Variables and Constants:

1. Declaration of Variable

|  |  |
| --- | --- |
| Pseudocode | Program Code |
| DECLARE Value1 AS INTEGER  DECLARE Sum AS REAL  DECALRE name AS STRING | Sub Main()  Dim Value1 as Integer  Dim Sum as Double  Dim name as String  End Sub |

1. Declaration of Constants (you do declaration and assignment in same line)

|  |  |
| --- | --- |
| Pseudocode | Program Code |
| CONSTANT pi ← 3.142 | Sub Main()  Const pi As Double = 3.142  End Sub |

1. Initialisation/Assignment

|  |  |
| --- | --- |
| Pseudocode | Program Code |
| Value1 🡨6  Sum 🡨0  Sum🡨Sum+Value1  Name 🡨”Harris” | Value1=6  Sum=0  Sum=Sum+Value1  name="Harris" |

1. Input

|  |  |
| --- | --- |
| Pseudocode | Program Code |
| DECLARE num AS INTEGER  OUTPUT “Input Number: “  INPUT num | dim num as Integer  Console.Write("Input number: ")  num=console.readline() |
|  |  |

1. Output

|  |  |
| --- | --- |
| Pseudocode | Program Code |
| OUTPUT “myvalue” | Moves to next line:  console.writeline("The number is " & num)  or  In same line:  console.write(“The number”) |

Selection (If statements)

1. IF….THEN

|  |  |
| --- | --- |
| Pseudocode | Program Code |
| IF num=5 THEN  OUTPUT “Value is 5”  END IF | if num=5 Then  console.writeline("Value is 5")  end if |

1. IF….ELSE IF….ELSE…..END IF

|  |  |
| --- | --- |
| Pseudocode | Program Code |
| IF num=5 THEN  OUTPUT "Value is 5"  ELSE  IF num=6 THEN  OUTPUT "Value is 6"  ELSE  OUTPUT "Other number"  END IF  END IF | if num=5 Then  console.writeline("Value is 5")  else  if num=6 Then  console.writeline("Value is 6")  else  console.writeline("Other number")  end if  end if |

Possibilities:

* If then end if
* If then else end if
* If then else if then end if end if
* If then else if then end if else end if

You can nest as much as you want.

1. CASE OF

|  |  |
| --- | --- |
| Pseudocode | Program Code |
| DECLARE grade AS CHAR  OUTPUT “what is your grade”  INPUT grade    CASE OF grade  “A”,”a”: OUTPUT “Excellent”  “B”,”C”: OUTPUT “Well done”  “D”: OUTPUT “You passed”  “F”: OUTPUT “Better try again”  OTHERWISE: OUTPUT “Invalid grade”  ENDCASE | Dim grade As Char  dim i as integer  i=0    console.write("Enter student grade: ")  grade=console.readline()      Select grade  Case "A", "a"  Console.WriteLine("Excellent!")  ‘This is just for demonstration  do  console.writeline("hello")  i=i+1  Loop Until i=10    Case "B", "C"  Console.WriteLine("Well done")  Case "D"  Console.WriteLine("You passed")  Case "F"  Console.WriteLine("Better try again")  Case Else  Console.WriteLine("Invalid grade")  End Select    Console.WriteLine("Your grade is {0}", grade)  Console.ReadLine()  End Sub |

Repetition (Loops)

1. Repeat…..Until (It is post condition, it runs atleast once), It is like a bouncer inside the bar.

|  |  |
| --- | --- |
| Pseudocode | Program Code |
| DECLARE I as INTEGER  I🡨0  REPEAT  OUTPUT “hello”  I🡨I+1  UNTIL I=10 | dim i as integer  i=0    do  console.writeline("hello")  i=i+1  Loop Until i=10 |

1. While…End while (This runs 0 or more times), this is like a security guard in front of the bar

|  |  |
| --- | --- |
| Pseudocode | Program Code |
| DECLARE I as INTEGER  I🡨0  WHILE I<10  OUTPUT “hello”  I🡨I+1  END WHILE | dim i as integer  i=0    do While i<10  console.writeline("hello")  i=i+1  Loop |

1. For… End for (remember it tends to be inclusive)

|  |  |
| --- | --- |
| Pseudocode | Program Code |
| DECLARE a AS INTEGER  FOR a 🡨1 TO 11  OUTPUT “Value of a:”, a  Next a  FOR a 🡨0 TO 10 STEP 2  OUTPUT “Value of a:”, a  Next a | Dim a As Integer  ' for loop execution    For a = 1 To 11  Console.WriteLine("value of a: ” & a)  Next      ' Step is used for determining the jump, default in for loop is step 1    For a = 0 To 10 step 2  Console.WriteLine("value of a: ” & a)  Next |

