**LEZIONE 1 (28\_9\_2020)**

**Variabili e tipi**

*Fare disegnino su VISIO.*

Tipi principali:

* Boolean 2 byte (True False)
* Byte 1 byte (0 – 256)
* Integer 4 byte (interi con segno)
* Long 8 byte (interi con segno)
* Single 4 byte (reali con segno)
* Double 4 byte (reali con segno)
* String 10 byte
* Object 4 byte
* Variant 16 byte
* Date 8 byte
* Currency 8 byte
* Array (vettori, matrici e versori)
* Tipi definiti dall’utente

**Operatori**

* + , - , \*, /, \, ^
* &
* AND, OR

**Procedure e funzioni**

* Similitudini e differenze
* Dove si scrivono? Crezione di un Modulo
* Funzioni da finestra immediata
	+ Left("Pippo", 2)
	+ Rnd()
	+ Int()
	+ …
* Esempi di funzioni di base, es. moltiplicazione, concatena stringhe
* Funzione Pi.Greco una funzione senza input
* Scrivere una procedura che fa le stesse cose, non restituisce output, ma scrive tutto a video. O su Excel (vedi sotto)

Esempio

Public Sub Print\_Out(Name As String, Surname As String)

 Dim To\_Prin As String

 To\_Print = "My Name Is " & Name

 Debug.Print To\_Print

 Application.Worksheets(1).Range("A1").Value = To\_Print

 To\_Print = "My Surname Is " & Surname

 Debug.Print To\_Print

 Application.Worksheets(1).Range("A2").Value = To\_Print

End Sub

* Esempio di Macro

Public Sub Hallo()

 ActiveCell.Value = "Hallo!"

End Sub

Osservare cosa succede nel caso dell’esempio successivo

Public Function Prova() As Integer

Dim A As Integer, B As Integer

A = 10

B = A

B = 5

Debug.Print B

Debug.Print A

End Function

**LEZIONE 2 (2\_10\_2020)**

Funzione Resto

Public Function Resto(Dividendo As Double, Divisore As Double) As Integer

 Resto = Dividendo - (Dividendo \ Divisore) \* Divisore

End Function

Public Function Resto2(Dividendo As Double, Divisore As Double) As Integer

 Resto2 = Dividendo - Int(Dividendo / Divisore) \* Divisore

End Function

**Variabili implicite ed interne ad una funzione**

Creare una variabile globale e poi provare le seguenti funzioni

Public Pip As Integer

Public Sub Crea\_Pip()

 Pip = 0

End Sub

Public Function Inc\_Pippo() As Integer

~~Dim Pip As Integer~~ ‘Provare con e senza quest’istruzione

Pip = Pip + 1

Inc\_Pippo = Pip

End Function

**ByRef ByVal**

Public Sub Reference\_Example()

Dim My\_Var As Integer, New\_Var As Integer

 My\_Var = 10

 New\_Var = Do\_Not\_Change(My\_Var)

 Debug.Print New\_Var

 Debug.Print My\_Var

 New\_Var = Change(My\_Var)

 Debug.Print New\_Var

 Debug.Print My\_Var

End Sub

Private Function Change(ByRef N As Integer) As Integer

 N = N \* 2

 Change = N

End Function

Private Function Do\_Not\_Change(ByVal M As Integer) As Integer

 M = M \* 2

 Do\_Not\_Change = M

End Function

Esempio non fatto in classe

Private Function Do\_ChangeL(ByRef S As String) As String

 S = Left(S, 2)

 Do\_ChangeL = S

End Function

Private Function Do\_ChangeR(ByRef S As String) As String

 S = Right(S, 2)

 Do\_ChangeR = S

End Function

Public Sub Prova\_Change()

Dim S As String, S1 As String

 S = "abcdefgjhiklmnopqrstuvwxyz"

 S1 = S

 Debug.Print Do\_ChangeL(S) & " " & Do\_ChangeR(S1)

End Sub

*Cosa si ottiene a video?*

In questo caso il risultato sarà “ad ad” come mai?

Entrambe le funzioni ricevono l’argomento byref, quindi lo modificano anche nello scopo esterno. In questo caso la chiamata è fatta nella prima riga, per cui prima viene eseguita DoChangeL e solo dopo Do\_ChangeR. Allora la prima restituisce “ab” e modifica in “ab” anche la variabile S. La variabili S così modificata viene passata alla DoChangeL che restituisce gli ultimi due caratteri di una stringa. Dato che S ha solo due caratteri, viene restituita S. In definitiva si ottiene “ab ab”.

**Variabili Opzionali -** funzione elevamento a potenza, default 2

Public Sub Potenza(X As Integer, Optional Y As Integer = 2 ) As Long

 Potenza = X^Y

End Sub

**Funzioni su stringa**

* LEN
* LEFT
* RIGHT
* MID

**Type Casting**

* CStr
* Cint
* CDbl
* …

**Istruzioni condizionate**

* **IF THEN ELSE**
* **SELECT CASE**

**CICLI**

* **For**
* **Do Loop**

Esempio 1 – Funzione Fattoriale

Public Function Fattoriale(Num As Integer) As Single ‘Perchè metto As Single?!?

Dim I As Integer

 If Num <= 1 Then

 Fattoriale = 1

 Else

 Fattoriale = Num

 End If

 For I = (Num – 1) To 1 Step -1

 Fattoriale = Fattoriale \* i

 Next i

End Function

Esempio 2 – Cambio di base da base inferiore a 10 a base 10

Spiegare il funzionamento logico e perché serve un ciclo Loop.

Public Function CB\_10(N As Integer, Optional B = 2) As Integer

Dim i As Integer, Counter As Integer

Dim S As String, Ch As String

 CB\_10 = 0

 If B > 9 Or B <= 1 Then Exit Function

 Counter = 0

 S = CStr(N)

 Do While Len(S) > 0 'Qui leggiamo la lunghezza

 Ch = Right(S, 1) 'Si parte sempre dall'ultima cifra, quella più a destra

 If Int(Ch) >= B Then

 CB\_10 = 0

 Exit Function

 End If

 CB\_10 = CB\_10 + Int(Ch) \* B ^ Counter 'La si eleva

 S = Mid(S, 1, Len(S) - 1) 'Si modifica la stringa

 Counter = Counter + 1

 Loop

End Function

Con For sarebbe possibile?

Public Function CB\_10(N As Integer, Optional B = 2) As Integer

Dim i As Integer

Dim S As String, Ch As String

 CB\_10 = 0

 S = CStr(N)

For i = 1 To Len(S)

Ch = Mid(S,1,i)

CB\_10 = CB\_10 + Int(Ch) \* B ^ (Len(L) – i)

Next i

End Function

**Tipi ENUM e TIPI Personalizzati**

Esempio 1 (il tipo enum va messo in un modulo a parte)

Enum Operatore

 Addizione = 1

 Sottrazione = 2

 Moltiplicazione = 3

 Divisione = 4

 Divisione\_Intera = 5

 Potenza = 6

End Enum

Public Function Operazione(A As Double, B As Double, Optional Op As Operatore = Addizione) As Double

 Select Case Op

 Case 1

 Operazione = A + B

 Case 2

 Operazione = A - B

 Case 3

 Operazione = A \* B

 Case 4

 Operazione = A / B

 Case 5

 Operazione = A \ B

 Case 6

 Operazione = A ^ B

 End Select

End Function

Esempio 2

Enum Prova

 A = 1

 B = 2

 c = 3

End Enum

Public Function Use\_Enum(E As Prova) As Prova

 Use\_Enum = E

End Function

Public Function Use\_Enum2(E As Prova) As String

 Select Case E

 Case A

 Use\_Enum2 = "A"

 Case B

 Use\_Enum2 = "B"

