

Date

COLLABORATORS

	TITLE : Date		
ACTION	NAME	DATE	SIGNATURE
WRITTEN BY		July 22, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	Date	1
1.1	Date.doc	1
1.2	Date/--background--	2
1.3	Date/--history--	4
1.4	Date/GregorianDayDiff	6
1.5	Date/GregorianDayGreater	7
1.6	Date/GregorianDaysAfterWeekday	8
1.7	Date/GregorianDaysBeforeWeekday	9
1.8	Date/GregorianDaySmaller	10
1.9	Date/GregorianDiffDate	10
1.10	Date/GregorianEaster	11
1.11	Date/GregorianLeapYear	12
1.12	Date/GregorianMonthDays	13
1.13	Date/GregorianMoonAge	13
1.14	Date/GregorianToJD	14
1.15	Date/GregorianWeek	15
1.16	Date/GregorianWeekday	16
1.17	Date/GregorianYearDays	16
1.18	Date/GSYearToJD	17
1.19	Date/GYearToScaliger	18
1.20	Date/HeisDayDiff	18
1.21	Date/HeisDayGreater	19
1.22	Date/HeisDaysAfterWeekday	20
1.23	Date/HeisDaysBeforeWeekday	21
1.24	Date/HeisDaySmaller	22
1.25	Date/HeisDiffDate	22
1.26	Date/HeisLeapYear	23
1.27	Date/HeisMonthDays	24
1.28	Date/HeisToJD	25
1.29	Date/HeisWeek	25

1.30	Date/HeisWeekday	26
1.31	Date/HeisYearDays	27
1.32	Date/HSYearToJD	28
1.33	Date/HYearToScaliger	28
1.34	Date/JDtoMJD	29
1.35	Date/JDToTime	30
1.36	Date/JSYearToJD	30
1.37	Date/JulianDayDiff	31
1.38	Date/JulianDayGreater	32
1.39	Date/JulianDaysAfterWeekday	32
1.40	Date/JulianDaysBeforeWeekday	33
1.41	Date/JulianDaySmaller	34
1.42	Date/JulianDiffDate	35
1.43	Date/JulianLeapYear	36
1.44	Date/JulianMonthDays	36
1.45	Date/JulianToJD	37
1.46	Date/JulianWeek	38
1.47	Date/JulianWeekday	39
1.48	Date/JulianYearDays	39
1.49	Date/JYearToScaliger	40
1.50	Date/LMT	41
1.51	Date/MJDtoJD	41
1.52	Date/ScaligerYearToG	42
1.53	Date/ScaligerYearToH	43
1.54	Date/ScaligerYearToJ	43
1.55	Date/SecToTime	44
1.56	Date/TimeToJD	44
1.57	Date/GregorianEaster	45
1.58	Date/TimeToSec	46
1.59	Date/TimeZoneFactor	46

Chapter 1

Date

1.1 Date.doc

```
--background--  
--history--()  
GregorianDayDiff()  
GregorianDayGreater()  
GregorianDaysAfterWeekday()  
GregorianDaysBeforeWeekday()  
GregorianDaySmaller()  
GregorianDiffDate()  
GregorianEaster()  
GregorianLeapYear()  
GregorianMonthDays()  
GregorianMoonAge()  
GregorianToJD()  
GregorianWeek()  
GregorianWeekday()  
GregorianYearDays()  
GYearToJD()  
GYearToScaliger()  
HeisDayDiff()  
HeisDayGreater()  
HeisDaysAfterWeekday()  
HeisDaysBeforeWeekday()  
HeisDaySmaller()  
HeisDiffDate()  
HeisLeapYear()  
HeisMonthDays()  
HeisToJD()  
HeisWeek()  
HeisWeekday()  
HeisYearDays()  
HYearToJD()  
HYearToScaliger()  
JDtoMJD()  
JDtoTime()  
JYearToJD()  
JulianDayDiff()  
JulianDayGreater()  
JulianDaysAfterWeekday()
```

```
JulianDaysBeforeWeekday()  
JulianDaySmaller()  
JulianDiffDate()  
JulianLeapYear()  
JulianMonthDays()  
JulianToJD()  
JulianWeek()  
JulianWeekday()  
JulianYearDays()  
JYearToScaliger()  
LMT  
MJDtoJD()  
ScaligerYearToG()  
ScaligerYearToH()  
ScaligerYearToJ()  
SecToTime()  
TimeToJD()  
GregorianEaster()  
TimeToSec()  
TimeZoneFactor()
```

1.2 Date/--background--

NAME

Date -- This module was designed to help calc. calendar dates (V33)

FUNCTION

I now about the date routines in the Amiga-OS(TM), but I decided not to use them, because of their limited functionalities and of the portability of this Module!

NOTES

A tropical year is 365.2422 days! / 365d, 5h, 48min, 46sec
A moon month is 29.53059 days! / 29d, 12h, 44min, 2.9 sec
A moon phase is 7.38265 days!

(German) Books which helped me creating this:

Kleine Naturwissenschaftliche Bibliothek, Band 23
Ewige Kalender
A.W. Butkewitsch & M.S. Selikson
5. Auflage
Teubner, Leipzig 1974
ISBN 3-322-00393-0

Tag und Woche, Monat und Jahr: eine Kulturgeschichte des
Kalenders
Rudolf Wendorff
Westdeutscher, Opladen 1993
ISBN 3-531-12417-X

Kalender und Chronologie: Bekanntes & Unbekanntes aus der
Kalenderwissenschaft
Heinz Zemanek
4. Auflage
Oldenbourg, München 1987

ISBN 3-486-20447-5

Meyers Handbuch
über das Weltall
Karl Schaifers & Gerhard Traving
5. Auflage
Bibliographisches Institut Mannheim 1973
ISBN 3-411-00940-3

(English) Books which helped me creating this:

Mathematical Astronomy with a Pocket Calculator
Aubrey Jones Fras
unknown(first) Edition
David & Charles Newton Abbot, London 1978
ISBN 0-7153-7675-6

COPYRIGHT

This module is Copyright 1994 by Kai Hofmann - all rights reserved!
For private use, Public Domain, Gift Ware, Freeware and Shareware
you could use this module under following conditions:

- You send me a little gift (money is very welcome :)
For Bank Account see below - but *ONLY* send in DM
to this Bank Account!!!
Other nice gifts: all Amiga hardware, and I am searching for a
good old 1541 (C64 floppy)
 - You include a notice in your product, that you use this library
and that it is Copyright by Kai Hofmann!
- If you want to redistribute this library read the following points:
- Redistribution warranty is given to:
Fred Fish for his great Amiga-Software-Library
The German SAAR AG PD-Library
The German AMOK PD-Library
All public accessible INTERNET servers and PHONE boxes!
All other who NOT take more than DM 5.- for one disk
ALL other who NOT take more than DM 50.- for one CD

For commercial use send me DM 200.-

But if you are Apple or Microsoft you have to send (20000.- US\$)

DISCLAIMER

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDER AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE
PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE
COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY REDISTRIBUTE THE
PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS
OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR
THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER
PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE

POSSIBILITY OF SUCH DAMAGES.

ADDITIONAL INFORMATION

I have tried to make portable/usefull and I hope bugfree software for eternity - but this seems to be impossible (sorry!) :)
So I hope you will pay a fee for this.

AUTHOR

Kai Hofmann
Arberger Heerstraße 92
28307 Bremen
Germany
EMail: i07m@alf.zfn.uni-bremen.de
(no phone - I hate it!)

