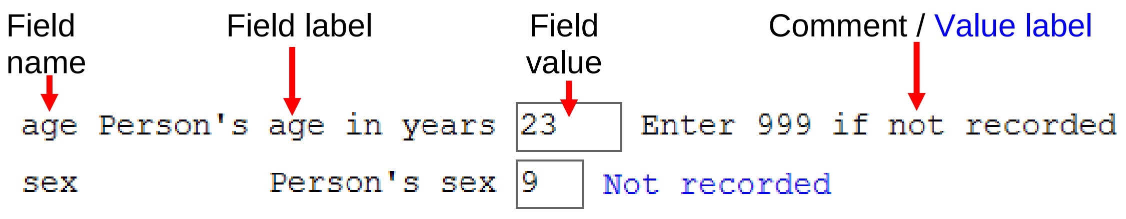
Exercise 2: The `QES-REC-CHK` triplet

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

a. Create and edit a questionnaire file (`*.qes`).
b. Make a record file (`*.rec`).
c. Make and edit a check file (`*.chk`).
d. Understand the `QES-REC-CHK` triplet and how the three are related to each other.

You are now ready to start with the design of the questionnaire in EpiData Entry, based on the data documentation sheet.

Before starting, let's clarify the terminology and where what is going to be placed. There are four elements that go with a field:

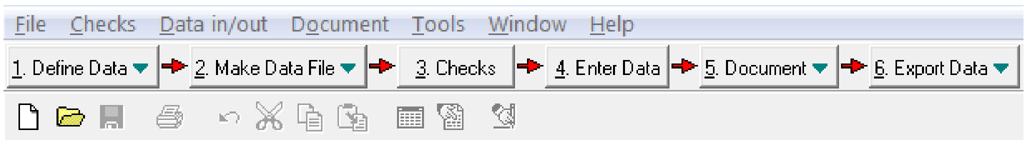


Field name
Field label
Field value
Comment / Value label
age Person's age in years 23 Enter 999 if not recorded
sex Person's sex 9 Not recorded

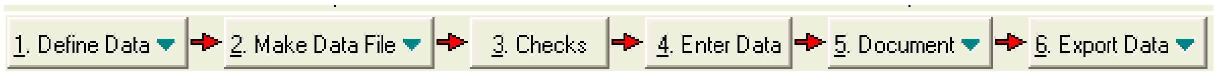
We call the variable the **Field name** (see Exercise 1). EpiData Entry interprets the first word in a line to be the Field name (see options). To have more explicit information than what can be conveyed by the length 10, single-word Field name, we supplement the Field name with a more explanatory **Field label**. Anything between the first word (Field name) and the Field definition is interpreted by EpiData to be the Field label. The third element on the line is the Field definition which receives the actual **Field value** during data entry. The field value is the actual datum (singular of "data") for that variable for that observation. For *continuous variables*, we might add an explanatory **Comment** to the right of the field definition, here for instance what to do if the primary source has no information on this variable for this observation. For *categorical variables*, the comment is not necessary: a label block will comprehensively provide information on what is entered. As we use numeric coding, it would, however, be a nice touch if after picking a value from a pick list (label block), the **Value label** would appear to the right of the field, as a kind of confirmation that what was picked was what was actually intended.

In the data documentation sheet we made two columns to distinguish "Comment" (for continuous variables) and "Value label" (for label blocks with categorical variables).

Open EpiData Entry by double-clicking the icon on your desktop or single-clicking the icon in your quick-launch task bar. You see the EpiData Entry has three rows on top of the screen - the top row (**Menu Bar**) with File Checks Data in/out etc, the second with a numbered sequence (**Work Process toolbar**), and the third (**Editor toolbar**) with mostly grayed out icons:



For the time being we concentrate only on the middle row that shows the following sequence (this is called the "Work Process toolbar"):



Each of these has a menu which you see when you click on the box. You can see immediately where you have to start.

Step 1: Creating the `*.QES` file

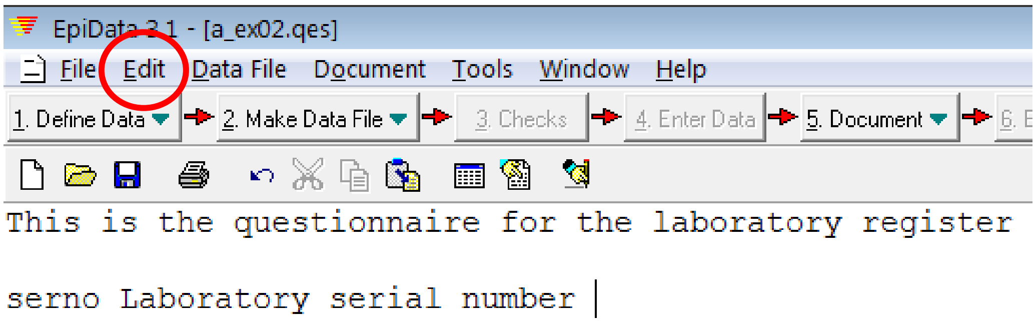
If you click on "1. Define data" (or using the shortcut **ALT+1**) the menu with two options pops up: "New .QES file" and "Open .QES file". EpiData questionnaire files have the extension "`*.QES`". As you must now create a new questionnaire file and not open an existing QES file, click "New .QES file" and the empty screen ready for writing opens.