 Case c

 Use\_Enum2 = "C"

 End Select

End Function

**LEZIONE 3 (5\_10\_2020)**

Esempio 3 - ROULETTE WHEEL

*Modulo Type*

Enum Num\_Str

 Stay = 1

 Casual = 2

 Last = 3

End Enum

Enum Bet\_Str

 Cautious = 1

 Wise = 2

 Random = 3

 Random\_Increase = 4

End Enum

Type Wh\_Game

 Wh\_Numbers As Integer

 N\_St As Num\_Str

 B\_St As Bet\_Str

 Min\_Bet As Long

 Min\_Win As Long

 Max\_Loss As Long

 Max\_Play As Long

 Number As Integer

 Bet As Long

 Win As Long

 Tot\_Win As Long

 Tot\_Loss As Long

 Tot\_Bet As Long

End Type

*Modulo principale*

Public Sub Main\_Wh(B As Bet\_Str, St As Num\_Str, Optional n As Integer = 37, \_

 Optional MB = 10, Optional MW = 1000, Optional ML = 1000000, \_

 Optional MP = 100, Optional Show As Boolean = True, Optional Random\_Sequence = -1)

Dim G As Wh\_Game

 G.Wh\_Numbers = n

 G.N\_St = St

 G.B\_St = B

 G.Max\_Play = MP

 G.Min\_Bet = MB

 G.Min\_Win = MW

 G.Max\_Loss = ML

 G.Number = Rand\_From(0, CLng(n))

 G.Bet = MB

 'To generate random sequence

 If Random\_Sequence <> -1 Then

 r = rnd(-1)

 Randomize (Random\_Sequence)

 Else

 Randomize

 End If

 Call Gumble(G, Show)

 Debug.Print "WIN: " & G.Tot\_Win

 Debug.Print "LOSS: " & G.Tot\_Loss

 Debug.Print "BET: " & G.Tot\_Bet

End Sub

Private Function Rand\_From(Low As Long, High As Long) As Long

Dim Rg As Long

Dim Rn As Double

 Rg = High - Low

 Rn = rnd() \* (Rg + 1) + Low

 Rand\_From = Int(Rn)

End Function

Private Function Choose\_Bet(G As Wh\_Game) As Long

 Select Case G.B\_St

 Case 1 'Cautious

 Choose\_Bet = Round((G.Min\_Bet + G.Tot\_Loss) / (G.Wh\_Numbers - 1)) ' per ripianare il debito

 Case 2 'Wise

 Choose\_Bet = Round((G.Min\_Win + G.Tot\_Loss) / (G.Wh\_Numbers - 1)) + 1

 Case 3 'Random

 Choose\_Bet = Rand\_From(G.Min\_Bet, G.Tot\_Loss)

 Case 4 'Random Increase

 Choose\_Bet = G.Bet + Rand\_From(1, G.Min\_Bet \* 10)

 End Select

 If Choose\_Bet < G.Min\_Bet Then Choose\_Bet = G.Min\_Bet

End Function

Private Function Choose\_Number(G As Wh\_Game, Optional L\_W) As Integer

' L\_W is the last winning number

 Select Case G.N\_St

 Case Stay: Choose\_Number = G.Number 'Useless, just to show ...

 Case Casual: Choose\_Number = CInt(Rand\_From(0, CLng(G.Wh\_Numbers)))

 Case Last: Choose\_Number = L\_W

 End Select

End Function

Private Sub Gumble(G As Wh\_Game, Pr\_Out As Boolean)

Dim W\_Num As Integer, i As Integer

Dim S1 As String, S2 As String

 i = 1

 Do Until (G.Tot\_Win >= G.Min\_Win) Or (G.Tot\_Loss >= G.Max\_Loss) Or (i > G.Max\_Play)

 W\_Num = Rand\_From(0, CInt(G.Wh\_Numbers))

 'W\_Num = 1000

 Call Update\_Record(G, W\_Num)

 'Printing section

 If Pr\_Out Then

 Debug.Print "Round: " & CStr(i)

 Debug.Print Show(G, W\_Num)

 End If

 i = i + 1

 'Select new Bet

 G.Bet = Choose\_Bet(G)

 G.Number = Choose\_Number(G, W\_Num)

 Loop

End Sub

Private Sub Update\_Record(G As Wh\_Game, W\_Num As Integer)

G.Tot\_Bet = G.Tot\_Bet + G.Bet

G.Win = 0

 If W\_Num = G.Number Then 'win

 G.Win = CDbl(G.Bet) \* (G.Wh\_Numbers - 1) ' Prize net of last bet

 G.Tot\_Loss = WorksheetFunction.Max(0, G.Tot\_Loss - G.Win)

 If G.Tot\_Loss = 0 Then

 G.Tot\_Win = G.Tot\_Win + (G.Win - G.Tot\_Loss)

 Else

 G.Tot\_Win = 0

 End If

 Else 'loss

 G.Tot\_Win = WorksheetFunction.Max(0, G.Tot\_Win - G.Bet)

 If G.Tot\_Win = 0 Then

 G.Tot\_Loss = G.Tot\_Loss + (G.Bet - G.Tot\_Win)

 Else

 G.Tot\_Loss = 0

 End If

 End If

End Sub

Private Function Show(G As Wh\_Game, W\_Num As Integer) As String

Dim S As String

 Show = "Winning number: " & W\_Num & " Your number: " & G.Number

 Show = Show & vbNewLine & "You Bet: " & G.Bet 'also vbcrlf

 If W\_Num = G.Number Then

 Show = Show & vbNewLine & "You win:" & G.Win

 Else

 Show = Show & vbNewLine & "You lose: " & G.Bet

 End If

End Function

**Vettori e Matrici**

* Dim A(1 To 3) As Integer , B(1 To 2, 1 To 4) As Double
* A(1) = 1
* …
* Dim A()
* Redim A(1 To 2)
* Redim Preserve
* UBound, LBound
* Esempietti vari (creare un vettore, leggere e mostrare a video il primo e l’ultimo, fare vedere errore)

Esempio Copia Vettore

Public Sub Copia\_Vettore(Original() As Integer, Copy() As Integer)

Dim i As Integer

 For i = LBound(Original) To UBound(Original)

 Copy(i) = Original(i)

 Next i

End Sub

Public Sub Random\_Vet(V() As Integer)

Dim i As Integer

 For i = LBound(V) To UBound(V)

 V(i) = Int(rnd() \* 101)

 Next i

End Sub

Public Sub Prova\_Copy(N As Integer)

Dim V\_Or() As Integer, V\_Cop() As Integer

 ReDim V\_Or(1 To N)

 ReDim V\_Cop(1 To N)

 Call Random\_Vet(V\_Or)

 Call Copia\_Vettore(V\_Or, V\_Cop)

 For i = 1 To N

 Debug.Print CStr(V\_Or(i)) & " " & CStr(V\_Cop(i))

 Next i

End Sub

**LEZIONE 4 (9\_10\_2020)**

**Assegnamento di vettori A = B si può fare solo con variabile di tipo variant**

Esempio banale

Public Sub Prova()

Dim V As Variant

Dim M(1 To 3) As Variant

M(1) = 1

M(2) = 2

M(3) = 3

V = M

For i = 1 To UBound(V)

 Debug.Print CStr(V(i))

Next i

End Sub

Questo permette di avere una funzione con più output: Esempio matrice identità

Public Function Mat\_I(N\_N As Integer) As Variant

Dim M() As Variant

Dim r As Integer, c As Integer

 ReDim M(1 To N\_N, 1 To N\_N)

 For r = 1 To N\_N

 For c = 1 To N\_N

 M(r, c) = 0

 If r = c Then M(r, c) = 1

 Next c

 Next r

 Mat\_I = M

End Function

Public Sub Print\_Mat(N\_N As Integer)

