Escuela Universitaria de Ingeniería Vitoria-Gasteiz
Ingeniaritzako Unibertsitate Eskola Vitoria-Gasteiz

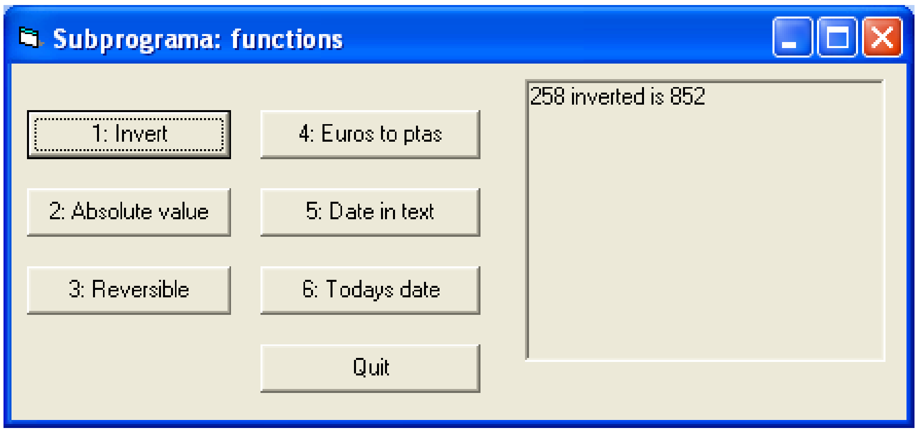
Fundamentals of Computer Science
2010-2011
Laboratory 8
Subprograms: functions

eman ta zabal zazu

Objectives:
- **Flowchart design** with **functions**
- **Implementation** of **functions** in Visual Basic
- Call to own and system functions

Program to demonstrate the use functions

Interface



258 inverted is 852

Figure 8.1 Objects present in the interface: command buttons and picture box

Operation

1. Each exercise has its own execution command button (**`cmdEj1`**, **`cmdEj2`**, ..., **`cmdEj6`**).
2. First thing after clicking one button will be removing the contents of the results picture box, **`pctRes`**. To do so we use the **`Cls`** method (`pctRes.`**`Cls`**).
3. When we click on the Quit button the program will finish.
4. An executable is provided to clarify the statements.

Exercise 8.1: Invert a number (solved)

Operation

We call **`cmdEx1`** to the command button associated to exercise 1. When the user clicks on this exercise labeled “1. Invert”, the program asks for a positive number not ending in 0 using *InputBox* and show the same number inverted in the picture box using **`Print`** (`pctRes.`**`Print`**), as exemplified in Figure 8.1.

