Programming In The Past

Daniel Gisolfi

Fortran

Code

```
program Cipher
  implicit none
  ! define a defualt value for the key
  integer :: offset = 10

  ! Input string to be encrypted
  character(3) :: str = 'HAL'
  character(3) :: og = ''
  character(3) :: encrypted = ''
  og = str

  write(*, '(2a)') 'Original string  = '//og

  ! Call the encrypt and concat using `//` <= why would they choose that!
  call encrypt(str, offset)
  write(*, '(2a)') 'Encrypt: '//og//'' => '//str
  ! Store the encrypted value
  encrypted = str
  ! Repeat the proccess for decryption
  call decrypt(str, offset)
  write(*, '(2a)') 'Decrypted: '//encrypted//'' => '//str

  ! encrypt for [0,26]
  call solve(str, og)


contains

subroutine encrypt(text, shift)
  character(*), intent(inout) :: text
  integer :: shift
  integer :: i

  ! do loop, iterate through the string
```

```
  do i = 1, len(text)
    select case(text(i:i))
      case ('A':'Z')
        ! use ascii code and add the shift....thank you stack overflow
        text(i:i) = achar(modulo(iachar(text(i:i)) - 65 + shift, 26) + 65)
      case ('a':'z')
        ! do the same for lowercase leters
        text(i:i) = achar(modulo(iachar(text(i:i)) - 97 + shift, 26) + 97)
    end select
  end do
end subroutine

subroutine decrypt(text, shift)
  character(*), intent(inout) :: text
  integer :: shift
  integer :: i

  do i = 1, len(text)
    select case(text(i:i))
      case ('A':'Z')
        ! subtract the same shift interval
        text(i:i) = achar(modulo(iachar(text(i:i)) - 65 - shift, 26) + 65)
      case ('a':'z')
        ! do the same for the lil letters
        text(i:i) = achar(modulo(iachar(text(i:i)) - 97 - shift, 26) + 97)
    end select
  end do
end subroutine

subroutine solve(text, og)
  character(*), intent(inout) :: text
  character(*), intent(inout) :: og
  character(4) :: key
  integer :: i

  ! The easy part.....call encrypt 27 times
  do i = 0, 26
    call encrypt(text, i)
    ! convert int to string.....I miss .lower()
    write(key, '(I2)') i
    ! trim the key because it gets weird otherwise
    write(*, '(2a)') 'Key:'//trim(key)//' => '//text
  end do
end subroutine

end program Cipher
```

Output

Case 1

```
Original string  = HAL
Encrypt: HAL => RKV
Decrypted: RKV => HAL
Key: 0 => HAL
Key: 1 => IBM
Key: 2 => JCN
Key: 3 => KDO
Key: 4 => LEP
Key: 5 => MFQ
Key: 6 => NGR
Key: 7 => OHS
Key: 8 => PIT
Key: 9 => QJU
Key:10 => RKV
Key:11 => SLW
Key:12 => TMX
Key:13 => UNY
Key:14 => VOZ
Key:15 => WPA
Key:16 => XQB
Key:17 => YRC
Key:18 => ZSD
Key:19 => ATE
Key:20 => BUF
Key:21 => CVG
Key:22 => DWH
Key:23 => EXI
Key:24 => FYJ
Key:25 => GZK
Key:26 => HAL
```

