Why a "case <string> of" statement in Delphi is a dangerous idea (http://www.sandon.it/node/109)

Posted on: Tue, 06/20/2017 - 16:15 **By:** ldsandon

Over and over, people ask the case .. of statement to be expanded to allow strings, i.e.

```
case S of
  'String one': DoSomething;
  'String two': DoSomethingElse;
end;
```

Looks nice, doesn't it? Sure, but the devil is in the details. Strings are complex data types, and comparing strings is not a straightforward tasks. If you use just a single, simple encoding, and a simple language like English, a comparison may be as simple as comparing bytes. But there are different encodings, and there are many different languages.

String comparison may need a collation, which is also language dependent - because in some languages the same word may have different string representations, a classic example is the German umlaut, which could be replaced by using an (e) after the vowel, i.e. *schön* and *schoen* are actually the same word. A user with a German keyboard could enter the former easily, a user without may use the latter.

Unicode adds another complication, combining characters (https://en.wikipedia.org/wiki/Combining_characters). i.e. accents (but there are many other diacritics), which may be used instead of precomposed character - i.e. using a + ' instead of à. Again, something which may happen very, very rarely in English, but may happen quite easily in other languages.

Thereby, a "case <string> of" statement would require a lot of hidden work. First, it should ensure any comparison is made in the same encoding, performing any required conversion, including any normalization to take care of combining characters and other similar issues. Then it would need to know in what language the comparison should happen, to use the proper collation, especially when strings to compare come from user input, or external data. And there's a last issue: should the case..of comparison case sensitive or case insensitive?

As long as the Delphi compilers has to cope with ordinals, it can generate efficient code for "case... of" statements. For string, it would need to create a lot of "hidden" functions calls, and it would also need information, like the comparison language, that could be ambiguous.

Does it looks so nice still? It could easily become code that brings more issues than it should. And it is really needed? Often strings are used instead of better, less ambiguous types because of laziness and bad habits (web programmers are among the worst offender when it comes to use strings as the "universal data type").

Often strings can be mapped to ordinals early, and these used through the code, avoiding the issues above and being tied to a given language.

Then there is the code the needs to perform real string processing. In these cases, the developers need to know that strings are not simple sequences of bytes. They are fairly more complex, and there are far better ways to process them properly.

If you're a native English speaker and developer, remember that English is not the only language on the planet. It may be the language of the internet, but software often has to process many more different languages. Some syntactic sugar to save some typing won't help to write better applications, it's just a way to write worse ones.

PS: that's I'm quite sure a bad implementation will be added to some upcoming version of Delphi.

- Log in (http://www.sandon.it/user/login?destination=/comment/reply/node/109/comment_node_blog%23comment-form) to post comments

Source URL: *http://www.sandon.it/node/109*