OBJETIVO

- Proporcionar los elementos básicos de la programación estructurada.

TEMARIO

- Metodología de la programación estructurada.

METODOLOGÍA DE LA PROGRAMACIÓN

- Secuencial

- Estructuras Cíclicas:
  - FOR – NEXT
  - REPITE – HASTA
  - MIENTRAS
- Decisión
  - SI – ENTONCES
  - SI – ENTONCES – CASO CONTRARIO
  - CASO <CASE>

METODOLOGÍA DE LA PROGRAMACIÓN ESTRUCTURADA.

- Tipos de datos
- Estructuras
- Arreglos
- Funciones

TIPOS DE DATOS

- ENTEROS
  - cortos
  - largos
- REALES
  - cortos
  - largos
- CARÁCTER
  - uno
  - cadena

COMPUESTOS
(ESTRUCTURAS)

TIPOS DE INSTRUCCIONES

SEUDOCÓDIGO

DIAGRAMA DE FLUJO

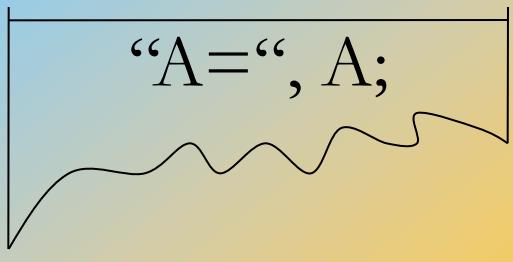
- INICIO

- FIN

- DECLARACIÓN DE VAR

- ASIGNACIÓN

- LECTURA



- ESCRITURA

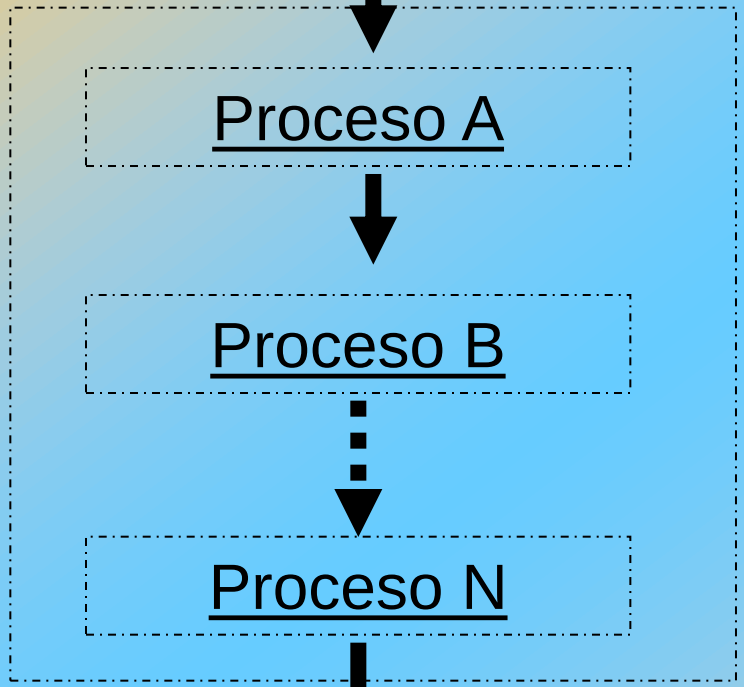
ALGORITMO

Conjunto finito de instrucciones lógicas a seguir para resolver un problema específico.

- Características de un algoritmo:
  - Finito : Instrucciones de INICIO y FIN
  - Claro
  - Eficaz
  - Eficiente
- Las instrucciones se ejecutan de arriba hacia abajo.

SECUENCIAL:

Diagrama de flujo



Seudocódigo

INICIO

Proceso A

Proceso B

Proceso N

FIN

MEMORIA PRINCIPAL

Lugar donde se almacena la información.

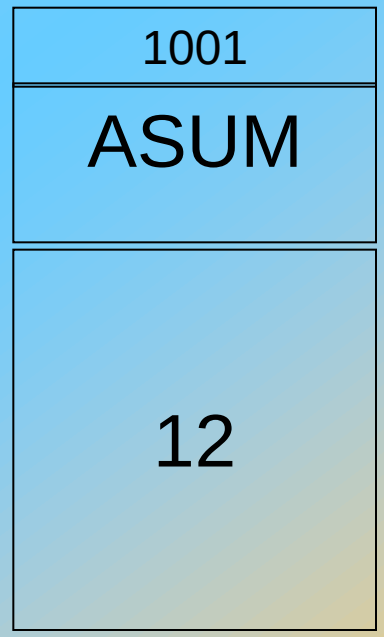
DIRECCIÓN DE MEMORIA

REAL

ASIGNADA POR

EL USUARO

CONTENIDO



DECLARACIÓN DE VARIABLES

Tipo de variable NOM_VAR1,NOM_VAR2 ;

Tipo de variable:

Entera int V1,V2,..Vn;

Real float A,B,C;

Carácter char a, B[5];

a ; almacena un carácter

B[5] ; almacena 5 caracteres.

DECLARACIÓN DE VARIABLES

Tipo de variable NOM_VAR1,NOM_VAR2 ;

Todas las variables tienen que declararse e inicializarlas.

Cuando el compilador crea la dirección de memoria le asigna un valor cualquiera (basura).

EXPRESIÓN ARITMÉTICA

- **Conjunto de constantes y/o variables unidas mediante operadores aritméticos.**

OPERADORES ARITMÉTICOS.

MULTIPLICACIÓN	*
DIVISIÓN	/
SUMA	+
RESTA	-

USO DE PARÉNTESIS PARA ROMPER LA JERARQUÍA.