Bank account : 1203 7503
Account owner: Kai Hofmann
Bank code : 290 501 01
Bank name : Sparkasse in Bremen
Bank address : 28307 Bremen / Germany

THANK

Thanx are going to the following people:
Danial Armor - For his hint about the Oberon-2 SHORT
command
Heinz Zemanek - For his great book
Christian Schaefer - For spending time on this lib with his
Borland C++ 4.0 compiler
Rita Reichl - For correcting my bad english ;-)
Jim Rickman - For reporting a bug

1.3 Date/--history--

NAME

history -- This is the development history of the Date module

VERSION

\$VER: Date 33.089 (30.11.1994)

HISTORY

16.01.1994 - Procedures: JulianLeapYear, GregorianLeapYear &
HeisLeapYear initiated.
22.01.1994 - Procedures: JulianMonthDays, GregorianMonthDays,
HeisMonthDays, JulianYearDays, GregorianYearDays,
HeisYearDays, JulianDayDiff, GregorianDayDiff,
HeisDayDiff, JulianDaySmaller, GregorianDaySmaller,
HeisDaySmaller, JulianWeekday, GregorianWeekday,
HeisWeekday, JulianDaysBeforeWeekday,
GregorianDaysBeforeWeekday, HeisDaysBeforeWeekday,
JulianDaysAfterWeekday, GregorianDaysAfterWeekday,
HeisDaysAfterWeekday JulianDiffDate, FreeDate
initiated.
Types: Weekdays, Date, DatePtr initiated.
Vars of Gregorian reform initiated
(for changing to different countries)

- 23.01.1994 - Procedures: JulianDiffDate finished,
GregorianDiffDate, HeisDiffDate, JYearToScaliger,
GYearToScaliger, HYearToScaliger, ScaligerYearToJ,
ScaligerYearToG, ScaligerYearToH, JSYearToJD,
GSYearToJD, HSYearToJD, JDtoMJD, MJDtoJD, JulianToJD,
GregorianToJD, HeisToJD, TimeToJD, JDToTime, FreeTime
initiated.
Types: Time, TimePtr initiated.
- 28.01.1994 - Procedures: GregorianMoonAge, MoonMonthAge,
GregorianEaster initiated.
- 30.01.1994 - Procedures: JulianDiffDate, GregorianDiffDate,
HeisDiffDate, JDtoTime, GregorianEaster edited
(changing return value from ptr to VAL variables).
Procedures: FreeDate, FreeTime deleted.
Types: Date, DatePtr, Time, TimePtr deleted (not
longer needed, because of the procedure changes).
Procedures: GregorianMoonAge, GregorianEaster changed
year parameter from CARDINAL to INTEGER (this is more
consistent to the rest of the library).
Bugs removed: GregorianWeekday, HeisWeekday
(before removing, the weekday for leapyears was
wrong)
Procedure: GregorianEaster finished.
- 30.01.1994 - Ported to Oberon-2
- 31.01.1994 - Compiled with Oberon-2 V3.11
- 12.02.1994 - Procedures: TimeZoneFactor, LMT, TimeToSec, SecToTime
initiated.
Version-String installed :)
- 12.02.1994 - Starting translation to SAS C 6.51
Date.h translated
- 13.02.1994 - Continuation of C translation
- 17.02.1994 - New Oberon-2 Port, because yesterday Daniel Armor
gives me a small hint about the SHORT command
(I did not know about this!)
- 17.02.1994 - Small bug in Autodocs removed
making this text as Date/--history-- autodoc
- 17.02.1994 - Continuation of C translation
- 18.02.1994 - Finished with C translation
- 19.02.1994 - C bugs removed (thanx to SAS for helping a C Lamer
like me!), some optimizations done too.
- 19.02.1994 - Oberon-2 version compiled with V40.17 includes
- 21.02.1994 - Writing Modula-II testmodule
Vars for the begining of Heis calculation initiated.
Fixed small bugs in GregorianWeekday, HeisWeekday,
TimeToSec, SecToTime
Return-value of LMT changed to LONGINT!
Converting testmodule to Oberon-2
- 22.02.1994 - Converting testmodule to C
- 23.02.1994 - I noticed, that I forgot the 3 functions
JulianWeek, GregorianWeek, HeisWeek
- 24.02.1994 - Initiated the 3 forgotten functions
- 26.02.1994 - Initiating new GregorianEastern with Gauß-algorithms
but ONLY for 1900-2099!
- 27.02.1994 - Bug fixed in JulianWeekday
Bugs fixed in JulianDayDiff, GregorianDayDiff,
HeisDayDiff
JulianDayGreater, GregorianDayGreater,
-

HeisDayGreater Initiated.

02.03.1994 - Small bug fixed in HeisdayDiff
Bugs from 27.02. fixed in Modula-II and Oberon-2 versions
I found the way to extend Gregorian Easter!
Small bug fixed in JulianWeek, GregorianWeek, HeisWeek (~(M2) is not !(C))

05.03.1994 - Some internal bugs removed
New internal procedures GregorianSB, GregorianJHSB, GregorianJHStartSB!
Extending GregorianEaster :)

11.03.1994 - Things from 05.03. done in Modula-II and Oberon

12.03.1994 - If __SASC is defined autoinitialization instead of _DateInit will be used!

13.03.1994 - After studying the SAS C Manual again I decided to check for __SASC_650 instead of __SASC because of the available of priorities!
Setting the priority of _DateInit for autoinitialization to 600!

15.03.1994 - Making Date as library

16.03.1994 - Some work on the Autodocs was done
Eliminating OldGregorianEaster by comments
(ANSI: STOP bad standards like that there are NO nestedcomments possible in C!!!)

19.03.1994 - Some work on the Autodocs was done in the M2 Code

20.03.1994 - Some work on the Autodocs was done in the Oberon Code

22.03.1994 - In JDtoMJD, MJD to JD an L was added to the constant
In GregorianWeekday(), HeisWeekday(), JulianDiffDate(), GregorianDiffDate(), HeisDiffDate(), JDToTime() I have inserted conversions (found with Borland C++ 4.0)

24.03.1994 - Making SunOS4.1.3, SunOS5.3(Solaris2.3) & RS6000 AIX3.2.? binaries with gcc
Eliminating nested comments by inserting a space between / and * (I hate this ANSI C standard feature for comments :(

27.03.1994 - Adding library register assignments to the autodocs

03.04.1994 - Small fixes for the SAS C++ Compiler
Small bug fixed in the M2 version of GregorianEaster

04.04.1994 - Adding some 'static' keywords

10.04.1994 - Changing from Shareware to Gift Ware ;-)

02.08.1994 - Small fixes in the Autodocs (thanks to Rita Reichl for correcting my bad english ;-)

11.08.1994 - Again small fixes in the Autodocs!

13.11.1994 - Small fix in JulianWeek(),GregorianWeek(),HeisWeek().
Thanks to Jim Rickman for reporting the bug!
Small changes in the Autodocs!

30.11.1994 - Fix the bug from 13.11. in M2 and Oberon code.

1.4 Date/GregorianDayDiff

NAME

GregorianDayDiff -- Calculates the days between 2 dates. (V33)

SYNOPSIS

```
days := GregorianDayDiff(day1,month1,year1,day2,month2,year2);
```

```
PROCEDURE GregorianDayDiff(day1,month1 : SHORTINT; year1 : INTEGER;
    day2,month2 : SHORTINT; year2 : INTEGER) : LONGINT;
```

FUNCTION

GregorianDayDiff gives you back the number of days between two specified dates.

INPUTS

```
day1    - day of the first date
month1  - month of the first date
year1   - year of the first date
day2    - day of the second date
month2  - month of the second month
year2   - year of the second date
```

RESULT

days - The number of days between the two dates
(positive if date1 <= date2).

EXAMPLE

```
...
days := GregorianDayDiff(18,9,1970,22,1,1994);
WriteString("Age of Kai Hofmann in days : ");
WriteInt(days,10); WriteLn;
...
```

NOTES

It is better only to use this function for years from -7 to 3200!

BUGS

If you use on of the dates 5.10.1582 to 14.10.1582 you will get a wrong output, because this days don't exist!

SEE ALSO

```
JulianDayDiff(), HeisDayDiff(), GregorianDaySmaller(),
GregorianDayGreater(), GregorianMonthDays(), GregorianYearDays()
```

1.5 Date/GregorianDayGreater

NAME

GregorianDayGreater -- Checks if date1 is great than date2. (V33)

SYNOPSIS

```
greater := GregorianDayGreater(day1,month1,year1,day2,month2,year2);
```

```
PROCEDURE GregorianDayGreater(day1,month1 : SHORTINT;
    year1 : INTEGER; day2,month2 : SHORTINT;
    year2 : INTEGER) : BOOLEAN;
```

FUNCTION

GregorianDayGreater test if date1 is greater than date2.

INPUTS

```

day1    - day of the first date
month1  - month of the first date
year1   - year of the first date
day2    - day of the second date
month2  - month of the second month
year2   - year of the second date

```

RESULT

greater - This is TRUE is date1 > date2 otherwise it's FALSE.

EXAMPLE

```

...
IF GregorianCalendar(18,9,1970,22,1,1994) THEN
  WriteString(">"); WriteLn;
ELSE
  WriteString("<="); WriteLn;
END;
...

