CMPT 331 - Spring 2022 - Dr. Labouseur

Programming In The Past

Jake Vissicchio
Jake.Vissicchio1@Marist.edu

September 27, 2022

1 Log

Log

Date:	Hours Spent:	Tasks/Accomplishments/Issues/Thoughts:
2/18	1	In Fortran you need to know the actual length of a string variable which could be really annoying if you have a long string
2/18	2	I kept getting this error ”non-conforming tab character”. I was able to fix it by removing all tabs but now the code is much harder to read so that is pretty annoying
2/18	3	Having trouble figuring out how to get each character in my string. Also just realized I am already off on my prediction. Also having trouble thinking of the math to handle when we reach the end of the alphabet to go back to the start after shifting
2/18	4	Tested ”THIS IS A STRING FROM ALAN” and realized I need to add an if statement to not shift the ASCII for spaces which converts to 32. So now it works. I think decrypt shouldn’t be too bad since its just encrypting backwards instead of forwards
2/18	4 1/2	I was right about it not being bad at all so that’s good. Onto solve
2/18	5	I thought I understood solve but I am getting a runtime error. My problem could stem from the fact that I have in encrypt that the string has a length of 26 while the string I want to solve is ”HAL” and that has a length of 3. I need to find a way for it to find out the length of the string passed to it without being hardcoded.
2/18	5 1/2	Solve now works with ”HAL”. I also set a toUpper subroutine which was funny because most languages just have an easy toUpperCase method or some sort, but this one effects the spaces too so it won’t work in this case.
2/19	6	Moving onto Pascal and thinking it won’t take nearly as long as Fortran.
2/19	6 1/2	Really weird issue in if/else statements and I finally figured out what was wrong. Turns out you put no semicolon for the if statement but you do for the else
2/19	6 1/2	Now I am getting the error ”Operator is not overloaded”
2/19	6 3/4	Turns out my parentheses were off in my math. ZABC now works as intended for Encrypt.

Log ctnd.

Date:	Hours Spent:	Tasks/Accomplishments/Issues/Thoughts:
2/19	7	Decrypt is not working as intended, I believe my math might be slightly off because I am getting weird characters.
2/19	7 1/4	For some reason the program believes -1 mod 26 = -1 when it should equal 25

2 Commentary

This section includes random thoughts that came to my head while I was doing the assignment (sort of like a diary).

2.1 Fortran

- Starting out with Fortran. I'm using ideone and just making a Hello World Program and so far not too complicated.

- What's the deal with all of these '*'s.

- Writing lines without a semicolon at the end just feels wrong.

- String variables are obnoxious. character(x)

- ifs aren't too bad considering it's kind of like how you would say it in so the "then" makes sense.
- I saw something that says implicit none should always be used in the beginning so I'm going to take their advice. It prevents confusion which is always good.

- do is the keyword for handling loops which is not too confusing

- subroutines seem a little confusing with intent(in) and intent(inout)

- I caught myself accidentally typing a bracket when making my encrypt subroutine. Makes me miss Java.

- '!' as a comment identifier is a little odd but it works

- non-conforming tab member is an extremely obnoxious error and the only way I found out how to fix it was making my code harder to read

- noticed you cannot declare i as an integer in the same line of the do statement unlike in Java when you can easily say int i = 0 right in the for statement

- I found achar and iachar to convert ascii and strings which will help with the shifting

- I found that you can use strings kind of like arrays allowing you to get a specific index or character with (x:x) with x being an integer

- So for encrypt I can loop through the string getting each character and shifting it then move to the next character.

- the math is kind of difficult so I started testing in visual studio with Java to find out how to make 90 go to 65 when adding 1.

- After doing some research on caesar ciphers and found a discrete math video. I saw the math without ASCII values would be the $(value + shift)\%26$. (26 represents the total number of letters in the alphabet) So we can do the math with ASCII values by first subtracting 65, which is 'A' in ASCII, from the value, then add the shift, then mod 26 which will give the remainder which would be number from 0 - 25 depending on our value, then adding the 65 back in, which should give us the final index for the shift.

- there is no x%y for mod in fortran, instead you put modulo(x, y) which is kind of weird

- a little after writing 4 3/4 in the log I realized the problem could be because I forgot to put write within solve so that could be why it is not printing anything. - nevermind, still getting a runtime error

- also realized I can call encrypt inside of decrypt and solve since it reuses a lot of code

- after doing a lot of research I found character(*) which assumes the length of the string passed. It must be a dummy argument or a PARAMETER or else I get an error so it won't actually work in the first declaration at the top.

