Декларативный язык FlexT – инструмент анализа и документирования бинарных форматов данных

Алексей Евгеньевич Хмельнов,
И.В. Бычков,
А.А. Михайлов
Институт динамики систем и теории управления СО РАН

{hmelnov,bychkov,mikhailov}@icc.ru

http://FlexT.icc.ru

Выбор: текстовые или бинарные файлы

Бинарные файлы:

- более компактные
- код для чтения/записи пишется проще
- возможен быстрый переход к любому месту файла (Seek) => чтение фрагмента

Текстовые файлы:

- более прозрачные
- легче редактируются
- не зависят от порядка байт платформы

Новое поколение выбирает текст (XML)

- сейчас часто жертвуют эффективностью ради упрощения контроля за правильностью структуры файла
- часто текстовые форматы основываются на синтаксисе XML, поскольку для этого синтаксиса существуют готовые библиотеки и инструментальные средства, облегчающие разработку алгоритмов чтения/записи информации,
- для XML можно автоматизировать контроль правильности структуры файла (используя XML-схемы)

Позиция авторов

- В небольших текстовых файлах удобно хранить информацию, которую необходимо часто редактировать (например, настройки программы). Иначе придётся писать специальные формы редактирования для таких файлов.
- Лучше избегать использования текстовых форматов для обмена большими объёмами информации.
- Для обеспечения прозрачности бинарных файлов и контроля правильности их создания используются спецификации форматов бинарных файлов

Цели разработки языков спецификации бинарных форматов данных

- Документирование форматов данных
- Проверка данных на соответствие спецификации с диагностикой ошибок
- Генерация кода чтения данных
- Метаинформация для библиотек обработки/передачи данных

Обзор средств спецификации бинарных форматов данных

Средство	Цель разработки	Вид	Описанные форматы	Битовые типы	Указатели	Варианты	Для человека	Дополнительно
BFF (Binary File Format Definition)	обратное проектирование формата AutoCAD DWG	Грамматика	DWG	–	?	?	?	
DFDL (Data Format Description Language)	описания текстовых и бинарных данных, используемых в GRID-системах, Open Grid, IBM	XML, простые записи	таблицы в специальных обменных форматах	–	–	?	–	
MFL (Message Format Language)	Описание обменных форматов в WebLogic Integration фирмы Oracle	XML, потоки данных	данные в специальных обменных форматах	–	–	+, необяза-тельные поля	–	
NetPDL	описание сетевых протоколов, документирования форматов пакетов	XML, потоки данных	сетевые пакеты	+, прими-тивные	–	+	–	
BinPAC	описание сетевых протоколов, генерация кода чтения	Модифицирован-ные файлы Bison, потоки данных	сетевые пакеты	–	–	+	+	
EAST (Enhanced Ada SubseT)	разработан CCSDS для облегчения обмена информацией между космическими системами	Типы данных на базе Ada, потоки данных	последова-тельно передаваемые данные космические данные	+	–	+, выбор по внешнему значению	+	только выраже-ния по одному значению
HUDDL	описание форматов гидрографических данных, генерация кода	XML, потоки данных	гидрографические форматы	?	–	?	–	
Advanced Language Processing Technology Applied to Digital Records	интеграции разнородной информации, используемой для принятия военных решений, военный НИИ США, генерация кода чтения	атрибутивные грамматики	Форматы с последова-тельной записью	?	–	?	+	
DataScript	генерации библиотек для чтения данных на языке Java	определения типов	Java CLASS (без маш. команд), DVI, ELF	+, прими-тивные	–, есть вычисляе мые метки	+, по условиям на поля в union	+	с 2003 г. исходные тексты проекта не обновлялись
Miraplacid Binary DOM	реализация универсальной библиотеки для доступа к бинарным и текстовым данным в стиле DOM	определения типов	Описан 21 формат, часть из них - текстовые	–	+	–	+	Просмотра дерева структур, выделение текущей в дампе
Kaitai Struct	создание парсеров бинарных данных по спецификациям	~JSON, последовательность с необязательными и повторяющимися фрагментами	Описано 36 форматов	–	+	+	–	Появился Switch
Synalyze It! Hexinator	Просмотр бинарных данных	XML, грамматики, Python	80 форматов, в основном для macOS	+	+	+	–	Просмотра дерева структур, раскраска дампа
FlexT	Просмотр бинарных данных	определения типов	100 форматов	+	+	+	+	

Обзор: выводы

- во многих предметных областях осознана необходимость использования спецификаций форматов данных
- иногда ставится задача не описания произвольных форматов, а лишь облегчения работы с некоторым их подмножеством
- иногда задача описания произвольных форматов ставится, но реализуются не все конструкции, необходимые для её решения
- часто для спецификаций выбирается представление XML, что затрудняет восприятие текста спецификации человеком
- в некоторых проектах явно ставится задача лёгкости восприятия текста человеком
- спецификацию бинарного формата рассматривают как грамматику, как структуру последовательности, или как набор типов данных
- многие языки спецификаций были созданы для облегчения написания кода для работы с данными
- разработчики очень редко изучают аналоги
- задача описания бинарных данных востребована

Виды спецификаций форматов данных

<table>
<tr><td colspan="3">Спецификация формата данных</td></tr>
<tr><td colspan="2">Спецификация интерпретации</td><td>Спецификация изменения</td></tr>
<tr><td>Идентификация типов данных</td><td>Графика, таблица, текст,...</td><td></td></tr>
</table>

Язык FlexT

- декларативные спецификации форматов бинарных данных
- **FlexT** – **Flex**ible **T**ypes
- Основными конструкциями языка FlexT являются определения типов данных, сложность которых может существенно превышать сложность аналогичных определений в традиционных языках программирования

Дамп файла LSKAT.DBF и содержащаяся в нём информация

```
LSKAT.DBF                                   DOS        241     Col 0      100%
00000000:  03 60 0B 1C 05 00 00 00   82 00 16 00 00 00 00 00   ♥`♂∟♣    B ▬
00000010:  00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
00000020:  4E 41 4D 45 00 69 B1 04   94 1D 00 43 38 64 0A 00   NAME i▒♦Φ↔ C8d◙
00000030:  12 00 D4 6F 34 69 B2 21   F1 52 0A 00 D4 6F 44 69   ↕ ╘o4i▓!ёR◙ ╘oDi
00000040:  47 43 4F 4C 00 69 B1 04   94 1D 00 4E 38 64 0A 00   GCOL i▒♦Φ↔ N8d◙
00000050:  02 00 D4 6F 34 69 B2 21   F1 52 0A 00 D4 6F 44 69   ☻ ╘o4i▓!ёR◙ ╘oDi
00000060:  54 4C 49 4E 45 00 B1 04   94 1D 00 4E 38 64 0A 00   TLINE ▒♦Φ↔ N8d◙
00000070:  01 00 D4 6F 34 69 B2 21   F1 52 0A 00 D4 6F 44 69   ☺ ╘o4i▓!ёR◙ ╘oDi
00000080:  0D 00 20 87 A5 AC AF AE   AB AE E2 AD AE 20 20 20   ♪  Земполотно
00000090:  20 20 20 20 20 31 30 20   20 88 91 91 8E 20 20 20        10  ИССО
000000A0:  20 20 20 20 20 20 20 20   20 20 20 20 39 20 20 82                9  В
000000B0:  91 8F 20 20 20 20 20 20   20 20 20 20 20 20 20 20   СП
000000C0:  20 31 35 30 20 8F E0 AE   E7 A8 A5 20 20 20 20 20    150 Прочие
000000D0:  20 20 20 20 20 20 20 31   34 20 20 90 A5 AC AE AD         14  Ремон
000000E0:  E2 20 20 20 20 20 20 20   20 20 20 20 20 20 31 32   т              12
000000F0:  1A                                                 →
```

