2) Func recursiva que emita la suma de los naturales hasta n inclusive

Func SumaN(Val n:entero): entero

Si n > 1 entonces

SumaN 🡨 n + SumaN(n - 1)

Sino

SumaN 🡨 1

Fsi

Ffunc

4) Func CantDigitos(Val N:entero):entero

Si n < 0 entonces

N 🡨 n\*(-1)

CantDigitos 🡨 cantdigitos(n)

Sino

Si n > 9 entonces

Cantdigitos 🡨 1 + cantdig(n div 10)

Sino

Cantdigitos 🡨 1

Fsi

Fsi

Ffunc

5)Func potencia( val n, Exp: entero): entero

Si Exp < 0 entonces

N 🡨 1/n

Exp 🡨 Exp \* (-1)

Potencia 🡨 potencia(n, exp)

Sino

Si exp = 0 entonces

Potencia 🡨 1

Sino

Potencia 🡨 n \* Potencia(n, exp - 1)

Fsi

Fsi

Ffunc

6) Func producto( val m, n: entero): entero

Si m < 0 y n < 0 entonces

Producto 🡨 producto(m, n)

Sino

Si n < 0 entonces

Producto 🡨 producto(-m, -n)

Sino

Si n > 0 entonces

Producto 🡨 m \* producto(m, n - 1)

Sino

Producto 🡨 1

Fsi

Fsi

Fsi

Ffunc

8) Pre = {x = 0, y = n}

proced pares(val n, x, y: entero)

Si n < 0 entonces

Si x >= y entonces

Mostrar(“(x, y)”)

Pares(n, x -1, y +1)

Fsi

Sino

Si x <= y entonces

Mostrar(“(x, y)”)

Pares(n, x + 1, y - 1)

fsi

Fsi

Fproced

9) pre = {pos = 1}

proced minelem(Ref A: arreglo[Max] de reales, val Tope: entero, ref pos: entero, ref min: reales)

si tope > 1 entonces

minelem(A, tope - 1, pos, min)

si a[tope] < min entonces

min 🡨 a[tope]

pos 🡨 tope

fsi

sino

min 🡨 a[tope]

pos 🡨 tope

fsi

FProced

11) pre = {n > 0}

func Especular(val n: entero): entero

Var local: cant: entero

Si n > 0 entonces

Cant 🡨 cantdigitos(n)

Especular 🡨 n Resto 10 \* potencia(10, cant - 1) + Especular(n div 10)

Sino

Especular 🡨 0

Fsi

Ffunc

13) pre = {Tope > 0}

Proced mostrarpares(ref A:Arreglo[max] de enteros, val tope: entero)

Si tope > 0 entonces

Si tope resto 2 = 0 entonces

Mostrar(A[pos])

fsi

mostrarPares(A, tope - 1)

fsi

Fproced

14) pre = {n > 0}

Func Raizcuadrada(val A: entero): real

Si A > 1 entonces

Raízcuadrada 🡨 1/ 2 \* (Raizcuadrada(A - 1)+ (A / raizCuadrada(A-1)))

Sino

RaizCuadrada 🡨1

Fsi

Ffunc

15) pre = {t1 = t2 > 0}

proced productoEscalar(ref A1, A2: arreglo[max] de enteros, val t1, t2: entero, ref Prod: entero)

si t1 > 0 entonces

productoEscalar(A1, A2, t1 – 1, t2 – 1, prod)

prod 🡨 prod + A1[t1] \* A2[t2]

sino

Prod 🡨 0

Fsi

Fproced

1. Func Existe(ref A: arreglo[Max] de enteros, val tope, x: entero): lógico

Si tope > 0 entonces

Si A[tope] = x entonces

Existe 🡨 verdadero

Sino

Existe 🡨 existe(A, tope – 1, x)

Fsi

Sino

Existe 🡨 falso

Fsi

Ffunc

1. Func BusquedaBinaria(ref A: arreglo[max] de enteros, Val pr, ult, X: entero): lógico

Var locales: central: entero

Si pr < = ult entonces

Si A[pr] > A[ult] entonces

Central 🡨 (pr + ult) div 2

Si A[central] = X entonces

BusquedaBinaria 🡨 verdadero

Sino

Si A[central] > x entonces

BúsquedaBinaria 🡨 BusquedaBinaria (A, central + 1, ult)

Sino

BusquedaBinaria 🡨 BusquedaBinaria (A, pr, ult - 1)

Fsi

16) Func SumaElem(refA: arreglo[max] de enteros, val tope: entero): entero

Si tope > 0 entonces

SumaElem 🡨 a[tope] + SumaElem(A, Tope - 1)

Sino

SumaElem 🡨 0

Fsi

Ffunc

17) proced Mostarelementos(Ref A:Arreglo[Max] de enteros, Val tope: entero)

Si tope > 0 entonces

Mostrar(A[tope])

MostrarElementos(A, Tope - 1)

Fsi

Fproced

18) Proced MostrarInverso(Ref A: Arreglo[max] de enteros, val tope: entero)

Si Tope > 0 entonces

MostrarElementos(A, tope - 1)

Mostrar(A[tope])