Case 2

```
Original string  = Daniel Craig is the best James Bond
Encrypt: Daniel Craig is the best James Bond => Nkxsov Mbksq sc dro locd Tkwoc Lyxn
Decrypted: Nkxsov Mbksq sc dro locd Tkwoc Lyxn => Daniel Craig is the best James Bond
Key: 0 => Daniel Craig is the best James Bond
Key: 1 => Ebojfm Dsbjh jt uif cftu Kbnft Cpoe
Key: 2 => Fcpkgn Etcki ku vjg dguv Lcogu Dqpf
```

```
Key: 3 => Gdqlho Fudlj lv wkh ehvw Mdphv Erqg
Key: 4 => Hermip Gvemk mw xli fiwx Neqiw Fsrh
Key: 5 => Ifsnjq Hwfnl nx ymj gjxy Ofrjx Gtsi
Key: 6 => Jgtokr Ixgom oy znk hkyz Pgsky Hutj
Key: 7 => Khupls Jyhpn pz aol ilza Qhtlz Ivuk
Key: 8 => Livqmt Kziqo qa bpm jmab Riuma Jwvl
Key: 9 => Mjwrnu Lajrp rb cqn knbc Sjvnb Kxwm
Key:10 => Nkxsov Mbksq sc dro locd Tkwoc Lyxn
Key:11 => Olytpw Ncltr td esp mpde Ulxpd Mzyo
Key:12 => Pmzuqx Odmus ue ftq nqef Vmyqe Nazp
Key:13 => Qnavry Penvt vf gur orfg Wnzrf Obaq
Key:14 => Robwsz Qfowu wg hvs psgh Xoasg Pcbr
Key:15 => Spcxta Rgpxv xh iwt qthi Ypbth Qdcs
Key:16 => Tqdyub Shqyw yi jxu ruij Zqcui Redt
Key:17 => Urezvc Tirzx zj kyv svjk Ardvj Sfeu
Key:18 => Vsfawd Ujsay ak lzw twkl Bsewk Tgfv
Key:19 => Wtgbxe Vktbz bl max uxlm Ctfxl Uhgw
Key:20 => Xuhcyf Wluca cm nby vymn Dugym Vihx
Key:21 => Yvidzg Xmvdb dn ocz wzno Evhzn Wjiy
Key:22 => Zwjeah Ynwec eo pda xaop Fwiao Xkjz
Key:23 => Axkfbi Zoxfd fp qeb ybpq Gxjbp Ylka
Key:24 => Bylgcj Apyge gq rfc zcqr Hykcq Zmlb
Key:25 => Czmhdk Bqzhf hr sgd adrs Izldr Anmc
Key:26 => Daniel Craig is the best James Bond
```

Log

Date	Hours	Tasks / Accomplishments / Issues / Thoughts
2018-09-20	2	Wrote a Encrypt and Decrypt Subroutine...The logic and syntax of encrypting is simmilar to pascal. The syntax is awful compared to pascal. I miss Pascal :(
2018-09-26	3	Once alan enlightened me on the wonders of page two of the assignment this became easier....Calling the encrypt function with a for loop isn't too hard. But who the hell chose // as the concatenation character.