Dim M As Variant

Dim S As String

Dim r As Integer, c As Integer

M = Mat\_I(N\_N)

For r = 1 To N\_N

 For c = 1 To N\_N

 If c = 1 Then S = ""

 S = S & CStr(M(r, c)) & " "

 Next c

 Debug.Print S

Next r

End Sub

Un altro modo per avere una funzione con più output è quello di utilizzare una funzione che restituisce una stinga, una sorta di CSV

Esempio output con più elementi tramite stringa

Public Function Random\_Code() As String

Dim r As Double

Dim S As String, Char As String

 Do

 r = rnd()

 If r < 0.5 Then

 Char = CStr(WorksheetFunction.RandBetween(0, 9))

 Else

 Char = Chr(WorksheetFunction.RandBetween(97, 122)) 'asc is the opposite

 End If

 S = S & Char & ","

 r = rnd()

 Loop Until r > 0.9

 Random\_Code = Left(S, Len(S) - 1)

End Function

Public Sub Decode()

Dim V As Variant

Dim S As String

 S = Random\_Code()

 V = Split(S, ",")

 For i = 0 To UBound(V)

 Debug.Print V(i)

 Next i

End Sub

**Esempio gioco con due dadi (non fatto in classe)**

Si lanciano due dati e si scommette sulla somma.

Enum Scommessa

 uguale = 1

 maggiore\_uguale = 2

 minore\_uguale = 3

End Enum

Public Probab(2 To 12, 1 To 3) As Double ‘La matrice con le probabilità

Tale matrice è scritta in un foglio Excel, di default nel range "E14:G24"

I valori sono quelli di seguito riportati. La prima colonna rappresenta la probabilità che esca un certo valore della somma, la seconda e la terza che il valore sia minore uguale o maggiore uguale.

|  |  |  |  |
| --- | --- | --- | --- |
| Somma | P(i) | P(<=i) | P(>=i) |
| 2 | 0.027778 | 0.027778 | 1 |
| 3 | 0.055556 | 0.083333 | 0.972222 |
| 4 | 0.083333 | 0.166667 | 0.916667 |
| 5 | 0.111111 | 0.277778 | 0.833333 |
| 6 | 0.138889 | 0.416667 | 0.722222 |
| 7 | 0.166667 | 0.583333 | 0.583333 |
| 8 | 0.138889 | 0.722222 | 0.416667 |
| 9 | 0.111111 | 0.833333 | 0.277778 |
| 10 | 0.083333 | 0.916667 | 0.166667 |
| 11 | 0.055556 | 0.972222 | 0.083333 |
| 12 | 0.027778 | 1 | 0.027778 |

Private Sub Setta\_Pr(Optional Address As Variant)

Dim Rng As Range

Dim r As Integer, c As Integer

 If IsMissing(Address) Then Address = "E14:G24"

 Set Rng = Application.Worksheets("Prob").Range(Address)

 For r = 2 To 12

 For c = 1 To 3

 Probab(r, c) = Rng.Cells(r - 1, c)

 Next c

 Next r

End Sub

Public Sub Gioco\_Dadi(Numero As Integer, Sc As Scommessa, Puntata As Double)

Dim Dado1 As Integer, Dado2 As Integer

Dim Vincita As Double

Dim Vittora As String

 If Probab(2, 1) = 0 Then Call Setta\_Pr

 Dado1 = Lancia\_Dado()

 Dado2 = Lancia\_Dado()

 Vincita = Puntata \* Moltiplicatore(Dado1 + Dado2, Numero, Sc)

 Debug.Print "Sono usciti i seguenti valori " & Dado1 & " "; Dado2

 If Vincita > 0 Then

 Debug.Print "Hai vinto " & (Vincita - Puntata)

 Else

 Debug.Print "Hai perso " & Vincita

 End If

End Sub

Private Function Lancia\_Dado() As Integer

 Lancia\_Dado = Int(1 + Rnd() \* 6)

End Function

Private Function Moltiplicatore(Somma As Integer, N As Integer, \_

 Sc As Scommessa) As Double

Dim c As Integer

 c = 0

 Moltiplicatore = -1

 Select Case Sc

 Case uguale

 If Somma = N Then c = 1

 Case minore\_uguale

 If Somma <= N Then c = 2

 Case maggiore\_uguale

 If Somma >= N Then c = 3

 End Select

 If c <> 0 Then Moltiplicatore = 1 / Probab(N, c)

End Function

Esempio vettore di studenti

Type Studente

 Name As String

 Surname As String

 Data\_Nascita As Date

 Corso\_Laurea As String

 Matricola As String

End Type

Dim Studenti() As Studente

Public Sub Prova\_St()

'Procedura di debug per verificare la creazione di uno studente

Dim St As Studente

 St.Name = "Franco"

 St.Surname = "Zeffirelli"

 St.Data\_Nascita = #12/23/1999#

 St.Matricola = "zmm\_fnc\_1234"

 St.Corso\_Laurea = "Ing\_Gestionale"

 Debug.Print "Sono lo studente " & St.Name & " " & St.Surname

 Debug.Print "Ho " & CStr(Age(St.Data\_Nascita)) & " anni"

 St.Age = Age(St.Data\_Nascita)

 Debug.Print CStr(St.Age)

End Sub

Public Function Age(Birth As Date) As Integer

'Funzione che calcola l'età

 Age = DateDiff("yyyy", Birth, Date)

 If Month(Birth) > Month(Date) \_

 Or (Month(Birth) = Month(Date) And Day(Birth) > Day(Date)) \_

 Then Age = Age - 1

End Function

Private Function Is\_Empty(Ar() As Studente) As Boolean

'Funzione che serve a vedere se un vettore non è ancora stato dimensionato

Dim L As Integer

On Error Resume Next

 Is\_Empty = True

 L = UBound(Ar)

 If L > 0 Then Is\_Empty = False

 'Se Ubound > 0 la funzione restituisce valore positivo

 'Altrimenti errore, il valore resta uno la funzione prosegue ed esce

End Function

Public Sub Add\_DB(New\_St As Studente)

'Aggiunge uno studente al vettore degli studenti

Dim Pos As Integer

 If Is\_Empty(Studenti) Then

 ReDim Studenti(1 To 1)

 Else

 ReDim Preserve Studenti(1 To UBound(Studenti) + 1)

 End If

 Pos = UBound(Studenti)

 Studenti(Pos) = New\_St

End Sub

Public Sub Main()

'Subroutine Main per verificare il corretto funzionamento

Dim S As Studente

 S.Name = "Francesco"

 S.Surname = "Zammori"

 Call Add\_DB(S)

 S.Name = "Silvia"

 S.Surname = "Rossi"

 Call Add\_DB(S)

 For i = 1 To 2

 Debug.Print Studenti(i).Name & " " & Studenti(i).Surname

 Next i

 Pippo = 1

End Sub

**LEZIONE 5 (11\_10\_2020) – RECUPERO LEZIONE SOLO REGISTRATA**

**Funzione Array e String**

Dim A As Variant

A = Array(10, 20, 30)

S = String(10, “a”)

**ParamArray**

Public Function Concat(ParamArray Stringhe() As Variant)

'Il numero di parole da concatenare è teoricamente infinito per cui usiamo un vettore variabile

'chiamato Stringhe(). Si noti che il vettore variabile deve necessariamente essere messo in fondo.