- Solve prints kind of weird, the numbers are very far right from "Caesar " but it works

2.2 PASCAL

- Still using ideone. Printed "Hello World"

- variables are interesting. I usually tend to put my variables at the top anyway for neatness so it actually is a really good idea to force neatness on programmers since it will really help in group settings.

- familiar with := because Alan++ uses it. I also like it because it adds a layer of readability since it allows you not to confuse "=" with "=="

- the for loops have a very "English" feel with it having keywords like "to" and "do" which increase readability but slows down writability

- begin and end are basically '' and '' in other languages like java but I do have to say I like the brackets better much more writable

- similar to what I did for Fortran I can treat my string as an array and go through each letter and shift them

- instead of using iachar and achar like I did for Fortran, it has chr and ord

- Pascal was a nice step up from Fortran.

2.3 Scala

- Still using ideone. Printed "Hello World". One weird thing I noticed is how long it takes to compile compared to the other languages.

- while researching I notice it being quite similar to Java with its use of objects and brackets.

- Scala for loops using "¡-" is good in a readable perspective however it's kind of annoying the write.

- The errors are really annoying and do not give a lot of context

- I was able to get encrypt to print properly but it took some hair pulling on the way.

- I was able to get decrypt to print properly after foolishly being confused why I was going back double until I realized it's because I was negatively shifting from the original string rather than the new encrypted string so I needed to make use of return in my methods.

- Honestly do not have a ton to say about Scala since using it just kind of felt like a "poor-man's" Java, but I do really enjoy it since it's readable and more writable than the languages I have worked on so far.

2.4 Ruby

- Wow Ruby looks incredibly simple at a glance. Has a similar feel to Python.

- I used a similar implementation to Scala and it was able to work well.

- I used str.bytes with str being a String which allows me to get an array of ASCII values to use for encrypt.

- The code is arguably very poorly readable but by far the easiest to write compared to the other languages in the project.

- This is most likely going in the top of my list because it was the least amount of hair pulled.

- I believe Ruby would be an amazing language to start with for beginners because of how lenient it is, however because it is too lenient I can see it causing a lot of problems with readability in a group setting.

3 Code Listings

3.1 Fortran

```
!The ide was having issues with indents so the code looks a little messy because
!of that
PROGRAM MAIN
implicit none
!variables
character(26) :: str = 'za bc'
integer :: shift = 1
!integer :: i
call toUpper(str)
print *, str
call encrypt(str, shift)
write(*,*) str
call decrypt(str, shift)
write(*,*) str
call solve('HAL', 26)

contains

subroutine encrypt(string, shiftAmount)
character(*) :: string
integer :: shiftAmount
integer :: i
!loop from 1 to the length of the given string
do i = 1, len(string)
!we need to skip spaces
if(iachar(string(i : i)) == 32) then
string(i : i) = achar(32)
else
string(i : i) = achar(modulo(iachar(string(i : i)) - 65 + shiftAmount, 26) + 65)
end if
end do
end subroutine

subroutine decrypt(string, shiftAmount)
character(*) :: string
!integer :: j
integer :: shiftAmount
!we need to inverse the shiftAmount to bring us back to where we started
shiftAmount = shiftAmount * (-1)
!integer :: j for some weird reason I was getting an error with
!declaring j here
!loop from 1 to the length of the given string
call encrypt(string, shiftAmount)
end subroutine

subroutine solve(string, maxShiftValue)
character(*) :: string
integer :: maxShiftValue
character(len(string)) :: tempString
integer :: k
!loop all the way to the maxShiftValue given, encrypting for
each value up until then
do k = 0, maxShiftValue
tempString = string
call encrypt(tempString, k)
write(*,*) 'Caesar', k, ':', tempString
end do
end subroutine

subroutine toUpper(string)
character(*) :: string
integer :: l
!loop from 1 to the length of the given string
do l = 1, len(string)
!we need to skip spaces
if(iachar(string(l : l)) >= 97 ) then
if (iachar(string(l : l)) <= 122 ) then
string(l : l) = achar(iachar(string(l : l)) - 32)
end if
end if
end do
end subroutine

END
```