COBOL

Code

```
IDENTIFICATION DIVISION.
```

```
PROGRAM-ID. Cipher.
ENVIRONMENT DIVISION.
DATA DIVISION.

*> Define all Global Variables
WORKING-STORAGE SECTION.
       01 str PIC x(3).
       01 og PIC x(3).
       01 encrypted PIC x(3).
       01 offset PIC 99.
       01 Counter PIC 99.
       01 i PIC 9(3).
       01 c PIC x(1).

*> Have no idea what im diving but the manuel says to do it
PROCEDURE DIVISION.
*> Subprogram (Basically my MAIN)
Begin.
       *> Assign the original values and default key
       SET str TO "HAL";
       SET og TO str;
       SET offset TO 3;

       *> For comparison
       DISPLAY FUNCTION CONCATENATE("Original ------> " str)
       *> Run Encrypt on the defualt
       PERFORM Encrypt.
       SET encrypted TO str;
       *> Using the encrypted version, decrypt
       PERFORM Decrypt.

       *> To solve run through the subprogram 26 times
       SET Counter TO 0;
       SET offset TO 0;
       DISPLAY "Solve:"
       PERFORM Solve UNTIL Counter = 26.

       STOP RUN.

Encrypt.
       MOVE Function Upper-case(og) to og
       *> If the offset is 26, cycle back to 0
       IF offset >= 26
           MOVE FUNCTION MOD(offset, 26) to offset
       END-IF
```

```
       *> This is basically a For loop I had to read the manual to find
this....
       PERFORM VARYING i FROM 1 BY 1 UNTIL i > FUNCTION LENGTH(og)
        *>    Get rid of extra spaces
           IF og(i:1) IS NOT EQUAL TO SPACE
            *>  For each char in OG(original) Add the offset
               MOVE og (i:1) to c
               IF (FUNCTION ORD(c) + offset) <= FUNCTION ORD("Z")
                   MOVE FUNCTION CHAR(FUNCTION ORD(c) + offset) to str
(i:1)
               ELSE
                   MOVE FUNCTION CHAR(FUNCTION ORD("A")
                   + ((FUNCTION ORD(c) + offset) - 1) - FUNCTION ORD("Z"))
to str (i:1)
               END-IF
            END-IF
        END-PERFORM
         *> Show the User the diff
        DISPLAY FUNCTION CONCATENATE("Encrypted " og " -> " str " with
Key:" offset).

Decrypt.
       MOVE Function Upper-case(str) to str

       IF offset >= 26
           MOVE FUNCTION MOD(offset, 26) to offset
       END-IF

       PERFORM VARYING i FROM 1 BY 1 UNTIL i > FUNCTION LENGTH(str)
          IF str(i:1) IS NOT EQUAL TO SPACE
            MOVE str (i:1) to c
            IF (FUNCTION ORD(c) - offset) >= FUNCTION ORD("A")
                MOVE FUNCTION CHAR(FUNCTION ORD(c) - offset) to str (i:1)
            ELSE
                MOVE FUNCTION CHAR(FUNCTION ORD("Z")
                - ((offset - 1) - (FUNCTION ORD(c) - FUNCTION ORD("A"))))
to str (i:1)
            END-IF
          END-IF
       END-PERFORM
       DISPLAY FUNCTION CONCATENATE("Decrypted " encrypted " -> " str "
with Key:" offset).

Solve.
       ADD 1 TO Counter;
       ADD 1 TO offset;
```

```
        PERFORM Encrypt.
        STOP RUN.
```

Output

Case 1

```
Original ------> HAL
Encrypted HAL -> KDO with Key:03
Decrypted KDO -> HAL with Key:03
Solve:
Encrypted HAL -> IBM with Key:01
Encrypted HAL -> JCN with Key:02
Encrypted HAL -> KDO with Key:03
Encrypted HAL -> LEP with Key:04
Encrypted HAL -> MFQ with Key:05
Encrypted HAL -> NGR with Key:06
Encrypted HAL -> OHS with Key:07
Encrypted HAL -> PIT with Key:08
Encrypted HAL -> QJU with Key:09
Encrypted HAL -> RKV with Key:10
Encrypted HAL -> SLW with Key:11
Encrypted HAL -> TMX with Key:12
Encrypted HAL -> UNY with Key:13
Encrypted HAL -> VOZ with Key:14
Encrypted HAL -> WPA with Key:15
Encrypted HAL -> XQB with Key:16
Encrypted HAL -> YRC with Key:17
Encrypted HAL -> ZSD with Key:18
Encrypted HAL -> ATE with Key:19
Encrypted HAL -> BUF with Key:20
Encrypted HAL -> CVG with Key:21
Encrypted HAL -> DWH with Key:22
Encrypted HAL -> EXI with Key:23
Encrypted HAL -> FYJ with Key:24
Encrypted HAL -> GZK with Key:25
Encrypted HAL -> HAL with Key:00
```