MOD and DIV

MOD gives remainder

DIV gives quotient

For example:

10 MOD 2 =0

5 DIV 1= 5

To check if something, say x is divisible by n, do

If X mod n=0 Then

OUTPUT “it is divisible”

ELSE

OUTPUT “it is not divisible”

END IF

For example, finding out if a number is even or not

DIV is similar to doing regular division and then rounding down the answer. It is best used for example if question is like your physics fringe question: “How many fringes can be seen on a screen during diffraction grating”

Notation for div in vb.net is: \

|  |  |
| --- | --- |
| Pseudocode | Program Code |
| If X mod n=0 Then  OUTPUT “it is divisible”  ELSE  OUTPUT “it is not divisible”  END IF | Dim x As Integer    console.writeline("What number do you want to know is divisible by 23")  x=console.readline()    if x mod 23 = 0 Then  console.writeline("It is divisible by 23")  console.writeline("The quotient is: " & x \ 23)  Else  console.writeline("It is not divisible by 23")  console.writeline("The quotient is: " & x \ 23)  console.writeline("The remainder is: " & x mod 23)  End if |

|  |  |
| --- | --- |
| Pseudocode | Program Code |
| DECLARE X,Y as INTEGER  X🡨10  Y🡨3  OUTPUT “X DIV Y”  (Ans is 3 here) | Example of div is given above, it is used to find quotient |

FUNCTIONS AND PROCEDURES

BYVAL: It is used when you don’t want the value in main to change (used most of the time)

BYREF: It is used when you want the value in main to change

PROCEDURE

|  |  |
| --- | --- |
| Pseudocode | Program Code |
|  |  |

FUNCTION

|  |  |
| --- | --- |
| Pseudocode | Program Code |
|  |  |

Function can be used directly, or it can be assigned to a variable and changed or printed or whatever you want to do with it

Procedure requires explicit “CALL” in Main

Example:



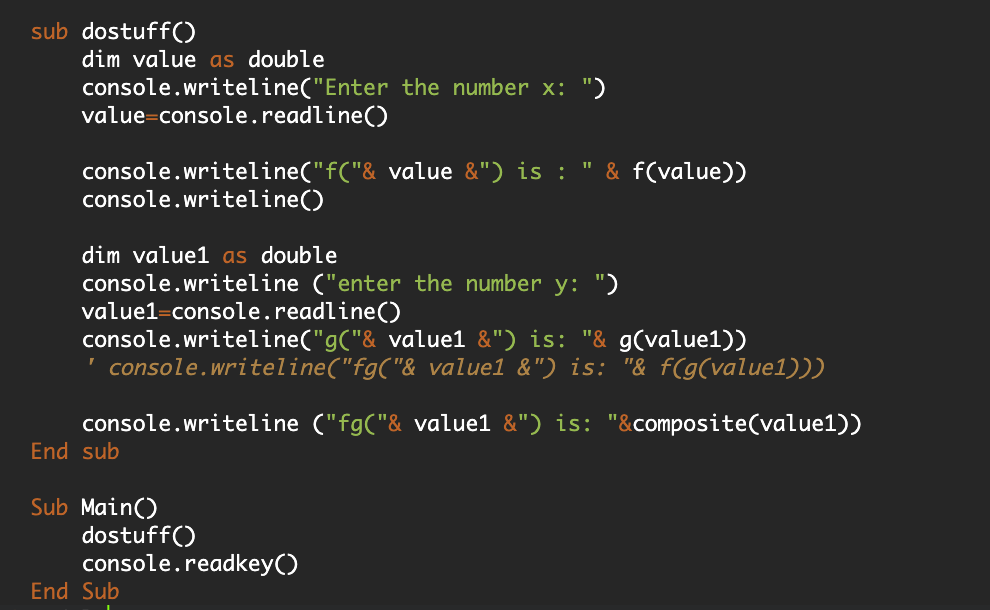
A picture containing graphical user interface