```

NOTES

It is better only to use this function for years from -7 to 3200!

BUGS

No known bugs.

SEE ALSO

JulianDayGreater(), HeisDayGreater()

1.6 Date/GregorianCalendarAfterWeekday

NAME

GregorianCalendarAfterWeekday -- Returns the diff to wday after. (V33)

SYNOPSIS

```
days := GregorianCalendarAfterWeekday(day,month,year,weekday);
```

```

PROCEDURE GregorianCalendarAfterWeekday(day,month : SHORTINT;
  year : INTEGER; weekday : Weekdays) : SHORTINT;

```

FUNCTION

Returns the days to the weekday after the specified date.
 So if you specify the 22.1.1994 (Saturday) and Thursday
 you get back 5!
 If you specify the 22.1.1994 and Saturday you get back 0
 (the same day)!

INPUTS

```

day      - day of the date
month    - month of the date
year     - year of the date
weekday  - weekday to search for building difference

```

RESULT

days - The days after to the searched weekday.

EXAMPLE

```
...
days := GregorianDaysAfterWeekday(22,1,1994,Thursday);
...
```

NOTES

It is better to use this function only from -7 to 3200!

BUGS

See GregorianWeekday()!

SEE ALSO

JulianDaysAfterWeekday(), HeisDaysAfterWeekday(), GregorianWeekday()

1.7 Date/GregorianDaysBeforeWeekday

NAME

GregorianDaysBeforeWeekday -- Returns the diff to wday before. (V33)

SYNOPSIS

```
days := GregorianDaysBeforeWeekday(day,month,year,weekday);
```

```
PROCEDURE GregorianDaysBeforeWeekday(day,month : SHORTINT;
    year : INTEGER; weekday : Weekdays) : SHORTINT;
```

FUNCTION

Returns the days to the weekday before the specified date.
 So if you specify the 22.1.1994 (Saturday) and Thursday
 you get back 2!
 If you specify the 22.1.1994 and Saturday you get back 0
 (the same day)!

INPUTS

```
day      - day of the date
month    - month of the date
year     - year of the date
weekday  - weekday to search for building difference
```

RESULT

```
days - The days back to the searched weekday (1-7)
      - If you get back 8 an error occurs!
```

EXAMPLE

```
...
days := GregorianDaysBeforeWeekday(22,1,1994,Thursday);
...
```

NOTES

It is better to use this function only from -7 to 3200!

BUGS

See GregorianWeekday()!

SEE ALSO

`JulianDaysBeforeWeekday(), HeisDaysBeforeWekday(), GregorianWeekday()`

1.8 Date/GregorianDaySmaller

NAME

`GregorianDaySmaller` -- Checks if date1 is smaller than date2. (V33)

SYNOPSIS

`smaller := GregorianDaySmaller(day1, month1, year1, day2, month2, year2);`

PROCEDURE `GregorianDaySmaller`(day1, month1 : SHORTINT;
 year1 : INTEGER; day2, month2 : SHORTINT;
 year2 : INTEGER) : BOOLEAN;

FUNCTION

`GregorianDaySmaller` test if date1 is smaller than date2.

INPUTS

day1 - day of the first date
month1 - month of the first date
year1 - year of the first date
day2 - day of the second date
month2 - month of the second month
year2 - year of the second date

RESULT

`smaller` - This is TRUE is date1 < date2 otherwise it's FALSE.

EXAMPLE

```
...
IF GregorianDaySmaller(18, 9, 1970, 22, 1, 1994) THEN
    WriteString("<"); WriteLn;
ELSE
    WriteString(">="); WriteLn;
END;
...
```

NOTES

It is better only to use this function for years from -7 to 3200!

BUGS

No known bugs.

SEE ALSO

`JulianDaySmaller(), HeisDaySmaller()`

1.9 Date/GregorianDiffDate

NAME

`GregorianDiffDate` -- Returns the diff date to another date. (V33)

SYNOPSIS

```
GregorianDiffDate(day,month,year,diffdays,dday,dmonth,dyear);
```

```
PROCEDURE GregorianDiffDate(day,month : SHORTINT;
    year,days : INTEGER; VAR dday,dmonth : SHORTINT;
    VAR dyear : INTEGER);
```

FUNCTION

Returns the date wich lies diffdays before/after the specified date.

INPUTS

day - day of the date
 month - month of the date
 year - year of the date
 diffdays - difference to the date in days

RESULT

dday - Destination day
 dmonth - Destination month
 dyear - Destination year

EXAMPLE

```
...
GregorianDiffDate(23,1,1994,7,dday,dmonth,dyear);
...
```

NOTES

It is better to use this function only from -7 to 3200!

BUGS

unknown.

SEE ALSO

JulianDiffDate(),HeisDiffDate(),GregoriandayDiff(),
 GregorianMonthDays()

1.10 Date/GregorianEaster

NAME

GregorianEaster -- Returns the date of eastern in a year (V33)

SYNOPSIS

```
GregorianEaster(year,dday,dmonth);
```

```
PROCEDURE GregorianEaster(year : INTEGER;
    VAR dday,dmonth : SHORTINT);
```

FUNCTION

Returns the date of eastern for a specified year.

INPUTS

year - eastern is calculated for this year

RESULT

dday - day of easter-Sunday
 dmonth - month of easter-Sunday

```
EXAMPLE
...
GregorianEaster(1994, dday, dmonth);
...

NOTES
Use this only for 1582 to 4100!
This is only a experimental version!

BUGS
In some years eastern lies one week behind!

SEE ALSO
GregorianMoonAge(), GregorianDaysAfterWeekday()
```

1.11 Date/GregorianLeapYear

```
NAME
GregorianLeapYear -- Checks if a year is a leap year. (V33)

SYNOPSIS
leapyear := GregorianLeapYear(year);

PROCEDURE GregorianLeapYear(year : INTEGER) : BOOLEAN;

FUNCTION
GregorianLeapYear checks if a year is a leap year.
For years after 1582 all years devideable by 4 are leap years,
without years devideable by 100, but years devideable by 400
are leap years again!
For years before 1582 see JulianLeapYear().

INPUTS
year - The year which should be checked (from -32768 to 32767)
      I think only values from -7 to 3200 are valid, because of
      the variant that was done on -8 by Augustus!

RESULT
leapyear - TRUE if the year is a leap year, otherwise false.

EXAMPLE
...
IF GregorianLeapYear(1994) THEN
  WriteString("leap year!");
ELSE
  WriteString("no leap year!");
END;
WriteLn;
...

NOTES
A year is 365.2425 days long!
Use this function only for values from -7 to 3200!
```

BUGS
No known bugs.

SEE ALSO
`JulianLeapYear()`, `HeisLeapYear()`

1.12 Date/GregorianCalendarMonthDays

NAME
`GregorianCalendarMonthDays` -- Gives back the number of days of a month. (V33)

SYNOPSIS
`days := GregorianCalendarMonthDays(month, year);`

PROCEDURE `GregorianCalendarMonthDays(month : SHORTINT;
year : INTEGER) : SHORTINT;`

FUNCTION
`GregorianCalendarMonthDays` gives you back the number of days a month in a specified year has.
For the year 1582 and the month 10 there are only 21 days, because of the Gregorian-reform 10 days are delete from the month (for more - look out for books about this!)

INPUTS
`month` - The month from wich you want to get the number of days.
`year` - The year in which the month is.

RESULT
`days` - The number of days the month uses, or 0 if you use a wrong month.

EXAMPLE
...
`days := GregorianCalendarMonthDays(1, 1994);`
`WriteString("Days of January 1994 : ");`
`WriteCard(days, 2); WriteLn;`
...

NOTES
Use this function only for years from -7 to 3200!

BUGS
If the reform in a country is not in the same month an error will occur!

SEE ALSO
`GregorianCalendarLeapYear()`, `JulianMonthDays()`, `HeisMonthDays()`

1.13 Date/GregorianCalendarMoonAge

NAME
GregorianMoonAge -- Returns the age of the moon (V33)

SYNOPSIS
ep := GregorianMoonAge(day,month,year);

PROCEDURE GregorianMoonAge(day,month : SHORTINT;
year : INTEGER) : SHORTINT;

FUNCTION
Returns the age of the moon on a specified date.

