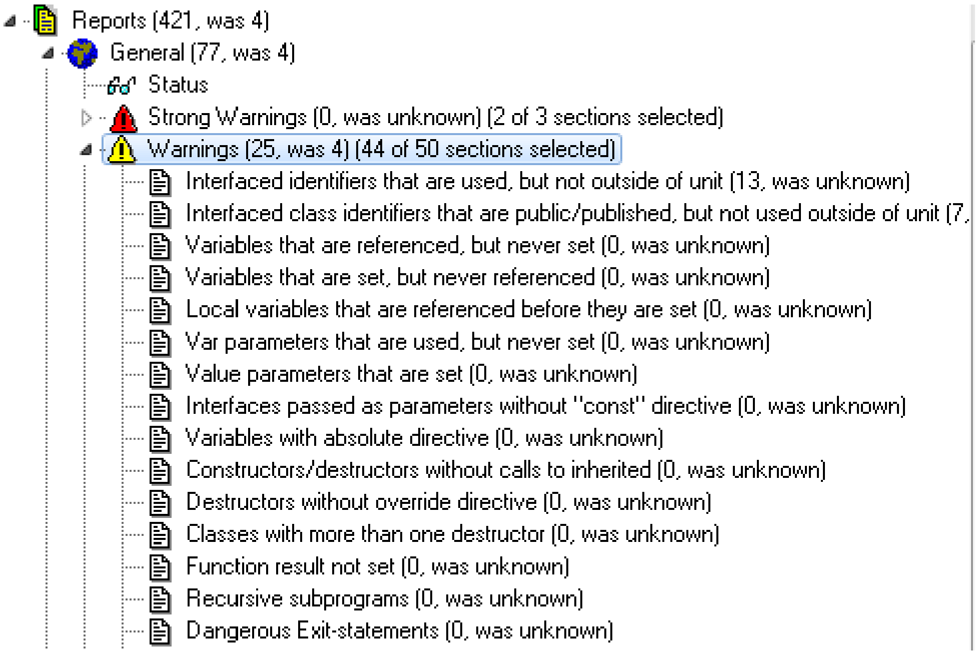
Obligatory exercise 4
Alexander Gausdal (120927)
Date: 2014-10-08

<u>Intro:</u>

The files are written in a programming language called Pascal. I don't have any experience with this language, and the files include a lot of poorly commented code. It would be hard for me to analyze all this code in a manual manner. I would need a tool to help me with this process. After hours of searching for good tools to help me with this task, I landed on a program called Pascal Analyzer, made by Pagenza. This is the only program that I could use for free (trial) that I managed to figure out how to use. It's actually the only program I could find that would analyze the files and give me warnings about security flaws in the code.

Pascal Analyzer has a very intuitive user interface, and grants warnings and notifications about security flaws in the code, among other things that could be smart to change. In this report I will take a look at some of the warnings that Pascal Analyzer generates and propose some code modifications that could fix the problems. Instead of making an unnecessarily long report with a bunch of screenshots from the program, I instead picked out some categories from the warnings section in the program and tried to create some extra insight of what the warnings actually mean and come up with some solutions to the problems.

The picture underneath is a screenshot of some of the warning categories in Pascal Analyzer [1].



Reports (421, was 4)
General (77, was 4)
Status
Strong Warnings (0, was unknown) (2 of 3 sections selected)
Warnings (25, was 4) (44 of 50 sections selected)
Interfaced identifiers that are used, but not outside of unit (13, was unknown)
Interfaced class identifiers that are public/published, but not used outside of unit (7,
Variables that are referenced, but never set (0, was unknown)
Variables that are set, but never referenced (0, was unknown)
Local variables that are referenced before they are set (0, was unknown)
Var parameters that are used, but never set (0, was unknown)
Value parameters that are set (0, was unknown)
Interfaces passed as parameters without "const" directive (0, was unknown)
Variables with absolute directive (0, was unknown)
Constructors/destructors without calls to inherited (0, was unknown)
Destructors without override directive (0, was unknown)
Classes with more than one destructor (0, was unknown)
Function result not set (0, was unknown)
Recursive subprograms (0, was unknown)
Dangerous Exit-statements (0, was unknown)

<u>Warnings:</u>

<u>Interfaced identifiers that are used, but not outside of unit:</u>

In the SDCommon.pas file there is a unit called SDCommon. The ApplicationDirectoryName (line 15) and the BackgroundBitmapFileName (line 23) identifiers are declared in the *interface* (public) section although they're only used inside this unit, never outside of it. These identifiers should be declared in the *implementation* (private) section instead.

<u>Interfaced class identifiers that are public/published, but not used outside of unit:</u>

The GetBitmap function in the dmScreenshot.pas file (line 15) and imgApplicationLogo in the fmSDAppInfo.pas file (line 16) are both declared in the public section of the class. These identifiers are never used outside of the unit, and should therefore be declared in the private/protected section instead.

<u>Redeclared identifiers from System unit:</u>

fmSDAppInfo.pas contains an identifier that uses the same name as an identifier from the System.pas unit for the compiler target (line 199). I would recommend changing the name of this identifier to avoid confusion in the future.

<u>Variables that are set, but never referenced:</u>

SDInfoProcesses.pas contains two variables that are set, but never referenced (line 625 and 626). This could be a false warning, but should be checked out.

<u>Empty finally-block:</u>

There's an empty finally-block in SDInfoProcess.pas (line 414), which is weird. If the *try* statement gets an error it's supposed to send control over to the *finally* statement which usually does something. In this case it does nothing. This should be looked at.

<u>Functions called as procedures:</u>

Many of the functions in the code are called as procedures without using the result value. For example at SDCommon.pas line 247 and 254. This could be coding errors, where the functions should be called as functions.

<u>Identifier with same name as keyword/directive:</u>

SDInfoProcesses.pas line 626 contains an identifier that conflicts with a keyword/directive ('Name'). This can be really confusing when maintaining the code. Could be smart to change it to something else.

Bibliography

[1] Peganza, [Online]. Available: http://www.peganza.com/index.htm.

[2] "Pascal Analyzer," Peganza, [Online]. Available: http://www.peganza.com/Files/PAL.PDF.