NAME	GCOL	TLINE
Земполотно	10	
ИССО	9	
ВСП	15	0
Прочие	14	
Ремонт	12	

[▪]═══ Структура D:...ODES\LSKAT.DBF ═══

Имя	Тип	Длина	Десятки
NAME	Символ	18	
GCOL	Число	2	0
TLINE	Число	1	0

Спецификация на FlexT формата DBF

```
type
  TBinDate array[3] of Byte //Дата в
                    двоичном формате(ГГММДД)
  TDBF3FldKind enum Char (
    fkChar='C', fkNumeric='N', fkLog='L',
    fkDate='D', fkMemo='M'
  )
  TDBF3FldDsc struc
    array[11] of Char,0; Name //Имя -
                    ASCIIZ строка
    TDBF3FldKind hType
    ulong DataP    //like Delphi Tag
    Byte Len       //Длина поля в байтах
    Byte DecNum    //Число знаков после
                    точки
    Word MUsrRsrv1//Зарезерв. для
                    многопольз. систем
    Byte WorkID    //ID рабочей области
    Word MUsrRsrv2//Зарезерв. для
                    многопольз. систем
    Byte SetFldData//Используется командой
                    SET FIELDS
    array[8] of Byte Reserved
  ends
  PDataArray ^TDataArray near
  TDBF3Hdr struc
    Byte Ver //$02-dBase II,$03-dBase III
             //$83-dBase III с Memo-полями
    TBinDate LastChangeDate
    ulong RecCnt  //Число записей
    PDataArray HdrLen //Длина заголовка
                      в байтах
    Word RecLen       //Длина записи в
                      байтах
    (array[20]of Byte) Reserved
  ends

  TDBF3HdrWithFields struc
    TDBF3Hdr H
    array[(@.H.HdrLen-@.H:Size-1) div
      32] of TDBF3FldDsc Fields
  ends
data
  0x0000 TDBF3HdrWithFields Hdr
type
  TFieldData array[Hdr.Fields[#].Len]of
                  Char
  TFieldsData array of TFieldData :
                [@:Size=Hdr.H.RecLen-1]
  TRecData struc
    Char F
    TFieldsData D
  ends
  TDataArray array[Hdr.H.RecCnt] of
                      TRecData
```

Результат разбора по спецификации файла LSKAT.DBF

```
00:Hdr: TDBF3HdrWithFields = (
  H:(Ver:dBase_III{03}; LastChangeDate:(Y:96; M:11; D:28); RecCnt:00000005; HdrLen:0082;
    RecLen:0016; Reserved:
    (0:00,1:00,2:00,3:00,4:00,5:00,6:00,7:00,8:00,9:00,10:00,11:00,12:00,13:00,14:00,
    15:00,16:00,17:00,18:00,19:00));
  Fields: (
    0:(Name:'NAME'; hType:fkChar{'C'}; DataP:000A6438; Len:12; DecNum:00;
       MUsrRsrv1:6FD4; WorkID:34; MUsrRsrv2:B269; SetFldData:21;
       Reserved:(0:F1,1:52,2:0A,3:00,4:D4,5:6F,6:44); InMDX:69),
    1:(Name:'GCOL'; hType:fkNumeric{'N'}; DataP:000A6438; Len:02; DecNum:00;
       MUsrRsrv1:6FD4; WorkID:34; MUsrRsrv2:B269; SetFldData:21;
       Reserved:(0:F1,1:52,2:0A,3:00,4:D4,5:6F,6:44); InMDX:69),
    2:(Name:'TLINE'; hType:fkNumeric{'N'}; DataP:000A6438; Len:01; DecNum:00;
       MUsrRsrv1:6FD4; WorkID:34; MUsrRsrv2:B269; SetFldData:21;
       Reserved:(0:F1,1:52,2:0A,3:00,4:D4,5:6F,6:44); InMDX:69),
    3:0D))
    81:00 ¦.¦
82:Hdr.H.HdrLen^: TDataArray =  (
  0:(F:' '; D: (0:'Земполотно        ',1:'10',2:' ')),
  1:(F:' '; D: (0:'ИССО              ',1:' 9',2:' ')),
  2:(F:' '; D: (0:'ВСП               ',1:'15',2:'0')),
  3:(F:' '; D: (0:'Прочие            ',1:'14',2:' ')),
  4:(F:' '; D: (0:'Ремонт            ',1:'12',2:' ')))
    F0:1A ¦.¦
```

Особенности динамических типов данных

Тип	Статический	Динамический
Размер	Фиксирован	Может зависеть от данных
Смещение составляющих	Фиксировано	Может зависеть от данных
Назначение	Тип переменных	Тип констант

Примеры на Паскале	`TName=array[0..31] of Char;`
	`TId = string[32]`
		`const CP: PChar = 'Hello';`
		`var S: String;`
		`...`
		`S := 'Hello';`

Пример динамического типа данных: представление RTTI

Определение типа данных из модуля TypInfo в Delphi	Функция для получения адреса поля `TypeData`	Описание этого типа на FlexT

`PTypeInfo = ^TTypeInfo;`
`TTypeInfo = record`
`  Kind: TTypeKind;`
`  Name: ShortString;`
`  {TypeData: TTypeData}`
`end;`
`PTypeData=^TTypeData;`
`TTypeData=packed record`
`  case TTypeKind of`
	`function GetTypeData(TypeInfo:`
	`  PTypeInfo): PTypeData;`
	`asm`
	`  XOR EDX,EDX`
	`  MOV DL,[EAX].TTypeInfo.`
	`  Name.Byte[0]`
	`  LEA EAX,[EAX].TTypeInfo.`
	`  Name[EDX+1]`
	`end;`
		`TTypeInfo struc`
		`  TTypeKind Kind`
		`  Str Name`
		`  TTypeData TypeData`
		`ends`

Особенности языка FlexT (1/2)

1. Составляющие переменного размера

```
FldNameTbl: array[@.numFields] of pchar;
```

2. Параметры и свойства типов

```
THdrData(Kind,Cnt) case THdrValType(@:Kind) of
  CHAR: array[@:Cnt] of Char
 ..........................
endc
```

3. Ссылки на свойства и составляющие в выражениях

`V[i]`- элемент массива, `V.A` - поле записи, `V.0x15` - случай варианта. `V:Size` - параметр или свойство. `@` - переменная определяемого типа. `V@` - родительский тип, `V:@` - содержащая переменная, `V:#` - порядковый номер переменной

Особенности языка FlexT (2/2)

4. Дополнительные блоки определений типов:

- Блок утверждений `:[@.offset:Cnt=@.count]`
- Блок условий корректности `:assert[@.Op>=0x80]`
- Блок отображения `:displ=(INT(2*@))`
- Блок именования `:autoname=(@.tag)`
- Блок доп. свойства типа `:let Val=(@.0)exc(@.1)`

5. Вызовы типов: `PSubSecData(Kind=@.subsec) lfo`

Основные конструкторы типов языка FlexT(1/2)

Целые числа	**num-**(6)
Пустой тип	**void**
Символьные типы	char, wchar
Перечислимые типы	
Собственно перечисление	**enum** byte (A=1,B,C)
Отображение выражением	**enum** byte = Tbl[@].Name;
Перечисление термов	**enum** TBit8 **fields** (R0: TReg @0.3,...) **of**(rts(R0) = 000020_,...)
Множество	**set** 8 **of** (OLD_SECURED ^ 0x02,...)
Запись	**struc** Byte Len **array**[@.Len]**of** Char S **ends**

Основные конструкторы типов языка FlexT(2/2)

