

EARIN Lab 7 Report

Krzysztof Rudnicki, 307585 Jakub Kliszko, 303866

June 4, 2023

1 Exercise Variant 2

Our task was to write Prolog program which returns number of days between two dates

Assumptions:

- Year is 2023
- Number of days is ≤ 365

Exemplary use:

```
?- interval("2205", "0506")
14
```

```
?- interval("0102", "1102")
10
```

Additional assumptions we made based on exemplary use is that we always receive date in "ddmm" format so first we receive 'd' (days) and then 'm' (month) If either days or month is a single digit we put '0' in front of the digit to force the string to be 4 characters wide

Examples:

1st of January is represented by "0101"

2nd of November is represented by "0211"

15th of January is represented by "1501"

Another assumption we made is that interval will be always positive, order of dates does not matter so those queries:

```
?- interval("2205", "0506")
14
```

```
?- interval("0506", "2205")
14
```

Will give the exact same results.

Last assumption is that we do not count first date as whole day so:

```
?- interval("0101", "0101")
0

?- interval("0101", "3112")
364
```

2 Program

Our program successfully performs its task and returns correct number of days based on our assumptions.

Program fails and prints out a message "Invalid input" if:

- The number of days in the month is too big (for example `interval("3205", "0506")`)
- Month does not exist (for example `interval("0113", "0506")`)
- Input does not make sense (for example `interval("xxyy", "0506")`)
- Day of month is smaller or equal to "00" (for example `interval("0011", "0506")`)

Program does NOT return error and tries to return correct output (thanks to Prolog magic) for example when given incomplete input like:

```
?- interval("106", "0506").
4
```

2.1 Prolog magic

Prolog is based on the idea of logic programming. It does not have the concept of functions, it rather operates on predicates and goals. A predicate (a fact or a rule) defines a state of the world and a goal tells Prolog to make that state of the world come true, if possible (in other case it fails).

For example, we can query our program with such command and it will return *true*, because it is a valid state:

```
?- month_days('09', 30, 243).
true
```

The following query will fail, because the predicate is not valid for these values:

```
?- month_days('10', 28, 123).
false
```

We can however provide Prolog with an unbound values. It will try to find a possible solution and assign them. This is called unification:

```

?- month_days('09', DaysInMonth, Days).
Days = 243,
DaysInMonth = 30

```

If there are multiple possible valid solutions, Prolog will remember them and backtrack to them in case it fails later. User can also ask for another solution if its available by pressing `;` for instance:

```

?- month_days(N, 30, X).
N = '04',
X = 90      ;
N = '06',
X = 151     ;
N = '09',
X = 243     ;
N = '11',
X = 304

```

2.2 Modules

The program consists of three main modules

1. `month_days` which defines facts about the months – how many days each month consists of and how many days it takes to reach this month
2. `day_of_year` which tells how many days have passed since the beginning of the year to a given date
3. `interval` which actually calculates the interval between two dates and writes it

2.3 Tested examples

```

% Wrong number of days in month
?- interval("3205", "0506")
Invalid input.
false.

% non existing month
?- interval("0113", "0506").
Invalid input.
false.

% erroneous input
?- interval("xxyy", "0506").
Invalid input.

```

```

false.

% Number of days equal or smaller than 00
?- interval("0011", "0506").
Invalid input.
false.

?- interval("-1011", "0506").
Invalid input.
false.

% Examples from project requirements
?- interval("2205", "0506").
14
true.

?- interval("0102", "1102").
10
true.

% Edge cases, first day of year and last day of
year
?- interval("0101", "3112").
364
true.

% Edge case, the same date in both inputs
?- interval("0101", "0101").
0
true.

% Edge case, input without first digit of day
?- interval("2205", "506").
14
true.

```

2.4 Differences

In exemplary use there is no *dot '.'* after query which was necessary for some interpreters (swipl) to actually run the query.

Also in exemplary use there is no 'true' statement after number of days which Prolog by default displays after successful queries; we decided to not forcefully change it as it would contradict regular Prolog use philosophy.

3 Challenges

The biggest challenge was understanding Prolog logic programming and its difference from functional programming paradigm.

Then it was about finding correct directive for handling characters of date, we decided on `atom_chars` and `atom_numbers`.

Last difficult part was writing the output exactly as in examples using `write(Interval)` and `nl`.