Library of routines supporting the input, storage and manipulation of dates.

General Information

Author: Paul B. Loux
Language: ACTION!
Compiler/Interpreter: ACTION!
Published: 1986

Requirements: EntryD() utilizes "EntryS()" (universal string entry routine), "PrintM()" (output formatter), and the "ValD()" function provided herein.

EntryS() is available under the name ENTRYS.ACT PrintM() is available under the name PRINTM.ACT

Four routines are provided to facilitate the storage and manipulation of dates. The CARD FUNC ValD(<string>) will convert a date in string format to a unique CARD value. The CARD returned by this function can be used to compute the number of calendar days between two dates. The string can have non-numeric characters; for instance "12/31/85" is legal. Used together with its converse, PROC StrD(CARD number,<string>), it is also possible to find the calendar date which falls a given number of days before or after a reference date. The string returned by StrD() contains only numbers; formatting must be performed separately.

PROC Day(CARD number,<string>) provides the day of the week corresponding to a given calender date, as represented by a CARD value generated by ValD().

CARD FUNC EntryD() obtains a date from the keyboard. It uses EntryS(), the universal string entry utility; therefore it has the associated features of error checking, timeout, etc. EntryD() will assure the validity of the entered date, check it against optional minimum and maximum dates, and echo succesful entry in mm-dd-yy format, by use of PrintM(). The calling program provides the entry buffer, so EntryD() can be used to return a CARD value (as with ValD()) or to obtain an unformatted string (as with StrD()).

PROC PrintM(<String>,<mask>) and its variants *ME,*MD,and *MDE can be used to print a date in any format desired, such as "mm-dd-yy".

To facilitate usage into the next century, the date computations include a 40-year offset. Thus, the date "043020" is presumed to mean April 30, 2020. Therefore, date computations are only valid for dates within the range from 1-1-1940 through 12-31-2039. ValD() and StrD() are consistent in this regard.

Note that more efficient storage results from use of CARD values (2 bytes) rather than strings (5 or 6 bytes plus length byte). This technique also facilitates ease in sorting data by date.

Technical note: in general, any string variable should be pre- extended to its maximum length prior to making a call which will use it to pass data.

;******************************************************
;* *(C)Copyright 1986 by Paul B. Loux *
;* These routines are in the public *
;* domain, and are not to be sold *
;* for a profit. They may be freely *
;* distributed, provided that this *
;******************************************************
File: DATES.LIB

Description: Library of routines supporting the input, storage and manipulation of dates.

Requirements: EntryD() utilizes "EntryS()" (universal string entry routine), "PrintM()" (output formatter), and the "ValD()" function provided herein.

EntryS() is available under the name ENTRYS.ACT

PrintM() is available under the name PRINTM.ACT

Four routines are provided to facilitate the storage and manipulation of dates. The CARD FUNC ValD(<string>) will convert a date in string format to a unique CARD value. The CARD returned by this function can be used to compute the number of calendar days between two dates. The string can have non-numeric characters; for instance "12/31/85" is legal. Used together with its converse, PROC StrD(CARD number,<string>), it is also possible to find the calendar date which falls a given number of days before or after a reference date. The string returned by StrD() contains only numbers; formatting must be performed separately.

PROC Day(CARD number,<string>) provides the day of the week corresponding to a given calendar date, as represented by a CARD
CARD FUNC EntryD() obtains a date from the keyboard. It uses EntryS(), the universal string entry utility; therefore it has the associated features of error checking, timeout, etc. EntryD() will assure the validity of the entered date, check it against optional minimum and maximum dates, and echo successful entry in mm-dd-yy format, by use of PrintM(). The calling program provides the entry buffer, so EntryD() can be used to return a CARD value (as with ValD()) or to obtain an unformatted string (as with StrD()).

PROC PrintM(<String>,<mask>) and its variants *ME,*MD,and *MDE can be used to print a date in any format desired, such as "mm-dd-yy".

To facilitate usage into the next century, the date computations include a 40-year offset. Thus, the date "043020" is presumed to mean April 30, 2020. Therefore, date computations are only valid for dates within the range from 1-1-1940 through 12-31-2039. ValD() and StrD() are consistent in this regard.

Note that more efficient storage results from use of CARD values (2 bytes) rather than strings (5 or 6 bytes plus length byte). This technique also facilitates ease in sorting data by date.

Technical note: in general, any string variable should be pre-extended to its maximum length prior to making a call which will use it to pass data.

************************************
"ValD()"

Convert a date string into a unique CARD value. Input expected:
CARD FUNC ValD(BYTE ARRAY dateS)

BYTE ARRAY digits(0)="......"
BYTE ARRAY month(0)="..",
        day(0)="..",
        year(0)=".."