Вариант	`case @.Kind of`
	`  vkByte: Byte`
	`else ulong`
	`endc`

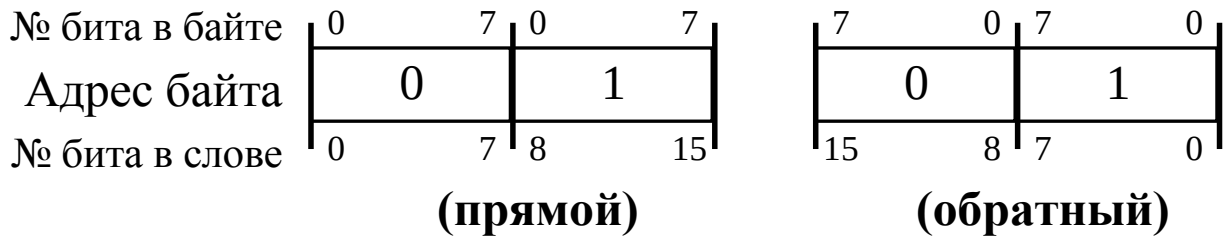
Проверка	`try`
	`  FN: TFntNum`
	`  Op: TDVIOp`
	`endt`

Массив	`array[@.Len]of str`
	`array of str ?@[0]=0!byte;`
	`array of str:[@:Size=@:ASize]`

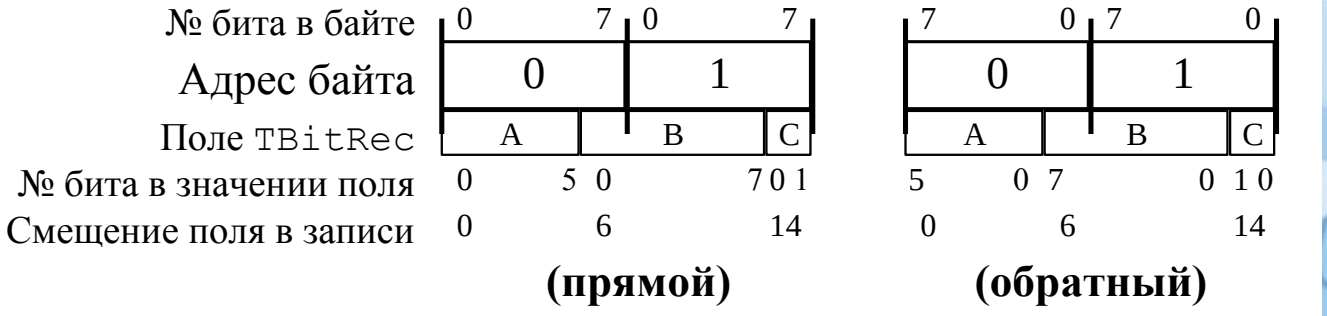
Сырые данные	`raw[@.S]`
Выравнивание	`align 16 at &@;`
Указатель	`^TTable near=DWORD, ref=@:Base+@;`
Предварительное объявление	`forward`
Машинные команды	`codes of TOpPDP ?(@.Op>=` `  TWOpCode.br)and...;`

Порядок байтов в слове и битовые типы данных

```
type bit
TBitRec struc
   num+(6) A
   num+(8) B
   num+(2) C
ends
```



Порядок байтов в слове



Побитовое размещение полей записи TBitRec

Типы данных ArcView Shape (общие для файлов SHP и SHX)

```
set byteorder rev
type
  longrev num-(4)
  ulongrev num+(4)
set byteorder norm
include Float.rfi

type
TShapeType enum long (
  Point=1, Arc=3,
  Polygon=5,MultiPoint=8)

TBBox struc
  TDouble Xmin
  TDouble Ymin
  TDouble Xmax
  TDouble Ymax
ends

TXPoint struc
  TDouble X
  TDouble Y
ends
```

```
TShpHeader struc
  longrev Magic
  longrev Rsrv0
  longrev Rsrv1
  longrev Rsrv2
  longrev Rsrv3
  longrev Rsrv4
  longrev FileLength
  long Ver
  TShapeType ShapeType
  TBBox BBox
  TDouble Zmin
  TDouble Zmax
  TDouble Mmin
  TDouble Mmax
ends:assert[@.Magic=9994,
    @.FileLength*2=FileSize,
    @.Ver=1000]
data
0 TShpHeader Hdr

assert Hdr:assert;
```

Формат файла SHP

```
include shp.rfi

type

TMultyPointData struc
  TBBox BBox
  long NumPoints
  array[@.NumPoints] of TXPoint
    Points
ends

TArcData struc
  TBBox BBox
  long NumParts
  long NumPoints
  array[@.NumParts] of long Parts
  array[@.NumPoints] of TXPoint
     Points
ends
```

```
TShapeRecData(Len) struc
  TShapeType ST
  case @.ST of
    Point: TXPoint
    Arc,Polygon: TArcData
    MultiPoint:
      TMultyPointData
  endc SD
  (raw[]at &@;) rest
ends:[@:Size=@:Len*2]:
   assert[@.ST=Hdr.ShapeType]

TShapeRec struc
  longrev RecNo
  ulongrev Len
  TShapeRecData(@.Len) Data
ends

data
Hdr:Size (array of TShapeRec:
   [@:Size=FileSize-&@]) Tbl
```

Float.rfi – представление действительных чисел

```
type bit
  TBit1 num+(1)
  TBit4 num+(4)
  TBit8 num+(8)
  TBit11 num+(11)
  TBit20 num+(20)
  TBit23 num+(23)
  TDoubleV struc
    array[13]of TBit4 M
    TBit11 P
    TBit1 S //Sign 1->Neg
  ends
  TSingle struc
    TBit23 M
    TBit8 P
    TBit1 S //Sign 1->Neg
  ends: displ=(FLOAT(@.M,@.P-0x7F,@.S))
type
  TDouble struc
    TDoubleV V
  ends: displ=(FLOAT(@.V.M,@.V.P-0x3FF,@.V.S))
```

Дамп небольшого Shape-файла

Результат разбора Shape-файла

```
000:Hdr: TShpHeader = (Magic:9994; Rsrv0:0; Rsrv1:0; Rsrv2:0; Rsrv3:0; Rsrv4:0;
  FileLength:186; Ver:1000; ShapeType:Polygon{5};
  BBox:(Xmin:27.0396; Ymin:31.1062; Xmax:27.0644; Ymax:31.1336); Zmin:0;
  Zmax:0; Mmin:0; Mmax:0)
064:Tbl: #Tf1_l44_c10 =  (
  0:(RecNo:1; Len:00000040;
    Data:(ST:Polygon{5};
      SD:(BBox:(Xmin:27.0396; Ymin:31.1062; Xmax:27.0618; Ymax:31.1336);
        NumParts:1; NumPoints:5; Parts: (0:0);
        Points: (0:(X:27.0568; Y:31.1336),1:(X:27.0618; Y:31.1302),
          2:(X:27.0448; Y:31.1062),3:(X:27.0396; Y:31.1096),
          4:(X:27.0568; Y:31.1336))); rest:[])),
  1:(RecNo:2; Len:00000040;
    Data:(ST:Polygon{5};
      SD:(BBox:(Xmin:27.055; Ymin:31.1156; Xmax:27.0644; Ymax:31.1246);
        NumParts:1; NumPoints:5; Parts: (0:0);
        Points: (0:(X:27.0578; Y:31.1246),1:(X:27.0644; Y:31.1198),
          2:(X:27.0616; Y:31.1156),3:(X:27.055; Y:31.1206),
          4:(X:27.0578; Y:31.1246))); rest:[])))
```

Источники информации о формате данных

- Описание на естественном языке

  неточность, неполнота, неоднозначность
- Исходные тексты программ чтения/записи

  информация о формате смешана с файловыми операциями, и другими командами обработки данных
- Примеры данных (reverse engineering)

  сначала спецификация отсутствует

Преимущества формальных спецификаций

- компактность, отсутствие не относящихся к способам хранения данных сведений, что облегчает их восприятие человеком;
- верифицируемость посредством их применения для анализа относящихся к формату корректных данных;
- возможность использования для локализации ошибок в сгенерированных файлах

Проверка соответствия данных спецификации

Результат разбора данных по описанию может отразить значение каждого бита исходного файла.