3.2 Pascal

```
program ideone;
procedure encrypt(var myStr: string; shiftAmount: integer);
var
        i: integer;
        charVal: integer;
begin
        for i := 1 to length(myStr) do
        begin
                charVal := ord(myStr[i]);
                if (charVal = 32) then
                        myStr[i] := chr(32)
                else
                begin
                        if (shiftAmount < 0) then
                                shiftAmount := shiftAmount + 26;
                        myStr[i] := chr(((ord(myStr[i]) - 65 + shiftAmount)
                            mod 26) + 65);
                end;
        end;
end;

procedure decrypt(var myStr: string; shiftAmount: integer);
begin
        shiftAmount := shiftAmount * -1;
        encrypt(myStr, shiftAmount);
end;

procedure solve(var myStr: string; maxShiftValue: integer);
var
        tempString: string;
        j: integer;
begin
        for j := 0 to maxShiftValue do
        begin
                tempString := myStr;
                encrypt(tempString, j);
                writeln('Caesar ', j, ': ', tempString);
        end;
end;

var
        str: string;
        shift: integer;
        solveStr: string;
        maxShift: integer;
        newStr: string;
begin
        str := 'za bc';
        shift := 1;
        newStr := upcase(str);
        writeln(newStr);
        encrypt(newStr, shift);
        writeln(newStr);
        decrypt(newStr, shift);
        writeln(newStr);
        solveStr := 'HAL';
        maxShift := 26;
        solve(solveStr, maxShift);
End.
```

3.3 Scala

```
object Main
{
        def main(args: Array[String])
        {
                var str: String = "zabc";
                var upperStr: String = "";
                upperStr = str.toUpperCase();
                var shift: Int = 1;
                println(upperStr);
                //this.encrypt(upperStr, shift);
                var encryptedString: String = this.encrypt(upperStr, shift);
                this.decrypt(encryptedString, shift);
                this.solve("HAL", 26);
        }//main

        def encrypt(myStr: String, shiftAmount: Int): String =
        {
                var i: Int = 0;
                var chars = myStr.toCharArray();
                var newShift: Int = shiftAmount;
                var newString: String = "";
                for (i <- chars)
                {
                        if (i.toInt == 32)
                        {
                                print(" ");
                                newString = newString + " ";
```

```
                    }
                    else
                    {
                    if (shiftAmount < 0)
                    {
                            newShift = shiftAmount + 26;
                    }
                    var charNum = i.toInt;
                    print((((charNum - 65 + newShift) % 26) + 65).toChar);
                    newString = newString + ((((charNum - 65 + newShift)
                       % 26) + 65).toChar);
                    }
            }//for
            println("");
            return newString;
        }//encrypt

        def decrypt (myStr: String, shiftAmount: Int)
        {
            var inverseShift: Int = shiftAmount * -1;
            this.encrypt(myStr, inverseShift);
        }//decrypt

        def solve (myStr: String, maxShiftValue: Int)
        {
            var tempStr: String = "";
            var j: Int = 0;
            for (j <- 0 to maxShiftValue)
            {
                    tempStr = myStr;
                    print ("Caesar " + j + ": ");
                    this.encrypt(tempStr, j);

            }
        }//solve
}//Main
```

3.4 Ruby

```
def encrypt (myStr, shiftAmount)
        i = 0;
        charNums = myStr.bytes;
        newStr = ""
        for i in 0...charNums.length do
                if charNums[i] == 32
```

```
                    print (" ");
                    newStr = newStr + " "
                else
                    print ((((charNums[i] - 65 + shiftAmount) % 26)
                        + 65).chr);
                    newStr = newStr + ((((charNums[i] - 65 + shiftAmount)
                        % 26) + 65).chr)
                end
        end
        puts (" ")
        return newStr
end

def decrypt (myStr, shiftAmount)
        encrypt myStr, -shiftAmount
end

def solve (myStr, maxShiftValue)
        j = 0
        for j in 0...maxShiftValue + 1
                tempStr = myStr
                print "Caesar "
                print j
                print ": "
                encrypt tempStr, j
        end
end


str = "za bc"
shift = 1
str = str.upcase
puts str
encryptedStr = encrypt str, shift
decrypt encryptedStr, shift
solve "HAL", 26
```