BYTE mm,dd,yy
BYTE dmax,yy,bad_date
BYTE len1
BYTE len2
BYTE ctr,tmp
BYTE xtmp,ztmp

CARD value
INT offset

len1=dateS(0)
len2=6

DO ; assure only digits
    tmp=dateS(len1)
    IF (tmp>47 AND tmp <58) THEN
        digits(len2)=tmp
        len2==-1
    FI
    len1==-1
UNTIL len1=0 OR len2=0
OD

IF len2>1 THEN ; 4 or less #'s
    RETURN(0)
FI

IF len2=1 THEN ; 5 #'s
    digits(1)=48 ; '0
FI
digits(0)=6
SCopyS(month,digits,1,2)
SCopyS(day,digits,3,4)
SCopyS(year,digits,5,6)

mm=ValB(month)
dd=ValB(day)
yy=ValB(year)
bad_date=0

IF mm>12 OR ; legal date
mm<1 OR ; checks
dd<1 THEN
    bad_date=1
FI

IF mm=2 THEN
    IF yy MOD 4 THEN
        dmax=28
    ELSE dmax=29
    FI
ELSEIF
    mm=4 OR
    mm=6 OR
    mm=9 OR
    mm=11 THEN dmax=30
ELSE dmax=31
FI

IF dd>dmax THEN
    bad_date=1
FI

IF bad_date THEN
    RETURN(0)
FI

IF yy<40 THEN ; 40 year offset
    yy=yy+100
FI

IF mm<3 THEN
    xtmp=0
    ztmp=(yy-1)/4
ELSE
    xtmp=(4*mm + 23)/10
    ztmp=yy/4
FI

mm=-1

value=365*yy+31*mm+dd+ztmp-xtmp

RETURN(value)

;*****************************************************************************
;
;   "StrD()"
;
;   Restores a date compressed
to a CARD value by ValD(),
to a fixed length string
of six digital characters;
no formating is performed.
Example output:
;
;   "010185"
;
;   Note: calling program must
; pre-extend string "dateS"
; to six places.
;
PROC StrD(CARD dateC
    BYTE ARRAY dateS)

BYTE ARRAY mm(0)="..",
            dd(0)="..",
            yy(0)=".."

BYTE POINTER ptr1,ptr2
INT m,d,y,r,s,t,y1,ly
BYTE dmax

y=0
y1=0

IF dateC>36524 THEN ; yy=1**
    dateC=-36525
FI

IF dateC>29220 THEN ; # too big
    dateC==-7305
    y1=20
FI

IF dateC<61 THEN ; handle yr=0
    dateC==+1461
    y1=-4
FI

y=dateC/365
r=dateC-(y*365)-y/4

IF r<31 THEN
    y=-1
    r=dateC-(y*365)-y/4
FI

IF r>59 then
    s=7
ELSE s=0
FI

m=(r+s)/31
ly=(y/4)-((y-1)/4)

IF m<3 THEN
    t=ly
ELSE
    t=(4*m+23)/10
FI

IF m=2 THEN
    IF y MOD 4 =0 THEN
        dmax=29
    
ELSE
  dmax=28
FI
ELSEIF m=4
  OR m=6
  OR m=9
  OR m=11 THEN
  dmax=30
ELSE
  dmax=31
FI

d=r-31*(m-1)+t

IF d>dmax THEN
  m==+1
  IF m<3 THEN
    t=ly
  ELSE
    t=(4*m+23)/10
  FI
  d=r-31*(m-1)+t
FI

IF m=13 THEN
  y==+1
  m==-12
FI

y==+y1

StrI(m,mm)
StrI(d/dd)
StrI(y,yy)

SCopy(dateS,"000000")

ptr1=mm+1
ptr2=dateS+1
IF mm(0)=1 THEN
  ptr2==+1
  ptr2 ^=ptr1^ 
ELSE
  ptr2 ^=ptr1^ 
  ptr1==+1
  ptr2==+1
  ptr2 ^=ptr1^ 
FI

ptr1=dd+1
ptr2=dateS+3
IF dd(0)=1 THEN
  ptr2==+1
  ptr2 ^=ptr1^ 
ELSE
  ptr2 ^=ptr1^ 
  ptr1==+1
  ptr2==+1
  ptr2 ^=ptr1^ 

FI

ptr1=yy+1
ptr2=dateS+5
IF yy(0)=1 THEN
  ptr2==+1
  ptr2^=ptr1^ 
ELSE 
  ptr2^=ptr1^ 
  ptr1==+1
  ptr2==+1
  ptr2^=ptr1^ 
FI

