УДК 519.6

СРЕДСТВА АВТОМАТИЗАЦИИ ДОКУМЕНТИРОВАНИЯ БОЛЬШИХ КОМПЛЕКСОВ ПРОГРАММ

О. Б. Арушанян[1], Н. А. Богомолов[1], Н. И. Волченскова[1], А. Д. Ковалев[1]

Рассматриваются проблемы документирования больших комплексов программ, а также подходы к автоматизации сопровождения программной документации, в том числе автоматизации создания программной документации в виде интерактивных документов и документов для печати.

Ключевые слова: библиотеки программ, сопровождение программных комплексов, средства автоматизации документирования, среда программирования Delphi.

1. Введение. В течение ряда лет в НИВЦ МГУ ведутся работы по развитию и сопровождению больших комплексов программ. Примерами таких комплексов программ могут служить Библиотека численного анализа [1] и программный инструментальный комплекс АДЕПТ, предназначенный для автоматизации создания приложений, ориентированных на интерактивную работу со сложными структурами данных [2].

Потребителей документации программного комплекса можно условно разделить на две категории: это пользователи программного комплекса и его разработчики. Для пользователей программного комплекса представляет интерес весь массив документации, содержащий как общие описания, включающие описания программного комплекса в целом и отдельных групп его программных элементов, соответствующих функциональной иерархии программного комплекса, так и описания "базовых" программных элементов (процедур, функций, типов данных, глобальных констант и переменных и т.д.). Для пользователей удобно, чтобы документация была представлена в виде комплексных документов, предназначенных как для печати, так и для интерактивного доступа к ним с широкой поддержкой гиперссылок.

Программистов-разработчиков программного комплекса в первую очередь интересуют описания "базовых" программных элементов, а не общие описания. Для эффективного развития и сопровождения программного комплекса эта информация должна обязательно присутствовать непосредственно в исходных текстах комплекса программ в виде комментариев. Это необходимо для облегчения согласованного изменения программных элементов и их описаний. Однако и для этой категории потребителей может оказаться полезной документация с описанием "базовых" программных элементов, представленная как в печатной, так и в интерактивной форме. Таким образом, информация о программных элементах должна одновременно существовать как в виде комментариев в исходных текстах программ, так и в виде различного рода пользовательской документации.

Описания "базовых" программных элементов составляют основной объем всего массива документации. Поэтому для разработчиков программного комплекса, осуществляющих сопровождение пользовательской документации, весьма актуальной является задача согласования описаний программных элементов, размещенных в исходных текстах программ, и описаний программных элементов, находящихся в пользовательской документации программного комплекса.

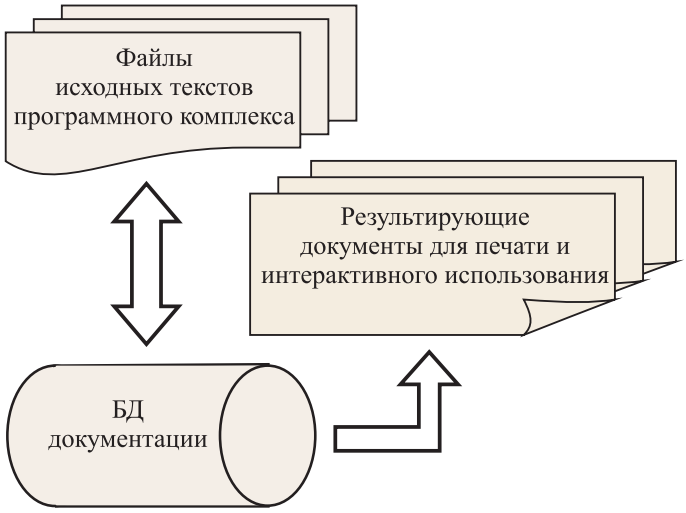
В настоящей статье описан подход к организации подготовки и сопровождения пользовательской документации, использованный для инструментального программного комплекса АДЕПТ, реализованного на языке Pascal в среде Delphi. Программные средства документирования были также реализованы на языке Pascal в среде Delphi на базе самого инструментального программного комплекса АДЕПТ.

2. Схема организации документации программного комплекса и программных средств сопровождения документации. Вся документация программного комплекса АДЕПТ строится в виде совокупности независимых фрагментов структурированного текста таким образом, чтобы каждому "базовому" программному элементу (процедуре, функции и т.д.), группе процедур, ориентированных на решение класса задач, а также более крупным разделам программного комплекса соответствовал бы отдельный фрагмент документации.