EXPRESIÓN ARITMÉTICA

- VARIABLE

  Es la dirección de memoria asignada por el usuario

  TODA VARIABLE TIENE QUE DECLARARSE

- CONSTANTE

  Cantidad que no cambia de valor.

  Contenido de una dirección de memoria.

EXPRESIÓN ARITMÉTICA

- Los nombres de variables dependen del lenguaje de Programación. (No se permite utilizar palabras reservadas por el compilador o intérprete como nombre de variable)
- Ejemplo.

```
INICIO /* inicio de algoritmo */
        ENTERAS A, B, C;
        A = 3;   B = A +2;   C = A+B;
FIN
```

ESTRUCTURA DE UN PROGRAMA EN LENGUAJE C

ARCHIVOS DE CABECERA

```
#include<stdio.h>
#include<conio.h>
```

```
void main(void)
{ /*  INICIO */
```

```
int main(void)
{
```

DECLARACIÓN DE VARIABLES

```
    int A;
    A =3;
```

BLOQUE DE INSTRUCCCIONES

```
    return(0);
}
```

```
}
```

ACUMULADOR DE SUMA

INICIO

ENTERO A;

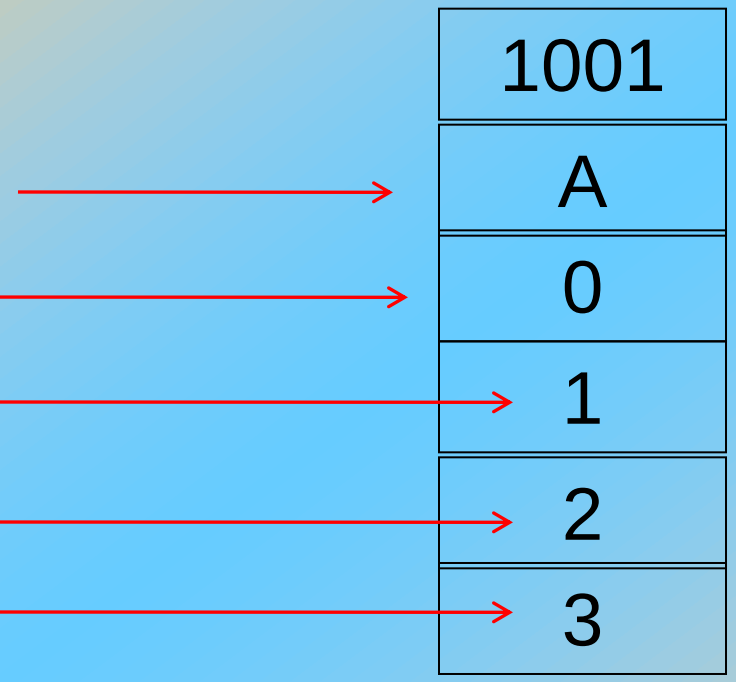
A = 0 ;

A = A + 1;

A = A + 1;

A = A + 1;

FIN



ACUMULADOR DE SUMA

INICIO

ENTERO A;

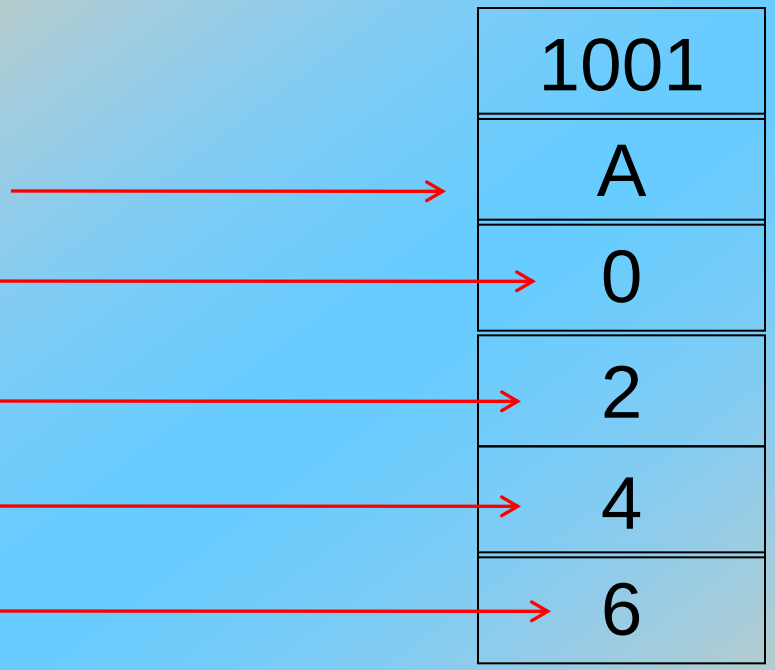
A = 0;

A = A + 2;

A = A + 2;

A = A + 2;

FIN



ACUMULADOR DE SUMA

INICIO

ENTERO A;