Fsi

Fproced

19) pre = {inicio = 1, tope > 0}

Func EsPalindromo(Ref A: arreglo[Max] de caracteres, val inicio, tope: entero): lógico

Si inicio < tope entonces

Si A[tope] = A[inicio] entonces

EsPalindromo 🡨 EsPalindromo(A, inicio, tope)

Sino

EsPalindromo 🡨 falso

Fsi

Sino

Espalindromo 🡨 verdadero

Fsi

Ffunc

20) pre = {t1 > 0, T2 = 1}

Proced InvertirArreglo(Ref A1,A2: arreglo[Max] de caracteres, val T1, T2: entero)

Si T1 > 0 entonces

InvertirArreglo(A1, A2, T1 - 1, T2 + 1)

A2[T2] 🡨 A1[T1]

FSi

Fproced

21)

a) func productoA(ref A:arreglo[Max] de enteros, ref tope: entero): entero

Si tope > 1 entonces

ProductoA 🡨 ProductoA \* A[tope] \* ProductoA(A, tope - 1)

Sino

ProductoA 🡨 A[tope]

Fsi

Ffunc

1. func Posicion(Ref A:arreglo[max] de enteros, Val Tope, x: entero): entero

si tope > 0 entonces

si a[tope] = x entonces

Posicion 🡨 Tope

Sino

Posicion 🡨 posición(A, tope – 1, x)

Fsi

Sino

Posición 🡨 0

Fsi

Ffunc

1. func CantApariciones(Ref A:Arreglo[Max] de tdato, val tope, x: entero): entero

si Tope > 0 entonces

si A[Tope] = x entonces

CantApariciones 🡨 1 + CantApariciones(A, tope - 1, x)

Sino

CantApariciones 🡨 CantApariciones(A, tope - 1, x)

Fsi

Sino

CantApariciones 🡨 0

Fsi

Ffunc

1. pre={Tope > 0, inicio = 1}

proced Imprimir(Ref A:arreglo[MAX] de enteros, val tope, inicio: entero)

si inicio < tope entonces

MostarDesde(A, inicio, tope)

Imprimir(A, inicio + 1, tope)

Sino

Mostrar(A[tope])

Fsi

Fproced

Proced mostrarDesde(Ref A: arreglo[Max] de enteros, val inicio, tope: entero)

Si inicio <= tope entonces

Mostrar(A[inicio])

MostrarDesde(A, inicio+1, tope)

Fsi

Fproced

1. pre = {T1 = T2 > 0}

Func arriguales(ref A1, A2: arreglo[Max] de enteros, val T1, T2: entero): lógico

si t1 > 0 entonces

si a1[t1] = a2[t2] entonces

Arriguales 🡨arriguales(A1, A2, T1 - 1, T2 - 2)

Sino

Arriguales 🡨 falso

Fsi

Sino

Arriguales 🡨 verdadero

Fsi

Ffunc

1. pre {Tope = pos > 0}

proced Promedio(A: arreglo[Max] de enteros, val tope, pos: enteros, ref Prom: real)

Var local: suma: enteros

si pos > 0 entonces

promedio(A, tope, pos - 1, prom )

suma 🡨 suma + A[Tope]

Si Tope = pos entonces

Prom 🡨 suma / tope

fsi

sino

Suma 🡨 0

Fsi

Fproced

22) a) Pre = {Aux = 0}

proced Mostrarlista(Ref L:tlista, val Aux: entero)

Var local: dato: Tdato

si Aux = 0 entonces

Si no Lvacia(L) entonces

Lppio(L)

Linfo(L, Dato)

Mostrar(Dato)

Lsig(L)

Mostrarlista(L, 1)

Fsi

Sino

Si no lfin(L) entonces

Linfo(L, Dato)

Mostrar(Dato)

Lsig(L)

MostrarLista(L, Aux)

Fsi

Fsi

Fproced

b) pre = {Aux = 0}

proced MostrarListaInverso(Ref L:Tlista, val Aux: entero)

si Aux = 0 entonces

si No Lvacia(L) entonces

Lppio(L)

Linfo(L, Dato)

MostrarListaInverso(L, 1)

Mostrar(Dato)

Fsi

Sino

Lsig(L)

Si No lfin(L) entonces

Linfo(L, dato)

MostrarListaInverso(L, Aux)

Mostrar(Dato)

Fsi

Fsi

Fproced

23) pre = {Aux = 0 }

Func Existe(Ref L:Tlista, val Aux: entero, val x: Tdato): lógico

Var local: dato: tdato

Si aux = 0 entonces

si No Lvacia(L) entonces

Lppio(L)

Linfo(L, dato)

Si Dato = x entonces

Existe 🡨 verdadero

Sino

Lsig(L)

Existe 🡨 existe(L, 1, x)

Fsi

Sino

Existe 🡨 falso

Fsi

Sino

Si No lfin(L) entonces

Linfo(L, dato)

Si Dato = x entonces

Existe 🡨 verdadero

Sino

Lsig(L)

Existe 🡨 existe(L, aux, x)