[1] Научно-исследовательский вычислительный центр, Московский государственный университет им. М. В. Ломоносова, 119991, Москва; О. Б. Арушанян, зав. лаб., e-mail: arush@srcc.msu.ru; Н. А. Богомолов, ст. науч. сотр., e-mail: nbogom@srcc.msu.ru; Н. И. Волченскова, ст. науч. сотр., e-mail: nad1946@mail.ru; А. Д. Ковалев, зав. лаб., e-mail: kovalev@srcc.msu.ru

Вся документация программного комплекса хранится в специально разработанной иерархической базе данных фрагментов документации. Фрагменты документации в базе данных делятся на фрагменты, хранящиеся только в базе данных, и фрагменты, имеющие копию в исходных текстах программного комплекса. Фрагменты документации с описанием “базовых” программных элементов (процедур, функций, типов данных, глобальных констант и переменных и др.) хранятся и в базе данных, и в исходных текстах программного комплекса.

Фрагменты документации с общими описаниями комплекса программ и его отдельных разделов могут храниться только в базе данных и не присутствовать в исходных текстах программного комплекса. Общая схема функционирования программного обеспечения (ПО) сопровождения документации приведена на рисунке. ПО документирования обеспечивает синхронизацию состава и содержимого фрагментов документации в базе данных и в файлах исходных текстов программного комплекса, а также генерацию необходимых результирующих документов, содержащих пользовательскую документацию для печати и интерактивного использования.



Общая схема функционирования ПО сопровождения документации

Средства базы данных позволяют организовывать фрагменты документации в различные иерархические структуры, которые можно использовать для последующей генерации результирующих документов с описаниями комплекса программ. При организации этих иерархических структур каждый экземпляр фрагмента документации может одновременно входить в состав неограниченного числа иерархических структур.

Программные средства сопровождения документации программного комплекса обеспечивают

— первоначальное автоматизированное наполнение базы данных фрагментами документации с описаниями “базовых” программных элементов на основе анализа исходных текстов программного комплекса;

— согласование фрагментов документации в базе данных с их копиями в исходных текстах комплекса программ на протяжении всего жизненного цикла программного комплекса;

— массовое редактирование содержимого фрагментов документации в базе данных средствами внешних текстовых редакторов (сквозная контекстная замена, поиск орфографических ошибок и др.);

— создание и хранение описаний результирующих документов в виде совокупности иерархических структур фрагментов документации;

— генерацию результирующих документов с описаниями программного комплекса в различной форме для печати и интерактивного доступа (в настоящее время реализована генерация результирующих документов в формате HTML, которые содержат необходимое стилевое оформление для последующей загрузки в Word и печати).

3. Синхронизация документации в базе данных и в исходных текстах программ. Процесс наполнения базы данных документации осуществляется следующим образом. Вначале создается пустая база, содержащая только связи с файлами исходных текстов программного комплекса. Эти связи организованы таким образом, что для каждого исходного файла программного комплекса создается соответствующий объект базы данных, в котором и накапливаются фрагменты документации, размещенные в этом файле исходных текстов.

Процедура согласования содержимого базы данных документации и файлов с исходными текстами программ осуществляет анализ каждого файла с учетом синтаксиса языка программирования. В файле с исходными текстами программ выявляются “базовые” программные элементы, которые должны быть документированы (классы, их поля и методы, процедуры и функции, записи, перечисления, множества и др.), и описывающие их фрагменты документации, которые размещаются в специальным образом организованных блоках комментариев непосредственно перед программными элементами или в одной строке с ними. Если фрагмент документации с описанием программного элемента отсутствует в базе данных, то процедура синхронизации автоматически создает в базе данных объект, соответствующий фрагменту документации в файле с исходными текстами программ. Если же фрагмент документации с описанием

программного элемента отсутствует и в исходном файле программного комплекса, то программа синхронизации создает в исходном файле программного комплекса шаблон описания найденного программного элемента, помещает его в виде комментариев в соответствующее место исходного файла и в базу для последующего заполнения разработчиком ПО. Пустой шаблон создается на основе синтаксического анализа соответствующего программного элемента.

При повторном запуске процедуры согласования осуществляется контроль содержимого фрагментов документации в базе данных и в файлах исходных текстов программ. На основе этого анализа осуществляется перенос измененных фрагментов документации из базы данных в файлы исходных текстов или из файлов исходных текстов в базу данных документации.