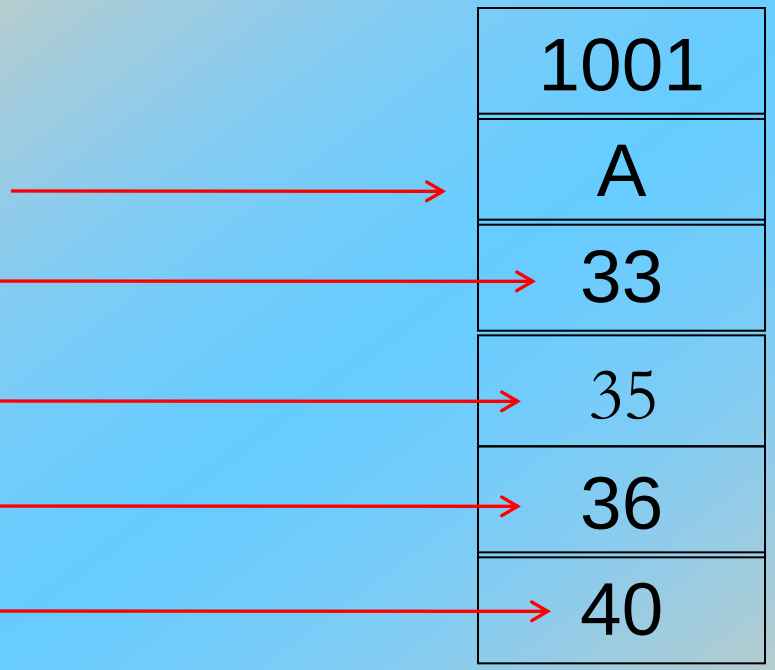
A = 33;

A = A + 2;

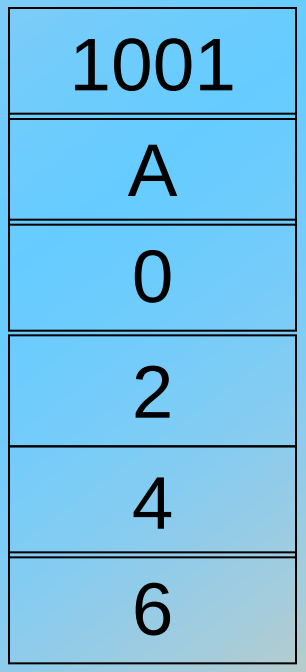
A = A + 1;

A = A + 4;

FIN



ACUMULADOR DE SUMA



NOTA:

EL PROCESO DE ASIGNACIÓN DE MEMORIA ES UN PROCESO DESTRUCTIVO.

VARIABLE TEMPORAL

A = 11;

B = 21;

UNO

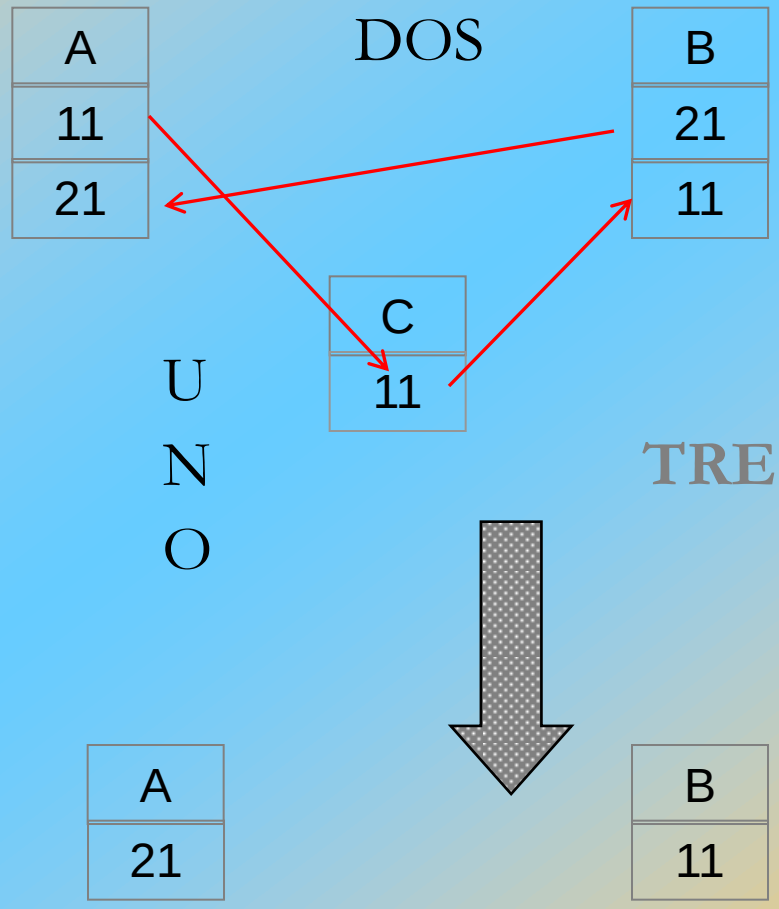
C = A;

DOS

A = B;

TRES

B = C;



INSTRUCCIÓN DESPLEGAR

(“tipo de formato y/o texto y/o códigos de control”, A, B, C);

Tipo de variable	Tipo de formato
Entera	%i o %d
Entera larga	%li o %ld
Real	%f
Real larga	%lf
Carácter	%c o %s

Códigos de control : \n, \t

INSTRUCCIÓN DESPLEGAR

Codificación:

printf(“tipo de formato y/o texto y/o códigos de control”, A, B, C);

INSTRUCCIÓN DE LECTURA

(“tipo de formato ”, &A, &B, &C);

INSTRUCCIÓN DE LECTURA

Codificación:

scanf(“tipo de formato”, &A, &B, &C);

EXPRESIÓN ARITMÉTICA

- EJEMPLO:

INICIO

REALES: A, B, C, X1, D;

Lee &A, &B, &C;

D = B * B-4*A

X1= (-B – sqrt(D))/(2*A);

Desplegar “X1=“,X1;

FIN

EXPRESIÓN LÓGICA

EXPRESIÓN ARITM. 1	OPERADOR LÓGICO	**EXPRESIÓN ARITM.2**

OP. LÓGICOS	CODIFICACIÓN	EJEMPLOS
>	>	A > 5
≥	>=	A >= B * C
<	<	SUMA < 9
≤	<=	7 <= 11
≠	!=	A != 7
=	==	B==4

ESTRUCTURAS

- Secuencial
- Selección
  - Si-entonces
  - Si-entonces-caso contrario
- Cíclicas
  - Mientras (while)
  - Repite-Hasta (do-while)
  - Para (for)