Start to type like in any word processor the following:

```
This is the questionnaire for the laboratory register
serno Laboratory serial number
```

Let's save this right away as **`A_EX02.QES`** (shortcut: **CTRL+S**).

We have to associate two things with this field, the type of field and the field length. In the top menu line you see "Edit".

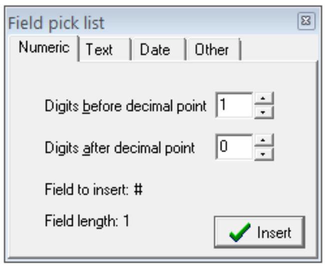

EpiData 3.1 - [a_ex02.qes]
File Edit Data File Document Tools Window Help
1. Define Data
2. Make Data File
3. Checks
4. Enter Data
5. Document
6. E

```
This is the questionnaire for the laboratory register

serno Laboratory serial number
```

Note: While EpiData Entry is not case-sensitive (that is, you may use upper-case or lower-case), some statistical packages are case-sensitive ("sex" is not identical to "Sex" or "SEX"). You are on the safe side if you make it a habit of using always lower-case for field names. See also "File" | "Options" | "Create Data File" to force lower-case for variable names.

Click on it to get the drop down menu. However, we encourage you to learn the shortcuts, they are often more efficient than the mouse. If you click the **Alt** key, you see that each of these menus has one letter underlined. In this case it is the E in Edit. Thus **Alt+E** is a fast way to see the drop down menu. In it, you see "Field pick list **CTRL+Q**". Pick it by typing "f" twice in sequence and the Field pick list box opens. You see four tabs, Numeric, Text, Date, and Other:



Digits before decimal point: 1

Digits after decimal point: 0

Field to insert: #

Field length: 1

Numeric fields

In numeric fields you define the number of digits before the decimal point. If there is none (zero) after the decimal point, the numeric field is an integer. If there is 1 or more digit after the decimal point, the numeric field is a float (real number). The symbol is the # (hash) sign:

`###`	Integer of length 3
`#.##`	Float of length 4 (the decimal point contributes to length)

For the field `SERNO`, we defined it to be an integer of length 4. Make sure that you have a space between the field name and the field type / field length:

```
serno Laboratory serial number ####
```

We add as per data documentation sheet the comment after the field definition what to do if one encounters a duplicate serial number.

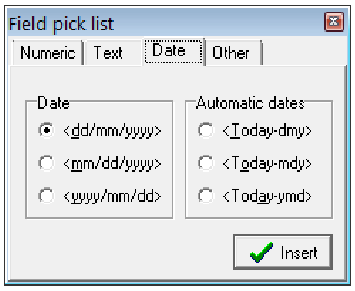
Text fields

Text: using this type you may enter upper or lower case letters or a mixture of cases, and the value will be exactly as you entered it. Commonly, these fields are also called "**String**" fields, as we may enter any string of characters that can be found on keyboard into such a field.

Upper-case text: if you enter a lower-case letter from your keyboard, it will automatically be converted to upper case. This is often very useful as the field value "`m`" is not the same as "`M`" and these would be considered as separate categories when you run a frequency table in analysis.

Encryption field: this is a powerful tool if you enter information like personal identifiers that should be openable only by persons with access to a password. More on data encryption will be discussed later in this course.

Date fields



Date: [✓ <dd/mm/yyyy>] <mm/dd/yyyy> <yyyy/mm/dd>

Automatic dates: <Today-dmy> <Today-mdy> <Today-ymd>

On the left hand side, we see that we have three options to visualize dates:

dd/mm/yyyy	"European" date
mm/dd/yyyy	"American" date
yyyy/mm/dd	"Chinese" date

> Note: It is of critical importance that you choose the style that is used in your setting else it is likely that many data entry errors will be made. To do it, click on start-control panel-regional and language options-choose English (UK). *Also, note that calculations by EpiData are made with European dates, what is chosen here is how dates are displayed (and entered), not how calculations are actually made internally.*