4 Output

4.1 Fortran

```
!shift of 1
 ZA BC
 AB CD
 ZA BC
 Caesar              0  :HAL
 Caesar              1  :IBM
 Caesar              2  :JCN
 Caesar              3  :KDO
 Caesar              4  :LEP
 Caesar              5  :MFQ
 Caesar              6  :NGR
 Caesar              7  :OHS
 Caesar              8  :PIT
 Caesar              9  :QJU
 Caesar             10  :RKV
 Caesar             11  :SLW
 Caesar             12  :TMX
 Caesar             13  :UNY
 Caesar             14  :VOZ
 Caesar             15  :WPA
 Caesar             16  :XQB
 Caesar             17  :YRC
 Caesar             18  :ZSD
 Caesar             19  :ATE
 Caesar             20  :BUF
 Caesar             21  :CVG
 Caesar             22  :DWH
 Caesar             23  :EXI
 Caesar             24  :FYJ
 Caesar             25  :GZK
 Caesar             26  :HAL

!shift of 8
 THIS IS A TEST STRING FROM ALAN
 BPQA QA I BMAB ABZQVO NZWU ITIV
 THIS IS A TEST STRING FROM ALAN
 Caesar              0  :GZZGIQ
 Caesar              1  :HAAHJR
 Caesar              2  :IBBIKS
 Caesar              3  :JCCJLT
 Caesar              4  :KDDKMU
 Caesar              5  :LEELNV
 Caesar              6  :MFFMOW
 Caesar              7  :NGGNPX
 Caesar              8  :OHHOQY
 Caesar              9  :PIIPRZ
 Caesar             10  :QJJQSA
 Caesar             11  :RKKRTB
 Caesar             12  :SLLSUC
 Caesar             13  :TMMTVD
 Caesar          14 :UNNUWE
 Caesar          15 :VOOVXF
 Caesar          16 :WPPWYG
 Caesar          17 :XQQXZH
 Caesar          18 :YRRYAI
 Caesar          19 :ZSSZBJ
 Caesar          20 :ATTACK
 Caesar          21 :BUUBDL
 Caesar          22 :CVVCEM
 Caesar          23 :DWWDFN
 Caesar          24 :EXXEGO
 Caesar          25 :FYYFHP
 Caesar          26 :GZZGIQ
```

4.2 Pascal

```
//shift of 1
ZA BC
AB CD
ZA BC
 Caesar 0: HAL
 Caesar 1: IBM
 Caesar 2: JCN
 Caesar 3: KDO
 Caesar 4: LEP
 Caesar 5: MFQ
 Caesar 6: NGR
 Caesar 7: OHS
 Caesar 8: PIT
 Caesar 9: QJU
 Caesar 10: RKV
 Caesar 11: SLW
 Caesar 12: TMX
 Caesar 13: UNY
 Caesar 14: VOZ
 Caesar 15: WPA
 Caesar 16: XQB
 Caesar 17: YRC
 Caesar 18: ZSD
 Caesar 19: ATE
 Caesar 20: BUF
 Caesar 21: CVG
 Caesar 22: DWH
 Caesar 23: EXI
 Caesar 24: FYJ
 Caesar 25: GZK
 Caesar 26: HAL

 //shift of 10
 THIS IS A TEST STRING FROM JAKE VISSICCHIO
DRSC SC K DOCD CDBSXQ PBYW TKUO FSCCSMMRSY
 THIS IS A TEST STRING FROM JAKE VISSICCHIO
 Caesar 0: IFMMP
 Caesar 1: JGNNQ
 Caesar 2: KHOOR
 Caesar 3: LIPPS
 Caesar 4: MJQQT
 Caesar 5: NKRRU
 Caesar 6: OLSSV
 Caesar 7: PMTTW
 Caesar 8: QNUUX
 Caesar 9: ROVVY
 Caesar 10: SPWWZ
 Caesar 11: TQXXA
 Caesar 12: URYYB
 Caesar 13: VSZZC
 Caesar 14: WTAAD
 Caesar 15: XUBBE
 Caesar 16: YVCCF
 Caesar 17: ZWDDG
 Caesar 18: AXEEH
 Caesar 19: BYFFI
 Caesar 20: CZGGJ
 Caesar 21: DAHHK
 Caesar 22: EBIIL
 Caesar 23: FCJJM
 Caesar 24: GDKKN
 Caesar 25: HELLO
 Caesar 26: IFMMP
```