ESTRUCTURAS SECUENCIAL

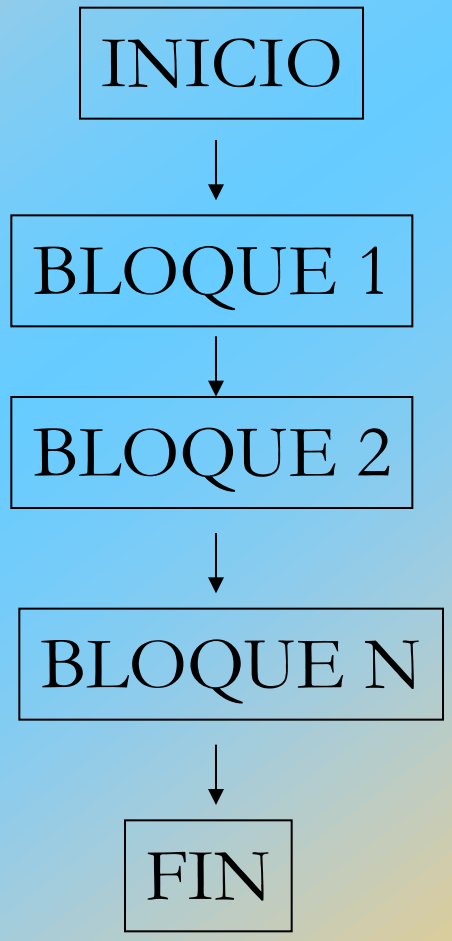
Inicio

Bloque 1

Bloque 2

Bloque N

Fin



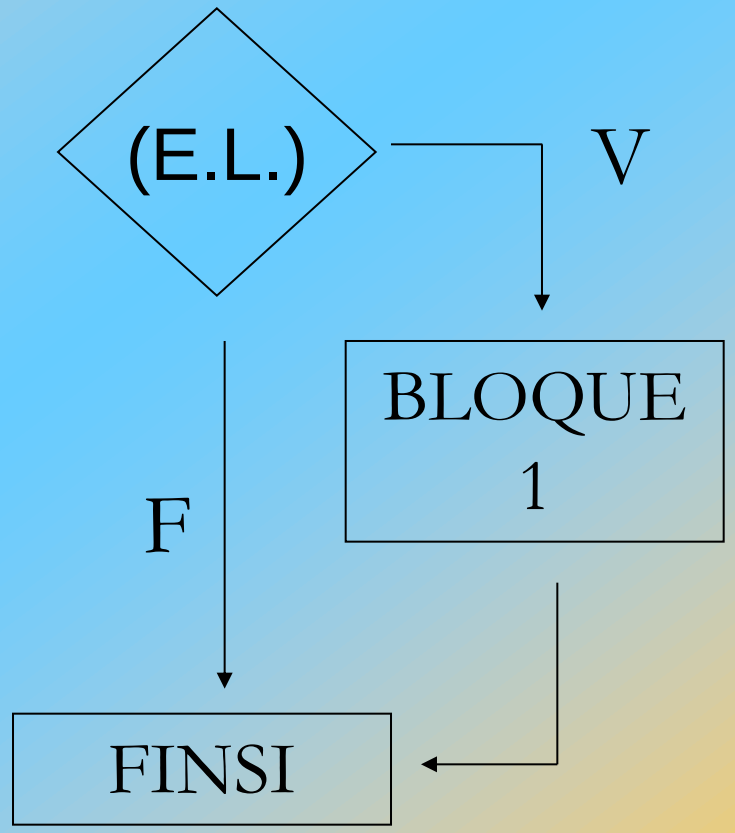
ESTRUCTURA SELECCIÓN SI ENTONCES

Si (Exp. Lógica)

Entonces

Bloque de instrucciones

Finsi



ESTRUCTURA SELECCIÓN SI ENTONCES-CASO CONTRARI0

Si (Exp. Lógica)