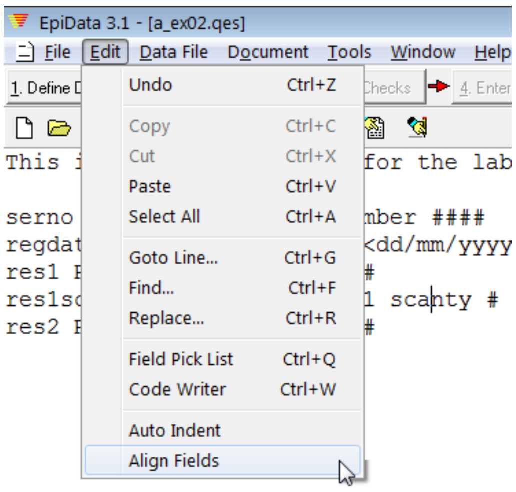
We are now only partially completing the questionnaire file, for the additional fields REGDATE, RES1, RES1SC, and RES2. For the remaining fields you will be doing doing as part of the task for this exercise. Thus, adding these four additional fields, you get:

```
This is the questionnaire for the laboratory register

serno Laboratory serial number ####   Enter 9001, 9002, ... if serial
regdate Registration date <dd/mm/yyyy>   Enter 01/01/1800 if missing
res1 Result of specimen 1 #
res1sc Result of specimen 1 scanty #
res2 Result of specimen 2 #
```

Aligning the fields

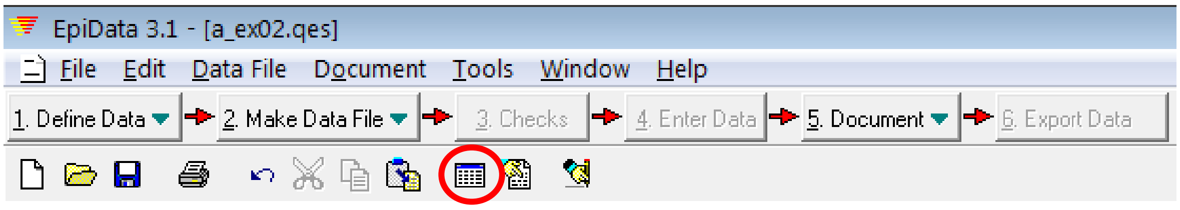
As you can observe, the fields are not properly lined up. Move your cursor anywhere to the field with the longest text before the field definition, here this is the line for the field RES1SC, then pick again Edit from the top line to see "Align fields" as the last on the drop-down menu and click it:



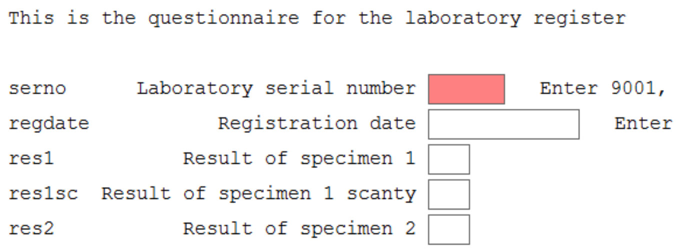
```
This i        for the lab

serno         mber ####
regdat        <dd/mm/yyyy
res1 R        #
res1sc        1 scanty #
res2 R        #
```

then the fields get aligned (sometimes you need to do it twice). To see how it will look for the data entry person, click the third icon from the right in the bottom third line:



and you get:



```
This is the questionnaire for the laboratory register

serno       Laboratory serial number [      ]    Enter 9001,
regdate            Registration date [          ]    Enter
res1             Result of specimen 1 [ ]
res1sc    Result of specimen 1 scanty [ ]
res2             Result of specimen 2 [ ]
```

F10 or **CTRL+F4** gets you out of this preview. For the time being, we leave this incomplete A_EX02.QES file and proceed with the next step.

Step 2: Making the *.REC file

We have now the QES file, and this provides the information on field definitions for the data file. **ALT+2** opens the drop down menu to pick the Make data file sub-menu which opens the "Create data file from *.QES file" dialogue box. The name of the *.QES file is already there and the same file name (with the *.REC extension) is proposed. That is very sensible, it should be so, and it must be so, and we accept this. **Note this very important principle – the names of the QES and REC files should have the same name to be linked with each other.** We are now prompted to enter a description. This is not necessary but it is helpful documentation. Thus we type in for instance "Exercise 2" and we are informed that the A_EX02.REC has been created.

Actually, this is enough to start data entry! You can open the REC file and start entering the data. However, you will notice that fields now can receive any values. The field definition itself poses some basic checks; for example, you will not be allowed to enter a text in a numeric field or you will not be able to enter an illegal date in a date field, but it can only go so far! *Rather than checking the data after all data have been entered, it may be useful to check the validity of the data during the data entry process.* Using a Check file (same name as the REC file but with the extension .CHK instead of .REC) makes this possible. It is in the CHK *file where you define all the rules of data entry control.* CHK files can do a range of things in ensuring data validity and include range checking, specification of legal values, making a field required, making conditional jumps between fields or any other conditional operations, using value labels, giving messages to the data entry operator and making complex calculations from the values entered in other fields.

