix<TMyTypes>.Get([ttB, ttC], SA\_Map));

WriteOut(Matrix<TMyTypes>.Get([ttB, ttD], SA\_Map));

WriteOut(Matrix<TMyTypes>.Get([ttC, ttD], SA\_Map));

WriteOut(Matrix<TMyTypes>.Get([ttA, ttB, ttC], SA\_Map));

WriteOut(Matrix<TMyTypes>.Get([ttA, ttB, ttD], SA\_Map));

WriteOut(Matrix<TMyTypes>.Get([ttA, ttC, ttD], SA\_Map));

WriteOut(Matrix<TMyTypes>.Get([ttA, ttB, ttC, ttD], SA\_Map));

**end**;

Результат исполнения тестового кода:

-

A

B

C

D

AB

AC

AD

BC

BD

CD

ABC

ABD

ACD

ABCD

**Windows**

1. Узнать, каким процессом занят файл:
   1. Монитор ресурсов – ЦП – отметить все процессы – набрать имя файла в поле «Поиск дескрипторов» панели «Связанные дескрипторы»
   2. Process Explorer – Find handle or DLL (Ctrl+F)

**Redmine**

1. Добавить новую статью в Wiki:
   1. Создать в родительской статье ссылку на новую статью
   2. Перейти и редактировать новую статью
2. Вставить картинку
   1. Скопировать картинку в буфер
   2. Редактировать – «Добавить изображение из буфера обмена»
   3. Скопировать название, например, «picture874-1.png»
   4. Кнопка «Изображение» (cid:image001.png@01D0F777.9DE4A280) в панели инструментов
   5. В поле «Ссылка» вставить название, например, «picture874-1.png»
   6. В режиме предпросмотра картинка не отобразится, нужно сохранить, чтобы она появилась

**SkinEditor**

Для загрузки картинки ее можно сохранить из Paint в PNG и потом сконвертировать с помощью \\tsd-fs\Work\Transfer\Zakirov\Png & Bmp. Для выгрузки нужно использовать ParamsEditor, потому что скин это наши параметры.

**GIT**

1. Исправить коммит на сервере: исправить локально, выложить с опцией Force: May discard – known changes.
2. Переключить head (активную ветку) на сервере: файл \*.git/HEAD.
3. Глобальный конфиг и игнор лежит на диске W:

**.gitconfig:**

[user]

name = VasilyevSM

email = [vasilev@tillypad.ru](mailto:vasilev@tillypad.ru)

[core]

excludesfile = w:/.gitignore

**.gitignore:**

\*.dproj

\*.dsk

\*.identcache

\*.dproj.local

\*.groupproj.local

\*.tpxl\_msg

\*.tvsconfig

\*.drc

1. Исключение файлов из индекса

Совсем удалить – в Проводнике на файле без изменений команда **Tortoise Delete and add to ignore list**. Удалится из индекса Git и из каталога.

Или просто исключить из обмена – в окне Tortiose Commit на измененном файле команда **Assume unchanged**. Git запишет это у себя в индексе, в игнор-файлах ничего не будет. Посмотреть можно в свойствах файла, вкладка Git – Assume valid/unchanged. Есть команда Git – вернуть список всех исключенных файлов.

**WinMerge**

1. Если включилось отображение только отличающихся строк вместо всех, вернуть: Вид - Контекст отличий — Все строки.