Case 2

```
Original ------> Lissome
Encrypted LISSOME -> OLVVRPH with Key:03
Decrypted OLVVRPH -> LISSOME with Key:03
Solve:
Encrypted LISSOME -> MJTTPNF with Key:01
```

```
Encrypted LISSOME -> NKUUQOG with Key:02
Encrypted LISSOME -> OLVVRPH with Key:03
Encrypted LISSOME -> PMWWSQI with Key:04
Encrypted LISSOME -> QNXXTRJ with Key:05
Encrypted LISSOME -> ROYYUSK with Key:06
Encrypted LISSOME -> SPZZVTL with Key:07
Encrypted LISSOME -> TQAAWUM with Key:08
Encrypted LISSOME -> URBBXVN with Key:09
Encrypted LISSOME -> VSCCYWO with Key:10
Encrypted LISSOME -> WTDDZXP with Key:11
Encrypted LISSOME -> XUEEAYQ with Key:12
Encrypted LISSOME -> YVFFBZR with Key:13
Encrypted LISSOME -> ZWGGCAS with Key:14
Encrypted LISSOME -> AXHHDBT with Key:15
Encrypted LISSOME -> BYIIECU with Key:16
Encrypted LISSOME -> CZJJFDV with Key:17
Encrypted LISSOME -> DAKKGEW with Key:18
Encrypted LISSOME -> EBLLHFX with Key:19
Encrypted LISSOME -> FCMMIGY with Key:20
Encrypted LISSOME -> GDNNJHZ with Key:21
Encrypted LISSOME -> HEOOKIA with Key:22
Encrypted LISSOME -> IFPPLJB with Key:23
Encrypted LISSOME -> JGQQMKC with Key:24
Encrypted LISSOME -> KHRRNLD with Key:25
Encrypted LISSOME -> LISSOME with Key:00
```

Log

Date	Hours	Tasks / Accomplishments / Issues / Thoughts
2018-10-02	3	Wow, this is awful. I was able to figure out Encrypt with the MOD FUNCTION thanks to old IBM Forums and Decrypt basically works now too.
2018-10-03	3	Using the limited and not very helpful Documentation I was able to create each into Sub-Programs(Don't know what I was doing instead it was all a mess). After 30 mins of attempting to run the Solve subprogram and getting caught in infinite loops, I am Victorious! I will never touch this language if I can manage. I've heard about projects to use python to replace COBOL in Data Science...Thank God.

BASIC

Code

```
Sub Encrypt(text As String, key As Integer)
  Dim temp As Integer
  For i As Integer = 0 To Len(text)
  ' get one char of the string
    Select Case As Const text[i]
    ' Check if char is upper case
      Case 65 To 90
        ' Shift and store in c as a placeholder
        temp = text[i] + key
        If temp > 90 Then temp -= 26
          text[i] = temp
      ' Check if char is lower case
      Case 97 To 122
        temp = text[i] + key
        If temp > 122 Then temp -= 26
          ' Assign the value back to its index in the string
          text[i] = temp
    End Select
  Next
End Sub

Sub Decrypt(text As String, key As Integer)
  Dim temp As Integer
  For i As Integer = 0 To Len(text)
  ' get one char of the string
  Select Case As Const text[i]
  Case 65 To 90
    ' Shift and store in c as a placeholder
    temp = text[i] - key
    If temp < 65 Then temp += 26
      text[i] = temp
  ' Check if char is lower case
  Case 97 To 122
    temp = text[i] - key
    If temp < 97 Then temp += 26
      ' Assign the value back to its index in the string
      text[i] = temp
  End Select
  Next
End Sub

Sub Solve(og As String, text As String)
  for i As Integer = 0 to 25
```

```
    ' Run the encrypt function 26 times and print each time
    text = og
    Encrypt text, i
    Print "Encrypted:" + str(i) + " " + og + " -> " + text
  Next
End Sub

Dim As String text = "HAL"
Dim As String og = text
Dim As Integer offset = 6
Dim As String Encrypted

Print "Original -> "; text
Encrypt text, offset
Print "Encrypted:" + str(offset) + " " + og + " -> " + text
Encrypted = text
Decrypt text, offset
Print "Decrypted:" + str(offset) + " " + Encrypted + " -> " + text
Print "SOLVE:"
Solve og, text
Sleeps
```