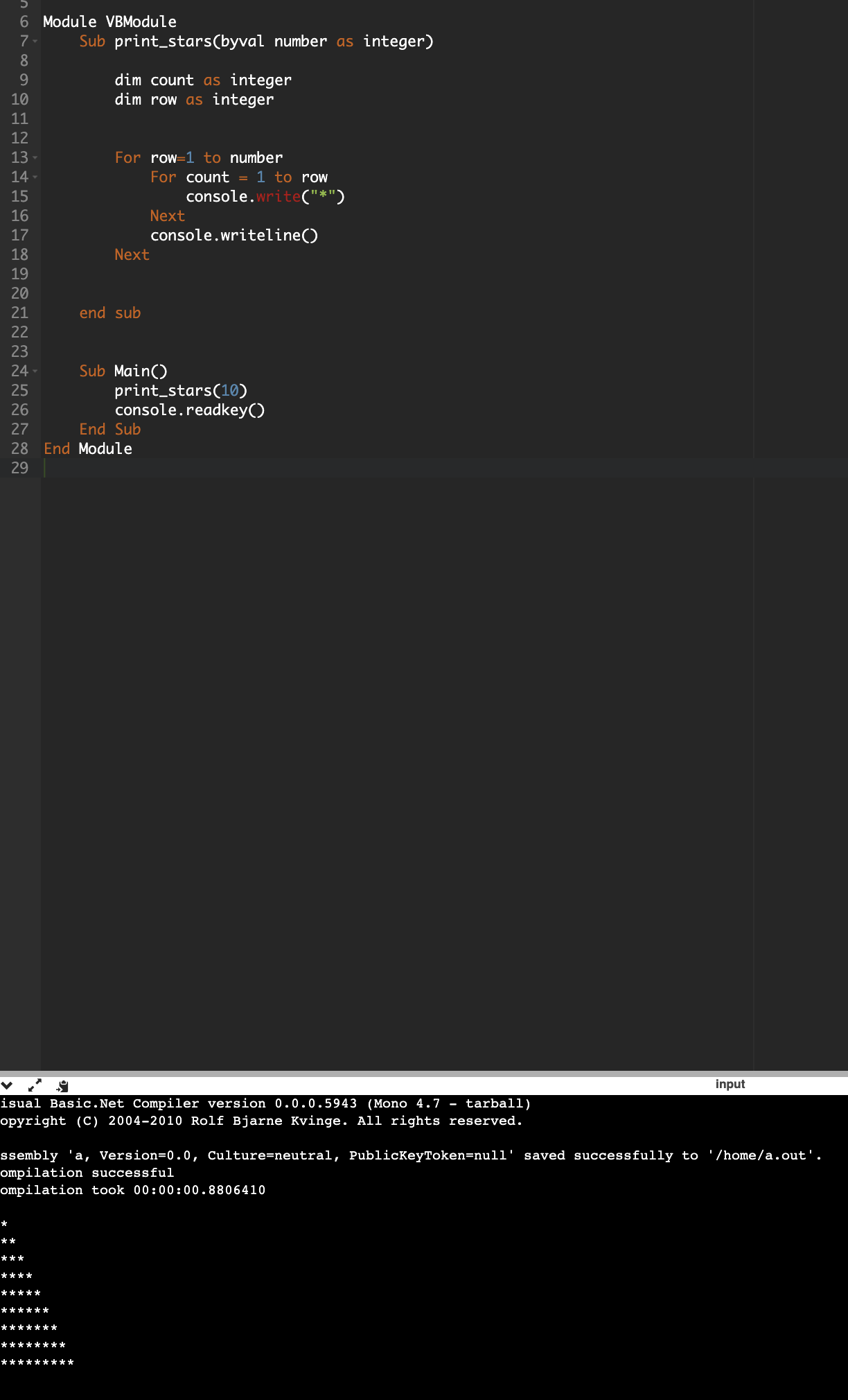
Description automatically