Dim i As Integer

 If UBound(Stringhe) <> -1 Then ' Se un vettore è vuoto Ubound = -1

 For i = LBound(Stringhe) To UBound(Stringhe)

 Concat = Concat & Stringhe(i) & " "

 Next i

 Else

 Concat = "Empty"

 End If

End Function

**IsMissing**

Public Function Missing(Optional Param As Variant) As Boolean

 If IsMissing(Param) Then

 Missing = True

 Else

 Missing = False

 End If

End Function

**Creazione di un proto Data Base: programma che popola una lista di studenti: vettore di studenti – Il tipo studente ha al suo interno un vettore contenente Esami (altro tipo personalizzato)**

Esempio tipo studente (2)

Type Studente

 Nome As String

 Cognome As String

 Id As String

 Data\_Iscrizione As Date

 Corso\_Laurea As String

 N\_Es\_Sg As Integer ' Seguiti

 N\_Es\_St As Integer ' Sostenuti

 Esami() As Esame

 Lista\_Es() As String

End Type

Type Esame

 Id As String

 N\_Tent As Integer

 Tentativi() As Variant 'Matrice contenente la data dell'esame e il voto conseguito

 Voto As Integer

 Data\_In As Variant 'Data Iscrizione

 Data\_Out As Variant ' Data Superamento

 Accettato As Boolean

End Type

Public Studenti() As Studente

Public Lista\_St() As String

Private Function Check\_St(Arr() As Studente) As Integer

On Error Resume Next

 Check\_St = -1 'Vettore non ancora dimensionato

 Check\_St = UBound(Arr)

End Function

Public Sub Aggiungi\_Esame(St As Studente, ID\_Esame As String)

Dim n As Integer

Dim E As Esame

 'Settiamo i parametri

 E.Id = ID\_Esame

 E.Accettato = False

 E.Data\_In = Date

 E.Data\_Out = Null

 E.N\_Tent = 0

 E.Voto = 0

 'Aggiorniamo il vettore

 n = St.N\_Es\_Sg + 1

 St.N\_Es\_Sg = n

 ReDim Preserve St.Esami(1 To n)

 ReDim Preserve St.Lista\_Es(1 To n)

 St.Esami(n) = E

 St.Lista\_Es(n) = E.Id

End Sub

Public Sub Aggiungi\_Voto(St As Studente, ID\_Esame As String, \_

 Voto As Integer, Optional D As Date = #1/1/1900#)

Dim Pos As Integer, n As Integer

Dim Es(1 To 2) As Variant

 If D = #1/1/1900# Then D = Date

 Pos = Find(ID\_Esame, St.Lista\_Es)

 If Pos = -1 Then Exit Sub

 If St.Esami(Pos).Accettato Then Exit Sub

 n = St.Esami(Pos).N\_Tent + 1

 St.Esami(Pos).N\_Tent = n

 ReDim Preserve St.Esami(Pos).Tentativi(1 To n)

 Es(1) = D

 Es(2) = Voto

 St.Esami(Pos).Tentativi(n) = Es

End Sub

Public Sub Accetta(St As Studente, ID\_Es As String)

Dim Pos As Integer, n As Integer

 Pos = Find(ID\_Es, St.Lista\_Es)

 If Pos = -1 Then Exit Sub

 If Not St.Esami(Pos).Accettato Then

 n = St.Esami(Pos).N\_Tent

 If St.Esami(Pos).Tentativi(n)(2) >= 18 Then

 St.Esami(Pos).Accettato = True

 St.N\_Es\_St = St.N\_Es\_St + 1

 St.Esami(Pos).Voto = St.Esami(Pos).Tentativi(n)(2)

 End If

 End If

End Sub

Public Function Media(St As Studente) As Double

Dim i As Integer, Totale As Integer

 For i = 1 To St.N\_Es\_Sg

 If St.Esami(i).Accettato Then Totale = Totale + St.Esami(i).Voto

 Next i

 Media = Totale / St.N\_Es\_St

End Function

Private Function Find(Id As String, Arr() As String)

Dim Pos As String

On Error GoTo Err:

 Pos = 1

 Do While Arr(Pos) <> Id

 Pos = Pos + 1

 Loop

 Find = Pos

 Exit Function

Err:

 Find = -1 'Non trovato

End Function

Public Sub Prova\_Studente()

Dim St As Studente

 St.Nome = "Filippo"

 St.Cognome = "Lippi"

 St.Corso\_Laurea = "Gestionale"

 St.Data\_Iscrizione = #9/1/2018#

 St.Id = "FL\_1234"

 Call Aggiungi\_Esame(St, "Analisi1")

 Call Aggiungi\_Esame(St, "Fisica1")

 Call Aggiungi\_Voto(St, "Analisi1", 15)

 Call Aggiungi\_Voto(St, "Analisi2", 15)

 Call Aggiungi\_Voto(St, "Analisi1", 22)

 Call Aggiungi\_Voto(St, "Analisi1", 25)

 Call Accetta(St, "Analisi1")

 Call Aggiungi\_Voto(St, "Fisica1", 20)

 Call Accetta(St, "Fisica1")

 Debug.Print Media(St)

End Sub

Public Sub Prova\_studenti()

Dim S As Studente

 Call Aggiungi\_Studente("Filippo", "Lippi")

 Call Aggiungi\_Studente("Veronica", "Lario")

 Call Aggiungi\_Studente("Fabio", "Rampini")

 Pos = Find("F.LI", Lista\_St) 'Presupponamo che ci sia

 Call Aggiungi\_Esame(Studenti(Find("F.LI", Lista\_St)), "Analisi1")

 Call Aggiungi\_Esame(Studenti(Find("F.LI", Lista\_St)), "Fisica1")

 Call Aggiungi\_Voto(Studenti(Find("F.LI", Lista\_St)), "Fisica1", 20)

 Call Accetta(Studenti(Find("F.LI", Lista\_St)), "Fisica1")

 S = Studenti(Find("F.LI", Lista\_St))

 Debug.Print Media(S)

End Sub

Public Sub Prova\_Check()

Dim St() As Studente

Dim n As Integer

 n = Check\_St(St)

 Debug.Print n

 ReDim St(1 To 10)

 n = Check\_St(St)

 Debug.Print n

End Sub

Public Sub Aggiungi\_Studente(Nome As String, Cognome As String)

Dim St As Studente

 St.Nome = Nome

 St.Cognome = Cognome

 St.Corso\_Laurea = "Gestionale"

 St.Id = UCase(Left(Nome, 1)) & "." & UCase(Left(Cognome, 2))

 Pos = Check\_St(Studenti) + 1

 If Pos = 0 Then

 ReDim Studenti(1 To 1)

 ReDim Lista\_St(1 To 1)

 Pos = 1

 Else

 ReDim Preserve Studenti(1 To Pos)

 ReDim Preserve Lista\_St(1 To Pos)

 End If

 Studenti(Pos) = St

 Lista\_St(Pos) = St.Id

End Sub

**CLASSI**

Tipo Quadrato e Classe Quadrato

Type quadrato

 Lato As Integer

 Perimetro As Integer

 Area As Integer

End Type

Public Function Area(Lato As Integer) As Integer

 Area = Lato ^ 2

End Function

Public Function Perimetro(Lato As Integer) As Integer

 Perimetro = Lato \* 4

End Function

Public Sub My\_Quadrato()

Dim Q As quadrato

Dim Q1 As Cls\_Quad

 Q.Lato = 10

 Q.Area = Area(Q.Lato)

 Q.Perimetro = Perimetro(Q.Lato)

 Debug.Print Q.Area

 Debug.Print Q.Perimetro

End Sub

**Classe quadrato (scritto in un modulo di classe Cls\_Quad)**

Public Nome

Public Lato

Public Function Area()

 Area = Lato ^ 2

End Function

Public Function Perimetro()

 Perimetro = Lato \* 4

End Function

Public Sub My\_Quadrato()

Dim Q1 As Cls\_Quad

Set Q1 = New Cls\_Quad

 Q1.Nome = "Quadrato"