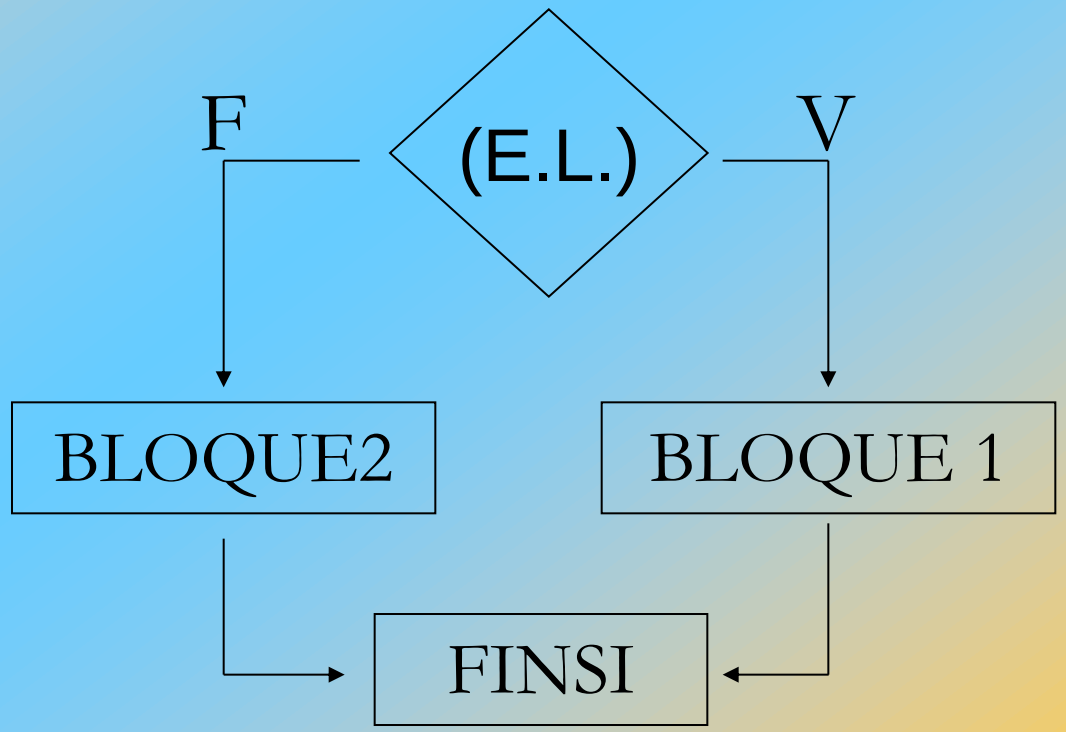
Entonces

Bloque 1

Caso Contrario

Bloque 2

Finsi



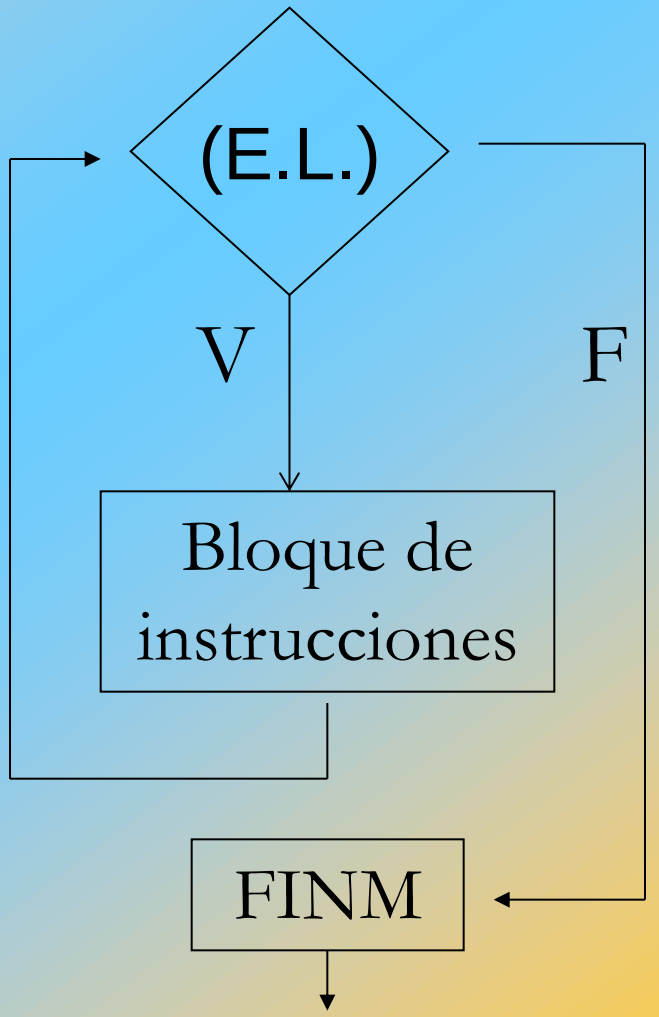
ESTRUCTURAS CÍCLICAS

Mientras (EXP LOG)