Flowchart

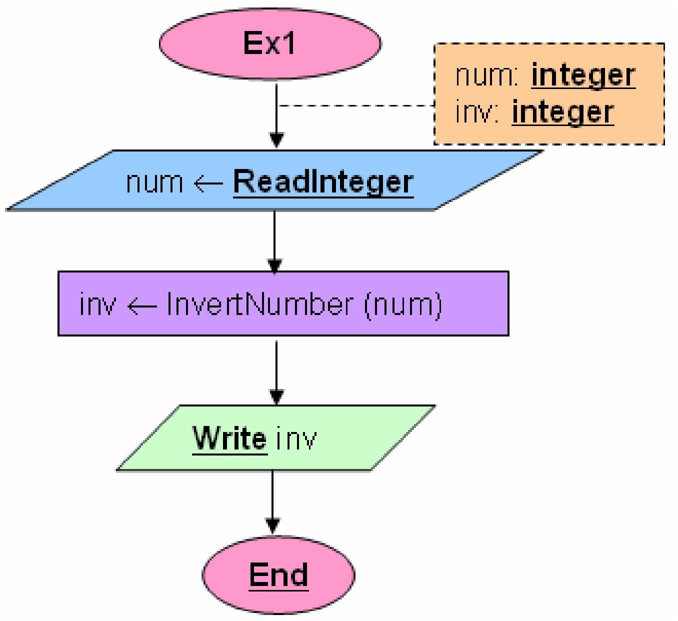


Figure 8.2 Flowchart of exercise 1

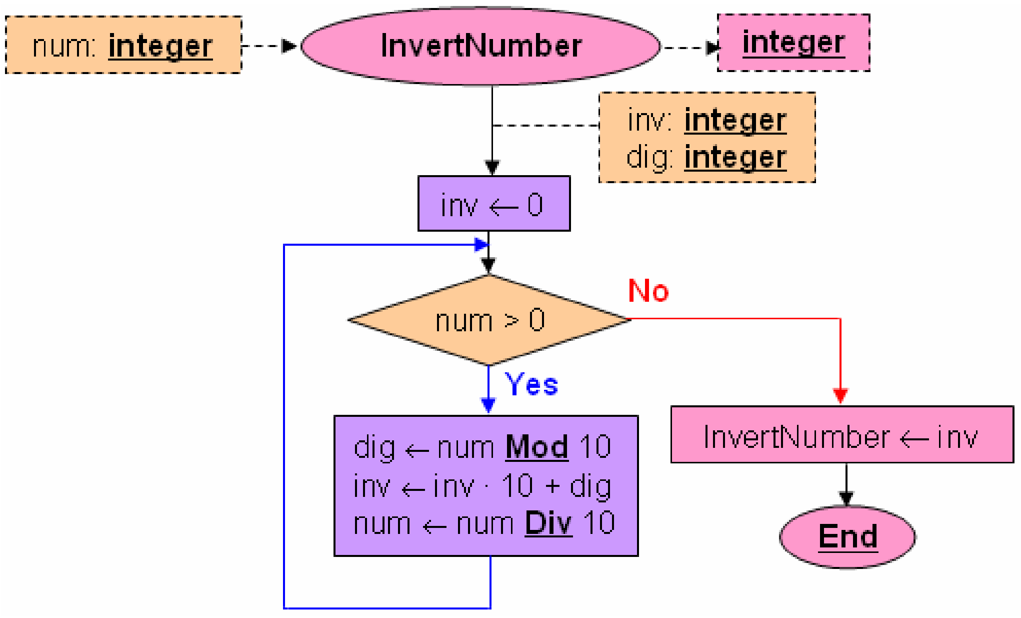


Figure 8.3 Flowchart of InvertNumber in exercise 1

Steps

1. We create the objects as in Figure 6.1. Save everything.
2. Add the code associated to the events. The code associated to the first exercise is shown in Figure 8.4.

- **Code for button "1: Invert"**: this is the procedure or subprogram associated to the click event, as we have been doing in previous exercises. Note that it calls `InvertNumber` function.

```
Sub cmdEj1_Click()
  Dim s As String
  Dim num As Integer
  Dim inv As Integer

  pctRes.Cls
  s = InputBox("Introduce a positive number not finished in 0")
  num = CInt (s)
  inv = InvertNumber (num)
  pctRes.Print CStr (num) & " inverted is " & CStr (inv)
End Sub
```

Figure 8.4 Code for the "1. Invert" button with a function call.

We will also need to define (just after this subprogram) the new function:

```
Function InvertNumber(ByVal num As Integer) As Integer
  Dim dig As Integer
  Dim inv As Integer

  inv = 0
  While num > 0
    dig = num Mod 10
    inv = inv * 10 + dig
    num = num \ 10
  Wend
  InvertNumber = inv
End Function
```

Figure 8.5 Code for function `InvertNumber`.

2. **Design** the flowchart and **implement** the VB program to read a number and calculate its absolute value, showing the result in a picture box. **Design** and use for it the `AbsVal` function to calculate the absolute value, using the header shown in Figure 8.6.