 Q1.Lato = 5

 Debug.Print Q1.Area

 Debug.Print Q1.Perimetro

End Sub

**Trasformare il tipo studente in Classe studente**

Classe di base studente

'Proprietà

Public Nome As String

Public Cognome As String

Public Data\_Iscrizione As Date

Public Corso\_Laurea As String

Private Id As String

Private N\_EsSg As Integer

Private N\_EsSt As Integer

Private Esami() As Esame

Private Sub Class\_Initialize()

 ReDim Esami(1 To 1)

 'ReDim Lista\_Es(1 To 1)

 Id = Int(Rnd() \* 999999)

End Sub

Public Sub Crea\_Matricola()

 If Nome <> "" And Cognome <> "" Then

 Id = UCase(Left(Nome, 1)) & "." & UCase(Left(Cognome, 1)) & "\_" & Id

 End If

End Sub

‘Funzioni già presenti nell’esercizio precedente, solo piccole modifiche

Public Property Get Matricola() As String

 Matricola = Id

End Property

Public Sub Aggiungi\_Esame(ID\_Esame As String, Optional D As Date = #1/1/1900#)

Dim E As Esame

 'Settiamo i parametri

 If D = #1/1/1900# Then D = Date

 E.Id = ID\_Esame

 E.Accettato = False

 E.Data\_In = D

 E.Data\_Out = Null

 E.N\_Tent = 0

 E.Voto = 0

 'Aggiorniamo il vettore

 N\_EsSg = N\_EsSg + 1

 ReDim Preserve Esami(1 To N\_EsSg)

 Esami(N\_EsSg) = E

End Sub

Public Sub Aggiungi\_Voto(ID\_Esame As String, Voto As Integer, Optional D As Date = #1/1/1900#)

Dim Pos As Integer, n As Integer

Dim Es(1 To 2) As Variant

 If D = #1/1/1900# Then D = Date

 Pos = Find(ID\_Esame)

 If Pos = -1 Then Exit Sub

 If Esami(Pos).Accettato Then Exit Sub

 n = Esami(Pos).N\_Tent + 1

 Esami(Pos).N\_Tent = n

 ReDim Preserve Esami(Pos).Tentativi(1 To n)

 Es(1) = D

 Es(2) = Voto

 Esami(Pos).Tentativi(n) = Es

End Sub

Private Function Find(Id As String) As Integer

Dim Pos As String

On Error GoTo Err:

 Pos = 1

 Do While Esami(Pos).Id <> Id

 Pos = Pos + 1

 Loop

 Find = Pos

 Exit Function

Err:

 Find = -1 'Non trovato

End Function

Public Function Esame(Id As String) As Boolean

 Esame = True

 If Find(Id) = -1 Then Esame = False

End Function

Public Sub Accetta(ID\_Es As String)

Dim Pos As Integer, n As Integer

 Pos = Find(ID\_Es)

 If Pos = -1 Then Exit Sub

 If Not Esami(Pos).Accettato And Esami(Pos).N\_Tent > 0 Then

 n = Esami(Pos).N\_Tent

 If Esami(Pos).Tentativi(n)(2) >= 18 Then

 Esami(Pos).Accettato = True

 N\_EsSt = N\_EsSt + 1

 Esami(Pos).Voto = Esami(Pos).Tentativi(n)(2)

 Esami(Pos).Data\_Out = Esami(Pos).Tentativi(n)(1)

 End If

 End If

End Sub

Public Function Media() As Double

Dim i As Integer, Totale As Integer

 If N\_EsSt = 0 Then Exit Function

 For i = 1 To N\_EsSg

 If Esami(i).Accettato Then Totale = Totale + Esami(i).Voto

 Next i

 Media = Totale / N\_EsSt

End Function

‘Nuovi metodi

Public Function Repr() As String

 Repr = Nome & " " & Cognome & vbNewLine

 Repr = Repr & "Matricola " & Id & vbNewLine

 Repr = Repr & "Esami seguiti " & N\_EsSg & vbNewLine

 If N\_EsSt > 0 Then Repr = Repr & "Media " & Media()

End Function

Public Function Trova\_Esame(Id As String) As String

Dim Pos As Integer

 Pos = Find(Id)

 If Pos <> -1 Then

 If Esami(Pos).Accettato Then Trova\_Esame = Id & " sostenuto il " & Esami(Pos).Data\_Out & " con voto " & Esami(Pos).Voto

 Else

 Trova\_Esame = "Esame non sostenuto"

 End If

End Function

Public Function Tentativi\_Esame(Id As String) As String

Dim Pos As Integer

 Pos = Find(Id)

 If Pos <> -1 Then

 For i = 1 To Esami(Pos).N\_Tent

 Tentativi\_Esame = Tentativi\_Esame & Esami(Pos).Tentativi(i)(1) & " " & Esami(Pos).Tentativi(i)(2) & vbNewLine

 Next i

 Else

 Tentativi\_Esame = "Esame non sostenuto"

 End If

End Function

Procedura di debug

Public Sub Prova\_Cls\_St()

Dim St As Cls\_Studente

 Set St = New Cls\_Studente

 St.Nome = "Fra"

 St.Cognome = "Zam"

 St.Crea\_Matricola

 St.Aggiungi\_Esame ("analisi1")

 St.Aggiungi\_Esame ("fisica1")

 St.Aggiungi\_Voto "analisi1", 15

 St.Aggiungi\_Voto "analisi1", 18

 St.Aggiungi\_Voto "analisi1", 22

 St.Accetta ("fisica1")

 St.Accetta ("analisi1")

 Debug.Print St.Repr

 Debug.Print St.Media

 Debug.Print St.Trova\_Esame("analisi1")

 Debug.Print St.Tentativi\_Esame("analisi1")

End Sub

Creazione del Data Base studenti con collection

Public Anagrafica As Collection

Public Sub Usa\_Collection()

Dim St As Cls\_Studente

Set St = New Cls\_Studente

Set Anagrafica = New Collection

 St.Nome = "Fra"

 St.Cognome = "Zam"

 St.Crea\_Matricola

 St.Aggiungi\_Esame ("analisi1")

 St.Aggiungi\_Esame ("fisica1")

 St.Aggiungi\_Voto "analisi1", 15

 St.Aggiungi\_Voto "analisi1", 18

 St.Aggiungi\_Voto "fisica1", 22

 St.Accetta ("fisica1")

 St.Accetta ("analisi1")

 Anagrafica.Add St, St.Cognome

 Set St = Nothing

 Set St = New Cls\_Studente

 St.Nome = "Filippo"

 St.Cognome = "Lippi"

 St.Crea\_Matricola

 St.Aggiungi\_Esame ("disegno")

 St.Aggiungi\_Esame ("algebra")

 St.Aggiungi\_Voto "algebra", 15

 St.Aggiungi\_Voto "disegno", 22

 St.Accetta ("disegno")

 Anagrafica.Add St, St.Cognome

 Set St = Nothing

 Debug.Print Anagrafica.Item("Lippi").Repr

 Debug.Print Anagrafica.Item("Zam").Tentativi\_Esame("analisi1")

End Sub

Private Function RB(low As Integer, High As Integer) As Integer

 RB = Application.WorksheetFunction.RandBetween(low, High)

End Function

Private Sub Crea\_St(C\_St As Cls\_Studente, Optional Nome As Variant, Optional Cognome As Variant, Optional Data\_Isc As Variant)

Dim Nomi() As Variant, Cognomi() As Variant, Esami() As Variant

Dim Esame As String

Dim N\_Mesi As Integer

 Nomi = Array("Francesco", "Giada", "Valentina", "Giovanni", "Filippo", "Gaetano", "Daniela")