Разбор данных по спецификации может использоваться для:

- проверки данных на соответствие спецификации (как это делается для XML с использованием XML-схем)
- проверки спецификации на соответствие примеру данных

Спецификация, предназначенная для проверки, может допускать частично соответствующие полной спецификации данные.

Уточнение спецификации в ходе её записи на FlexT

1. Описать фрагмент формата в соответствии с тем, как понята спецификация
2. Применить спецификацию для разбора файла
3. При неудачном разборе данных исправить спецификацию, go to 2
4. Если не закончено, go to 1

Спецификация по примерам данных (reverse engineering)

- Является крайним случаем уточнения спецификации
- Лучше всего применять при наличии программы-генератора примеров данных с задаваемыми свойствами, например, компилятора
- Может применяться просто при наличии достаточного числа примеров

Команды блока displ

Строковая константа	вывод строки
'NL'[<Целочисл. выражение>]	перевод строки с отступом по уровню вложенности конструкции
'NL_'	перевод строки без отступа
'SHOWARRAY' '('<Адресное выражение> ',' '(' <Список команд>,','* ')' ')'	вывод массива
'COND' '(' <Логическое выражение> ',' '(' <Список команд>,','* ')' **[** ',' '(' <Список команд>,','* ')' **]** ')'	условный оператор
'FLOAT' '(' M:<Целочисл. выражение> ',' Pwr:<Целочисл. выражение> ',' Sign:<Целочисл.выражение> ')'	вывод действительного числа, заданного мантиссой, порядком, и знаком
'INT' <Целочисл. выражение>	вывод целого числа
'HEX' '(' V:<Целочисл. выражение> **[**',' Prec:<Целочисл. выражение>**]** ')'	вывод целого числа в шестнадцатеричном представлении
'CHR' <Целочисл. выражение>	вывод символа, заданного кодом ASCII
'WCHR' <Целочисл. выражение>	вывод символа, заданного кодом Unicode
<Адресное выражение>	вывод элемента данных
'ADDRNAME' <Целочисл. выражение>	вывод имени переменной с указанным адресом

Перевод из Shape в MIF при помощи запроса на отображение Shp2Mif.displ

```
Hdr ('Version 300',NL,'Charset "WindowsCyrillic"',NL,'Delimiter ","',NL,'CoordSys NonEarth Units "km" Bounds
      (',@.BBox.Xmin,', ',@.BBox.Ymin,') ','(',@.BBox.Xmax,', ',@.BBox.Ymax,')',NL,
      'Columns 1',NL,'  ID Integer',NL,'Data',NL)
Tbl (ShowArray(@,(
   COND(@.Data.ST=TShapeType.Polygon,(
    'REGION ',INT(@.Data.SD.Polygon.NumParts),NL_,
    ShowArray(@.Data.SD.Polygon.Points,
     (INT(@.T:N),NL_,
      ShowArray(@.T,('  ',@.X,' ',@.Y,NL_))
     ) ) ),(
   COND(@.Data.ST=TShapeType.Arc,(
    COND(@.Data.SD.Arc.NumParts=1, (
      'PLINE ',INT(@.Data.SD.Arc.NumPoints),NL_,
      ShowArray(@.Data.SD.Arc.Points[0].T,('  ',@.X,' ',@.Y,NL_))
     ),('PLINE MULTIPLE ',INT(@.Data.SD.Arc.NumParts),NL_,
      ShowArray(@.Data.SD.Arc.Points, (INT(@.T:N),NL_, ShowArray(@.T,('  ',@.X,' ',@.Y,NL_)))))),
    (COND(@.Data.ST=TShapeType.Point,(
     'POINT ',@.Data.SD.Point.X,' ',@.Data.SD.Point.Y,NL
    ),(
     'Other',NL
    )))))))))
```

Использование спецификаций для анализа машинного кода

- Машинные команды используются в:
  - исполняемых и объектных файлах
  - коде виртуальных машин
  - сценариях в: TTF, файлах инсталляторов
- исследование файлов многих форматов будет неполным без отображения содержащихся там машинных инструкций
- представление машинной команды состоит из постоянной части, задающей вид команды, и переменной части, задающей её аргументы, если таковые имеются
- Постоянная часть для байт-кодов виртуальных машин, как правило, является перечислимым типом данных
- Для машинных команд реальных процессоров используется более плотное кодирование, поэтому в постоянной части могут выделяться битовые поля, содержащие часть аргументов, например, номера регистров
- Будем называть сочетания таких битовых полей с кодом инструкции термами
- Кодовый тип данных позволяет использовать статический дизассемблер для любых спецификаций кодирования машинных команд
- Использование спецификаций кодирования машинных команд позволяет применять все методы анализа, отображения и поиска данных для анализа кода

Тип данных перечисление термов

- Задаётся список всех битовых полей, применяемых для кодирования команд
- Каждое поле характеризуется битовыми смещением, размером и типом
- Затем описываются сами термы с указанием того, какие из этих полей они используют
- Вместе с термом указывается его база – число, задающее значения не входящих в поля битов
- В записи базы терма допускается использовать подчерки для обозначения переменной части - битов, входящих в состав полей (такая маска проверяется на соответствие упоминаемым в терме полям)
- В реализации типа данных «перечисление термов» используется дерево решений, позволяющее быстро найти терм по его коду
- В выражениях можно обращаться к любым полям перечисления термов, как к полям записи, однако, если у выбранного терма это поле не используется, то такое обращение вызывает ошибку вычисления этой части выражения
- Реализована разновидность типа данных «вариант» с выбором содержимого по значению перечисления термов для описания переменной части команд. Для выбора варианта в этом случае также строится дерево решений

Кодовый тип данных

- Реализуется на базе массива со стоп-условием и записывается аналогичным образом
- Используется идентификатор codes вместо array
- Стоп-условие используется для выделения команд, завершающих последовательность, например, команд безусловного перехода или выхода из процедуры
- В описаниях команд, вызывающих переход, должны использоваться указатели на определяемый кодовый тип
- Может быть определено вычисляемое свойство isCall, которое в этом случае используется дизассемблером для различения команд перехода и вызовов процедур

Фрагменты спецификации кодирования команд процессора ARM

```
type bit
TBit num+(1)
TBit2 num+(2)
…
TsBit24 num-(24)
TRN enum TBit4 (R0,R1,… R12,SP,LR,PC)
…
TShiftOp enum TBit2 (SHL,SHR,ASR,ROR)
TBrOfs24 ^TOpARMSeq NIL- =TsBit24 REF=
  (&(@:@)+4+@*4);
type
TARMOpCode enum TBit32 fields (
  Cond: TCond @28.4, //Op.execution condition
  Rn: TRN @16.4, //Source register
  Rd: TRN @12.4, //Destination register
  Shift: TShiftOp @5.2,
…
  SM: TBit @20.1
) of (
 //Data processing rotate right with extend
  _DPRRX(cond,ArOp,S,Rn,Rd,Rm) =
    0b____000_I________I____0000I0110____,
//Move status register to register
  MSR(cond,R,Rd) =
    0b____0001I0_001111I____0000I00000000,
  …
 //Branch and Branch with link
  Branch(cond,BL,BrOfs) =
    0b____101_I________I________I________,
 //Software interrupts
  SWI(cond,SWIOfs) =
    0b____1111I________I________I________
):let IsStop=(((@.BL=0)and(@.cond>=
  TCond.AL)) exc 0
);:let isCall=((@.BL=1) exc 0);

TOpARMSeq codes of TARMOpCode ?@:isStop;:
  displ=('(', ShowArray(@,(NL,HEX(&@,4),':
',
  @)),NL,')')
```