generated

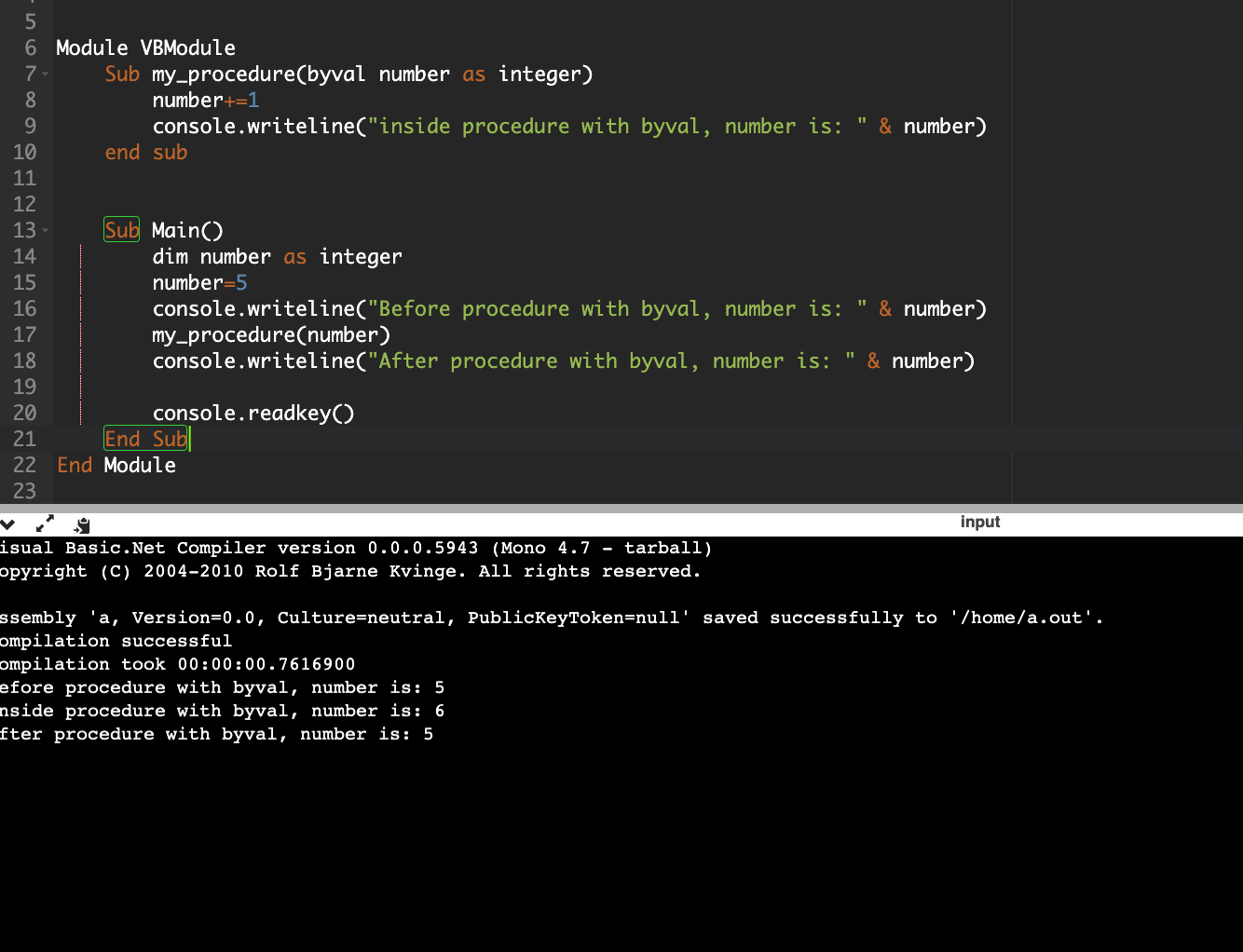
c) Find fg(-2)