 Cognomi = Array("Bianchi", "Rossi", "Verdi", "Romano", "Franchi", "Giusti", "Debiagi")

 Esami = Array("Analisi1", "Analisi2", "Fisica1", "Fisica2", "Algebra", "Informatica", "Data\_Base")

 Set C\_St = New Cls\_Studente

 C\_St.Nome = Nomi(RB(0, 6))

 C\_St.Cognome = Cognomi(RB(0, 6))

 C\_St.Data\_Iscrizione = DateSerial(RB(2011, 2020), 9, 1)

 If Not IsMissing(Nome) Then C\_St.Nome = Nome

 If Not IsMissing(Cognome) Then C\_St.Cognome = Cognome

 If Not IsMissing(Data\_Isc) Then C\_St.Data\_Iscrizione = Data\_Isc

 C\_St.Crea\_Matricola

 For i = 1 To 4

 Esame = Esami(RB(0, 6))

 If Not C\_St.Esame(Esame) Then

 C\_St.Aggiungi\_Esame (Esame)

 N\_Mesi = RB(6, 24)

 C\_St.Aggiungi\_Voto Esame, RB(10, 17), DateAdd("m", N\_Mesi, C\_St.Data\_Iscrizione)

 C\_St.Aggiungi\_Voto Esame, RB(18, 33), DateAdd("m", N\_Mesi + RB(3, 12), C\_St.Data\_Iscrizione)

 C\_St.Accetta (Esame)

 End If

 Next i

End Sub

Public Sub Crea\_Lista(Optional n As Integer = 5)

Dim St As Cls\_Studente, St2 As Cls\_Studente

Dim key As String

Set Anagrafica = New Collection

 For i = 1 To n

 Set St = New Cls\_Studente

 Call Crea\_St(St)

 key = Left(St.Nome, 1) & "." & St.Cognome

 If Not Is\_In(Anagrafica, key) Then Anagrafica.Add St, key

 Set St = Nothing

 Next i

End Sub

Public Sub Show\_All()

‘Osservare il ciclo For Each

 For Each St In Anagrafica

 Debug.Print

 Debug.Print St.Repr

 Next St

End Sub

Private Function Is\_In(C As Collection, key As String) As Boolean

‘Simile alla funzione scritta per verificare che un vettore sia dimensionato correttamente

On Error Resume Next

 C.Item key

 Is\_In = (Err.Number = 0)

End Function

**LEZIONE 6 (16\_10\_2020)**

Classe Poligono

Public Nome As String

Private nl As Integer

Private lt As Double

Private ap As Double

Private rg As Double

Private an As Double

Private pi As Double

Public Property Let N\_Lati(n As Integer)

 If n <= 3 Then n = 3

 pi = Application.WorksheetFunction.pi

 nl = n

 an = 2 \* pi / nl

End Property

Public Property Get N\_Lati() As Integer

 N\_Lati = nl

End Property

Public Property Let Lato(l As Double)

 If l < 1 Then l = 1

 lt = l

 rg = l / (2 \* Sin(an / 2))

 ap = rg \* Cos(an / 2)

End Property

Public Property Get Lato() As Double

 Lato = lt

End Property

Public Property Get Raggio() As Double

 Raggio = rg

End Property

Public Property Get Apotema() As Double

 Apotema = ap

End Property

Public Function Area() As Double

 Area = Perimetro \* ap \* 0.5

End Function

Public Function Perimetro() As Double

 Perimetro = nl \* lt

End Function

Esempio 1 Uso della classe

Public Sub QuEsad()

Dim Q As Cls\_Poligoni

Set Q = New Cls\_Poligoni

Q.Nome = "Triangolo"

Q.N\_Lati = 3

Q.Lato = 10

Debug.Print Q.Area

Debug.Print Q.Apotema

End Sub

Esempio 2 – Copia di oggetto

Public Sub Quad()

Dim Q As Cls\_Poligoni, Q2 As Cls\_Poligoni

Set Q = New Cls\_Poligoni

Q.Nome = "Triangolo"

Q.N\_Lati = 3

Q.Lato = 10

Debug.Print Q.Area

Debug.Print Q.Apotema

Set Q2 = Q

Q2.N\_Lati = 4

Debug.Print Q2.Area

Debug.Print Q2.Apotema

Debug.Print Q.Area

Debug.Print Q.Apotema

End Sub

Utilizzo di una collection

Public Poligoni As Collection

Public Sub Usa\_Poligoni()

Dim Pl As Cls\_Poligoni

Dim P() As Variant

Set Poligoni = New Collection

Set Pl = New Cls\_Poligoni

 P = Array("Triangolo", "Quadrato", "Pentagono", "Esagono")

 Crea Pl, CStr(P(0)), 3, 10

 Poligoni.Add Pl, P(0)

 Crea Pl, CStr(P(1)), 4, 10

 Poligoni.Add Pl, P(1)

 Crea Pl, CStr(P(2)), 5, 10

 Poligoni.Add Pl, P(2)

 Crea Pl, CStr(P(3)), 6, 10

 Poligoni.Add Pl, P(3)

 Set Pl = Nothing

 For Each Pl In Poligoni

 Debug.Print Pl.Nome

 Debug.Print Pl.Area

 Next Pl

 Set Pl = Poligoni.Item(P(1))

 Debug.Print Pl.Nome

End Sub

Public Sub Crea(Pl As Cls\_Poligoni, Nome As String, Lati As Integer, Lato As Integer)

 Set Pl = Nothing

 Set Pl = New Cls\_Poligoni

 Pl.Nome = Nome

 Pl.N\_Lati = Lati

 Pl.Lato = Lato

End Sub

Class\_Initialize

Private Sub Class\_Initialize()

 Nome = Triangolo

 N\_Lati = 3

 Lato = 10

End Sub

**Una Classe che fa la media mobile di ordine n**

Attenzione che se si cambia l’ordine della media mobile ci sono dei problemi… Bisogna azzerare tutto!!!

Private SS() As Double, MM() As Double, TOT As Double

Private S\_SS As String, S\_MM As String

Private N\_TOT As Single, N\_MM As Single

Private Ordine As Integer

Private Sub Class\_Initialize()

 Call Init

End Sub

Private Sub Init()

 ReDim SS(1 To 100)

 ReDim MM(1 To 100)

 S\_SS = "Serie Storica" & vbNewLine & "[]" ‘lista contenente la serie storica

 S\_MM = "Media Mobile" & vbNewLine & "[]" ‘lista contenente la media mobile

 N\_TOT = 0

 N\_MM = 0

 TOT = 0

 Ordine = 3

End Sub

Public Function Media\_Tot() As Double ‘Restituisce la media totale della serie storica

 Media\_Tot = 0

 If N\_TOT > 0 Then Media\_Tot = TOT / N\_TOT

End Function

Public Function Show(Optional MM As Boolean = True) ‘mostra la serie storica o la media mobile

 If MM Then

 Show = S\_MM

 Else

 Show = S\_SS

 End If

End Function

Public Sub Set\_Up(Ord As Integer, ParamArray NewData() As Variant)

Dim i As Integer

 Call Init

 Ordine = Ord

 If UBound(NewData) >= 1 Then

 For i = LBound(NewData) To UBound(NewData)

 If Check(NewData(i)) Then Call Add(NewData(i))

 Next i

 End If

End Sub

Public Sub Add\_Data(ParamArray NewData() As Variant)

 If UBound(NewData) < 1 Then Exit Sub

 For i = LBound(NewData) To UBound(NewData)

 If Check(NewData(i)) Then Call Add(NewData(i))

 Next i

End Sub

Private Sub Add(D As Variant)

 N\_TOT = N\_TOT + 1

 TOT = TOT + CDbl(D)