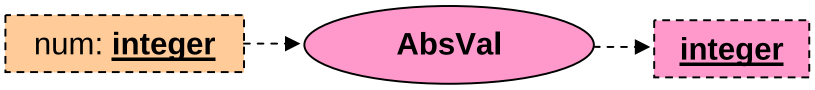


Figure 8.6 Header for the function to calculate the absolute value

3. **Design** the flowchart and **implement** the VB program to read a number and calculate if the number is reversible, that is, it does not change when inverted. **Design** and use for it the `IsReversible` function using the header shown in Figure 8.7, which calls the `InvertNumber` function seen in exercise 8.1.

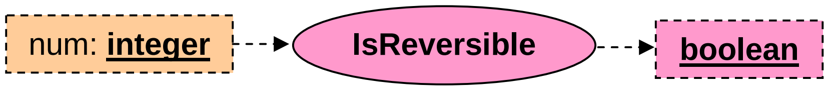


Figure 8.7 Header for the function to say if a number is reversible

4. **Design** the flowchart and **implement** the VB program to read a quantity in euros and show its corresponding value in pesetas, knowing that 1 € correspond with 166.386 pesetas. **Design** and use for it the `EurosPtas` using the header shown in Figure 8.8.

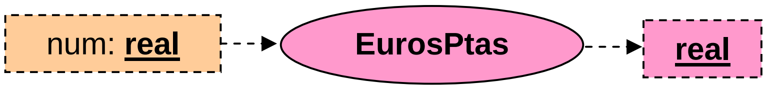


Figure 8.8 Header for the function to convert euros to pesetas

5. **Design** the flowchart and **implement** the VB program to read a day, month and year (using calls to **`InputBox`**) and show the string for the date using the following format: “22nd of April 2009”. It will obtain the string using the `DateString` function to be designed using the header shown in Figure 8.9. In order to obtain the string for the month it will use a specific `MonthString` function to be designed using the header shown in Figure 8.10. This function does not verify if the day, month and year correspond to a correct date, so that it may return a string “42nd of no-month -123”. Use another function `DaySuffix` to add the suffix (“st”, “nd”, “rd” or “th”) to the day.

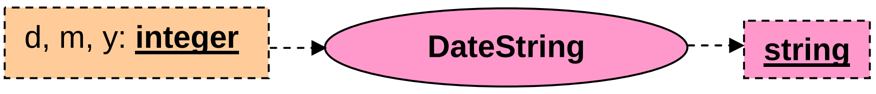


Figure 8.9 Header for the function to obtain the string for a date

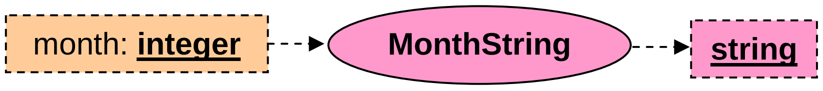


Figure 8.10 Header for the function to obtain the string for a month

6. **Design** the flowchart and **implement** the VB program to show the system date. Design and use a function `TodayStr` with the header given in Figure 8.11. It will make use of the system functions given in Table 8.1 as required.

More in detail, you must declare a variable called `today` of type **`Date`**, which will be initialized with the system date by calling the **`Date`** function (note that type and function have the same name). This variable `today` will be the input to functions **`Day`**, **`Month`** y **`Year`** to obtain day, month and year respectively in numeric format. With these we can call directly the DateString developed in the previous exercise.

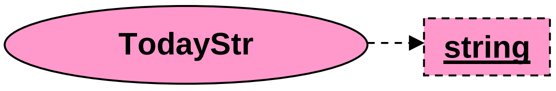


Figure 8.11 Header for the function to obtain the string for the system date

Quick reference table

Function	Description
`Date () As Date`	Current system date dd/mm/yyyy
`Day (ByVal dat As Date) As Integer`	Day (dd) of a date
`Month (ByVal dat As Date) As Integer`	Month (mm) of a date
`Year (ByVal dat As Date) As Integer`	Year (yyyy) of a date

Table 8.1 List of date functions in Visual Basic.