Output

Case 1

```
Original -> HAL
Encrypted:6 HAL -> NGR
Decrypted:6 NGR -> HAL
SOLVE:
Encrypted:0 HAL -> HAL
Encrypted:1 HAL -> IBM
Encrypted:2 HAL -> JCN
Encrypted:3 HAL -> KDO
Encrypted:4 HAL -> LEP
Encrypted:5 HAL -> MFQ
Encrypted:6 HAL -> NGR
Encrypted:7 HAL -> OHS
Encrypted:8 HAL -> PIT
Encrypted:9 HAL -> QJU
Encrypted:10 HAL -> RKV
Encrypted:11 HAL -> SLW
Encrypted:12 HAL -> TMX
Encrypted:13 HAL -> UNY
Encrypted:14 HAL -> VOZ
```

```
Encrypted:15 HAL -> WPA
Encrypted:16 HAL -> XQB
Encrypted:17 HAL -> YRC
Encrypted:18 HAL -> ZSD
Encrypted:19 HAL -> ATE
Encrypted:20 HAL -> BUF
Encrypted:21 HAL -> CVG
Encrypted:22 HAL -> DWH
Encrypted:23 HAL -> EXI
Encrypted:24 HAL -> FYJ
Encrypted:25 HAL -> GZK
```

Case 2

```
Original -> Python is fun and powerful
Encrypted:6 Python is fun and powerful  -> Veznut oy lat gtj vuckxlar
Decrypted:6 Veznut oy lat gtj vuckxlar  -> Python is fun and powerful
SOLVE:
Encrypted:0 Python is fun and powerful  -> Python is fun and powerful
Encrypted:1 Python is fun and powerful  -> Qzuipo jt gvo boe qpxfsgvm
Encrypted:2 Python is fun and powerful  -> Ravjqp ku hwp cpf rqygthwn
Encrypted:3 Python is fun and powerful  -> Sbwkrq lv ixq dqg srzhuixo
Encrypted:4 Python is fun and powerful  -> Tcxlsr mw jyr erh tsaivjyp
Encrypted:5 Python is fun and powerful  -> Udymts nx kzs fsi utbjwkzq
Encrypted:6 Python is fun and powerful  -> Veznut oy lat gtj vuckxlar
Encrypted:7 Python is fun and powerful  -> Wfaovu pz mbu huk wvdlymbs
Encrypted:8 Python is fun and powerful  -> Xgbpwv qa ncv ivl xwemznct
Encrypted:9 Python is fun and powerful  -> Yhcqxw rb odw jwm yxfnaodu
Encrypted:10 Python is fun and powerful  -> Zidryx sc pex kxn zygobpev
Encrypted:11 Python is fun and powerful  -> Ajeszy td qfy lyo azhpcqfw
Encrypted:12 Python is fun and powerful  -> Bkftaz ue rgz mzp baiqdrgx
Encrypted:13 Python is fun and powerful  -> Clguba vf sha naq cbjreshy
Encrypted:14 Python is fun and powerful  -> Dmhvcb wg tib obr dcksftiz
Encrypted:15 Python is fun and powerful  -> Eniwdc xh ujc pcs edltguja
Encrypted:16 Python is fun and powerful  -> Fojxed yi vkd qdt femuhvkb
Encrypted:17 Python is fun and powerful  -> Gpkyfe zj wle reu gfnviwlc
Encrypted:18 Python is fun and powerful  -> Hqlzgf ak xmf sfv hgowjxmd
Encrypted:19 Python is fun and powerful  -> Irmahg bl yng tgw ihpxkyne
Encrypted:20 Python is fun and powerful  -> Jsnbih cm zoh uhx jiqylzof
Encrypted:21 Python is fun and powerful  -> Ktocji dn api viy kjrzmapg
Encrypted:22 Python is fun and powerful  -> Lupdkj eo bqj wjz lksanbqh
Encrypted:23 Python is fun and powerful  -> Mvqelk fp crk xka mltbocri
Encrypted:24 Python is fun and powerful  -> Nwrfml gq dsl ylb nmucpdsj
Encrypted:25 Python is fun and powerful  -> Oxsgnm hr etm zmc onvdqetk
```

Log

Date	Hours	Tasks / Accomplishments / Issues / Thoughts
2018-09-03	3	Was able to create a basic main sub as well as a help screen and begun making a dedicated sub for encrypting, decrypting and solving. I used to be very familiar with this syntax as Visual basic was my first ever programming language. Sadly this was 4 years ago and I only find the word "dim" used to define familiar. Why is dim the keyword....it stands for dimension?. Was able to find the ascii integer of a character and print the char based on the ascii
2018-10-03	2	Coming back to Basic was so nice after the hell that is COBOL. Why did i put this easy one off?