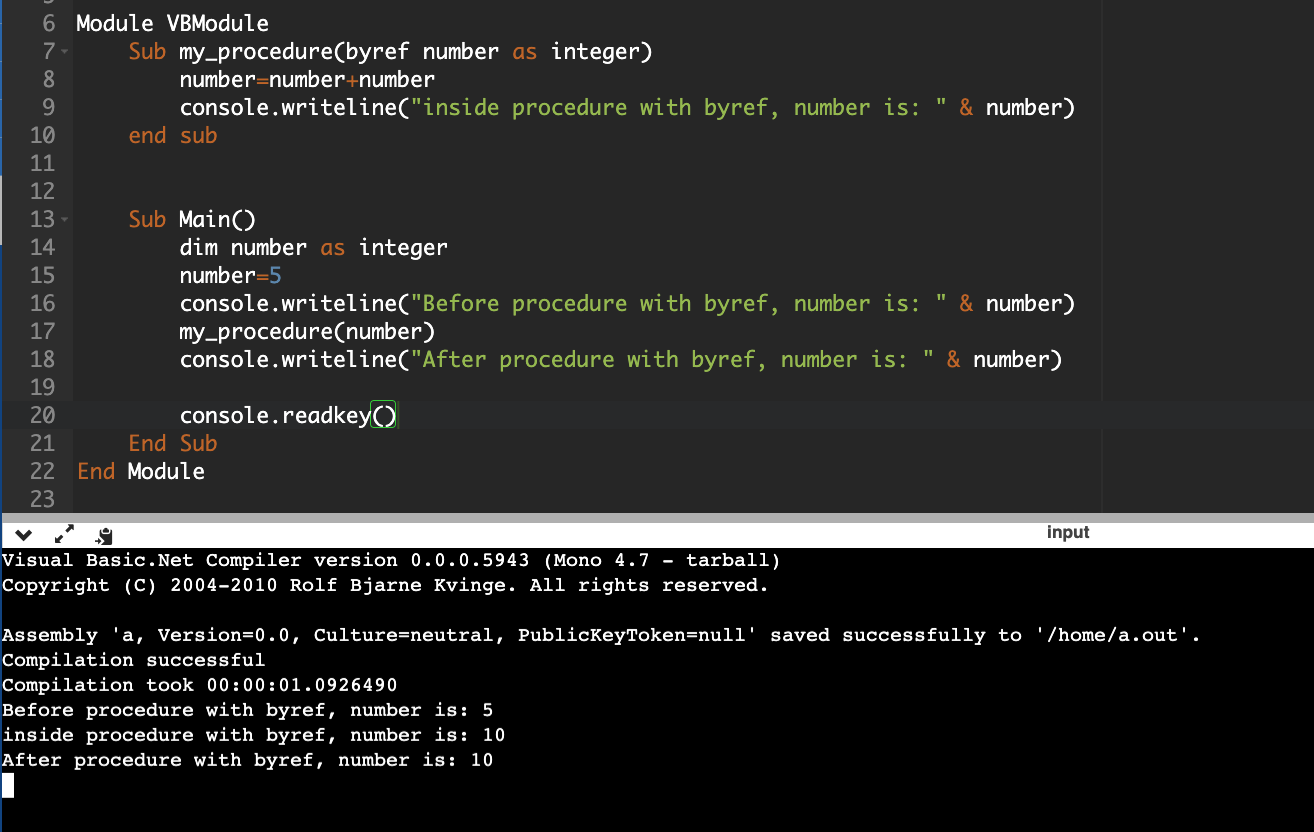
First make code for functions for f and g. Then find the values of the above.



