|  |  |  |
| --- | --- | --- |
| **ACTIES OP HET SCHERM** | **VOICE-OVER** | **DUUR** |
| Start in empty terminal window | Say we want to implement a function that checks whether a given year is a leap year and use it on a couple of examples. |  |
| 1. open skeleton file in editor 2. write contains 3. write function-end function 4. write implicit none 5. write argument declaration 6. write result assignment 7. save file | 1. We don’t start quite from scratch. 2. We will discuss functions in more detail later, but lets write one now. Functions go into the program compilation unit, after a “contains” keyword. 3. The function will return a logical and its name is “is\_leap\_year”. It takes a single argument that represents a year. The function definition ends with “end function”. 4. Since a function is a compilation unit, we start by declaring “implicit none”. 5. Our first implementation is straightforward, the function will always return false. This is quite good actually, since the function will already work correctly for a little over 75 % of the years. The value the function is supposed to return is assigned to the function name. |  |
| 1. switch to terminal 2. compile file, use -o 3. run application | 1. When we compile... 2. And run this, we see that the output is correct for 1899 and 2000. |  |
| 1. Switch to editor 2. Add if statement for mod 4 3. Add end if 4. Add else 5. Save file | 1. Years that are divisible by 4 are leap years, so lets add a condition to that effect. Fortran has the intrinsic function “mod” that returns the remainder after division. So if “mod(year, 4)” is equal to zero, the year is divisible by 4. 2. An if-statement is closed by “end if”. 3. So if the year is divisible by 4, the function should return true, else it should return false. |  |
| 1. Switch to terminal 2. Compile file 3. Run application | 1. Great, now the result is correct for 1899 and 1936, but not for 1900 and 2000. |  |
| 1. Switch to editor 2. Add if statements for mod 100 and mod 400 3. Save file | 1. Lets add some more conditions. A year is not a leap year if it is divisible by 100, but not by 400. As you can see, nested conditions are straightforward and if you use proper indentation of your code, it is always clear which if and “else” or and “end if” belongs to. |  |
| 1. Switch to terminal 2. Compile file 3. Run application | 1. Now it works perfectly. |  |
| 1. Switch to editor 2. Add if for 1582, add extra case 3. Save file | 1. If-statements do not necessarily have an “else” part. For instance, lets add a check whether the Gregorian calendar was actually introduced, because there were no leap years before that. We will just print a warning. |  |
| 1. Switch to terminal 2. Compile file 3. Run applicatoin | 1. Now we get a warning for 1502. |  |
|  | Of course this function could have been implemented without the nested if-statement by using a logical expression, but that makes for an exercise if you’ve not done that before.  Also, we will have to discuss proper warning messages and error handling later on. And as I already mentioned, there is more to discuss about functions. |  |
| **TOTALE DUUR** | | *Maak je screencast niet langer dan ca. 6 minuten.* |