 If UBound(SS) < N\_TOT Then ReDim Preserve SS(1 To UBound(SS) + 100)

 SS(N\_TOT) = CDbl(D)

 Call AddS(D, S\_SS, N\_TOT)

 If UBound(MM) < N\_MM Then ReDim Preserve MM(1 To UBound(MM) + 100)

 If N\_TOT >= Ordine Then

 N\_MM = N\_MM + 1

 MM(N\_MM) = Med() ‘funzione

 Call AddS(MM(N\_MM), S\_MM, N\_MM)

 End If

End Sub

Private Function Med() As Double

Dim i As Integer

 For i = N\_TOT To (N\_TOT + 1 - Ordine) Step -1

 Med = Med + CDbl(SS(i))

 Next i

 Med = Med / Ordine

End Function

Private Function Check(D As Variant) As Boolean

On Error Resume Next

 Dim j As Double

 Check = True

 j = CDbl(D)

 If Err.Number = 13 Then Check = False

End Function

Private Sub AddS(D As Variant, S As String, N As Single)

‘Aggiunge un valore alla lista della serie o della media

 S = Left(S, Len(S) - 1) 'Cancelliamo l'ultimo elemento

 If (N \ 10) = (N / 10) Then

 S = S & " ..." & vbNewLine

 Else

 If Right(S, 1) <> "[" Then S = S & "; "

 End If

 S = S & D & "]"

End Sub

**LEZIONE 7 (19\_10\_2020)**

**RICORSIONE**

Fattoriale

Public Function Factorial(N As Integer) As Single

 Factorial = 1

 If N <= 1 Then Exit Function

 Factorial = N \* Factorial(N - 1)

End Function



Fibonacci

Serie di Fibonacci

1,1,2,3,5,8,13,

Public Function Fib(N As Integer) As Single

 If N <= 2 Then

 Fib = 1

 Exit Function

 End If

 Fib = Fib(N - 1) + Fib(N - 2)

End Function



Notare la ripetizione inutile di operazioni!!!

*Memoization*

Public FibSeq(1 To 1000) As Single ‘Al Massimo arriviamo alla millesima cifra

Public Function FibMem(N As Integer) As Single

 If N <= 2 Then

 FibMem = 1

 If FibSeq(1) = 0 Then

 FibSeq(1) = 1

 FibSeq(2) = 1

 End If

 Exit Function

 End If

 If FibSeq(N) <> 0 Then

 FibMem = FibSeq(N)

 Exit Function

 End If

 FibMem = FibMem(N - 1) + FibMem(N - 2)

 FibSeq(N) = FibMem

End Function

Public Sub Show\_Fib(N As Integer)

Dim F As Single

Dim i As Integer

 F = FibMem(N)

 For i = 1 To 1000

 If FibSeq(i) <> 0 Then

 Debug.Print FibSeq(i)

 Else

 Exit For

 End If

 Next i

End Sub

Funzione ricorsiva che impila una stringa, ossia che la scrive in verticale carattere per carattere

Esempio Impila(“abcd”)



si ottiene:

a

b

c

d

Private Function Impila(V As Variant) As String

Dim S As String

Dim SubS As Variant 'La sotto sequenza

 S = CStr(V)

 If Len(S) = 1 Then

 Impila = S

 Else

 SubS = Right(S, Len(S) - 1)

 Impila = Left(S, 1) & vbNewLine & Impila(SubS)

 End If

End Function

Public Sub Print\_impila(V As Variant)

Dim S As String

 If Notnumeric(V) Then 'Verifica valore di input

 MsgBox "input non numerico", vbCritical, "errore"

 Else

 S = Impila(V)

 Debug.Print S

 End If

End Sub

Stessa Funzione non ricorsiva

Public Sub Impila\_NR(V As Variant)

Dim S As String

 If Notnumeric(V) Then 'Verifica valore di input

 MsgBox "input non numerico", vbCritical, "errore"

 Else

 S = CStr(V)

 Do While Len(S) > 0

 Debug.Print Left(S, 1)

 S = Right(S, Len(S) - 1)

 Loop

 End If

End Sub

Private Function Notnumeric(V As Variant) As Boolean

Dim L As Long

Dim S As String

On Error GoTo Err:

 Notnumeric = False

 S = CStr(V)

 L = CLng(S)

Err:

 If Err.Number = 13 Or Err.Number = 94 Then Notnumeric = True

End Function

Funzione ricorsiva Full\_Instring che restituisce tutte le sottostringhe di cui è composta una certa stringa (le sotto-stringhe vanno identificate a partire da un separatore)

In VBA esiste InStr() che però trova solo la prima occorrenza!

Ad esempio: “pippo;franco;topolino;minnie”, con separatore punto e virgola, restituisce:

pippo franco topolino minnie

Public Function Separa(Testo As String, Optional Separatore As String = ",") As Variant

Dim Positions() As String

Dim S\_Positions As String

Dim Words() As Variant

Dim i As Integer, NChar As Integer, NWrd As Integer

On Error GoTo Err

 S\_Positions = F\_InStr(Testo, Separatore)

 Positions = Split(S\_Positions, "-")

 NWrd = UBound(Positions) + 1

 ReDim Words(0 To NWrd)

 Words(0) = Trim(Left(Testo, Positions(0) - 1))

 For i = 0 To UBound(Positions) - 1

 NChar = Positions(i + 1) - Positions(i) - 1

 Words(i + 1) = Trim(Mid(Testo, Positions(i) + 1, NChar))

 Next i

 Words(NWrd) = Trim(Mid(Testo, Positions(UBound(Positions)) + 1))

 Separa = Words

Err:

If Err.Number = 9 Then Separa = Testo

End Function

Restituisce la posizione in cui si trova il separatore. Lo restituisce come stringa concatenata. Esempio F\_InStr(“qb,cccc,ff,g”, “,”, “-”) restituisce 3-8-11

Public Function F\_InStr(Testo As String, Sotto\_Stringa As String, Optional Concatenatore As String = "-", Optional Start = 1) As String

Dim pos As Integer

 pos = InStr(Start, Testo, Sotto\_Stringa)

 If pos = 0 Then

 F\_InStr = ""

 Exit Function

 Else

 F\_InStr = F\_InStr & pos & Concatenatore & F\_InStr(Testo, Sotto\_Stringa, Concatenatore, pos + 1)

 If Right(F\_InStr, 1) = Concatenatore Then F\_InStr = Left(F\_InStr, Len(F\_InStr) - 1)

 End If

End Function

**FIBONACCI COME CLASSE CON MEMOIZATION**

Private Fib\_Series() As Single

Private Sub Class\_Initialize()

 ReDim Fib\_Series(1 To 6)

 Fib\_Series(1) = 1

 Fib\_Series(2) = 1

 Fib\_Series(3) = 2

 Fib\_Series(4) = 3

 Fib\_Series(5) = 5

 Fib\_Series(6) = 8

 ' ...