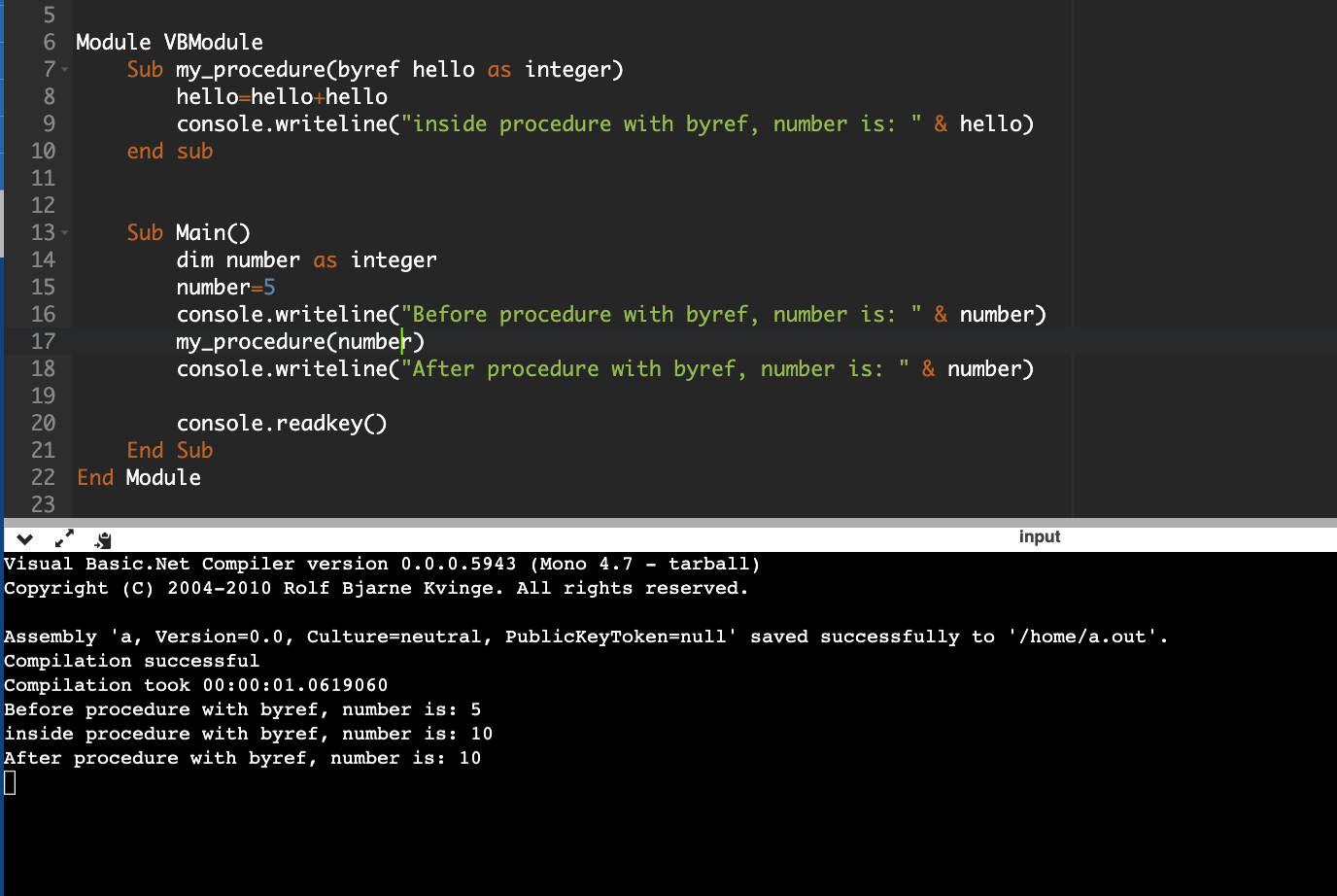








Changing variable name in procedure doesn’t matter, it maps the variable in main to the variable in procedure



1D Array: refer to the menu codes with procedure and without procedure. They have everything.

2D array:

Pseudocode:

Declaration:

DECLARE myArray : ARRAY[0:8,0:2] OF INTEGER

Assignment:

myArray[7,0] ← 16

For the program code, look at the 2D array files in the folder sent.

Files:

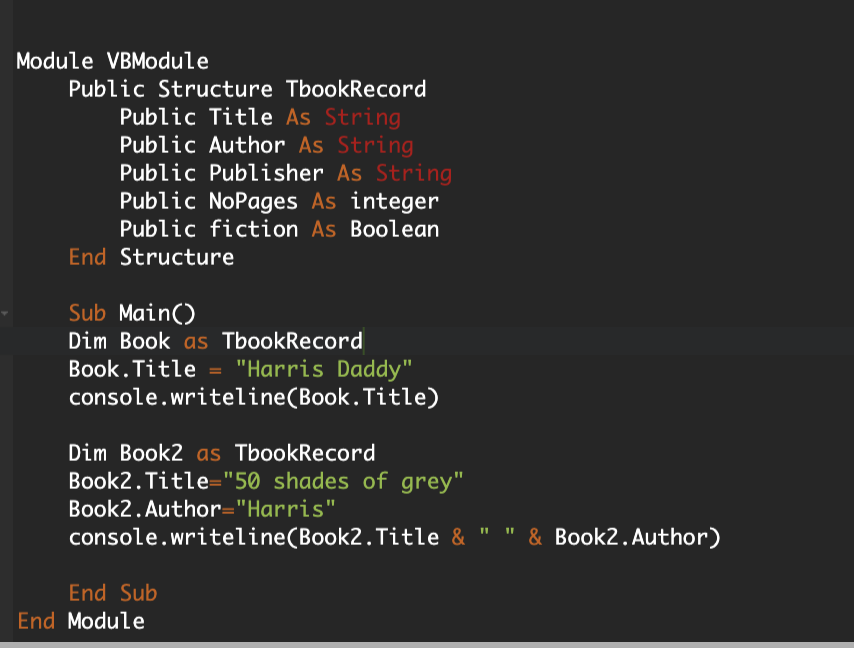
For files refer to the Files pdf

Type End Type (Structures)

They store a “record” related to one person, thing etc.

So it is one variable that has multiple information about a single thing.

|  |  |
| --- | --- |
| Pseudocode |  |
|  |  |
| Program code |  |
| Public Structure TbookRecord  Public Title As String  Public Author As String  Public Publisher As String  Public NoPages As integer  Public fiction As Boolean  End Structure |  |



Same thing but with OOP