Pascal

Code

```
// Daniel Nicolas Gisolfi
// 2018-09-03
program Cipher;
uses Crt;

// Define procedures
procedure intro;
begin
    writeln('**Welcome to the Pascal Ceaser Cipher**');
end;

procedure help;
begin
    ClrScr;
    writeln('Pascal Ceaser Cipher Help');
    writeln('The following are all possible commands: ');
    writeln('> help');
    writeln('> encrypt');
    writeln('> decrypt');
    writeln('> solve[break]');
    writeln('> exit');
end;

```

```
function encrypt(var str: string; var offset: Integer): string;
var
  i: Integer;
begin
    // each character is incremented 6 characters within ASCII
    for i := 1 to Length(str) do
        // check if value of current char is any lower or uppercase char
        case str[i] of
            'A'..'Z': str[i] := chr(ord('A') + (ord(str[i]) - ord('A') + offset) mod 26);
            'a'..'z': str[i] := chr(ord('a') + (ord(str[i]) - ord('a') + offset) mod 26);
        end;
        encrypt := str;
end;

function decrypt(var str: string; var offset: Integer): string;
var
  i: Integer;
begin
    // each character is unincremented 6 characters within ASCII
    for i := 1 to Length(str) do
        // check if value of current char is any lower or uppercase char
        case str[i] of
            'A'..'Z': str[i] := chr(ord('A') + (ord(str[i]) - ord('A') - offset + 26) mod 26);
            'a'..'z': str[i] := chr(ord('a') + (ord(str[i]) - ord('a') - offset + 26) mod 26);
        end;
    decrypt := str;
end;

procedure main;
var
action: string;
offset: Integer;
str: string;
og: string;
i: Integer;
maxShift: Integer;

begin
    // defualt value for key
    offset := 10;
    repeat
        writeln('Run "help" for a list of possible commands');
```

```
66          write('What would you like to do => ');
67          readln(action);
68          if (action = 'help') then
69              begin
70                  help;
71                  writeln('Press Enter to Continue');
72                  readln;
73                  ClrScr;
74              end;
75          if (action = 'encrypt') then
76              begin
77                  // request input to be encrypted
78                  write('Please input a string to be encrypted => ');
79                  readln(str);
80                  og := str;
81                  // request the desired key for encryption
82                  write('Please input the key to be used => ');
83                  readln(offset);
84
85                  // show original vs result
86                  writeln(og + ' -> ' + encrypt(str, offset));
87                  writeln('Press Enter to Continue');
88                  readln;
89              end;
90          if (action = 'decrypt') then
91              begin
92                  // request input to be encrypted
93                  write('Please input a string to be decrypted => ');
94                  readln(str);
95                  og := str;
96                  // request the desired key for decryption
97                  write('Please input the key to be used => ');
98                  readln(offset);
99
100                 // show original vs result
101                 writeln(og + ' -> ' + decrypt(str, offset));
102                 writeln('Press Enter to Continue');
103                 readln;
104             end;
105
106         if (action = 'solve') then
107             begin
108                 // request input to be encrypted
109                 write('Please input a string to be solved for => ');
110                 readln(str);
111                 og := str;
```

```
                // request the desired key for decryption
                write('Please input the Max Shift Value => ');
                readln(offset);
                maxShift:= offset;


                for i := 0 to maxShift do
                    begin
                        offset := i;
                        str := og;
                        writeln('Caesar ', offset, ' -> ' + encrypt(str, offset));
                    end;

                writeln('Press Enter to Continue');
                readln;
            end;

    until (action = 'exit');
end;

begin
    ClrScr;
    intro;
    main;
end.
```

Output

Case 1

```
Original -> HAL
Encrypted:10 HAL -> RKV
Decrypted:10 RKV -> HAL
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
```

