|  |  |  |
| --- | --- | --- |
| **ACTIES OP HET SCHERM** | **VOICE-OVER** | **DUUR** |
|  | The compiler is your friend when you want to prevent bugs in your code. However, it is important to be explicit about the semantics so the compiler has more information and help catch potential issues. |  |
| 1. Show start situation 2. Add assignment to year 3. Save file | 1. We have the function “is\_leap\_year” that we discussed previously. We use it to determine whether some years stored in an array are leap years. When we are done, we also print the array containing the years. 2. Now suppose we have a bug in our code. By accident, the value of the function argument “year” is modified. |  |
| 1. Switch to terminal 2. Compile code 3. Run application | 1. When we compile and run the code, we get the expected output for the for the tests of the years, but the array “years” no longer contains the original data. |  |
| 1. Add intent(in) to argument year | 1. Giving the compiler a little extra information would catch this bug. Our intension is to use the value passed to the function as an argument, but not to modify it. This can be formalized by declaring the intent of the argument. In this case, the intent would be “in” since we want to use the value passed to the function, not modify it. |  |
| 1. Switch to terminal 2. Compile code | 1. When we compile an error is shown. We assign a value to the argument year that was declared as intent “in”. |  |
| 1. Add error argument 2. Add argument to function call and check | 1. Suppose we add an argument “error” of type “logical” that we set to false when the year is after 1582, and to true otherwise. This would give the user of our function extra feedback, because it only makes sense to check for leap years after the introduction of the Gregorian calendar. Whatever the value of the argument that is passed to it, we do not really care since we will not use it in the function “is\_leap\_year”. Hence we declare the intent of the “error” argument “out”. 2. Of course, we also have to adapt the function call and check the result in the program unit. |  |
| 1. Switch to terminal 2. Compile source code 3. Run application | 1. Again, compiling and running shows the expected output |  |
| 1. Add intent(inout) | 1. So the intent of a procedure argument can be “in” and “out”, but also “inout”. To illustrate its use, consider the subroutine “norm” that will normalize an array so that the sum of its arguments is 1. 2. The normalization requires the values of the array’s elements passed as the argument to the subroutine, and it will update them appropriately. |  |
|  | Although strictly speaking it is not required to specify the intent of procedure arguments, it will help the compiler spot potential errors in your code, and may even help optimize your code better. |  |
| **TOTALE DUUR** | | *Maak je screencast niet langer dan ca. 6 minuten.* |