INPUTS
day - For this day the age is calculated.
month - For this month the age is calculated.
year - For this year the age is calculated.

RESULT
ep - The age of the moon on the specified date.

EXAMPLE
...
ep := GregorianMoonAge(18,9,1994);
...

NOTES
Use this only for 1582 to 4100!
This is only a experimental version!

BUGS
unknown.

SEE ALSO
MoonMonthAge(),GregorianEP()

1.14 Date/GregorianToJD

NAME
GregorianToJD -- Returns the JD for a date. (V33)

SYNOPSIS
jd := GregorianToJD(day,month,year);

PROCEDURE GregorianToJD(day,month : SHORTINT;
year : INTEGER) : LONGINT;

FUNCTION
Returns the JD for a Gregorian date.

INPUTS
day - day of the date to convert
month - month of the date to convert
year - year of the date to convert

```

    RESULT
jd - This is the JD

    EXAMPLE
...
jd := GregorianToJD(23,1,1994);
...

    NOTES
It is better to use this function only from -7 to 3200!

    BUGS
unknown.

    SEE ALSO
JulianToJD(), HeisToJD(), GYearToJD(), GYearToScaliger(),
GregorianDayDiff()

```

1.15 Date/GregorianWeek

```

    NAME
GregorianWeek -- Gets the weeknumber of a specified date. (V33)

    SYNOPSIS
weeknr := GregorianWeek(day,month,year);

    PROCEDURE GregorianWeek(day,month : SHORTINT;
    year : INTEGER) : SHORTINT;

    FUNCTION
GregorianWeek gets the weeknumber for a specified date.

    INPUTS
day    - day of the date
month  - month of the date
year   - year of the date

    RESULT
week - This is the number of the week the specified date lies in.
      If the first day in a new year is a Friday, Saturday or
      Sunday, this would be the last week of the last year!
      If the 29.12. is a Monday, the 30.12. is a Monday or a Tuesday,
      the 31.12. is a Monday, Tuesday or a Wednesday this is the
      first week of the next year!

    EXAMPLE
...
weeknr := GregorianWeek(4,10,1582);
...

    NOTES
It is better only to use this function for years from 0 to 3000!

    BUGS
For years < 0 errors could occur.

```

SEE ALSO
 JulianWeek(), HeisWeek(), GregorianWeekday(), GregorianDayDiff(),
 GregorianDaySmaller()

1.16 Date/GregorianWeekday

NAME
 GregorianWeekday -- Gets the weekday of a specified date. (V33)

SYNOPSIS
 weekday := GregorianWeekday(day, month, year);

PROCEDURE GregorianWeekday(day, month : SHORTINT;
 year : INTEGER) : Weekday;

FUNCTION
 GregorianWeekday gets the weekday for a specified date.

INPUTS
 day - day of the date
 month - month of the date
 year - year of the date

RESULT
 weekday - This result is of type:
 Weekdays = (dayerr, Monday, Tuesday, Wednesday, Thursday, Friday,
 Saturday, Sunday);
 dayerr will show you, that an error occurs!

EXAMPLE
 ...
 weekday := GregorianWeekday(22, 1, 1994);
 IF weekday = dayerr THEN
 ...
 END;
 ...

NOTES
 It is better only to use this function for years from -7 to 3200!
 In this version dayerr will only occur for the lost days :)

BUGS
 It's not possible to use years < 0 (for more see JulianWeekday()).

SEE ALSO
 JulianWeekday(), HeisWeekday(), GregorianDaySmaller(),
 GregorianLeapYear()

1.17 Date/GregorianYearDays

NAME

GregorianYearDays -- Gives back the number of days in a year. (V33)

SYNOPSIS

```
days := GregorianYearDays(year);
```

```
PROCEDURE GregorianYearDays(year : INTEGER) : INTEGER;
```

FUNCTION

GregorianYearDays gives you back the number of days in a specified year.

INPUTS

year - The year in which to count the days.
(I think its better not to use years before -7!)

RESULT

days - The number of days the year uses.

EXAMPLE

```
...
days := GregorianYearDays(1994);
WriteString("Days of 1994 : ");
WriteCard(days,3); WriteLn;
...
```

NOTES

It is better only to use this function for years from -7 to 3200!

BUGS

No known bugs.

SEE ALSO

GregorianMonthDays(), JulianYearDays(), HeisYearDays()

1.18 Date/GSYearToJD

NAME

GSYearToJD -- Calcs the JD from a Scaliger year. (V33)

SYNOPSIS

```
jd := GSYearToJD(syear);
```

```
PROCEDURE GSYearToJD(syear : INTEGER) : LONGINT;
```

FUNCTION

Returns the Julianday of a Scaliger year.

INPUTS

syear - Scaliger year

RESULT

jd - The Julianday

```
EXAMPLE
...
jd := GYearToJD(4800);
...

NOTES
It is better to use this function only from 4707 to 7981!

BUGS
unknown.

SEE ALSO
JYearToJD(), HYearToJD()
```

1.19 Date/GYearToScaliger

```
NAME
GYearToScaliger -- Returns the year as Scaliger year. (V33)

SYNOPSIS
syear := GYearToScaliger(year);

PROCEDURE GYearToScaliger(year : INTEGER) : INTEGER;

FUNCTION
Returns the Scaliger year.

INPUTS
year      - Gregorian year

RESULT
syear - The Scaliger year

EXAMPLE
...
syear := GYearToScaliger(1994);
...

NOTES
It is better to use this function only from -7 to 3200!

BUGS
unknown.

SEE ALSO
JYearToScaliger(), HYearToScaliger()
```

1.20 Date/HeisDayDiff

```
NAME
HeisDayDiff -- Calculates the days between 2 dates. (V33)
```

SYNOPSIS

```
days := HeisDayDiff(day1,month1,year1,day2,month2,year2);
```

```
PROCEDURE HeisDayDiff(day1,month1 : SHORTINT; year1 : INTEGER;
    day2,month2 : SHORTINT; year2 : INTEGER) : LONGINT;
```

FUNCTION

HeisDayDiff gives you back the number of days between two specified dates.

INPUTS

```
day1    - day of the first date
month1  - month of the first date
year1   - year of the first date
day2    - day of the second date
month2  - month of the second month
year2   - year of the second date
```

RESULT

days - The number of days between the two dates
(positive if date1 <= date2).

EXAMPLE

```
...
days := HeisDayDiff(18,9,1970,22,1,1994);
WriteString("Age of Kai Hofmann in days : ");
WriteInt(days,10); WriteLn;
...
```

NOTES

It is better only to use this function for years from -7 to 8000!

BUGS

If you use on of the dates 5.10.1582 to 14.10.1582 you will get a wrong output, because this days don't exist!

SEE ALSO

JulianDayDiff(),GregorianDayDiff(),HeisDaySmaller(),HeisDayGreater(),
HeisMonthDays(),HeisYearDays()

1.21 Date/HeisDayGreater

NAME

HeisDayGreater -- Checks if date1 is greater than date2. (V33)

SYNOPSIS

```
greater := HeisDayGreater(day1,month1,year1,day2,month2,year2);
```

```
PROCEDURE HeisDayGreater(day1,month1 : SHORTINT; year1 : INTEGER;
    day2,month2 : SHORTINT; year2 : INTEGER) : BOOLEAN;
```

FUNCTION

HeisDayGreater test if date1 is great than date2.

INPUTS

```

day1    - day of the first date
month1  - month of the first date
year1   - year of the first date
day2    - day of the second date
month2  - month of the second month
year2   - year of the second date

```

RESULT

greater - This is TRUE is date1 > date2 otherwise it's FALSE.

EXAMPLE

```

...
IF HeisDaySmaller(18,9,1970,22,1,1994) THEN
    WriteString(">"); WriteLn;
ELSE
    WriteString("<="); WriteLn;
END;
...

```

NOTES

It is better only to use this function for years from -7 to 8000!

BUGS

No known bugs.

SEE ALSO

JulianDayGreater(),GregorianDayGreater()

1.22 Date/HeisDaysAfterWeekday

NAME

HeisDaysAfterWeekday -- Returns the diff to the wday after. (V33)

SYNOPSIS

```
days := HeisDaysAfterWeekday(day,month,year,weekday);
```

```

PROCEDURE HeisDaysAfterWeekday(day,month : SHORTINT;
    year : INTEGER; weekday : Weekdays) : SHORTINT;

```

FUNCTION

Returns the days to the weekday after the specified date.
 So if you specify the 22.1.1994 (Saturday) and Thursday
 you get back 5!
 If you specify the 22.1.1994 and Saturday you get back 0
 (the same day)!