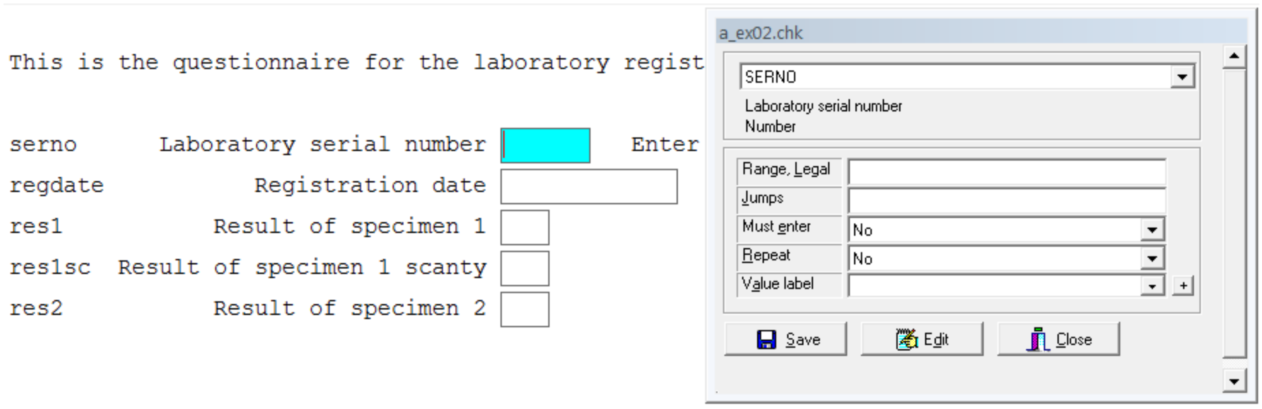
Let us create a CHK file right away!

There are two ways of creating a CHK file: By using menu-based option or by using the editor and writing all the CHK commands manually. While both is possible, preferentially the menu should be used for all field-level checks, because EpiData checks here your grammar. Let us thus begin with the menu and postpone the learning of how to edit the CHK file in a text editor when we create Checks that apply to the record or the file level (rather than a specific field), where using the menu is not possible.

3. Checks Step 3: Making the `*.CHK` file

With **ALT+3** we open the dialog box "Select data file for checks". The A_EX02.REC is suggested and that is the file we need.

The entry screen appears with one variable and a box showing that we have indeed now the third file A_EX02.CHK:



```
This is the questionnaire for the laboratory regist

serno       Laboratory serial number [      ]     Enter
regdate            Registration date [          ]
res1              Result of specimen 1 [  ]
res1sc    Result of specimen 1 scanty [  ]
res2              Result of specimen 2 [  ]
```

a_ex02.chk

SERNO
Laboratory serial number
Number

Range, Legal	
Jumps	
Must enter	No
Repeat	No
Value label	

The cursor is in the first field SERNO and we can see that this is a "Number" (numeric) field.

All Checks added with:

3. Checks

refer to the specific field which has the focus during data entry. **To add checks with reference to a specific field, you have to select the field by moving the cursor into that field**. Right now, the cursor is in the field SERNO (which is highlighted) and any checks you add will be for that field.

In the pop-up box you see different types of Checks we can apply to the field on which the focus is:

Range, Legal Here you can enter all legal values, i.e., it defines what values the data entry person is allowed to enter. In the case of the unique identifier, there will, by definition, be as many as there are individuals on whom information has to be entered. It does therefore not make any sense to define legal values for the variable SERNO. **Legal values are used for continuous variables. *Do not use Legal values for categorical variables*.** We have two fields for continuous variables, the registration date REGDATE and the patient's AGE:

```
01/01/2000-31/12/2005,01/01/1800
0-125,999
```

You can only have a single range, but you may add multiple single legal values, separated by commas.

Jumps You could determine here if a subsequent field should be skipped if the current value takes a certain value. For instance, you may have the value "2" for male and the value "1" for female in the field SEX. If the person is female, one might ask how many pregnancies (variable, e.g., PREGNO) she has had, but if the person is male, one would obviously not ask this question. Thus, you would

	jump (bypass) the question about pregnancies in case the study subject is male and go in that case to the field after pregnancies which might be AGE. To tell that a jump is needed in case the value of the field SEX is “2”, you would simply type: `2>age` In the case of the field SERNO, no jump is needed, and we will leave this open. We actually prefer an alternative to Jumps as shown specifically below.
Must enter	Here you define whether information must be entered or not. If you define a field as “MUSTENTER”, then once you enter into the field, you will not be able to get out of the field without entering a value. In the case of the field SERNO, we will require that it is entered. Choose thus “Yes” from the drop-down menu at the right. You will therefore not have any missing values for this variable. ***Note: in this course, we will always use*** MUSTENTER ***fields*** (except for automatically calculated variables).
Repeat	Here you can specify whether a value for the field should be repeated in all subsequent records, unless you choose to overwrite the value. This comes in handy when you make a field for capturing district name, and the district name in the field is the same for a set of consecutive records. For the field SERNO this is obviously not the case.
Value label	If we code, e.g., the value “female” for the field SEX with “1” and the value “male” with “2”, then it could prove useful to label it so that an explanation appears that “1” stands for “female” and “2” for “male” to reduce data entry errors. It will make the life of the data entry person so much easier.

You are reminded to save by clicking on ‘Save and close’ after working on checks.