{

Bloque de instrucciones

}



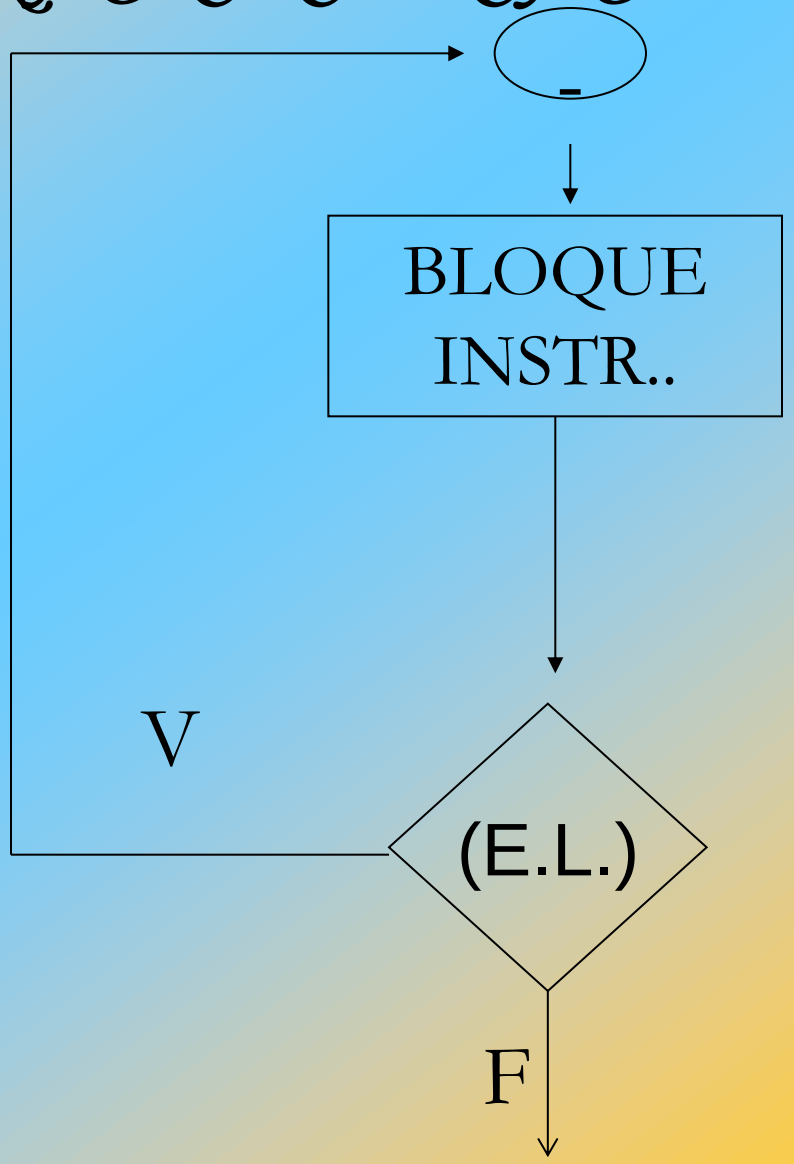
ESTRUCTURAS CÍCLICAS

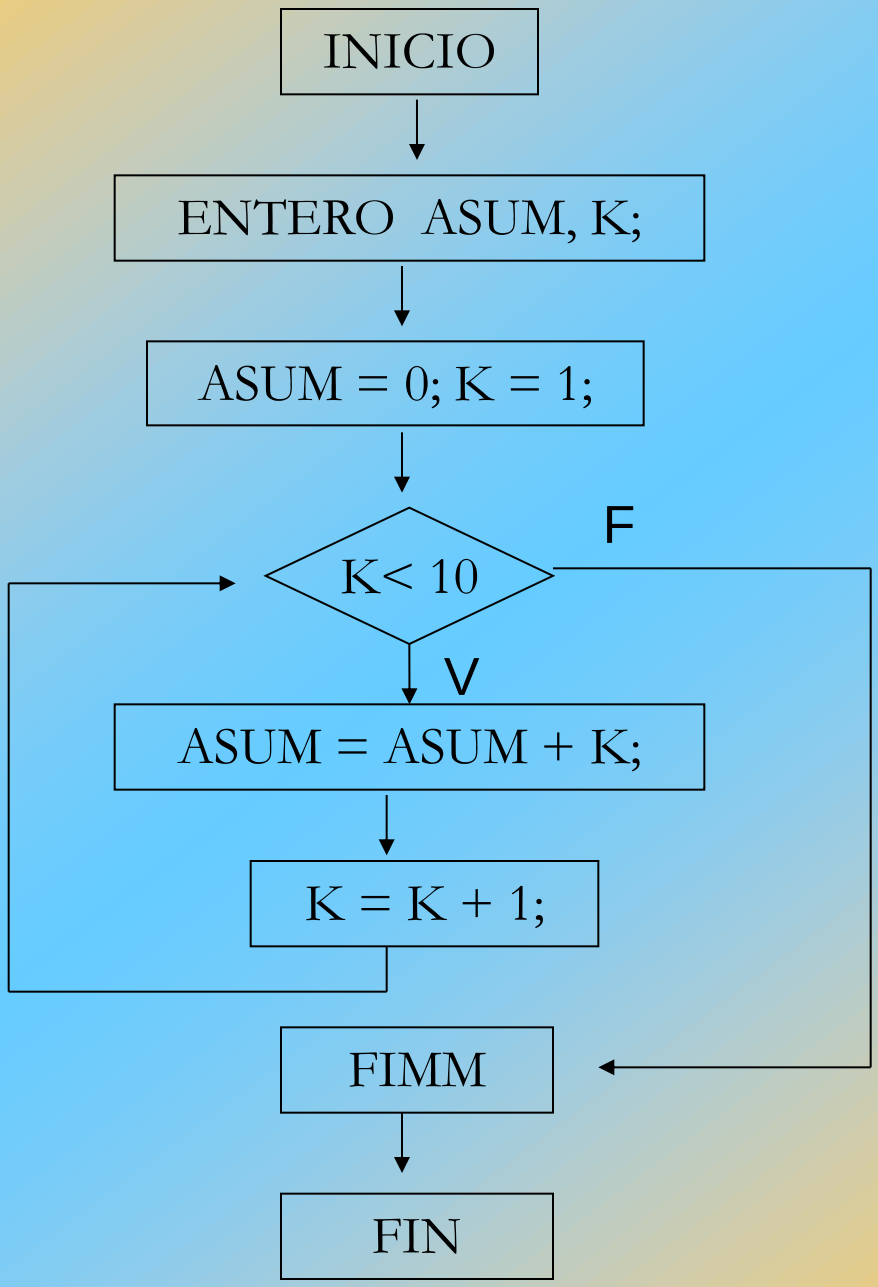
Realiza

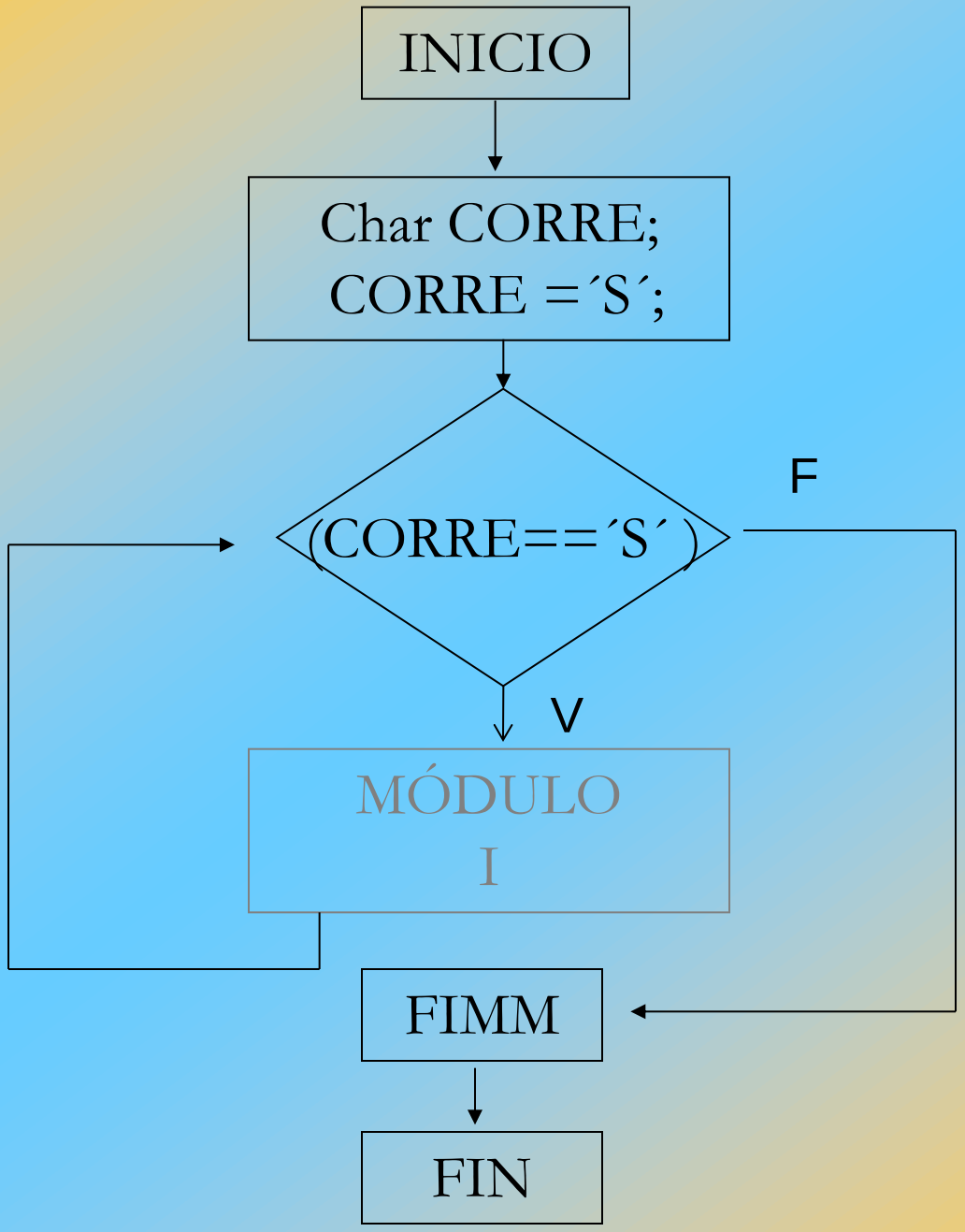
{

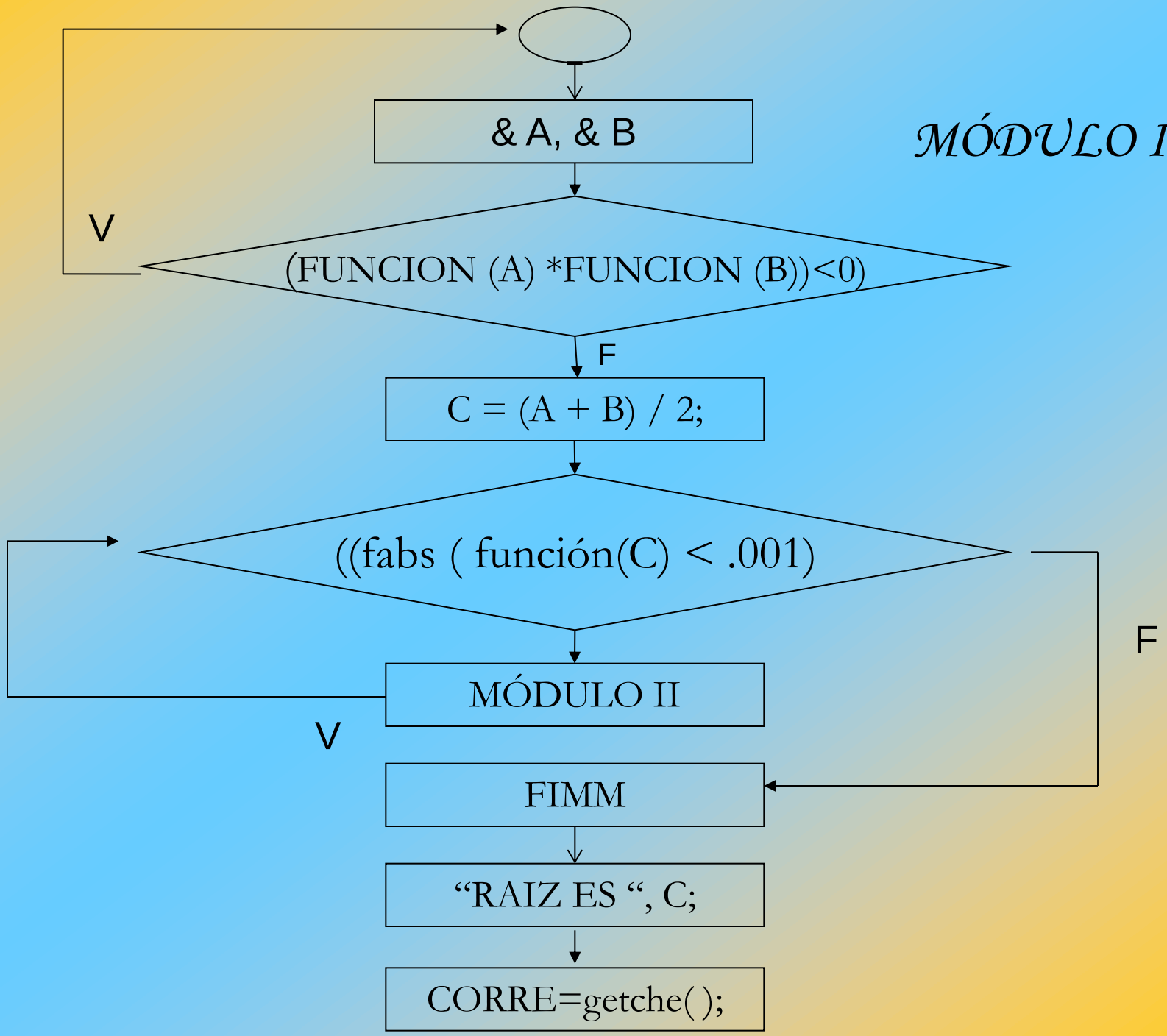