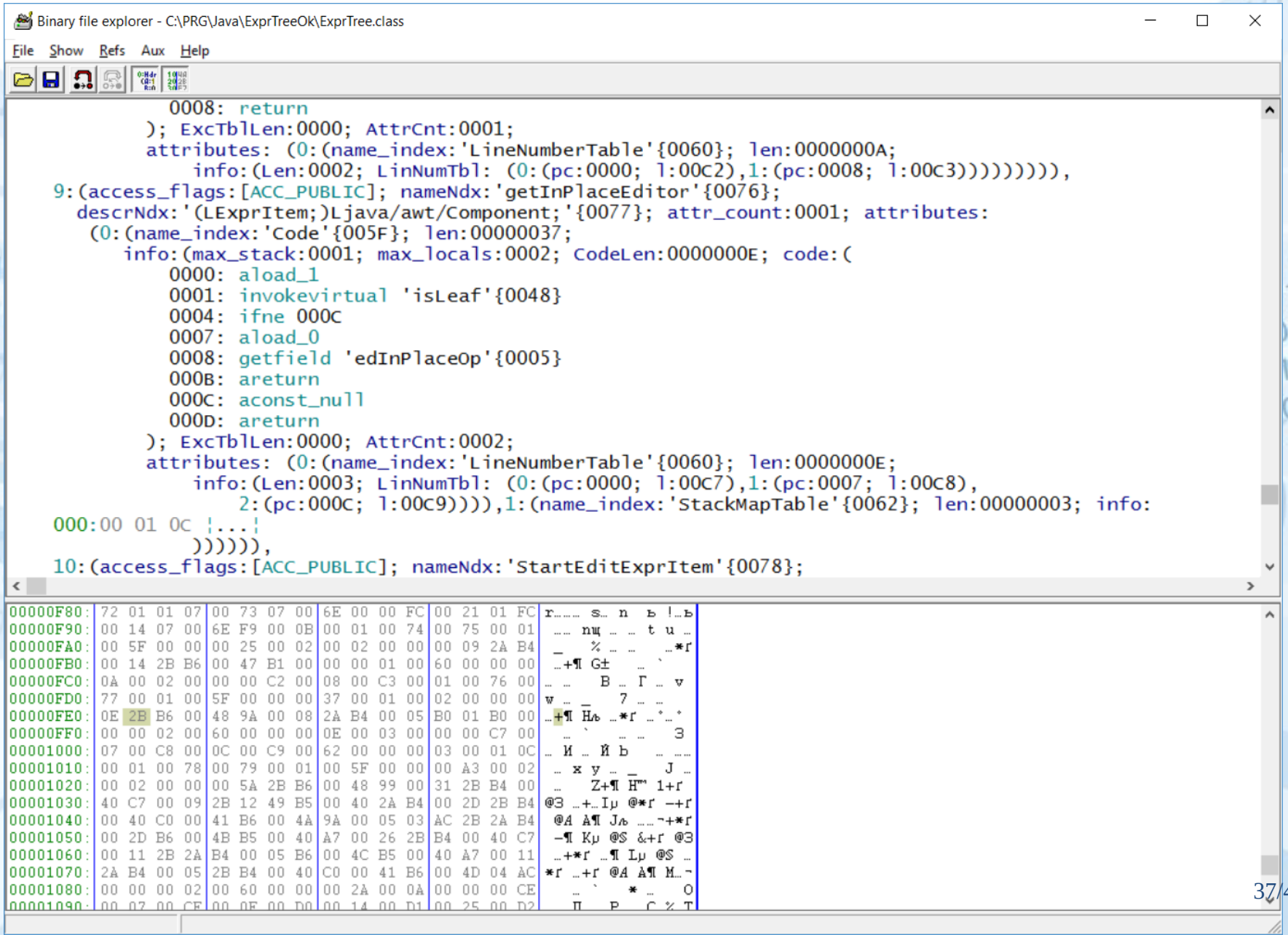
Разбор файла класса Java

Binary file explorer - C:\PRG\Java\ExprTreeOk\ExprTree.class

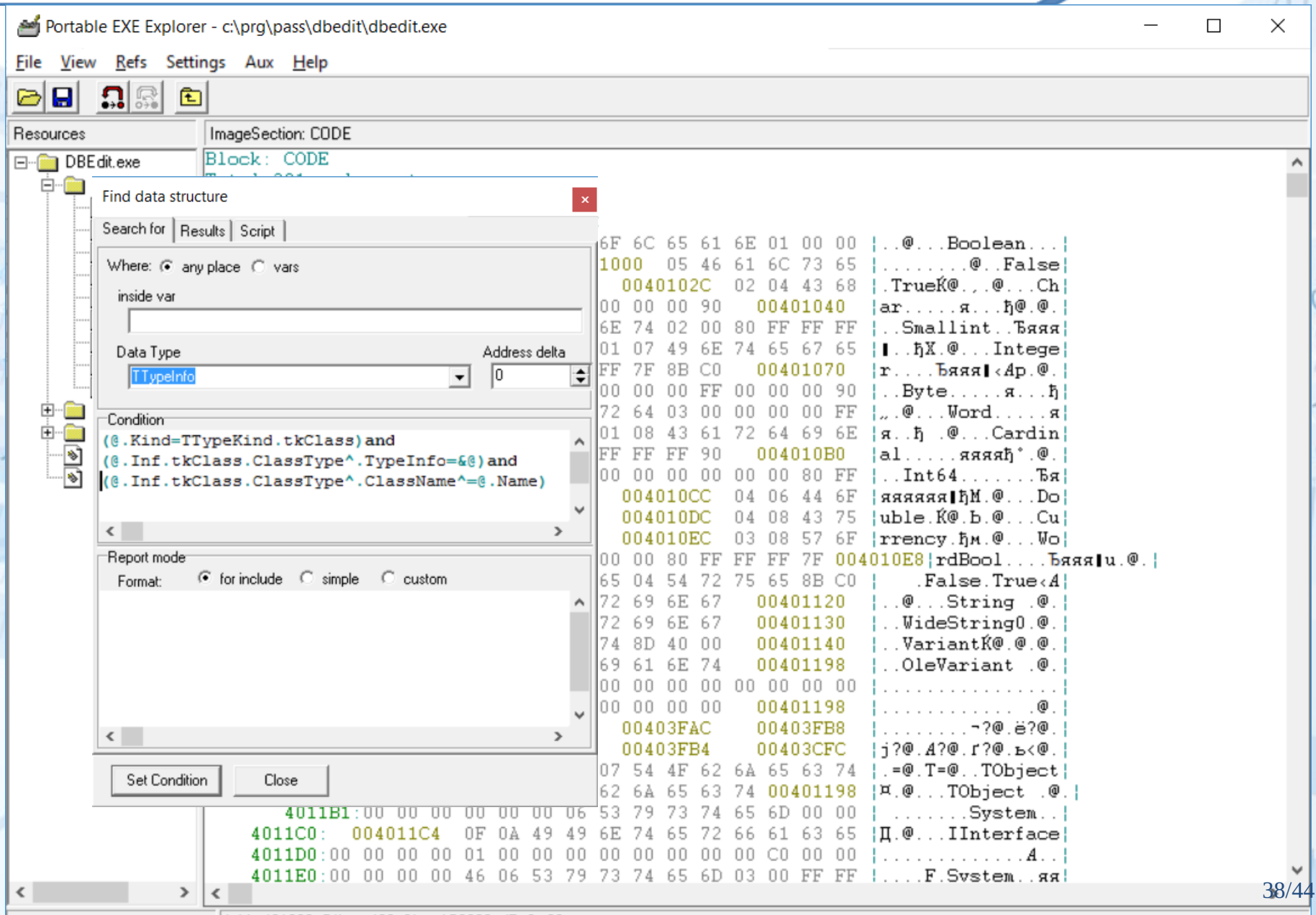
File Show Refs Aux Help

```
    descrNdx:'(LExprItem;)Ljava/awt/Component;'{0077}; attr_count:0001; attributes:
     (0:(name_index:'Code'{005F}; len:00000037;
        info:(max_stack:0001; max_locals:0002; CodeLen:0000000E; code:(
            0000: aload_1
            0001: invokevirtual 'isLeaf'{0048}
            0004: ifne 000C
            0007: aload_0
            0008: getfield 'edInPlaceOp'{0005}
            000B: areturn
            000C: aconst_null
            000D: areturn
          ); ExcTblLen:0000; AttrCnt:0002;
          attributes: (0:(name_index:'LineNumberTable'{0060}; len:0000000E;
              info:(Len:0003; LinNumTbl: (0:(pc:0000; l:00C7),1:(pc:0007; l:00C8),
                  2:(pc:000C; l:00C9)))),1:(name_index:'StackMapTable'{0062}; len:00000003; info:
000:00 01 0C |...|
            ))))),
10:(access_flags:[ACC_PUBLIC]; nameNdx:'StartEditExprItem'{0078};
  descrNdx:'(LExprItem;)Z'{0079}; attr_count:0001;
  attributes: (
    0:(name_index:'Code'{005F}; len:000000A3;
      info:(max_stack:0002; max_locals:0002; CodeLen:0000005A; code:(
            0000: aload_1
            0001: invokevirtual 'isLeaf'{0048}
            0004: ifeq 0035
            0007: aload_1
            0008: getfield 'h'{0040}
            000B: ifnonnull 0014
            000E: aload_1
            000F: ldc (Tag:C_String{08}; info:'?'{00C3}){49}
            0011: putfield 'h'{0040}
            0014: aload_0
            0015: getfield 'edVal'{002D}
            0018: aload_1
            0019: getfield 'h'{0040}
            001C: checkcast 'java/lang/String'{0041}
            001F: invokevirtual 'EditText'{004A}
            0022: ifne 0027
```

