Fortran

Commentary

I started with a search for “Fortran examples.” At this point, a new problem presented itself; it became apparent that there are enough versions of fortran that my search wasn’t all that enlightening. I then went to the official documentation, which was somewhat helpful. I was able to quickly learn a lot of the syntax and how common things like variables and loops are implemented in Fortran. I noticed I didn’t see a section for “string methods” or “helper functions.” At this point I knew I would be writing those. I started programming a hello world on IDEone.com and got it on my first try but the compiler gave me a warning about using tabs. Did Fortran come before tabs? I wind up googling about how to get rid of it and I’m told to use "-Wtabs" as a flag when compiling. I ignore this because I’m not sure if I can control that on IDEone. My approach for creating the Caesar cipher is to make a function to handle shifting a single letter, and then carry that to the whole word in the encrypt and decrypt functions. I’m happily coding along at this point and while trying to get a single letter to work, I come across a runtime error. I wish I could tell you what it was for, but I’m just passing along the same message Fortran gave me, “Runtime Error.” This was painful to debug, I fiddled around for 15 minutes and deleted everything I changed and re-implemented it, somehow working the second time. Getting the loop to work was a pain. The syntax for accessing the index of a string is a bit strange, getting a slice of one element. After implementing the for loop, my function would successfully compute the encrypted cipher-text but there were random characters like “���□�� ◻V@◻◻V” following the result. I was able to fix this after some help from a classmate, and figuring out I can just replace the empty space at the end of a string with space characters.

Google Searches

- fortran examples
- fortran hello world
- fortran getting started
- fortran string methods
- fortran helper functions
- fortran compiler warning tabs
- ideone compiler settings
- fortran character to int
- fortran strings
- fortran "runtime error"
- fortran index of
- fortran illegible characters end of string

Caesar Implementation

```
! compile and run with gfortran caesar.f90 && ./a.out
program Caesar
```

```
    character(1) :: shift
    character(64) :: encrypt
    character(64) :: decrypt

    print *, encrypt('ATTACK AT ONCE', 4)
    print *, decrypt('EXXEGO EX SRGI', 4)

    call solve("abcdeFGHIJKLmnopqrstuvwxyz ,?;{[()]}", 26)

    stop
end

function encrypt(str, shiftAmt) result(res)
    character(*) :: str
    character(1) :: shift
    integer :: shiftAmt
    character(*) :: res
    integer :: i
    integer :: j

    do i = 1, len_trim(str)
        res(i:i) = shift(str(i:i), shiftAmt)
    end do
    do j = len_trim(str) + 1, 64
       res(j:j) = ' '
    end do


end function

function decrypt(str, shiftAmt) result(res)
    character(*) :: str
    integer :: shiftAmt
    character(64) :: res
    character(64) :: encrypt


    res = encrypt(str, -shiftAmt)

end function

subroutine solve(str, maxShiftValue)
    implicit none
    character(*) :: str
    integer :: maxShiftValue
    character(64) :: decrypt
    integer :: i

    do i = 1, maxShiftValue
        print *, 'Shift: ', i, decrypt(str, i)
    end do

end subroutine
```

```
function toUpper(chr) result(upper)
    implicit none
    character :: chr
    character :: upper
    integer :: charcode

    charcode = iachar(chr)

    if(charcode >= 97 .and. charcode <= 122) then
        ! lowercase
        upper = achar(charcode - 32)
    else
        upper = chr
    end if

end function

function shift(chr, num) result(shifted)
    implicit none
    character :: chr
    integer :: num
    character :: shifted
    integer :: shiftedCode
    character :: toUpper

    shiftedCode = iachar(toUpper(chr))

    if(shiftedCode >= 64 .and. shiftedCode <= 90) then
        shiftedCode = modulo(shiftedCode - 65 + num, 26) + 65
    end if

    shifted = achar(shiftedCode)

    !print *, chr, ' -> ', shifted


end function
```