Case 2

```
Original -> Coffee is yummy
Encrypted:10 Coffee is yummy -> Iullkk oy easse
Caesar 0 -> Coffee is yummy
Caesar 1 -> Dpggff jt zvnnz
Caesar 2 -> Eqhhgg ku awooa
Caesar 3 -> Friihh lv bxppb
Caesar 4 -> Gsjjii mw cyqqc
Caesar 5 -> Htkkjj nx dzrrd
Caesar 6 -> Iullkk oy easse
Caesar 7 -> Jvmmll pz fbttf
Caesar 8 -> Kwnnmm qa gcuug
Caesar 9 -> Lxoonn rb hdvvh
Caesar 10 -> Myppoo sc iewwi
Caesar 11 -> Nzqqpp td jfxxj
Caesar 12 -> Oarrqq ue kgyyk
Caesar 13 -> Pbssrr vf lhzzl
Caesar 14 -> Qcttss wg miaam
Caesar 15 -> Rduutt xh njbbn
Caesar 16 -> Sevvuu yi okcco
Caesar 17 -> Tfwwvv zj plddp
Caesar 18 -> Ugxxww ak qmeeq
Caesar 19 -> Vhyyxx bl rnffr
Caesar 20 -> Wizzyy cm soggs
Caesar 21 -> Xjaazz dn tphht
Caesar 22 -> Ykbbaa eo uqiiu
Caesar 23 -> Zlccbb fp vrjjv
Caesar 24 -> Amddcc gq wskkw
Caesar 25 -> Bneedd hr xtllx
```

Log

Date	Hours	Tasks / Accomplishments / Issues / Thoughts
2018-09-03	5	Was able to define a few procedures and being creating my main loop for the program. Not a fan of using a weird character such as ":" to define variables. After a few hours of writing a longer and longer file, the heavy structured syntax is becoming helpful. Recurssion is not always the best.....infinite loops can become issues.
2018-09-05	2	Finally have something ressembling a cipher. Shifting returns capital letters for some reason. May have to do with ASCII code and me simply iterating it. Decrypting doesnt decypt but encypts even further(for now I will consider this a feature.
2018-09-20	1	Rewrote all Procedures as functions in order to get a return value. After looking at the second page of the assigment I was happy to see my program was already cappable of solving. Pascal is complete.
2018-09-20	na	Final thoughts: Pascal is not as tedious as java. There are a few oddities that slowed me down however I would consider writing a command line program in Pascal if time was not a factor.

Scala

Code

```scala
// Cipher Program
object Cipher {
    def main(args: Array[String]): Unit = {
        val text = "HAL"
        // backup the text var
        val og = text
        // show we got the corrct input
        println("Original  => " + text)

        // encode the text and show results
        val encoded = Cipher.encode(text, 10)
        println("Encrypt:   " + og +  " => " + encoded)

        //d ecode the text and show results
```

```
        val decoded = Cipher.decode(encoded, 10)
        println("Decrypted: " + og +  " => " + decoded)

        // pass thw text to the solve program
        Cipher.solve(text)
    }
    // Upper case alphabet
    private val alphaU='A' to 'Z'
    // Lower case alphabet
    private val alphaL='a' to 'z'

    // Encode
    def encode(text:String, key:Int)=text.map{
        // This is way cooler than how I originally did it...once again thank you stack overflow
        // shift uppers
        case str if alphaU.contains(str) => rot(alphaU, str, key)
        // shift normal case
        case str if alphaL.contains(str) => rot(alphaL, str, key)
        // assign the new val
        case str => str
    }
    // Im not this clever, googling is an art
    def decode(text:String, key:Int)=encode(text,-key)
    private def rot(a:IndexedSeq[Char], c:Char, key:Int)=a((c-a.head+key+a.size)%a.size)

    // Just call the encode function 27 times, 0-26 and print it nicley
    def solve(text:String){
        for(i <- 0 to 26){
            val encoded = Cipher.encode(text, i)
            println("Key " + i + " => " + encoded )
        }
    }
}
```