Note: As you choose some check options in the pop-up menu, commands are written in the background. For example, if you click on Edit**,** you will notice the following:

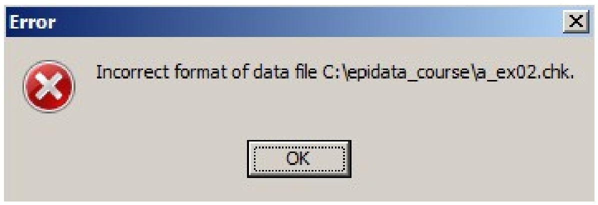
```
serno
  MUSTENTER
END
```

As you notice, the command MUSTENTER is written as we chose the option in the menu. Alternately, we can write MUSTENTER here directly and it is the same as choosing the menu option. Add and delete here to observe what happens on the menu. We recommend that you notice as you choose options from the pop-up menu.

Similarly for regdate, the following will appear as we define the range and legal values

```
regdate
  RANGE 01/01/2000 31/12/2005
  LEGAL
    01/01/1800
  END
  MUSTENTER
END
```

Note: When you are adding or revising checks in this way, that is via ‘3. Checks’, you have to click on the REC file for which the checks have to be defined. Sometimes, going by intuition, if you try to open a CHK file through this path by selecting the CHK file, you will encounter an error:



```
Error
Incorrect format of data file C:\epidata_course\a_ex02.chk.
```

At first it might appear counter-intuitive. If we think about it a bit, it becomes clear that we create here controls of what can be entered into a data (i.e. REC) file: we "make Checks for a REC file". We will introduce to you another way of opening the CHK file where you will follow the usual method – click on the CHK file to open it.

Specific points, exemplified with the five fields

MUSTENTER

We will make every field a MUSTENTER field, unless it is an automatically calculated field. The reason for this choice is that leaving a field empty could mean two different things:

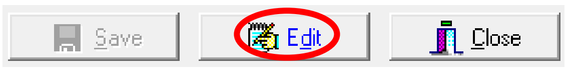
1) The field in the original data source was empty;

2) the data entry person forgot to enter the information.

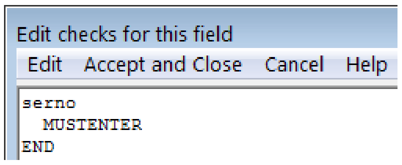
These two are obviously fundamentally different. To reduce the risk of the second, we make each field a MUSTENTER field. This can be done by clicking and selecting or with the short-cut key combination **CTRL+E**. For this exercise, there are no calculated fields, all will thus have to become MUSTENTER.

KEY UNIQUE

The most important variable in any dataset is a unique identifier for every record. The identifier must be unique because if it is not, it will not ever be possible to identify the record with certainty. The recommendation for the tuberculosis microscopy laboratory register is that the laboratory serial number is to start with 1 each calendar year and be strictly sequential throughout that year. It is thus a unique identifier for each record in a given year in a given laboratory. Often, identifiers are not that neatly sequential and it will then not be possible to easily ascertain whether the identifier had already been used earlier in the data set. We therefore need EpiData to check after entering the identifier (here the serial number) in every record whether it had been used before or not and if so to report where and to prevent the data entry person to continue to work before the problem is addressed. As you might have noticed, there is no fixed menu option to make a variable "unique"; thus, we have to write a command. At the bottom of the Check file box, you can see "Edit":



If you select "Edit" we get:



```
serno
  MUSTENTER
END
```

with the field `serno`, followed by the command `MUSTENTER` in the second line and finishing with the command `END`. We add here:

```
serno
  KEY UNIQUE 1
  MUSTENTER
END
```

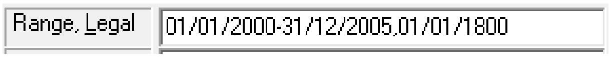
If we don't number the `KEY`, EpiData will do that. Using keys will make EpiData to automatically create an index file with the same name as the `*.QES`, `*.REC`, and `*.CHK` file, but with the extension `*.EIX` (**E**piData **I**nde**X** file). Index files are single-purpose files. In this case, the only purpose is to keep the information on all `SERNO`s ever used and to determine after a new one has been entered whether it had not been used before. If it has, it must raise an alert with information in which record it had been used. This single-purpose action makes index files very fast in search function and the user doesn't even notice that they do their job except if there is a problem.

RANGE, LEGAL

The legal registration dates are constrained to be allowed only from 1 January 2000 through 31 December 2005 plus a value of 1 January 1800 if the registration date is not recorded. The range is indicated by a hyphen separating the minimum and the maximum, and legal values follow, separated by commas, thus:

```
01/01/2000-31/12/2005,01/01/1800
```

Is entered into the appropriate space:

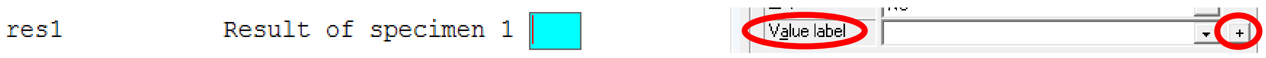


Range, Legal 01/01/2000-31/12/2005,01/01/1800

Note that you cannot define more than one range!

