Exercise 7: Keeping track of data entry time

At the end of this exercise you should be able to:

a. Edit the check file, and use temporary variables to calculate the amount of time required for data entry.

In this exercise you will learn to calculate the number of seconds which are required to complete data entry for one record. A field to retain this value as part of the database thus needs to be added to the questionnaire.

If you prepare a study, you should make it a rule to enter several hundred records by yourself (with the help of a colleague). This is important for two reasons:

- Identify weaknesses in the data entry form and the `CHK` file
- Recording the time needed to enter the information

You need the information on recording time to know what you can expect from another data entry person. This is critical for making your budget. It is obviously not satisfactory to pay a data entry person according to the time the person claims to need: some people are slow and they should not receive the same remuneration as the faster person. If you take your achievement as the basis, you can objectively pay for the time required if a data entry person works at your speed: those slower than you will loose, those faster will win, and that is how it should be.

We can use EpiData Entry to monitor our data entry time and make a permanently available record that we can later analyze. In the `EpiData\samples\` folder you find three files:

```
DateTime.qes
DateTime.rec
DateTime.chk
```

These three files are the basis to adjust our own sample files together with the **About time** in the EpiData Entry Help file.

There are two important things you need to know about dates.

- EpiData works internally with dates as date numbers, counting the number of days since 31st December 1899, which has the day number 1. This is referred to as the anchor date. The date 15th October 2000 has, for example, the day number 36814. The advantage of day numbers is that it makes it easy to perform calculations with dates, e.g. adding 7 days to a date or counting the number of days from a specific date to today.
- EpiData understands dates in European date format only i.e, `dd/mm/yyyy`. So a date 01/07/2012 is understood as 1 July, 2012 and not 7 January, 2012. Date constants can be defined in two ways: either by "`15/03/1977`" or by `date(15,3,1977)` which both will produce the date 15th March 1977.

We are going to make use of computer's clock. EpiData Entry has a function `NOW` which records the exact time of the computer. This time consists of the date and the time of the day. `NOW` writes the information on the date into the integer part of a field and the time of the day

as a fraction of the 24-hour clock into the fractional part of the field. If we define thus such a field for the file and start counting the time when a new record is opened:

```
BEFORE FILE
  DEFINE StartTime #####.######
END

BEFORE RECORD
  StartTime=NOW
END
```

We could of course write the field `StartTime` into the questionnaire as a `NOENTER` field and then we would get, as an example:

`39650.864672`

`NOW` gives the computer time right at this point, writing the date as the integer part and the fraction of the day as the fractional part. The computer stores the date as the number of days passed since the internal anchor date, 31 December 1899. If we divide the above integer part of the number by the average number of days per year we get 39650/365.25=108.56, that is a bit more than 108.5 years after the anchor date, or some time in July 2008. What about the fractional part? If we multiply the 24-hour day by this fraction we get 0.864672*24=20.75 which is roughly a quarter to nine in the evening, but it is much more precise than to the quarter of the hour (6 digits!) because what we really want is to have it expressed in seconds and one day has 86,400 seconds. To get the rounding properly we use even 6 rather than the bare minimum of 5 digits.

As the database only needs to retain the number of seconds required to complete one record, the number above is thus just something we need the `CHK` file to calculate in the background.

Assuming that we have a field `SECONDS` in the data file, we would then write after entering the value for the last field:

```
res3sc
  COMMENT LEGAL USE label_scanty SHOW
  MUSTENTER
  TYPE COMMENT
  AFTER ENTRY
    IF seconds=. THEN
      seconds=(NOW-StartTime)*86400
    ENDIF
  END
END
```

This `NOW` is a different `NOW` from the beginning (computer clock!) and all we do is to revert the difference between the two time points into seconds.

If we do it as above, the clock will stop right after the value of the last field `RES3SC` has been entered and that value (number of seconds) will be written into the field `SECONDS`. Even when returning to the record, the original value in the field `SECONDS` will not change. If we were to take out the:

```
    IF seconds=. THEN
      seconds=(NOW-StartTime)*86400
    ENDIF
```

then the number of seconds would change to a new value if going again later through the record, and the value then might be lower. Ideally, however, the number of SECONDS would need to be cumulative, i.e. if a record is re-visited, the additional time during the revisit would be added to the pre-existing time. In other words, the time for corrections after validation would be added. The change required to accomplish the latter is to delete it from the AFTER ENTRY statement in the RES3SC field:

```
res3sc
  COMMENT LEGAL USE label_scanty SHOW
  MUSTENTER
  TYPE COMMENT
  AFTER ENTRY
    IF seconds=. THEN
      seconds=(NOW-StartTime)*86400
    ENDIF
  END
END
```

Instead, the statements are integrated into the AFTER RECORD statement and extended to provide accumulation of time. *Note that this is a better choice than the previous option as the amount of time required to save the record would also be included in the calculations now.*

```
AFTER RECORD
  IF idcode=. THEN
    HELP "You cannot save a record without an identifier\n  Please enter
all required information" TYPE=WARNING
    GOTO labname
  ENDIF
  IF seconds=. THEN
    seconds=(NOW-StartTime)*86400
   ELSE
    seconds=seconds+(NOW-StartTime)*86400
  ENDIF
END
```

Let us summarize the rationale of this approach once again.

Before the entry of a record is started, StartTime gets a value corresponding to the time of the clock at that point in time in your computer using the function NOW. It will be in the format 39814.25678 – the integer part refers to the date (converted into a number after subtracting the anchor date from current date) and the decimal part refers to the fraction of time in the day. Similarly, after the entry is complete, we will record the time at that point, again in the same format 39814.26677. The difference between the two gives the time in terms of fraction of the day = (39814.26677-39814.25678)=0.00999; this fraction, when multiplied by 24 (number of hours in a day) will give time duration in hours; when multiplied by 1440 (number of minutes in a day) will give time duration in minutes. Since we need the duration in seconds, we have multiplied the fraction by 86400 (the number of seconds in a day).

However, an even more informative approach is to have both the information about time required to enter one record for the first time and the information about the total amount of time (cumulatively) spent on a records, i.e. including the time spent on corrections. In the analysis one can then thus determine how much time is spent on entering one record, and how

much additional time on correcting the record if a discordance must be resolved after validation (by subtracting the value in the field for first entry from that of the cumulative time). This is important because it has been shown that working faster is accompanied by making more errors, thus requiring revisiting more records again.

Note on formatting: The choice of a mixture of lower-case and upper-case letters (eg, in `StartTime`) is for easier visual recognition only. As mentioned earlier, EpiData Entry is not case-sensitive. This makes it particularly powerful as you can make use of formatting to visual ease. As a general rule, if you let EpiData Entry make the choices for you, commands are capitalized, all the rest is not. As for field names, lower-case is always preferred. While it does not matter in formatting, you can force its format in the REC file to any option you prefer, but we give preference to lower case (go to "File" "Options" "Create data file").

Task:

- ***Start with the*** `A_EX06 QES-REC-CHK` ***files, save them as*** `A_EX07 QES-REC-CHK` ***and revise them accordingly. Don't just retype the above, try to consider the logic of it!***
- ***Make two*** `NOENTER` ***fields, one that calculates the data entry time for the first entry (that will not change by revisiting the records) and another field that calculates the cumulative time resulting from one or more re-visits of the record.***
- ***Enter some data to check the functionality.***