End Sub

Public Function F\_Compute(N As Integer) As Single

 If N > UBound(Fib\_Series) Then ReDim Preserve Fib\_Series(1 To N)

 F\_Compute = CFib(N)

End Function

Private Function CFib(N As Integer) As Single

 If Fib\_Series(N) <> 0 Then

 CFib = Fib\_Series(N)

 Exit Function

 End If

 CFib = CFib(N - 1) + CFib(N - 2)

 Fib\_Series(N) = CFib

End Function

Public Sub Show\_Sequence()

 For i = 1 To UBound(Fib\_Series)

 Debug.Print Fib\_Series(i)

 Next i

End Sub

**SPLIT RICORSIVO**



Public Function My\_Split(St As String, \_

 Optional Sep As String = ",") As Variant

Dim V() As Variant

 ReDim V(0 To 0)

 Call Spt(V, St, Sep)

 'If V(UBound(V)) = "" Then ReDim Preserve V(0 To UBound(V) - 1)

 My\_Split = V

End Function

Private Sub Spt(V() As Variant, St As String, \_

 Sep As String)

Dim Pos As Integer, N As Integer

 If St = "" Then

 ReDim Preserve V(0 To UBound(V) - 1)

 Exit Sub

 End If

 N = UBound(V)

 Pos = InStr(St, Sep)

 If Pos = 0 Then

 V(N) = St

 Else

 V(N) = Left(St, Pos - 1)

 St = Mid(St, Pos + 1)

 ReDim Preserve V(0 To N + 1)

 Call Spt(V, St, Sep)

 End If

End Sub

Public Sub Try\_My\_Split(St As String, Optional Sep As String = ",")

Dim X As Variant, Y As Variant

Dim S As String

 S = St

 X = My\_Split(St, Sep)

 Y = My\_Split2(S, Sep)

 For i = 0 To UBound(X)

 Debug.Print X(i) & " " & Y(i)

 Next i

End Sub

Public Function My\_Split2(St As String, \_

 Optional Sep As String = ",") As Variant

Dim V() As Variant

Dim Pos As Integer, N As Integer

 ReDim V(0 To 0)

 Pos = InStr(1, St, Sep)

 Do While Pos > 0

 N = UBound(V)

 V(N) = Left(St, Pos - 1)

 St = Mid(St, Pos + 1)

 Pos = InStr(1, St, Sep)

 ReDim Preserve V(0 To N + 1)

 Loop

 If St <> "" Then

 V(N + 1) = St

 Else

 ReDim Preserve V(0 To N)

 End If

 My\_Split2 = V

End Function

**FIBONACCI COME CLASSE CON MEMOIZATION**

Private Fib\_Series() As Single

Private Sub Class\_Initialize()

 ReDim Fib\_Series(1 To 6)

 Fib\_Series(1) = 1

 Fib\_Series(2) = 1

 Fib\_Series(3) = 2

 Fib\_Series(4) = 3

 Fib\_Series(5) = 5

 Fib\_Series(6) = 8

 ' ...

End Sub

Public Function F\_Compute(N As Integer) As Single

 If N > UBound(Fib\_Series) Then ReDim Preserve Fib\_Series(1 To N)

 F\_Compute = CFib(N)

End Function

Private Function CFib(N As Integer) As Single

 If Fib\_Series(N) <> 0 Then

 CFib = Fib\_Series(N)

 Exit Function

 End If

 CFib = CFib(N - 1) + CFib(N - 2)

 Fib\_Series(N) = CFib

End Function

Public Sub Show\_Sequence(Optional Limit As Variant)

Dim N As Integer

 N = UBound(Fib\_Series)

 If IsMissing(Limit) Then Limit = N

 If Limit > N Then Limit = N

 For i = 1 To Limit

 Debug.Print Fib\_Series(i)

 Next i

End Sub

**CLASSE LISTA RICORSIVA LIFO**



**Classe nodo**

Public ID As String

Public Content As Variant

Public P\_Node As Cls\_Node

Public N\_Node As Cls\_Node

Private Sub Class\_Initialize()

 Set P\_Node = Nothing

 Set N\_Node = Nothing

 Content = Null

End Sub

Public Sub Link\_Node(N As Cls\_Node, Optional Nxt As Boolean = True)

 If Nxt Then

 Set N\_Node = N 'Il nodo viene linkato a valle

 Set N.P\_Node = Me 'Il nodo linkato vede Me ossia il nodo originario come nodo a monte

 Else

 Set P\_Node = N

 Set N.N\_Node = Me

 End If

End Sub

Public Function Representation() As String

Dim s As String

 On Error Resume Next

 If IsNull(Content) Or IsEmpty(Content) Then

 s = "Missing"

 Else

 s = CStr(Content)

 End If

 Representation = "Node " & ID & ", Value is : " & s

End Function

Public Sub Show()

 Debug.Print Representation

End Sub

**Classe LIFO**

Private Head As Cls\_Node

Private Tail As Cls\_Node

Private Sub Class\_Initialize()

 Set Head = Nothing

 Set Tail = Nothing

End Sub

Public Sub Push(Node As Cls\_Node)

 If Head Is Nothing Then

 Set Head = Node

 Set Tail = Node

 Else

 Tail.Link\_Node Node

 Set Tail = Node

 End If

 Set Node = Nothing 'Che succederebbe senza questa riga?!?

End Sub

Public Function Pop() As Cls\_Node

 If Tail Is Nothing Then

 Set Pop = Nothing

 Else

 Set Pop = Tail

 If Pop.P\_Node Is Nothing Then

 Set Head = Nothing

 Set Tail = Nothing

 Else

 Set Tail = Pop.P\_Node

 End If

 End If

End Function

Public Sub Remove\_Last()

Dim Last\_Node As Cls\_Node

 Set Last = Me.Pop

 Set Last = Nothing

End Sub

Public Sub Bkw\_View()

Dim s As String

 If Tail Is Nothing Then

 Debug.Print "The List is empty"

 Else

 s = View(Tail)

 Debug.Print s

 End If

End Sub

Private Function View(Node As Cls\_Node) As String

 View = Node.Representation

 If Not (Node.P\_Node Is Nothing) Then

 View = View & vbCrLf & View(Node.P\_Node)

 Else

 View = View & vbCrLf & "--- End ---"

 End If

End Function

**Prova LISTA**

Public Sub Prova\_Nodi()

Dim N1 As Cls\_Node, N2 As Cls\_Node, N3 As Cls\_Node

 Set N1 = New Cls\_Node

 Set N2 = New Cls\_Node

 Set N3 = New Cls\_Node

 N1.ID = "N1"

 N1.Content = 1

 N2.ID = "N2"

 N2.Content = "A"

 N3.ID = "N3"

 N1.Show

 N2.Show

 N3.Show

 N2.Link\_Node N3

 N2.Link\_Node N1, False

 Debug.Print "The linked nodes of N2"

 N2.N\_Node.Show

 N2.P\_Node.Show

 On Error Resume Next

 Debug.Print "The linked nodes of N1"

 N1.N\_Node.Show

 N1.P\_Node.Show

 Debug.Print "The linked nodes of N3"

 N3.N\_Node.Show

 N3.P\_Node.Show

End Sub

Public Sub Prova\_Lista()

Dim N1 As Cls\_Node, N2 As Cls\_Node, N3 As Cls\_Node, N4 As Cls\_Node, N5 As Cls\_Node, LastNode As Cls\_Node

Dim N\_List As Cls\_LL

 Set N1 = New Cls\_Node

 Set N2 = New Cls\_Node

 Set N3 = New Cls\_Node

 Set N4 = New Cls\_Node

 Set N5 = New Cls\_Node

 Set N\_List = New Cls\_LL

 N1.ID = "A"

 N1.Content = 1

 N2.ID = "B"

 N2.Content = 2

 N3.ID = "C"

 N3.Content = 3

 N4.ID = "D"

 N4.Content = 4

 N5.ID = "E"

 N5.Content = 5

 N\_List.Push N1

 Set LastNode = N\_List.Pop

 Debug.Print "Last"

 LastNode.Show

 'Ora la lista è nuovamente vuota

 N\_List.Bkw\_View

 'Set LastNode = N\_List.Pop

 N\_List.Push LastNode

 N\_List.Push N2

 N\_List.Push N3

 N\_List.Push N4

 N\_List.Push N5

 N\_List.Bkw\_View

 N\_List.Remove\_Last

 N\_List.Remove\_Last

 N\_List.Remove\_Last

 N\_List.Remove\_Last

 N\_List.Remove\_Last

End Sub