Помимо синхронизации информации в базе данных и в исходных текстах, процедура согласования осуществляет форматирование фрагментов документации в файлах исходных текстов программ для повышения их "читаемости", исправляя "недоделки" форматирования, оставшиеся после ручного редактирования.

4. Синтаксис разметки фрагментов документации в файлах исходных текстов. Для обеспечения возможности автоматизированной синхронизации информации в базе данных и в исходных текстах был разработан специальный язык разметки текстов комментариев, согласованный с синтаксисом языка программирования, на котором реализован комплекс программ. Разработанный язык разметки позволяет автоматически распознавать в исходных текстах программного комплекса фрагменты документации и их внутреннюю структуру. Большие усилия были предприняты для того, чтобы разметка исходных текстов не приводила к снижению уровня их "читаемости".

Фрагменты документации, описывающие программные элементы, всегда размещаются в блоке комментариев непосредственно перед соответствующим программным элементом и состоят из нескольких разделов. Все строки фрагмента документации начинаются с признака комментария ("//") с первой позиции. Ниже приведен фрагмент исходного файла на языке Паскаль с фрагментом документации, описывающим функцию K_DeleteFile.

```
//##path K_Delphi\SF\K_clib\K_CLib0.pas\K_DeleteFile
//********************************************************** K_DeleteFile ***
// Удаление файла
//
//     Parameters
// AFileName       - имя удаляемого файла
// ADeleteReadOnly - признак безусловного удаления файла, имеющего признак
//                   ''Только Чтение''
// Result          - Возвращает TRUE, если файл удален или он не существует
//
// Если файл не удалось удалить, так как он занят другим приложением, его имя
// заносится в глобальный список неудаленных файлов для последующего удаления
//
function K_DeleteFile( const AFileName : string;
                        ADeleteReadOnly : Boolean = false ) : Boolean;
begin
...
...
end; //*** end of K_DeleteFile
```

Началом фрагмента документации служит строка следующего вида:

```
//****************** <Идентификатор программного элемента> ***
```

Эта строка должна содержать после признака комментария ("//") не менее 10 символов "*" и далее не менее одного пробела, за которыми следует лексема (без пробелов), содержащая идентификатор программного элемента. В приведенном примере идентификатором программного элемента служит имя описываемой функции K_DeleteFile.

Если фрагмент уже содержится в базе данных документации, то непосредственно перед начальной строкой фрагмента располагается строка, содержащая идентификатор фрагмента документации в базе данных. Эта строка начинается с символов "//##path", за которыми через пробел следует путь к этому фрагменту в иерархической базе данных, однозначно идентифицирующий этот фрагмент документации.

Если же строка с идентификатором отсутствует, то это означает, что данный фрагмент отсутствует в базе данных и он будет добавлен в базу в процессе согласования.

Строки, следующие за начальной строкой фрагмента документации, содержат раздел с описанием назначения этого программного элемента, который состоит из одного абзаца. Концом этого раздела фрагмента документации является строка, содержащая только "//" с первой позиции.

Если описываемый программный элемент содержит внутренние элементы, то за разделом с описанием назначения программного элемента размещается раздел с описанием этих элементов. Началом этого раздела служит строка, содержащая в качестве первой лексемы ключевое слово `Elements` или `Parameters`. В приведенном примере этими элементами являются параметры и возвращаемое значение функции.

Описание каждого внутреннего элемента начинается с имени этого элемента, за которым через один или несколько пробелов следует разделитель "-", а затем через пробел начинается текст с описанием этого элемента, который может занимать несколько строк. Признаком продолжения описания элемента на следующей строке является начальная позиция текста в этой строке, сдвинутая вправо относительно начала имени элемента.

Раздел описания элементов завершается строкой, содержащей только "//" с первой позиции.

Далее фрагмент документации может содержать раздел с детальным описанием и замечаниями по использованию этого программного элемента. Этот раздел может содержать допустимые структурные элементы текста (абзацы, списки и др.). Разделителем абзацев является строка, содержащая только "//" с первой позиции. Этот же разделитель завершает описание программного элемента.

Содержащееся непосредственно за фрагментом документации описание программного элемента на языке программирования также попадает в базу данных документации и используется при генерации текста документации для пользователя.