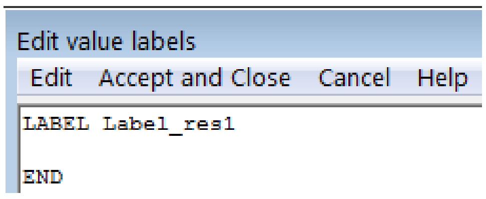
LABELBLOCK

The field `RES1` is a variable with results expressed categorically and has thus been defined as an integer field of length 1. If we look at the Check box:



```
res1          Result of specimen 1
```

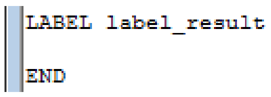
We see on the last line "`Value label`" and a plus sign at the end of the line. If you click on the plus sign you get:



```
LABEL Label_res1

END
```

EpiData has prepared the beginning and end of the label block for this field. The command `LABEL` is followed by a proposed Label name `Label_res1` for the label and the command finishes with an `END`, while the cursor is blinking in between, ready for you to write. The name for the label is a suggestion, and commonly it makes sense to just use it, but in this case, we will choose to modify it and will call it `label_result`:

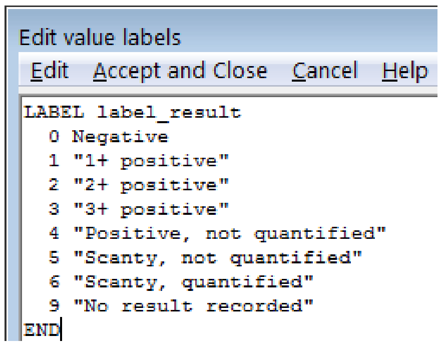


```
LABEL label_result

END
```

This is not necessary. It is on one hand just to show that you may modify the proposed name, but also because there are in the end 3 results for which we can use the same label. While it is not wrong, it might look a bit strange to use `label_res1` for `RES2` and `RES3`.

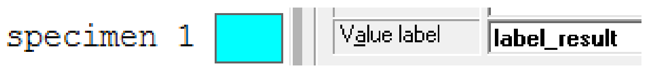
We now faithfully type into the space between `LABEL` and `END` exactly what we have defined in the data documentation sheet:



```
LABEL label_result
  0 Negative
  1 "1+ positive"
  2 "2+ positive"
  3 "3+ positive"
  4 "Positive, not quantified"
  5 "Scanty, not quantified"
  6 "Scanty, quantified"
  9 "No result recorded"
END
```

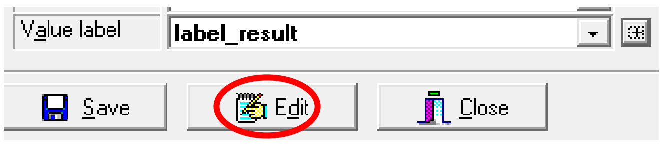
Note that `Negative` is not in quotation marks, but all other Value labels are. The rule is that a label consisting of more than one word (defined as a space character separating two strings) must be put into quotation marks. Single-word labels can but don't need to be placed in quotation marks.

We accept and close (**ALT+A**) and can see now the label that is being used for the field:

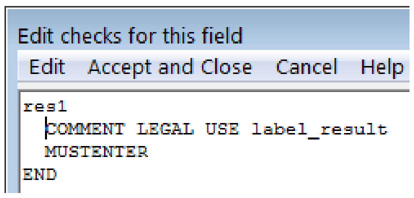


SHOW and TYPE COMMENT

Below the line Value label, we see the tab Edit:



Upon clicking, we get:



```
res1
  COMMENT LEGAL USE label_result
  MUSTENTER
END
```

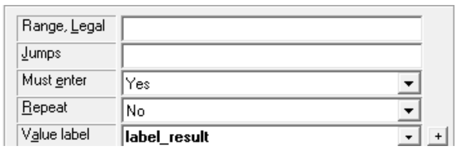
This is the actual paragraph written into the Check file. Let's take it apart.

The paragraph starts with the field name, `res1`, and finishes with `END`, and then there are two lines of commands in between. This is the principle for Checks for any field in EpiData: if there are any Check commands, then there is first the field name and it concludes with an `END`. If there are no Check commands, then nothing is written.

The **indent** has no functional value but it greatly facilitates a quick ascertainment where something (the field) begins and where it ends.

In between we have two lines of commands. Again, **commands are capitalized**, but the name of the label or the name of the field is not: lower case or capitals have no functional meaning: **EpiData is not case-sensitive**. It is for visual guidance only that EpiData capitalizes all commands but not names of fields and labels. It helps to quickly make distinctions. As a user you will not have to follow this convention, but it is actually tremendously useful and there is nothing wrong with adopting these conventions.