Output

```
EXXEGO EX SRGI
ATTACK AT ONCE
Shift:            1 ZABCDEFGHIJKLMNOPQRSTUVWXY ,?;{[()]}
Shift:            2 YZABCDEFGHIJKLMNOPQRSTUVWX ,?;{[()]}
Shift:            3 XYZABCDEFGHIJKLMNOPQRSTUVW ,?;{[()]}
Shift:            4 WXYZABCDEFGHIJKLMNOPQRSTUV ,?;{[()]}
Shift:            5 VWXYZABCDEFGHIJKLMNOPQRSTU ,?;{[()]}
Shift:            6 UVWXYZABCDEFGHIJKLMNOPQRST ,?;{[()]}
Shift:            7 TUVWXYZABCDEFGHIJKLMNOPQRS ,?;{[()]}
Shift:            8 STUVWXYZABCDEFGHIJKLMNOPQR ,?;{[()]}
Shift:            9 RSTUVWXYZABCDEFGHIJKLMNOPQ ,?;{[()]}
```

```
 Shift:           10 QRSTUVWXYZABCDEFGHIJKLMNOP ,?;{[()]}
 Shift:           11 PQRSTUVWXYZABCDEFGHIJKLMNO ,?;{[()]}
 Shift:           12 OPQRSTUVWXYZABCDEFGHIJKLMN ,?;{[()]}
 Shift:           13 NOPQRSTUVWXYZABCDEFGHIJKLM ,?;{[()]}
 Shift:           14 MNOPQRSTUVWXYZABCDEFGHIJKL ,?;{[()]}
 Shift:           15 LMNOPQRSTUVWXYZABCDEFGHIJK ,?;{[()]}
 Shift:           16 KLMNOPQRSTUVWXYZABCDEFGHIJ ,?;{[()]}
 Shift:           17 JKLMNOPQRSTUVWXYZABCDEFGHI ,?;{[()]}
 Shift:           18 IJKLMNOPQRSTUVWXYZABCDEFGH ,?;{[()]}
 Shift:           19 HIJKLMNOPQRSTUVWXYZABCDEFG ,?;{[()]}
 Shift:           20 GHIJKLMNOPQRSTUVWXYZABCDEF ,?;{[()]}
 Shift:           21 FGHIJKLMNOPQRSTUVWXYZABCDE ,?;{[()]}
 Shift:           22 EFGHIJKLMNOPQRSTUVWXYZABCD ,?;{[()]}
 Shift:           23 DEFGHIJKLMNOPQRSTUVWXYZABC ,?;{[()]}
 Shift:           24 CDEFGHIJKLMNOPQRSTUVWXYZAB ,?;{[()]}
 Shift:           25 BCDEFGHIJKLMNOPQRSTUVWXYZA ,?;{[()]}
 Shift:           26 ABCDEFGHIJKLMNOPQRSTUVWXYZ ,?;{[()]}
```

Log

Prediction: 2 hours

Date	Hours Spent	Accomplishments
1/26	2	Get familiar with Fortran, Create a function to shift a single letter
1/27	1	Figure out how to apply it to a variable length string
1/27	1	Handle non alphabetic characters
1/27	0.5	Create “solve” subroutine
2/18	0.5	Fix the random characters caused by overallocating
2/21	0.5	Test thoroughly

Discrepancy of time

I think the discrepancy in time taken to complete this program is just how different this language is from most of what I've done in the past. Also, IDEone definitely added time, my development was much smoother once I removed them from the equation. This was also my first iteration through creating a caesar cipher, at least in a little while, so I had to actually debug my code to make sure my standard algorithm was working. This increasingly led to extra time with the terrible error messages given in the event of a runtime error.

Overall Review

Fortran was tough to develop mainly because of how different some of the syntax was. The readability and writability are both not that great. One thing in particular that was annoying was having to declare the types of everything multiple times when it should be able to inferred by the context of the program. This may have increased some readability by having the types located close to the functions but I feel the writability was hurt more than that. I'm not a huge

fan of the use of end in place of curly braces, they tend to blend into the rest of the program. I think programming languages made the right move going forward with the use of symbols. It's also unclear to me why the for loops don't have parentheses but the if statements do, this seems like something that should be consistent throughout he language.

Ratings

Readability: 5/10

Writability: 4/10

Ranking: 4th