INPUTS

```

day      - day of the date
month    - month of the date
year     - year of the date
weekday  - weekday to search for building difference

```

RESULT

days - The days after to the searched weekday.

EXAMPLE

```
...
days := HeisDaysAfterWeekday(22,1,1994,Thursday);
...
```

NOTES

It is better to use this function only from -7 to 8000!

BUGS

See HeisWeekday()!

SEE ALSO

JulianDaysAfterWeekday(), GregorianDaysAfterWeekday(), HeisWeekday()

1.23 Date/HeisDaysBeforeWeekday

NAME

HeisDaysBeforeWeekday -- Returns the diff to wday before. (V33)

SYNOPSIS

```
days := HeisDaysBeforeWeekday(day,month,year,weekday);
```

```
PROCEDURE HeisDaysBeforeWeekday(day,month : SHORTINT;
    year : INTEGER; weekday : Weekdays) : SHORTINT;
```

FUNCTION

Returns the days to the weekday before the specified date.
 So if you specify the 22.1.1994 (Saturday) and Thursday
 you get back 2!
 If you specify the 22.1.1994 and Saturday you get back 0
 (the same day)!

INPUTS

```
day      - day of the date
month    - month of the date
year     - year of the date
weekday  - weekday to search for building difference
```

RESULT

```
days - The days back to the searched weekday (1-7)
      - If you get back 8 an error occurs!
```

EXAMPLE

```
...
days := HeisDaysBeforeWeekday(22,1,1994,Thursday);
...
```

NOTES

It is better to use this function only from -7 to 8000!

BUGS

See HeisWeekday()!

SEE ALSO

JulianDaysBeforeWeekday(),GregorianDaysBeforeWeekday(),HeisWeekday()

1.24 Date/HeisDaySmaller

NAME

HeisDaySmaller -- Checks if date1 is smaller than date2. (V33)

SYNOPSIS

smaller := HeisDaySmaller(day1,month1,year1,day2,month2,year2);

PROCEDURE HeisDaySmaller(day1,month1 : SHORTINT; year1 : INTEGER;
day2,month2 : SHORTINT; year2 : INTEGER) : BOOLEAN;

FUNCTION

HeisDaySmaller test if date1 is smaller than date2.

INPUTS

day1 - day of the first date
month1 - month of the first date
year1 - year of the first date
day2 - day of the second date
month2 - month of the second month
year2 - year of the second date

RESULT

smaller - This is TRUE is date1 < date2 otherwise it's FALSE.

EXAMPLE

```
...
IF HeisDaySmaller(18,9,1970,22,1,1994) THEN
  WriteString("<"); WriteLn;
ELSE
  WriteString(">="); WriteLn;
END;
...
```

NOTES

It is better only to use this function for years from -7 to 8000!

BUGS

No known bugs.

SEE ALSO

JulianDaySmaller(),GregorianDaySmaller()

1.25 Date/HeisDiffDate

NAME

HeisDiffDate -- Returns the date for a diff to another date. (V33)

SYNOPSIS

HeisDiffDate(day,month,year,diffdays,dmonth,dyear);

```

PROCEDURE HeisDiffDate(day,month : SHORTINT; year,days : INTEGER;
    VAR dday,dmonth : SHORTINT; VAR dyear : INTEGER);

FUNCTION
Returns the date wich lies diffdays before/after the specified date.

INPUTS
day      - day of the date
month    - month of the date
year     - year of the date
diffdays - difference to the date in days

RESULT
dday     - Destination day
dmonth   - Destination month
dyear    - Destination year

EXAMPLE
...
HeisDiffDate(23,1,1994,7,dday,dmonth,dyear);
...

NOTES
It is better to use this function only from -7 to 8000!

BUGS
unknown.

SEE ALSO
JuliandiffDate(),GregorianDiffdate(),HeisDayDiff(),HeisMonthDays()

```

1.26 Date/HeisLeapYear

```

NAME
HeisLeapYear -- Checks if a year is a leap year. (V33)

SYNOPSIS
leapyear := HeisLeapYear(year);

PROCEDURE HeisLeapYear(year : INTEGER) : BOOLEAN;

FUNCTION
HeisLeapYear checks if a year is a leap year.
For years after 1582 see GregorianLeapYear(),
The correction from N. Heis says, that all years devideable by
3200 are no longer leap years!
For years before 1582 see JulianLeapYear

INPUTS
year - The year which should be checked (from -32768 to 32767)
      I think only values from -7 to 8000 are valid, because of
      the variant that was done on -8 by Augustus!

RESULT

```

leapyear - TRUE if the year is a leap year, otherwise false.

EXAMPLE

```
...
IF HeisLeapYear(1994) THEN
  WriteString("leap year!");
ELSE
  WriteString("no leap year!");
END;
WriteLn;
...
```

NOTES

A year is now 365.2421875 days!
Use this function only for values from -7 to 8000!

BUGS

No known bugs.

SEE ALSO

JulianLeapYear(), GregorianLeapYear()

1.27 Date/HeisMonthDays

NAME

HeisMonthDays -- Gives back the number of days of a month. (V33)

SYNOPSIS

```
days := HeisMonthDays(month,year);
```

```
PROCEDURE HeisMonthDays(month : SHORTINT;
  year : INTEGER) : SHORTINT;
```

FUNCTION

HeisMonthDays gives you back the number of days a month in a specified year has.

For the year 1582 and the month 10 there are only 21 days, because of the Gregorian-reform 10 days are delete from the month (for more - look out for books about this!)

INPUTS

month - The month from wich you want to get the number of days.
year - The year in which the month is.

RESULT

days - The number of days the month uses, or 0 if you use a wrong month.

EXAMPLE

```
...
days := HeisMonthDays(1,1994);
WriteString("Days of January 1994 : ");
WriteCard(days,2); WriteLn;
...
```

NOTES

Use this function only for years from -7 to 8000!

BUGS

See `GregorianMonthDays()`!

SEE ALSO

`HeisLeapYear()`, `JulianMonthDays()`, `GregorianMonthDays()`

1.28 Date/HeisToJD

NAME

`HeisToJD` -- Returns the JD for a date. (V33)

SYNOPSIS

```
jd := HeisToJD(day,month,year);
```

```
PROCEDURE HeisToJD(day,month : SHORTINT;  
                  year : INTEGER) : LONGINT;
```

FUNCTION

Returns the JD for a Heis date.

INPUTS

```
day      - day of the date to convert  
month    - month of the date to convert  
year     - year of the date to convert
```

RESULT

jd - This is the JD

EXAMPLE

```
...  
jd := HeisToJD(23,1,1994);  
...
```

NOTES

It is better to use this function only from -7 to 3268!

BUGS

unknown.

SEE ALSO

`JulianToJD()`, `GregorianToJD()`, `HSYearToJD()`, `HYearToScaliger()`,
`HeisDayDiff()`

1.29 Date/HeisWeek

NAME

`HeisWeek` -- Gets the weeknumber of a specified date. (V33)

SYNOPSIS

```
weeknr := HeisWeek(day,month,year);
```

```
PROCEDURE HeisWeek(day,month : SHORTINT;
  year : INTEGER) : SHORTINT;
```

FUNCTION
HeisWeek gets the weeknumber for a specified date.

INPUTS
day - day of the date
month - month of the date
year - year of the date

RESULT
week - This is the number of the week the specified date lies in.
If the first day in a new year is a Friday, Saturday or Sunday, this would be the last week of the last year!
If the 29.12. is a Monday, the 30.12. is a Monday or a Tuesday, the 31.12. is a Monday, Tuesday or a Wednesday this is the first week of the next year!

EXAMPLE
...
weeknr := HeisWeek(4,10,1582);
...

NOTES
It is better only to use this function for years from 0 to 8000!

BUGS
For years < 0 errors could occur.

SEE ALSO
JulianWeek(),GregorianWeek(),HeisWeekday(),HeisDayDiff(),
HeisDaySmaller()

1.30 Date/HeisWeekday

NAME
HeisWeekday -- Gets the weekday of a specified date. (V33)

SYNOPSIS
weekday := HeisWeekday(day,month,year);

```
PROCEDURE HeisWeekday(day,month : SHORTINT;
  year : INTEGER) : Weekday;
```

FUNCTION
HeisWeekday gets the weekday for a specified date.