Output

Case 1

```
Original => HAL
Encrypt: HAL => RKV
Decrypted: RKV => HAL
Key 0 => HAL
Key 1 => IBM
```

```
Key 2 => JCN
Key 3 => KDO
Key 4 => LEP
Key 5 => MFQ
Key 6 => NGR
Key 7 => OHS
Key 8 => PIT
Key 9 => QJU
Key 10 => RKV
Key 11 => SLW
Key 12 => TMX
Key 13 => UNY
Key 14 => VOZ
Key 15 => WPA
Key 16 => XQB
Key 17 => YRC
Key 18 => ZSD
Key 19 => ATE
Key 20 => BUF
Key 21 => CVG
Key 22 => DWH
Key 23 => EXI
Key 24 => FYJ
Key 25 => GZK
```

Case 2

```
Original  => This code is made for running and thats just what itll do
Encrypt:   This code is made for running and thats just what itll do =>
Drsc myno sc wkno pyb bexxsxq kxn drkdc tecd grkd sdvv ny
Decrypted: This code is made for running and thats just what itll do =>
This code is made for running and thats just what itll do
Key 0 => This code is made for running and thats just what itll do
Key 1 => Uijt dpef jt nbef gps svoojoh boe uibut kvtu xibu jumm ep
Key 2 => Vjku eqfg ku ocfg hqt twppkpi cpf vjcvu lwuv yjcv kvnn fq
Key 3 => Wklv frgh lv pdgh iru uxqqlqj dqg wkdwv mxvw zkdw lwoo gr
Key 4 => Xlmw gshi mw qehi jsv vyrrmrk erh xlexw nywx alex mxpp hs
Key 5 => Ymnx htij nx rfij ktw wzssnsl fsi ymfyx ozxy bmfy nyqq it
Key 6 => Znoy iujk oy sgjk lux xattotm gtj zngzy payz cngz ozrr ju
Key 7 => Aopz jvkl pz thkl mvy ybuupun huk aohaz qbza doha pass kv
Key 8 => Bpqa kwlm qa uilm nwz zcvvqvo ivl bpiba rcab epib qbtt lw
Key 9 => Cqrb lxmn rb vjmn oxa adwwrwp jwm cqjcb sdbc fqjc rcuu mx
Key 10 => Drsc myno sc wkno pyb bexxsxq kxn drkdc tecd grkd sdvv ny
Key 11 => Estd nzop td xlop qzc cfyytyr lyo esled ufde hsle teww oz
Key 12 => Ftue oapq ue ympq rad dgzzuzs mzp ftmfe vgef itmf ufxx pa
Key 13 => Guvf pbqr vf znqr sbe ehaavat naq gungf whfg jung vgyy qb
```

```
Key 14 => Hvwg qcrs wg aors tcf fibbwbu obr hvohg xigh kvoh whzz rc
Key 15 => Iwxh rdst xh bpst udg gjccxcv pcs iwpih yjhi lwpi xiaa sd
Key 16 => Jxyi setu yi cqtu veh hkddydw qdt jxqji zkij mxqj yjbb te
Key 17 => Kyzj tfuv zj druv wfi ileezex reu kyrkj aljk nyrk zkcc uf
Key 18 => Lzak ugvw ak esvw xgj jmffafy sfv lzslk bmkl ozsl aldd vg
Key 19 => Mabl vhwx bl ftwx yhk knggbgz tgw matml cnlm patm bmee wh
Key 20 => Nbcm wixy cm guxy zil lohhcha uhx nbunm domn qbun cnff xi
Key 21 => Ocdn xjyz dn hvyz ajm mpiidib viy ocvon epno rcvo dogg yj
Key 22 => Pdeo ykza eo iwza bkn nqjjejc wjz pdwpo fqop sdwp ephh zk
Key 23 => Qefp zlab fp jxab clo orkkfkd xka qexqp grpq texq fqii al
Key 24 => Rfgq ambc gq kybc dmp psllgle ylb rfyrq hsqr ufyr grjj bm
Key 25 => Sghr bncd hr lzcd enq qtmmhmf zmc sgzsr itrs vgzs hskk cn
```

Log

Date	Hours	Tasks / Accomplishments / Issues / Thoughts
2018-09-23	2	Scala is to EZ. I love the `=>` syntax for passing a parameter by name, clean and resembles and arrow. The proccess of finding the correct function calls for shifting the text were tricker this time, required extensive googling.