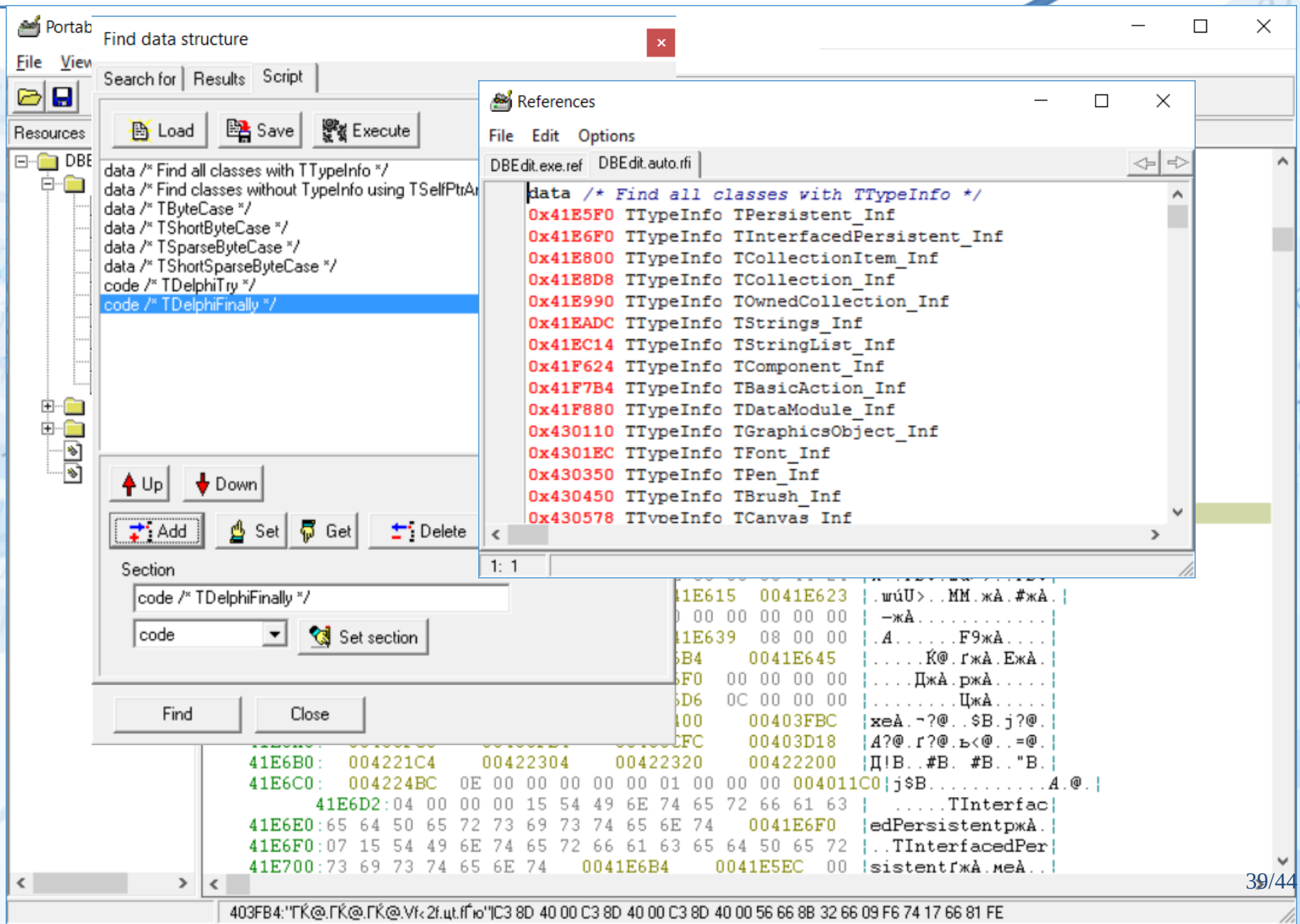
Просмотр результатов разбора и дампа



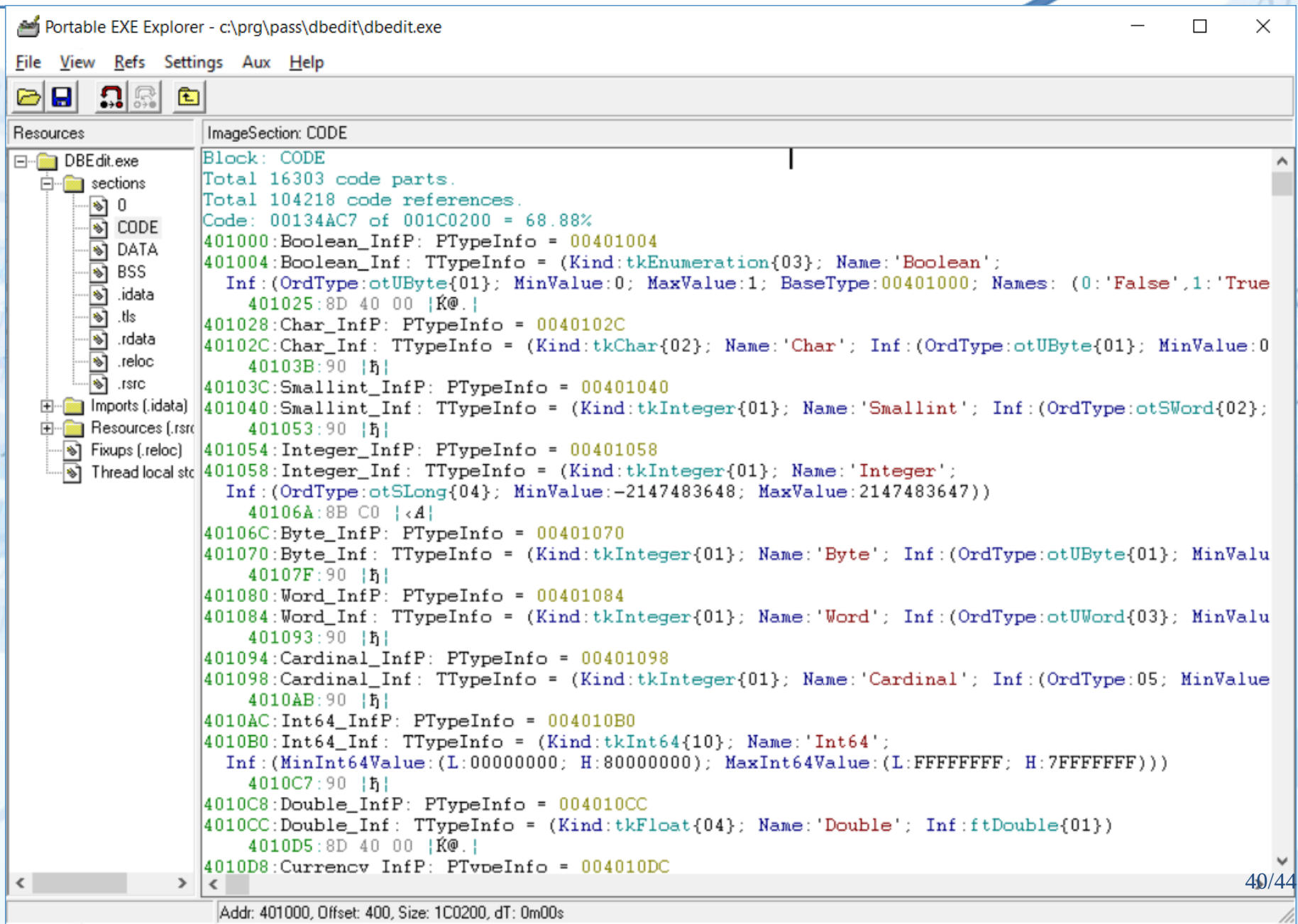
Поиск структур данных 1/4



Поиск структур данных 2/4



Поиск структур данных 3/4



Поиск структур данных 4/4

Portable EXE Explorer - c:\prg\pass\dbedit\dbedit.exe

File View Refs Settings Aux Help

Resources

- DBEdit.exe
  - sections
    - 0
    - CODE
    - DATA
    - BSS
    - .idata
    - .tls
    - .rdata
    - .reloc
    - .rsrc
  - Imports (.idata)
  - Resources (.rsr
  - Fixups (.reloc)
  - Thread local stc

ImageSection: CODE

```
clTInterfacedPersistent.IntfTable^.Entries[0].VTable^[2]:
@d29812(0041E641):
  41E62D: 83 44 24 04 F8        |ƒD$.ш  |ADD DWORD PTR [ESP+4],-8
  41E632: E9 55 3E 00 00        |úU>..  |JMP +15957; (42248C) -> @j31664
    41E637:CC CC |ММ|
41E639:clTInterfacedPersistent.IntfTable^.Entries[0].VTable^: TVirtMethodTbl =  (
  0:0041E615{->clTInterfacedPersistent.IntfTable^.Entries[0].VTable^[0]},
  1:0041E623{->clTInterfacedPersistent.IntfTable^.Entries[0].VTable^[1]},
  2:0041E62D{->clTInterfacedPersistent.IntfTable^.Entries[0].VTable^[2]},3:)
41E645:clTInterfacedPersistent.IntfTable^: TInterfaceTable = (EntryCount:1;
  Entries: (0:(IID:{00000000-0000-0000-C000-000000000046}; VTable:0041E639; IOffset:8; ImplGette
    41E665:8D 40 00 |Ќ@.|
41E668:P_TInterfacedPersistent: TClass = 0041E6B4
41E66C:clTInterfacedPersistent: TTClassData = (IntfTable:0041E645; AutoTable:00000000; InitTable
  FieldTable:00000000; MethodTable:00000000; DynamicTable:00000000; ClassName:0041E6D6; Instance
  SafeCallException:00403FAC{->clTObject.SafeCallException};