INPUTS
day - day of the date
month - month of the date
year - year of the date

RESULT
 weekday - This result is of type:
 Weekdays = (dayerr,Monday,Tuesday,Wednesday,Thursday,Friday,
 Saturday,Sunday);
 dayerr will show you, that an error occurs!

EXAMPLE
 ...
 weekday := HeisWeekday(22,1,1994);
 IF weekday = dayerr THEN
 ...
 END;
 ...

NOTES
 It is better only to use this function for years from -7 to 8000!
 In this version dayerr will only occur for the lost days :)

BUGS
 It is not possible to use year < 0 (see JulianWeekday() for more).

SEE ALSO
 JulianWeekday(),GregorianWeekday(),HeisDaySmaller(),HeisLeapYear()

1.31 Date/HeisYearDays

NAME
 HeisYearDays -- Gives back the number of days in a year. (V33)

SYNOPSIS
 days := HeisYearDays(year);

PROCEDURE HeisYearDays(year : INTEGER) : INTEGER;

FUNCTION
 HeisYearDays gives you back the number of days in
 a specified year.

INPUTS
 year - The year in which to count the days.
 (I think its better not to use years before -7!)

RESULT
 days - The number of days the year uses.

EXAMPLE
 ...
 days := HeisYearDays(1994);
 WriteString("Days of 1994 : ");
 WriteCard(days,3); WriteLn;
 ...

NOTES
 It is better only to use this function for years from -7 to 8000!

BUGS
No known bugs.

SEE ALSO
HeisMonthDays(), JulianYearDays(), GregorianYearDays()

1.32 Date/HSYearToJD

NAME
HSYearToJD -- Calcs the JD from a Scaliger year. (V33)

SYNOPSIS
jd := HSYearToJD(syear);

PROCEDURE HSYearToJD(syear : INTEGER) : LONGINT;

FUNCTION
Returns the Julianday of a Scaliger year.

INPUTS
syear - Scaliger year

RESULT
jd - The Julianday

EXAMPLE
...
jd := HSYearToJD(6700);
...

NOTES
It is better to use this function only from 4707 to 7981!
In this version only GYearToJD() is called, because the
Scaliger period is only valid to 3268

BUGS
unknown.

SEE ALSO
JYearToJD(), GYearToJD()

1.33 Date/HYearToScaliger

NAME
HYearToScaliger -- Returns the year as Scaliger year. (V33)

SYNOPSIS
syear := HYearToScaliger(year);

PROCEDURE HYearToScaliger(year : INTEGER) : INTEGER;

FUNCTION

Returns the Scaliger year.

INPUTS

year - Heis year

RESULT

syear - The Scaliger year

EXAMPLE

...

```
syear := HYearToScaliger(1994);
```

...

NOTES

It is better to use this function only from -7 to 8000!

BUGS

The Scaliger period is defined to 3268!!!.

SEE ALSO

JYearToScaliger(), GYearToScaliger()

1.34 Date/JDtoMJD

NAME

JDtoMJD -- Switches from JD to MJD. (V33)

SYNOPSIS

```
mjd := JDtoMJD(jd);
```

```
PROCEDURE JDtoMJD(jd : LONGINT) : LONGINT;
```

FUNCTION

Returns the Modified Julianday of a Julianday.

INPUTS

jd - Julianday

RESULT

mjd - The Modified Julianday

EXAMPLE

...

```
mjd := JDtoMJD(2449354);
```

...

NOTES

none

BUGS

Only use this function for $jd > 2400001$, because mjd is only defined for this, otherwise system will crash!

SEE ALSO

MJDtoJD()

1.35 Date/JDToTime

NAME

JDToTime -- Returns the real time for a JD time. (V33)

SYNOPSIS

JDToTime(jd,rhour,rmin,rsec);

PROCEDURE JDToTime(jd : REAL; VAR rhour,rmin,rsec : SHORTINT);

FUNCTION

Returns the real time for a JD time.

INPUTS

jd - JD time

RESULT

rhour - 24 hour real time

rmin - real minutes

rsec - real seconds

EXAMPLE

...

JDToTime(0.76543,rhour,rmin,rsec);

...

NOTES

none.

BUGS

If jd is > 0 (including days) there will be occur arithmetic bugs!

SEE ALSO

TimeToJD()

1.36 Date/JSYearToJD

NAME

JSYearToJD -- Calcs the JD from a Scaliger year. (V33)

SYNOPSIS

jd := JSYearToJD(syear);

PROCEDURE JSYearToJD(syear : INTEGER) : LONGINT;

FUNCTION

Returns the Julianday of a Scaliger year.

INPUTS

syear - Scaliger year

```
RESULT
jd - The Julianday

EXAMPLE
...
jd := JSYearToJD(4800);
...

NOTES
It is better to use this function only from 4707 to 6295!

BUGS
unknown.

SEE ALSO
GSYearToJD(), HYearToJD()
```

1.37 Date/JulianDayDiff

```
NAME
JulianDayDiff -- Calculates the days between 2 dates. (V33)

SYNOPSIS
days := JulianDayDiff(day1,month1,year1,day2,month2,year2);

PROCEDURE JulianDayDiff(day1,month1 : SHORTINT; year1 : INTEGER;
    day2,month2 : SHORTINT; year2 : INTEGER) : LONGINT;

FUNCTION
JulianDayDiff gives you back the number of days between
two specified dates.

INPUTS
day1    - day of the first date
month1  - month of the first date
year1   - year of the first date
day2    - day of the second date
month2  - month of the second month
year2   - year of the second date

RESULT
days - The number of days between the two dates
      (positive if date1 <= date2).

EXAMPLE
...
days := JulianDayDiff(18,9,1970,22,1,1994);
WriteString("Age of Kai Hofmann in days : ");
WriteInt(days,10); WriteLn;
...

NOTES
It is better only to use this function for years from -7 to 1582!
```

BUGS
No known bugs.

SEE ALSO
GregorianDayDiff(), HeisDayDiff(), JulianMonthDays(), JulianYearDays()

1.38 Date/JulianDayGreater

NAME
JulianDayGreater -- Checks if date1 is greater than date2. (V33)

SYNOPSIS
greater := JulianDayGreater(day1,month1,year1,day2,month2,year2);

PROCEDURE JulianDayGreater(day1,month1 : SHORTINT; year1 : INTEGER;
 day2,month2 : SHORTINT; year2 : INTEGER) : BOOLEAN;

FUNCTION
JulianDayGreater test if date1 is greater than date2.

INPUTS
day1 - day of the first date
month1 - month of the first date
year1 - year of the first date
day2 - day of the second date
month2 - month of the second month
year2 - year of the second date

RESULT
greater - This is TRUE is date1 > date2 otherwise it's FALSE.

EXAMPLE
...
IF JulianDayGreater(18,9,1970,22,1,1994) THEN
 WriteString(">"); WriteLn;
ELSE
 WriteString("<="); WriteLn;
END;
...

NOTES
It is better only to use this function for years from -7 to 1582!

BUGS
No known bugs.

SEE ALSO
GregorianDayGreater(), HeisDayGreater()

1.39 Date/JulianDaysAfterWeekday

NAME

JulianDaysAfterWeekday -- Returns the diff to the wday after. (V33)

SYNOPSIS

days := JulianDaysAfterWeekday(day,month,year,weekday);

PROCEDURE JulianDaysAfterWeekday(day,month : SHORTINT;
year : INTEGER; weekday : Weekdays) : SHORTINT;

FUNCTION

Returns the days to the weekday after the specified date.
So if you specify the 22.1.1994 (Saturday) and Thursday
you get back 5!
If you specify the 22.1.1994 and Saturday you get back 0
(the same day)!

INPUTS

day - day of the date
month - month of the date
year - year of the date
weekday - weekday to search for building difference

RESULT

days - The days after to the searched weekday.

EXAMPLE

...
days := JulianDaysAfterWeekday(22,1,1994,Thursday);
...

NOTES

It is better to use this function only from -7 to 1582!

BUGS

See JulianWeekday()!