Now what do these commands mean? The first line has the command `COMMENT LEGAL USE` followed by the name of the label we just created before, while the label block is actually nowhere to be seen. We mentioned above that `RANGE` and `LEGAL` should not be used for categorical variables, and perhaps now it is becoming clearer why not: instead of legal values indicated in the line above:

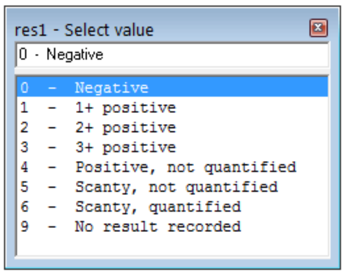


Range, Legal	
Jumps	
Must enter	Yes
Repeat	No
Value label	label_result

we use a label block to provide what is legal to use and this is expressed by this command `COMMENT LEGAL USE label_result`: it is legal to use the label block `label_result`. We cannot see the actual label block here: EpiData keeps all label blocks separately in one "super-block" at which we will look in a later exercise. Suffice it here to know that this command refers to it.

You recognize `MUSTENTER`: by choosing that all fields including the field `res1` are Mustenter fields, EpiData recorded this as a separate command on a second line.

The command `COMMENT LEGAL` … defines that no value other than what was defined in the label block is legal. This is not quite sufficient for our convenient use. What we want is that when entering the field `RES1`, a Label block pops up from which we can pick the pair of field value and field label as here:



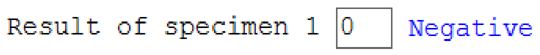
```
res1 - Select value
0 - Negative

0  -  Negative
1  -  1+ positive
2  -  2+ positive
3  -  3+ positive
4  -  Positive, not quantified
5  -  Scanty, not quantified
6  -  Scanty, quantified
9  -  No result recorded
```

The default of EpiData is not to show the label block. To invoke the labelblock during data entry and make it show up requires adding the command `SHOW` after the name of the label block:

```
COMMENT LEGAL USE label_result SHOW
```

We also wanted that the value label is actually (temporarily) written to the right of the field after entry:



```
Result of specimen 1 [0   ] Negative
```

This requires the additional command `TYPE COMMENT` written into a new line:

```
res1
```

```
  COMMENT LEGAL USE label_result SHOW
  TYPE COMMENT
  MUSTENTER
END
```

AFTER ENTRY

There is one more issue to be addressed. We have a second variable for each result, and that is a field to be completed only if the result is '`scanty, quantified`'. The value we give for a quantified scanty result in the field `RES1` is "`6`". In other words, only if the value in `RES1` is `6` should the cursor go into the field `RES1SC`. In any other case, it should enter the value `0` into the field `RES1SC`, skip the field and place the cursor into the field `RES2`. This is where the command `AFTER ENTRY` comes into play. This is a block command and *specifies a block of commands that are executed after a value has been entered in this field. This is a very useful command for making calculations in the* `CHK` *file.* `AFTER ENTRY` must be terminated with an `END` by syntax, and we thus have to make use of space between `AFTER ENTRY` and `END` to enter the commands to be executed:

```
res1
  COMMENT LEGAL USE label_result SHOW
  MUSTENTER
  TYPE COMMENT
  AFTER ENTRY
    Something goes in here
  END
END
```

What needs to be entered in between the `AFTER ENTRY … END` statements is conditions of the type "If this is the case, then do that". In EpiData, this is done with:

```
IF …. THEN
  Something to happen
ENDIF
```

We note here three things: first, the `IF` and `THEN` are on the same line; second, the thing or things that should happen are on another line, and third, an `IF` command finishes with an `ENDIF`. We thus get in extension:

```
res1
  COMMENT LEGAL USE label_result SHOW
  MUSTENTER
  TYPE COMMENT
  AFTER ENTRY
    IF … THEN
      xxxx
    ENDIF
  END
END
```

The "`xxxx`" doesn't need to be a single command, it can be multiple ones, each written into a separate line. Specifically, what we want is that if the value that was entered for the field `RES1` is not equal to `6`, then the value for the field `RES1SC` should become `0` and the cursor should move to the field `RES2`. This is formulated as (note the sequence for execution):

```
res1
  COMMENT LEGAL USE label_result SHOW
  MUSTENTER
  TYPE COMMENT
  AFTER ENTRY
    IF res1<>6 THEN
      res1sc=0
      GOTO res2
    ENDIF
  END
END
```

Note:

1. The additional command GOTO followed by the field name. GOTO can be used alternatively to JUMPS and we generally give it preference. It would actually not be that simple to formulate an efficient JUMPS command in this case. Try to use JUMPS command to set this up as an exercise.
2. When you have connecting conditions (two or more conditions connected by AND or OR), then it will be mandatory to use parentheses. More about this later.