Fsi

Sino

Existe 🡨 falso

Fsi

Fsi

Ffunc

24) pre = {tf=tc > 0}

Func MinimoMat(Ref m: Matriz[Maxf][Maxc] de enteros, val Tf, tc: entero): entero

Var local: min1, min2: entero

Si tf > 1 entonces

Min1 🡨 minimomat(m, tf – 1, tc - 1)

Min2 🡨 minimoarr(M, tf, tc)

Si Min 1 < min 2 entonces

Minimomat 🡨 Min1

Sino

MinimoMat 🡨 min2

Fsi

Sino

Minimomat 🡨 minimoarr(m, Tf, tc)

Fsi

Ffunc

Func minimoarr(ref M:Matriz[MaxF][MaxC] de enteros, val tf, tc: entero): entero

Var local: min1, min2: entero

Si tc > 1 entonces

Min1 🡨 minimoarr(M, tf, tc - 1)

Min2 🡨 M[tf][tc]

Si Min1 < Min2 entonces

Minimoarr 🡨 min1

Sino

Minimoarr 🡨 min2

Fsi

Sino

Minimoarr 🡨 m[tf][tc]

Fsi

Ffunc

25) pre={tf > 0, tc > 0, ta = 0}

Proced Maximosmat(ref M:matriz[Maxf][MaxC] de enteros, val tf, tc: entero, ref a: arreglo[Max] de enteros, val ta: entero)

Si tf > 1 entonces

MaximosMat(M, tf – 1, tc, a, ta)

Max 🡨 maximoarr(M, tf, tc)

Ta 🡨 ta + 1

A[ta] 🡨 max

Sino

Max 🡨 maximoarr(M, tf, tc)

Ta 🡨 ta + 1

A[ta] 🡨max

Fsi

Fproced

Func MaximoArr(ref M:matriz[maxf][maxc] de enteros, val tf, tc: entero)

Var local: max1, Max2: enteros

Si tc > 1 entonces

Max1 🡨 Maximoarr(M, tf, tc - 1)

Max2 🡨 M[tf][tc]

Si Max1 > Max2 entonces

Maximoarr 🡨 max1

Sino

Maximoarr 🡨 max2

Fsi

Sino

Maximoarr 🡨 m[tf][tc]

Fsi

Ffunc

26) pre ={}

Func essimetrica(Ref M:matriz[Maxf][Maxc] de enteros, val tf, tc: entero): lógico

Si tf > 0 entonces

si SonIguales(M, tf, tc- 1, tf-1, tc) entonces

EsSimetrica 🡨 essimetrica(M, tf - 1, tc - 1)

Sino

EsSimetrica 🡨 falso

Fsi

Sino

EsSimetrica 🡨 verdadero

Fsi

Ffunc

Func SonIguales(ref M: matriz[Maxf][Maxc] de enteros, val tf1, tc1, tf2, tc2: entero): lógico

Si tc1 > 0 entonces

Si m[tf1][tc1] = m[tf2][tc2] entonces

Soniguales 🡨 sonigusles(m, tf1, tc1 – 1, tf2 – 1, tc2)

Sino

Soniguales 🡨 falso

Fsi

Sino

Soniguales 🡨 verdadero

Fsi

Ffunc

27) func Maxtrianginf(ref M: matriz[maxf][maxc] de enteros, val tf, tc: entero): entero

Si tf > 2 entonces

Max1 🡨Maxtrianginf(M, tf-1, tc-1)

Max2 🡨 maximoarr(M, tf, tc - 1)

Si Max1 > Max2 entonces

Maxtrinaginf 🡨 Max1

Sino

Maxtrianginf 🡨 max2

Fsi

Sino

Maxtrianginf 🡨 maximoarr(M, tf, tc - 1)

Fsi

Ffunc

29) pre = {n = 0}

func taylorseno(val x: real, val Tol: real, Val n: entero): real

var local: t: real

T 🡨 potencia(-1, n) \* potencia(x, 2\*n +1) / factorial(2\*n+1)

Si modulo(T) > tol entonces

TaylorSeno 🡨 taylorseno(x, tol, n + 1)

Sino

TalorSeno 🡨 t

Fsi

Ffunc

30) proced caractRepetidos(ref M:matriz[maxf][Maxc] de caracteres, val tf, tc: enteros, ref A: arreglo[max] de enteros, val tope: entero)

Si Tc > 1 entonces

Caractrepetidos(M, tf, tc – 1, A, tope)

Tope 🡨 tope + 1

Cant 🡨 Cantrepeticiones(M, tf, tc)

A[Tope] 🡨 cant

Sino

Tope 🡨 tope + 1

Cant 🡨 cantrepeticiones(M, tf, tc)

A[tope] 🡨 cant

Fsi

Fproced

Func Cantrepeticiones(ref M: matriz[Maxf, MaxC] de caracteres, val tf, tc: enteros)

Si tc > 0 entonces

Si M[Tf][Tf] = m[tf][Tc] entonces

CantRepeticiones 🡨 cantrepeticiones(M, tf, tc - 1) + 1

Sino

CantRepeticiones 🡨 cantrepeticiones(M, tf, tc)

Fsi

Sino

Cantaparciciones 🡨 0

fsi

Ffunc