SEE ALSO

GregorianDaysAfterWeekday(),HeisDaysAfterWeekday(),JulianWeekday()

1.40 Date/JulianDaysBeforeWeekday

NAME

JulianDaysBeforeWeekday -- Returns the diff to the wday before. (V33)

SYNOPSIS

days := JulianDaysBeforeWeekday(day,month,year,weekday);

PROCEDURE JulianDaysBeforeWeekday(day,month : SHORTINT;
year : INTEGER; weekday : Weekdays) : SHORTINT;

FUNCTION

Returns the days to the weekday before the specified date.
So if you specify the 22.1.1994 (Saturday) and Thursday
you get back 2!

If you specify the 22.1.1994 and Saturday you get back 0 (the same day)!

INPUTS

day - day of the date
 month - month of the date
 year - year of the date
 weekday - weekday to search for building difference

RESULT

days - The days back to the searched weekday (0-6)
 If you get back 8 an error occurs!

EXAMPLE

```
...
days := JulianDaysBeforeWeekday(22,1,1994,Thursday);
...
```

NOTES

It is better to use this function only from -7 to 1582!

BUGS

See JulianWeekday()!

SEE ALSO

GregorianDaysBeforeWeekday(), HeisDaysBeforeWeekday(), JulianWeekday()

1.41 Date/JulianDaySmaller

NAME

JulianDaySmaller -- Checks if date1 is smaller than date2. (V33)

SYNOPSIS

```
smaller := JulianDaySmaller(day1,month1,year1,day2,month2,year2);
```

```
PROCEDURE JulianDaySmaller(day1,month1 : SHORTINT; year1 : INTEGER;
    day2,month2 : SHORTINT; year2 : INTEGER) : BOOLEAN;
```

FUNCTION

JulianDaySmaller test if date1 is smaller than date2.

INPUTS

day1 - day of the first date
 month1 - month of the first date
 year1 - year of the first date
 day2 - day of the second date
 month2 - month of the second month
 year2 - year of the second date

RESULT

smaller - This is TRUE is date1 < date2 otherwise it's FALSE.

EXAMPLE

```
...
IF JulianDaySmaller(18,9,1970,22,1,1994) THEN
```

```
    WriteString("<"); WriteLn;
ELSE
    WriteString(">="); WriteLn;
END;
...

NOTES
It is better only to use this function for years from -7 to 1582!

BUGS
No known bugs.

SEE ALSO
GregorianDaySmaller(), HeisDaySmaller()
```

1.42 Date/JulianDiffDate

```
NAME
JulianDiffDate -- Returns the date for a diff to another date. (V33)

SYNOPSIS
JulianDiffDate(day,month,year,diffdays,dday,dmonth,dyear);

PROCEDURE JulianDiffDate(day,month : SHORTINT; year,days : INTEGER;
    VAR dday,dmonth : SHORTINT; VAR dyear : INTEGER);

FUNCTION
Returns the date wich lies diffdays before/after the specified date.

INPUTS
day      - day of the date
month    - month of the date
year     - year of the date
diffdays - difference to the date in days

RESULT
dday     - Destination day
dmonth   - Destination month
dyear    - Destination year

EXAMPLE
...
JulianDiffDate(23,1,1994,7,dday,dmonth,dyear);
...

NOTES
It is better to use this function only from -7 to 1582!

BUGS
unknown.

SEE ALSO
GregorianDiffDate(), HeisDiffDate(), JulianDayDiff(), JulianMonthDays()
```

1.43 Date/JulianLeapYear

NAME

JulianLeapYear -- Checks if a year is a leap year for jj. (V33)

SYNOPSIS

```
leapyear := JulianLeapYear(year);
```

```
PROCEDURE JulianLeapYear(year : INTEGER) : BOOLEAN;
```

FUNCTION

JulianLeapYear checks if a year is a leap year in the julian calendar
For years after Chr. it checks if the year is devideable by 4.
For years before Chr. a leap year must have a modulo 4 value of 1

INPUTS

year - The year which should be checked (from -32768 to 32767)
I think only values from -7 to 1582 are valid, because of
the variant that was done on -8 by Augustus!

RESULT

leapyear - TRUE if the year is a leap year, otherwise false.

EXAMPLE

```
...
IF JulianLeapYear(1994) THEN
  WriteString("leap year!");
ELSE
  WriteString("no leap year!");
END;
WriteLn;
...
```

NOTES

A year is 365.25 days long!
Use this function only for values from -7 to 1582!

BUGS

No known bugs.

SEE ALSO

GregorianLeapYear(), HeisLeapYear()

1.44 Date/JulianMonthDays

NAME

JulianMonthDays -- Gives back the number of days of a month. (V33)

SYNOPSIS

```
days := JulianMonthDays(month, year);
```

```
PROCEDURE JulianMonthDays(month : SHORTINT;
  year : INTEGER) : SHORTINT;
```



```

FUNCTION
JulianMonthDays gives you back the number of days a month in
a specified year has.

INPUTS
month - The month from wich you want to get the number of days.
year  - The year in which the month is.

RESULT
days - The number of days the month uses, or 0 if you use
       a wrong month.

EXAMPLE
...
days := JulianMonthDays(1,1994);
WriteString("Days of January 1994 : ");
WriteCard(days,2); WriteLn;
...

NOTES
It is better only to use this function for years from -7 to 1582!

BUGS
No known bugs.

SEE ALSO
JulianLeapYear(),GregorianMonthDays(),HeisMonthDays()

```

1.45 Date/JulianToJD

```

NAME
JulianToJD -- Returns the JD for a date. (V33)

SYNOPSIS
jd := JulianToJD(day,month,year);

PROCEDURE JulianToJD(day,month : SHORTINT;
                    year : INTEGER) : LONGINT;

FUNCTION
Returns the JD for a Julian date.

INPUTS
day      - day of the date to convert
month    - month of the date to convert
year     - year of the date to convert

RESULT
jd - This is the JD

EXAMPLE
...
jd := JulianToJD(23,1,1994);
...

```

NOTES

It is better to use this function only from -7 to 1582!

BUGS

unknown.

SEE ALSO

GregorianToJD(), HeisToJD(), JSYearToJD(), JYearToScaliger(),
JulianDayDiff()

1.46 Date/JulianWeek

NAME

JulianWeek -- Gets the weeknumber of a specified date. (V33)

SYNOPSIS

```
weeknr := JulianWeek(day,month,year);
```

```
PROCEDURE JulianWeek(day,month : SHORTINT;  
    year : INTEGER) : SHORTINT;
```

FUNCTION

JulianWeek gets the weeknumber for a specified date.

INPUTS

day - day of the date
month - month of the date
year - year of the date

RESULT

week - This is the number of the week the specified date lies in.
If the first day in a new year is a Friday, Saturday or
Sunday, this would be the last week of the last year!
If the 29.12. is a Monday, the 30.12. is a Monday or a Tuesday,
the 31.12. is a Monday, Tuesday or a Wednesday this is the
first week of the next year!

EXAMPLE

```
...  
weeknr := JulianWeek(4,10,1582);  
...
```

NOTES

It is better only to use this function for years from 0 to 1582!

BUGS

For years < 0 errors could occur.

SEE ALSO

GregorianWeek(), HeisWeek(), JulianWeekday(), JulianDayDiff(),
JulianDaySmaller()

1.47 Date/JulianWeekday

NAME

JulianWeekday -- Gets the weekday of a specified date. (V33)

SYNOPSIS

```
weekday := JulianWeekday(day,month,year);
```

```
PROCEDURE JulianWeekday(day,month : SHORTINT;  
    year : INTEGER) : Weekday;
```

FUNCTION

JulianWeekday gets the weekday for a specified date.

INPUTS

day - day of the date
month - month of the date
year - year of the date

RESULT

weekday - This result is of type:
Weekdays = (dayerr,Monday,Tuesday,Wednesday,Thursday,Friday,
Saturday,Sunday);
dayerr will show you, that an error occurs!

EXAMPLE

```
...  
weekday := JulianWeekday(4,10,1582);  
IF weekday = dayerr THEN  
...  
END;  
...
```

NOTES

It is better only to use this function for years from 0 to 1582!
In this version no dayerr will occur!

BUGS

For years < 0 errors could occur, or systemcrashes(?).

SEE ALSO

GregorianWeekday(),HeisWeekday()

1.48 Date/JulianYearDays

NAME

JulianYearDays -- Gives back the number of days in a year. (V33)

SYNOPSIS

```
days := JulianYearDays(year);
```

```
PROCEDURE JulianYearDays(year : INTEGER) : INTEGER;
```

FUNCTION

JulianYearDays gives you back the number of days in a specified year.