If you are in the field RES3, there is only the field RES3SC to follow, thus the above GOTO command directing to another following field becomes inappropriate. EpiData has a command GOTO WRITE. After this command, the user gets the prompting whether the record should be written to disk. We can thus use:

```
res3
  COMMENT LEGAL USE label_result SHOW
  MUSTENTER
  TYPE COMMENT
  AFTER ENTRY
    IF res3<>6 THEN
      res3sc=0
      GOTO WRITE
    ENDIF
  END
END
```

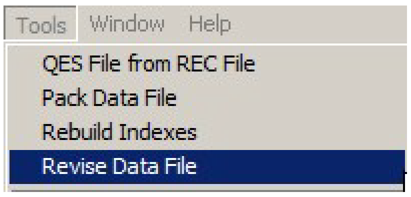
There is one final thing to be done. As has been mentioned in the data documentation sheet, if the value of the res1 is 6 (scanty, quantified), then it should not be possible to enter 0 (not applicable) in res1sc for obvious reasons. The same is the case for res2 and res3. If this happens, then a message should appear, "Values of res1 and res1sc are not compatible. Please verify" and the cursor should go back to res1. How do we achieve this? EpiData has a command HELP (which can be used to deliver messages to the person while doing data entry) after which one can write anything in quotation marks:

```
res1sc
  COMMENT LEGAL USE label_scanty SHOW
  MUSTENTER
  TYPE COMMENT
  AFTER ENTRY
    IF (res1=6) AND (res1sc=0) THEN
      HELP "Values of res1 and res1sc not compatible. Please verify"
      GOTO res1
    ENDIF
  END
END
```

Note: **Check commands added here can only be executed if and only if the cursor is in the field in question.** You can circumvent Check file commands for a field if you use the mouse to skip a few fields: the mouse should thus not be used for data entry. Actually, there is really no need for the mouse during data entry as one proceeds – and must proceed – from one field to the next which happens automatically or by confirming an entry with the Enter key. Nevertheless, people cannot be prevented from using the mouse and thus circumventing your carefully crafted Check file. For key variables, we will thus show in a later exercise how some control can still be retained.

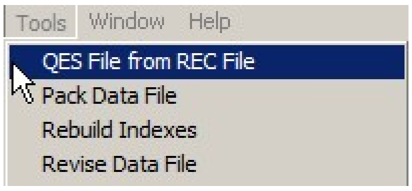
Now that you have learnt how to create QES, REC and CHK files, let us pause and reinforce some of the important principles of the EpiData triplet files.

1. The data structure is defined in the QES file. So, if you want to alter the structure of the database, that is, if **you want to add or delete fields**, change the type of the field, increase/decrease the length of the field, you have to do it in the QES file.
2. **Any change you make to the QES file requires that you update the REC file** with the changes you have made to the structure of the database. Many a time, you will forget this step and expect to see the revised REC file. Note the changes in the QES file are not automatically updated to the REC file – you have to update it. The usual way you have learnt to create a REC file, (**ALT+2** and clicking 'Make data file') though gets you an updated REC file, it will erase/overwrite the previously created file with an empty REC file. Hence, if there was any data entered previously, it will be lost. There is another way of revising the REC file without losing any data. This can be accomplished through 'Tools-Revise Data File'.

3. An even simpler method is just to open the REC file with 4. Enter Data: whenever you do that, EpiData checks whether there is a QES file of the same name in the same folder, and if there is, whether the date of the QES file is older or newer than that of the REC file. If it is newer, then obviously the QES file has been revised after last data entry. Then EpiData prompts you asking whether you want to adapt the REC file to the revised QES file. If you had made a non-lossy change (adding a field, making a field definition longer, or trivial changes like correcting a Field label) you can safely confirm. Should you have made a lossy change (removing a field, shortening a field definition), EpiData will alert you that you are about to loose data and where you will loose them.
4. Data entry is done in the REC file. It is the REC file which contains all the data.
5. Every time you add/delete the checks in the CHK file, check the functionality by entering some dummy data into the REC file to ensure that checks are actually functional.
6. The names of the QES, REC and CHK files have to be the **same** for them to be linked and work properly. You now realize the value of making the file extensions visible!
7. **Always place the triplet files together in the same folder.** Whenever you have to share the files, share all the three together. Since the QES file is integrated into the REC file, REC and CHK files are enough for functional data entry. However, if you

have to change the structure of the database, then you will need the QES file. Though there is an option to create a QES from the REC file, (look under Tools – QES file from REC file) it is a good practice to keep all the three files together, not least because a QES file created from a REC file may loose some precious formatting beauty.



Familiarize yourself with opening, editing and closing the three files.

Tasks:

- ***Open the existing*** `A_EX02.QES` ***file and complete it.***
- ***Create the*** `A_EX02.REC` ***file (overwrite the existing one).***
- ***Edit the*** `A_EX02.CHK` ***file, make all fields*** `MUSTENTER` ***fields***
- ***Make Range and legal and label blocks where appropriate, but not both for the same variable.***
- ***Edit the field to ensure that label blocks are shown and add other key commands as appropriate such as*** `AFTER ENTRY` ***with*** `IF … THEN … ENDIF` ***and*** `HELP` ***statements.***