4.3 Scala

```
//shift of 1
ZA BC
AB CD
ZA BC
Caesar 0: HAL
Caesar 1: IBM
Caesar 2: JCN
Caesar 3: KDO
Caesar 4: LEP
Caesar 5: MFQ
Caesar 6: NGR
Caesar 7: OHS
Caesar 8: PIT
Caesar 9: QJU
Caesar 10: RKV
Caesar 11: SLW
Caesar 12: TMX
Caesar 13: UNY
Caesar 14: VOZ
Caesar 15: WPA
Caesar 16: XQB
Caesar 17: YRC
Caesar 18: ZSD
Caesar 19: ATE
Caesar 20: BUF
Caesar 21: CVG
Caesar 22: DWH
Caesar 23: EXI
Caesar 24: FYJ
Caesar 25: GZK
Caesar 26: HAL

//shift of 15
ALAN IS THE BEST
PAPC XH IWT QTHI
ALAN IS THE BEST
Caesar 0: KBLF
Caesar 1: LCMG
Caesar 2: MDNH
Caesar 3: NEOI
Caesar 4: OFPJ
Caesar 5: PGQK
Caesar 6: QHRL
Caesar 7: RISM
Caesar 8: SJTN
Caesar 9: TKUO
Caesar 10: ULVP
Caesar 11: VMWQ
Caesar 12: WNXR
Caesar 13: XOYS
Caesar 14: YPZT
Caesar 15: ZQAU
Caesar 16: ARBV
Caesar 17: BSCW
```

```
 Caesar 18: CTDX
 Caesar 19: DUEY
 Caesar 20: EVFZ
 Caesar 21: FWGA
 Caesar 22: GXHB
 Caesar 23: HYIC
 Caesar 24: IZJD
 Caesar 25: JAKE
 Caesar 26: KBLF
```

4.4 Ruby

```
#shift of 1
ZA BC
AB CD
ZA BC
 Caesar 0: HAL
 Caesar 1: IBM
 Caesar 2: JCN
 Caesar 3: KDO
 Caesar 4: LEP
 Caesar 5: MFQ
 Caesar 6: NGR
 Caesar 7: OHS
 Caesar 8: PIT
 Caesar 9: QJU
 Caesar 10: RKV
 Caesar 11: SLW
 Caesar 12: TMX
 Caesar 13: UNY
 Caesar 14: VOZ
 Caesar 15: WPA
 Caesar 16: XQB
 Caesar 17: YRC
 Caesar 18: ZSD
 Caesar 19: ATE
 Caesar 20: BUF
 Caesar 21: CVG
 Caesar 22: DWH
 Caesar 23: EXI
 Caesar 24: FYJ
 Caesar 25: GZK
 Caesar 26: HAL

#shift of 20
```

```
HOW MUCH WOOD CAN A WOODCHUCK CHUCK IF A WOODCHUCK COULD CHUCK WOOD
BIQ GOWB QIIX WUH U QIIXWBOWE WBOWE CZ U QIIXWBOWE WIOFX WBOWE QIIX
HOW MUCH WOOD CAN A WOODCHUCK CHUCK IF A WOODCHUCK COULD CHUCK WOOD
Caesar 0: TUBST
Caesar 1: UVCTU
Caesar 2: VWDUV
Caesar 3: WXEVW
Caesar 4: XYFWX
Caesar 5: YZGXY
Caesar 6: ZAHYZ
Caesar 7: ABIZA
Caesar 8: BCJAB
Caesar 9: CDKBC
Caesar 10: DELCD
Caesar 11: EFMDE
Caesar 12: FGNEF
Caesar 13: GHOFG
Caesar 14: HIPGH
Caesar 15: IJQHI
Caesar 16: JKRIJ
Caesar 17: KLSJK
Caesar 18: LMTKL
Caesar 19: MNULM
Caesar 20: NOVMN
Caesar 21: OPWNO
Caesar 22: PQXOP
Caesar 23: QRYPQ
Caesar 24: RSZQR
Caesar 25: STARS
Caesar 26: TUBST
```

5 Official Ranking of Jake Vissicchio

1. Ruby
- Very beginner friendly and extremely writable. I will admit the readability does have some issues.

2. Scala
- As a big fan of Java, it's easy to see why I would like Scala as well. It being Java based allowed me to feel at home with slight modifications.

3. Pascal
- I really enjoy the neatness of the code since it is very readable.

4. Fortran

- I have a soft spot for Fortran since it's the language I used to figure out the caesar cipher but a lot of it is kind of weird. One annoying thing was that it caused errors when I indented which hurt the readability a ton.

6 Resources

- Here are a few resources that I used for this project.

https://ideone.com/

https://fortran-lang.org/learn/quickstart

https://www.pascal-programming.info/lesson1.php

https://www.tutorialspoint.com/scala/index.htm

https://www.geeksforgeeks.org/scala-programming-language/

https://www.geeksforgeeks.org/ruby-tutorial/