Pascal

Commentary

After doing a few of these I feel like I have a good grasp of what I need from a language to make a caesar cipher. I start by finding a compiler and getting hello world to run. From there, I look up how to create a function. After that I create essentially an “echo” function, which just returns one of the parameters I give it. So far, Pascal has had very helpful online documentation and it is easy for me to find all of the resources I need. Next I get into the meat of the program, needing helper functions like toUpper, if the language has it, a way to do modulus, a looping structure, and a char to int conversion and vice versa. Once again, this info is readily available. I assemble this into the way that I know how to by now, and voila, a working program. No big problems arose during the development of this program.

Google Searches

- Pascal getting started
- Freepascal ubuntu
- Pascal touppercase
- Pascal for loop syntax
- Pascal if syntax
- Pascal modulus
- Pascal character conversion
- Pascal check if character is a letter

Caesar Implementation

```
{compile and run with fpc -v0 caesar.pas && ./caesar}
program Hello;
uses sysutils;
function encrypt(str:string; shift:integer): string;
var i: integer;
begin
    str:=upcase(str);
    for i:=1 to length(str) do
        begin
            {test for if it is a letter, at this point all is uppercase}
            if (str[i] in ['A'..'Z']) then
                begin
                    str[i]:= chr((ord(str[i]) -65 + shift) mod 26 + 65);
                end;
        end;
    encrypt:=str;
end;

function decrypt(str:string; shift:integer): string;
var i: integer;
begin
    decrypt:=encrypt(str, 26-shift);
```

```
end;

procedure solve(str:string; maxShiftValue:integer);
var i: integer;
begin
    str:=upcase(str);
    for i:=1 to 26 do
        begin
            writeln('Shift: ', i, ' Result: ', decrypt(str, i));
        end;
end;

begin
    writeln(encrypt('ATTACK AT ONCE',4));
    writeln(decrypt('EXXEGO EX SRGI',4));
    solve('abcdeFGHIJKLmnopqrstuvwxyz ,?;{[()]}', 26)
end.
```

Output

```
EXXEGO EX SRGI
ATTACK AT ONCE
Shift: 1 Result: ZABCDEFGHIJKLMNOPQRSTUVWXY ,?;{[()]}
Shift: 2 Result: YZABCDEFGHIJKLMNOPQRSTUVWX ,?;{[()]}
Shift: 3 Result: XYZABCDEFGHIJKLMNOPQRSTUVW ,?;{[()]}
Shift: 4 Result: WXYZABCDEFGHIJKLMNOPQRSTUV ,?;{[()]}
Shift: 5 Result: VWXYZABCDEFGHIJKLMNOPQRSTU ,?;{[()]}
Shift: 6 Result: UVWXYZABCDEFGHIJKLMNOPQRST ,?;{[()]}
Shift: 7 Result: TUVWXYZABCDEFGHIJKLMNOPQRS ,?;{[()]}
Shift: 8 Result: STUVWXYZABCDEFGHIJKLMNOPQR ,?;{[()]}
Shift: 9 Result: RSTUVWXYZABCDEFGHIJKLMNOPQ ,?;{[()]}
Shift: 10 Result: QRSTUVWXYZABCDEFGHIJKLMNOP ,?;{[()]}
Shift: 11 Result: PQRSTUVWXYZABCDEFGHIJKLMNO ,?;{[()]}
Shift: 12 Result: OPQRSTUVWXYZABCDEFGHIJKLMN ,?;{[()]}
Shift: 13 Result: NOPQRSTUVWXYZABCDEFGHIJKLM ,?;{[()]}
Shift: 14 Result: MNOPQRSTUVWXYZABCDEFGHIJKL ,?;{[()]}
Shift: 15 Result: LMNOPQRSTUVWXYZABCDEFGHIJK ,?;{[()]}
Shift: 16 Result: KLMNOPQRSTUVWXYZABCDEFGHIJ ,?;{[()]}
Shift: 17 Result: JKLMNOPQRSTUVWXYZABCDEFGHI ,?;{[()]}
Shift: 18 Result: IJKLMNOPQRSTUVWXYZABCDEFGH ,?;{[()]}
Shift: 19 Result: HIJKLMNOPQRSTUVWXYZABCDEFG ,?;{[()]}
Shift: 20 Result: GHIJKLMNOPQRSTUVWXYZABCDEF ,?;{[()]}
Shift: 21 Result: FGHIJKLMNOPQRSTUVWXYZABCDE ,?;{[()]}
Shift: 22 Result: EFGHIJKLMNOPQRSTUVWXYZABCD ,?;{[()]}
Shift: 23 Result: DEFGHIJKLMNOPQRSTUVWXYZABC ,?;{[()]}
Shift: 24 Result: CDEFGHIJKLMNOPQRSTUVWXYZAB ,?;{[()]}
Shift: 25 Result: BCDEFGHIJKLMNOPQRSTUVWXYZA ,?;{[()]}
Shift: 26 Result: ABCDEFGHIJKLMNOPQRSTUVWXYZ ,?;{[()]}
```

Log

Prediction: 2 hours

Date	Hours Spent	Accomplishments
2/7	0.5	Set up dev env
2/10	0.5	Find toUpper, modulus, for loops, and char int conversion documentation
2/14	1	Implement encrypt, decrypt, solve
2/21	0.5	Test thoroughly

Discrepancy of time

This language went by really quick. Largely due to good documentation, which I am learning to be one of the most important aspect of a language. I still went a little bit over budget due to extensive testing, but nothing really arose during the testing period except for a few minor tweaks.

Overall Review

This language is a pretty good language. It has a lot of strange syntactical choices, I think. For example, the walrus operator for equals is not a common thing in other languages and adds to the cost of using the language. I know you have mentioned that you like this decision, but on my personal machine I use font ligatures which change the == into a different symbol so this was never a problem for me. They do similar odd things like this in a few spots... the use of writeln vs println, comments, semicolons in the function typing parameters instead of commas, thing like that do not some to add anything to the language and are just being done to be different. One thing that was kind of strange to me is the need to declare variables at the function header but the compiler gives a good enough error message so that it is something I can search for, and once again quickly resolve the problem. I found that the toUpper function is located in the sysutils library, so that is something new, I haven’t had to import anything in the other languages. I kind of like that extra functionality can be located in external libraries to reduce the size of the compiled program. This makes the language more orthogonal by default, while still allowing for extra convenience where it makes sense.

Ratings

Readability: 7/10

Writability: 6/10

Ranking: 3/5