  AfterConstruction:00422400{->clTInterfacedPersistent.AfterConstruction};
  BeforeDestruction:00403FBC{->clTObject.BeforeDestruction}; Dispatch:00403FC0{->clTObject.Dispa
  DefaultHandler:00403FB4{->clTObject.DefaultHandler}; NewInstance:00403CFC{->clTObject.NewInsta
  FreeInstance:00403D18{->clTObject.FreeInstance}; Destroy:004221C4{->clTPersistent.Destroy};
  VMT: (0:00422304{->clTPersistent.VMT[0]},1:00422320{->clTPersistent.VMT[1]},2:00422200{->clTPe
    3:004224BC{->clTInterfacedPersistent.VMT[3]}))
41E6C4:_Inf: TTypeInfo = (Kind:tkRecord{0E}; Name:''; Inf:(Size:00000000; Cnt:00000001; Fields:
41E6D6:clTInterfacedPersistent.ClassName^: str = 'TInterfacedPersistent'
41E6EC:TInterfacedPersistent_InfP: PTypeInfo = 0041E6F0
41E6F0:TInterfacedPersistent_Inf: TTypeInfo = (Kind:tkClass{07}; Name:'TInterfacedPersistent';
  Inf:(ClassType:0041E6B4; ParentInfo:0041E5EC; PropCount:0; UnitName:'Classes'; PropData:(Count
    41E71B:90 |ђ|
41E71C:P_TRecall: TClass = 0041E768
41E720:clTRecall: TTClassData = (IntfTable:00000000; AutoTable:00000000; InitTable:00000000; Typ
  MethodTable:00000000; DynamicTable:00000000; ClassName:0041E768; InstanceSize:0000000C; Parent
  SafeCallException:00403FAC{->clTObject.SafeCallException}; AfterConstruction:00403FB8{->clTObj
  BeforeDestruction:00403FBC{->clTObject.BeforeDestruction}; Dispatch:00403FC0{->clTObject.Dispa
  DefaultHandler:00403FB4{->clTObject.DefaultHandler}; NewInstance:00403CFC{->clTObject.NewInsta
  FreeInstance:00403D18{->clTObject.FreeInstance}; Destroy:00422668{->clTRecall.Destroy})
```