INPUTS

year - The year in which to count the days.

RESULT

days - The number of days the year uses.

EXAMPLE

```
...
days := JulianYearDays(1994);
WriteString("Days of 1994 : ");
WriteCard(days,3); WriteLn;
...
```

NOTES

It is better only to use this function for years from -7 to 1582!

BUGS

No known bugs.

SEE ALSO

JulianMonthDays(), GregorianYearDays(), HeisYearDays()

1.49 Date/JYearToScaliger

NAME

JYearToScaliger -- Returns the year as Scaliger year. (V33)

SYNOPSIS

```
syear := JYearToScaliger(year);
```

```
PROCEDURE JYearToScaliger(year : INTEGER) : INTEGER;
```

FUNCTION

Returns the Scaliger year.

INPUTS

year - Julian year

RESULT

syear - The Scaliger year

EXAMPLE

```
...
syear := JYearToScaliger(1582);
...
```

NOTES

It is better to use this function only from -7 to 1582!

BUGS

unknown.

SEE ALSO
GYearToScaliger(), HYearToScaliger()

1.50 Date/LMT

NAME
LMT -- Calculates your local time in your timezone (V33)

SYNOPSIS
secs := LMT(secs, meridian, pos);

PROCEDURE LMT(secs : LONGINT; meridiandegree,
posdegree : REAL) : LONGINT;

FUNCTION
Calculates your Local Mean Time of you place!

INPUTS
secs - Seconds of the running day (hours*3600+min*60+sec)
meridian - Degrees of your timezone-meridian
pos - Degrees of your place

RESULT
secs - Local seconds of the running day

EXAMPLE
...
secs := LMT(76080, 15.0, 8.923055556);
...

NOTES
none

BUGS
No errorcheck, if you put in valid degrees (-180 to +180)

SEE ALSO

1.51 Date/MJDtoJD

NAME
MJDtoJD -- Switches from MJD to JD. (V33)

SYNOPSIS
jd := MJDtoJD(mjd);

PROCEDURE MJDtoJD(mjd : LONGINT) : LONGINT;

FUNCTION
Returns the Julianday of a Modified Julianday.

INPUTS
mjd - Modified Julianday

RESULT
jd - The Julianday

EXAMPLE
...
jd := JDtoMJD(49353);
...

NOTES
none

BUGS
unknown.

SEE ALSO
MJDtoJD()

1.52 Date/ScaligerYearToG

NAME
ScaligerYearToG -- Returns the Scaliger year as Gregorian year. (V33)

SYNOPSIS
year := ScaligerYearToG(syear);

PROCEDURE ScaligerYearToG(syear : INTEGER) : INTEGER;

FUNCTION
Returns the Gregorian year of a Scaliger year.

INPUTS
syear - Scaliger year

RESULT
year - The Gregorian year

EXAMPLE
...
year := ScaligerYearToG(6400);
...

NOTES
It is better to use this function only from 4707 to 7981!

BUGS
unknown.

SEE ALSO
ScaligerYearToJ(), ScaligerYearToH()

1.53 Date/ScaligerYearToH

NAME

ScaligerYearToH -- Returns the Scaliger year as Heis year. (V33)

SYNOPSIS

```
year := ScaligerYearToH(syear);
```

```
PROCEDURE ScaligerYearToH(syear : INTEGER) : INTEGER;
```

FUNCTION

Returns the Heis year of a Scaliger year.

INPUTS

syear - Scaliger year

RESULT

year - The Heis year

EXAMPLE

```
...
year := ScaligerYearToH(7000);
...
```

NOTES

It is better to use this function only from 4707 to 7981!

BUGS

unknown.

SEE ALSO

ScaligerYearToJ(), ScaligerYearToG()

1.54 Date/ScaligerYearToJ

NAME

ScaligerYearToJ -- Returns the Scaliger year as Julian year. (V33)

SYNOPSIS

```
year := ScaligerYearToJ(syear);
```

```
PROCEDURE ScaligerYearToJ(syear : INTEGER) : INTEGER;
```

FUNCTION

Returns the Julian year of a Scaliger year.

INPUTS

syear - Scaliger year

RESULT

year - The Julian year

EXAMPLE

```
...
```

```
year := ScaligerYearToJ(4800);  
...  
  
NOTES  
It is better to use this function only from 4707 to 6295!  
  
BUGS  
unknown.  
  
SEE ALSO  
ScaligerYearToG(), ScaligerYearToH()
```

1.55 Date/SecToTime

```
NAME  
SecToTime -- Returns the time from seconds (V33)  
  
SYNOPSIS  
SecToTime(secs, hour, min, sec);  
  
PROCEDURE SecToTime(secs : LONGINT; VAR hour, min, sec : SHORTINT);  
  
FUNCTION  
Gives you back the time from the specified seconds  
  
INPUTS  
secs - Time in seconds  
  
RESULT  
hour - hours (0-23)  
min - minutes (0-59)  
sec - seconds (0-59)  
  
EXAMPLE  
...  
SecToTime(76860, hour, min, sec);  
...  
  
NOTES  
Don't forget to convert 24h time to AM/PM time if needed!  
  
BUGS  
No errorcheck, if you use a valid time  
  
SEE ALSO  
TimeToSec()
```

1.56 Date/TimeToJD

```
NAME  
TimeToJD -- Returns the JD for a time. (V33)
```

```
SYNOPSIS
jd := TimeToJD(hour,min,sec);

PROCEDURE TimeToJD(hour,min,sec : SHORTINT) : REAL;

FUNCTION
Returns the JD for a specified time.

INPUTS
hour - hour of the time to convert
min  - minute of the time to convert
sec  - sec. of the time to convert

RESULT
jd - This is the JD time

EXAMPLE
...
jd := TimeToJD(16,33,0);
...

NOTES
none

BUGS
There is no check, if the specified time is a valid time!

SEE ALSO
JDToTime()
```

1.57 Date/GregorianEaster

```
NAME
GregorianEaster -- Returns the date of eastern in a year (V33)

SYNOPSIS
GregorianEaster(year,dday,dmonth);

PROCEDURE GregorianEaster(year : INTEGER;
    VAR dday,dmonth : SHORTINT);

FUNCTION
Returns the date of eastern for a specified year.

INPUTS
year - eastern is calculated for this year

RESULT
dday  - day of easter-Sunday
dmonth - month of easter-Sunday

EXAMPLE
...
GregorianEaster(1994,dday,dmonth);
...
```

NOTES

Use this only for 1900 to 2099!

Tested for 1977-1994! But this formula is from Gauß - so it must be correct :)

BUGS

None.

SEE ALSO

GEP(),GregorianJHSB()

1.58 Date/TimeToSec

NAME

TimeToSec -- Returns the time in seconds (V33)

SYNOPSIS

secs := TimeToSec(hour,min,sec);

PROCEDURE TimeToSec(hour,min,sec : SHORTINT) : LONGINT;

FUNCTION

Gives you back the time in seconds

INPUTS

hour - hours you want (0-23)

min - minutes you want (0-59)

sec - seconds you want (0-59)

RESULT

secs - Time in seconds

EXAMPLE

...

secs := TimeToSec(21,15,00);

...

NOTES

Don't forget to convert AM/PM time to 24h time!

BUGS

No errorcheck, if you use a valid time

SEE ALSO

SecToTime()

1.59 Date/TimeZoneFactor

NAME

TimeZoneFactor -- Returns the value you have to add to GMT time (V33)

SYNOPSIS

```
addhours := TimeZoneFactor(degrees);
```

```
PROCEDURE TimeZoneFactor(degree : SHORTINT) : SHORTINT;
```

FUNCTION

This gives you the hours you have to add to GMT time, specified on the fact, that a timezone is 15 degrees and that GMT is centered on 0 degrees!

INPUTS

degrees - Position of timezone you live in (from -180 to +180)

RESULT

addhours - Time to add to GMT time to get your locale zone time (-12 to +12)

EXAMPLE

```
...
```

```
addhours := TimeZoneFactor(-8);
```

```
...
```

NOTES

none

BUGS

No errorcheck, if you put in valid degrees (-180 to +180)

Only full degrees are supportet, keep sure that you round in the right way for 0.x degree places

I am not sure about the correct +/- behaviour!!!

SEE ALSO