Синтаксис представления фрагментов документации для таких программных элементов, как структуры данных, перечисления и множества, позволяет размещать описания их внутренних элементов как вне "тела" фрагмента документации, так и непосредственно "внутри" описания типа на языке программирования.

```
//##path K_Delphi\SF\K_clib\K_UDT2.pas\TK_VFile
//************************************************************ TK_VFile ***
// Описатель виртуального файла
//
// Используется во всех процедурах работы с виртуальными файлами
//
type TK_VFile = record
  VFType    : TK_VFileType;    // тип виртуального файла
  UDMemName : string;          // путь к объекту встроенной базы данных; актуален,
                               // если виртуальный файл отображается на объект БД
  UDMem : TN_UDMem;            // ссылка на объект встроенной базы данных,
                               // на который отображается виртуальный файл
  UDCBFRoot : TN_UDBase;       // корневой объект поддерева составного
                               // буферизованного файла
  DFName : string;             // имя файла данных; актуально, если виртуальный файл
                               // отображается на файл данных
  DFile  : TK_DFile;           // описатель файла данных
  DFCreatePars: TK_DFCreateParams; // параметры создания файла данных
  DFOpenFlags : TK_DFOpenFlags;    // параметры открытия файла данных
end;
```

Начало описания каждого внутреннего элемента размещается в виде комментария в строке этого элемента после разделителя ";", завершающего описание этого элемента на языке программирования. Это описание может занимать несколько строк. Признаком продолжения описания на следующей строке является признак комментария "//" размещенный не с первой позиции этой строки.

Такой комплексный программный элемент, как класс, представляется в базе данных в виде двухуровневого дерева фрагментов документации: "головной" фрагмент содержит описание назначения класса, описание его полей и замечания по использованию. В исходных текстах программного комплекса "головной" фрагмент документации размещается непосредственно перед программным объявлением класса. Описания полей класса оформляются так же, как и описания полей структур данных, элементов перечисления или множества.

```
//##path K_Delphi\SF\K_clib\K_CLib0.pas\TK_LineSegmFunc
//******************************************************* TK_LineSegmFunc ***
// Генератор значений кусочно-линейной функции
//
type TK_LineSegmFunc = class
  LSFArgVals : TN_DArray;   // массив значений аргумента
  LSFFuncVals : TN_DArray;  // массив значений функции
  LSFLastSegmInd : Integer; // номер сегмента, которому принадлежит предыдущее
                            // сгенерированное значение; задается для ускорения поиска
  constructor Create( AArgFuncPoints : array of Double );
  function Arg2Func( AArgVal  : Double ): Double;
  function Func2Arg( AFuncVal : Double ): Double;
end;
```

Фрагменты документации с описанием методов класса размещаются в базе данных документации как дочерние объекты “головного” фрагмента документации класса. В исходном файле эти фрагменты документации размещаются непосредственно перед описанием методов класса (так же как и для обычных процедур и функций):

```
//##path K_Delphi\SF\K_clib\K_CLib0.pas\TK_LineSegmFunc\Arg2Func
//********************************************** TK_LineSegmFunc.Arg2Func ***
// Вычислить значение кусочно-линейной функции по заданному значению аргумента
//
//     Parameters
// AArgVal - значение аргумента
// Result  - Возвращает значение функции, соответствующее заданному значению
//           аргумента
//
function TK_LineSegmFunc.Arg2Func( AArgVal: Double ): Double;
begin
...
end; //*** end of TK_LineSegmFunc.Arg2Func
```

5. Заключение. Программные средства документирования инструментального программного комплекса АДЕПТ реализованы на языке Pascal в среде Delphi на базе средств самого комплекса АДЕПТ. Общий объем ПО документирования составляет примерно 8000 строк программного кода. Общий объем самого программного комплекса АДЕПТ составляет примерно 350 000 строк.

В настоящее время документировано примерно 70% программного кода комплекса АДЕПТ. Результирующая документация представляет собой документ объемом примерно 760 страниц.

Опыт использования программных средств документирования комплекса АДЕПТ показал, что их применение существенно снижает затраты на создание и сопровождение пользовательской документации комплекса. Программные средства документирования инструментального программного комплекса АДЕПТ могут быть практически использованы для документирования любых комплексов программ, реализованных на языке Pascal в среде Delphi.

СПИСОК ЛИТЕРАТУРЫ

1. *Арушанян О.Б., Волченскова Н.И.* Библиотека программ НИВЦ МГУ для решения типовых задач численного анализа // Вычислительные методы и программирование. 2002. **3**, № 2. 158–163.
2. *Арушанян О.Б., Богомолов Н.А., Ковалев А.Д., Волченскова Н.И.* Об одном подходе к автоматизации создания приложений, ориентированных на работу со сложными структурами данных // Вычислительные методы и программирование. 2005. **6**, № 1. 115–123.

Поступила в редакцию
15.12.2009