Addr: 401000, Offset: 400, Size: 1C0200, dT: 0m00s

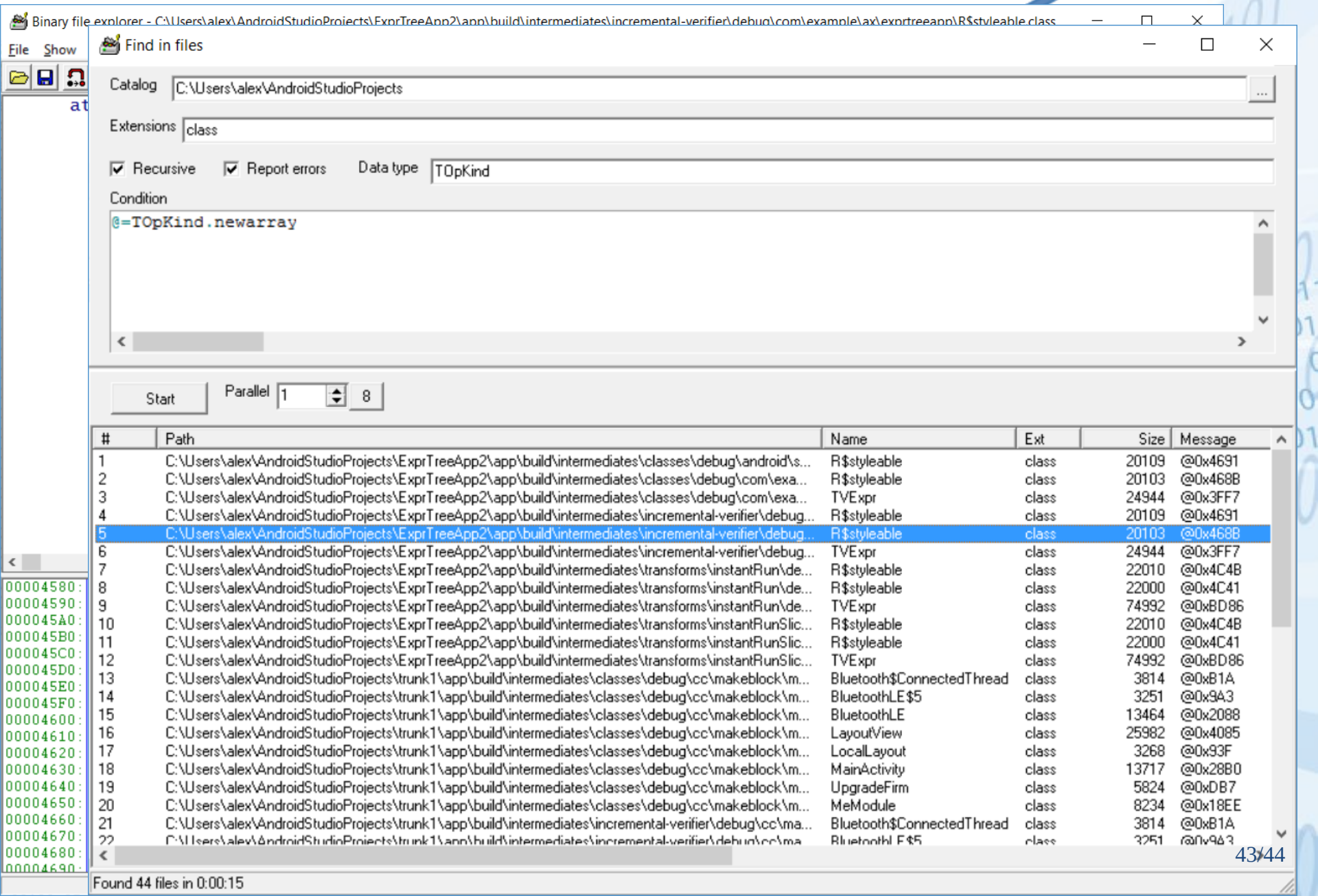
Пример применения поиска структур: отчёт по анализу кода интерпретатора => описание типов данных для конвертера

```
TPushHParentOrZ try
  P: TPushHParent
  Z: TPushZ
  Fail: TFail
endt

TRegClassCall struc
  TPushBOrD pushSize
  TSkipToPushConst Skip1
  TPushFieldTblOrZ pushF
  TPushMethodTblOrZ pushM
  TPushHParentOrZ pushP
  byte bPushClName
  PPChar ClName
  byte bCallReg
  long CallRegAddr
  byte bAdd
  word wAddArg
  TSkipToA3 Other
  byte bMov
  ulong hClAddr
ends:assert[@.pushSize.*:assert, @.pushF.*:assert,@.pushM.*:assert,
  @.pushP.*:assert, @.bPushClName=0x68, IsFixup @.ClName,
  @.bCallReg=0xE8, @.CallRegAddr+&@.bCallReg+5=0x50ED90,
  @.bAdd=0x83,@.wAddArg=0x14C4, @.bMov=0xA3]

fldlist.txt:
@.ClName^,
ShowArray(@.pushF.F.FT^,(NL,'|',@.Name^,'|',Int(@.hDT))),NL

AVCl.txt
@.ClName^,'|',AddrName(@.pushP.P.CallPar^.Addr),
ShowArray(@.pushF.F.FT^,(NL,@.Name^,'|',Int(@.hDT))),NL
```

```
DocWin|Obj
  Owner|11
  Open|18
  X|0
  Y|0
  W|0
  H|0

Doc|Obj
  Name|4
  Creator|5
  CreationDate|5
  Comments|5
  Open|5
  Update|5
  Close|5
  GUIName|4
  GroupGUIName|4
  Win|11
  CSMgr|11
  DocExts|12
  ObjectTag|11
  IsPrinting|18

ExtObj|Obj

InvRec|Obj
  Object|11
  Client|11
```

```
Project|Doc
  VisGUIWidth|0
  Flags|3
  Doc|12
  ActiveDoc|11
  Buttons|12
  Dependencies|12
  Available|12
  Scripts|11
  Prefs|11
  GUI|12
  WorkDir|11
  WinX|0
  WinY|0
  WinW|0
  WinH|0
  StartUp|5
  ShutDown|5
  SerialNumber|5
  SelColor|11
  FN|11
  Windows|13
  Modified|18
  Codepage|2
  GUINames|12
  LegendLineFlatness|2
  TocWidthScale|6
  ODBVersion|2
  ODBGUIs|12
```

Поиск в структурах данных файлов



Ограничения текущей версии FlexT

- спецификации на FlexT являются чисто декларативными, поэтому они не предназначены для описания таких форматов, при интерпретации которых необходимо строить сложные вспомогательные структуры данных, непосредственно не представленные в файле
- взаимосвязь нескольких файлов можно описать только через генерацию REF-файлов

Заключение

- при наличии спецификации на FlexT бинарный файл становится не менее прозрачным, чем текстовый
- спецификации форматов бинарных данных предоставляют альтернативу использованию текстовых форматов для упрощения контроля за правильностью работы алгоритмов чтения/записи
- возможность автоматического контроля правильности структуры файла делает спецификации на языке FlexT альтернативой использованию форматов на основе XML

Декларативный язык FlexT – инструмент анализа и документирования бинарных форматов данных

Алексей Евгеньевич Хмельнов,
И.В. Бычков,
А.А. Михайлов
Институт динамики систем и теории управления СО РАН

{hmelnov,bychkov,mikhailov}@icc.ru

http://FlexT.icc.ru

Спецификация формата BCRL

```
const
  BitCntBits = 4;
  LogMaxVal = 3;
  MinIHdrSz=BitCntBits+LogMaxVal+1;

type bit

TBit num+(1)
TBitCnt num+(BitCntBits):displ=(int(@))
TLPart num+(LogMaxVal)

TILen(Cur) struc pas
  Cont: TBit
  Part: TLPart
  Rest: case @.Cont of
   1: TILen(@@:Cur shl LogMaxVal+@@.Part)
  endc
ends:let Val=(@.Rest.1:Val exc(@:Cur shl
   LogMaxVal+@.Part));:displ=(Int(@:val))
```

```
TInterval(Lim) struc pas
  BitCnt: TBitCnt
  Len: TILen(0)
  D: array[@.Len:val+1]of
       num-(@@.BitCnt)
ends

TBCRLData(PkSz,Sz) struc pas
  D: array of TInterval(&@+
       @@:PkSz)?&@>=@:Lim!void;
  Al: align 8;
ends:[@:Size=@:PkSz*8]
```

Спецификация кодировки UTF-8

```
type
T1ByteChar char():assert[(@>0)and(@<0x80)]
T2ByteChar array[2]of Byte:assert[((@[0]and 0xE0)=0xC0)
  and((@[1]and 0xC0)=0x80)]:displ=(HEX((@[0]and 0x1F)shl 6+(@[1]and
   0x3F)))
T3ByteChar array[3]of Byte:assert[((@[0]and 0xF0)=0xE0)
  and((@[1]and 0xC0)=0x80)and((@[2]and
   0xC0)=0x80)]:displ=(HEX((@[0]and 0x0F)shl 12
  +(@[1]and 0x3F)shl 6+(@[2]and 0x3F)))

TUtf8Char try
  B1: T1ByteChar
  B2: T2ByteChar
  B3: T3ByteChar
  X: Byte
endt

TUtf8Array(Sz) array of TUtf8Char:[@:Size=@:Sz]:displ=(
  '"',ShowArray(@,(WCHR(@.B1 exc(
    ((@.B2[0]and 0x1F)shl 6+(@.B2[1]and 0x3F)) exc(
    ((@.B3[0]and 0x0F)shl 12+(@.B3[1]and 0x3F)shl 6+(@.B3[2]and 0x3F))
   exc @.X))))),'"')
```