RETURN

;*****************************************************
;    "Day()"
;    Day of the week computation
;    Returns variable-length string
;    containing corresponding day
;    of the week for the CARD value
;    supplied. String can be easily
;    massaged to obtain upper case
;    only, first three letters,etc.
;    Note: string "day" must be
;    pre-xtended to 9 places by the
;    the calling program, to allow
;    room for "Wednesday" response.
;
PROC Day(CARD dateC BYTE ARRAY day)

CARD ref=[31412] ; Wednesday 1/1/86
INT dif
BYTE num,dir
BYTE ARRAY ptr
CARD ARRAY dow(7)

dow(0)="Wednesday"
dow(1)="Thursday"
dow(2)="Friday"
dow(3)="Saturday"
dow(4)="Sunday"
dow(5)="Monday"
dow(6)="Tuesday"
dow(7)="Wednesday"

dir=0
dif=dateC-ref
IF dif<0 THEN
  dif=-dif
  dir=1
FI
IF dir THEN
    num=7-num
FI
ptr=dow(num)
SCopy(day,ptr)
RETURN

;***************************************************************
;
; CARD FUNC EntryD()
;
; Data entry utility used to gather a calendar date from the keyboard in the "mmddyy" format. The routine performs checks for illegal dates and echoes a valid response in "mm-dd-yy" format. Returns date as a CARD value as per ValD(), or as an unformatted string as per StrD().
;
; This function uses both the EntryS() data entry utility and the PrintM() formatter.
;
; Calling options include the screen coordinates; high and low checks; null-entry flag; and exit flag, per EntryS().
;
;***************************************************************

INCLUDE "ENTRYS.ACT"
INCLUDE "PRINTM.ACT"

***************************************************************

MODULE

CARD FUNC EntryD(BYTE ARRAY field
BYTE col,row,nullok,xit
CARD min_date,max_date
BYTE POINTER err_ptr)

BYTE bad_date,accept,ctr,
    min,max,typec

CARD value

BYTE POINTER ptr1,ptr2
INT chk

min=5
IF nulllok THEN
    min=0
FI

IF max_date=0 THEN
    max_date=51134 ; 12-31-39
FI

max=6
typec=5 ; pos int
accept=0
chk=0

DO
    POSITION(row,col)
    PRINT("            ")
    ENTRIES(field,min,max,typec,xit,col,row,err_ptr)

    IF err_ptr^#0 THEN RETURN(0) FI

    bad_date=0

    IF field(0)=0 THEN
        IF nulllok=1
            THEN RETURN(0)
        ELSE bad_date=1
        FI
    FI

    value=ValD(field)

    IF value=0 THEN bad_date=1
    ELSEIF value<min_date
        OR value>max_date
        THEN bad_date=2
    FI

    IF
        bad_date=1 THEN
            MSG(8)
        ELSEIF
            bad_date=2 THEN
            MSG(7)
        ELSE accept=1
    FI

    UNTIL accept
OD

POSITION(col,row)
PRINTM(field,"<Z/<Z/ZZ")

RETURN(value)

;************************************
PROC Test5()

BYTE ARRAY date_field="......."
BYTE ARRAY dow="........" 

BYTE x,y
CARD date_val,min_date,max_date

BYTE errcde
BYTE POINTER err_ptr

errcde=0
err_ptr=@errcde
min_date=0
max_date=0

PUT(125)
POSITION(1,5)
x=22 y=5

PRINT("Enter date (mmddyy): ")

date_val=EntryD(date_field,x,y,0,0,
min_date,max_date,
err_ptr)

PUTE()
PRINT("The CARD value representing ")
PRINT("this date is ")
PRINTC(date_val)
PRINT(".")

PUTE()
PRINT("StrD() gives us back the string")
StrD(date_val,date_field)
PRINT("representation, ")
PRINT(date_field)
PRINT(".")

PUTE()
PRINT("Adding 31 days to this date")
PRINT("gives us a CARD value of ")
date_val==+31
PRINTC(date_val)
PRINT(".")

PUTE()
PRINT("The date corresponding to this")
PRINT("CARD value is ")
StrD(date_val,date_field)
PRINT(date_field)
PRINT(".")

PUTE()
PRINT("The ValD() of this date gives")
PRINT("back the CARD value, ")
date_val=ValD(date_field)
PRINTC(date_val)
PRINT(".")
PUTE()
PRINT("The day of the week for these")
PRINT("two days is as follows:")
PUTE()
PRINT(date_field)
PRINT(" ")
Day(date_val,dow)
PRINT(dow)
date_val=-31
StrD(date_val,date_field)
PRINT(date_field)
PRINT(" ")
Day(date_val,dow)
PRINT(dow)
PUTE()
PRINT("Done...")
RETURN