Bloque de instrucciones

}Mientras (Exp. Lógica);



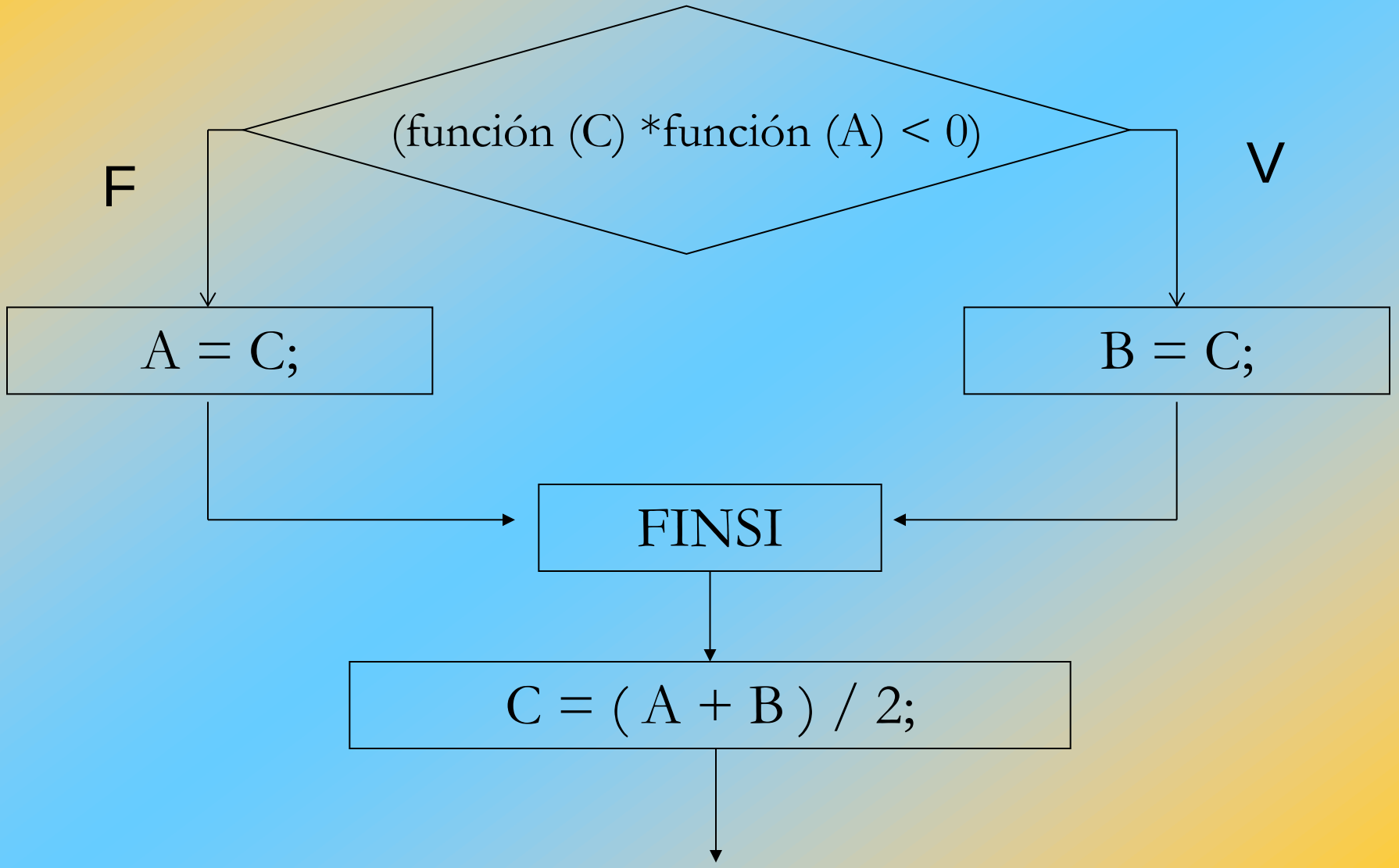

INICIO
ENTERO ASUM, K;
ASUM = 0; K = 1;
K< 10
F
V
ASUM = ASUM + K;
K = K + 1;
FIMM
FIN


INICIO
Char CORRE;
CORRE =´S´;
(CORRE==´S´)
F
V
MÓDULO
I
FIMM
FIN


& A, & B
V
(FUNCION (A) *FUNCION (B))<0)
F
C = (A + B) / 2;
((fabs (función(C) < .001)
F
MÓDULO II
V
FIMM
“RAIZ ES “, C;
CORRE=getche( );

MÓDULO I

MÓDULO II



CURSO DE MÉTODOS NUMÉRICOS

FUNCIONES

FUNCIONES

- Una función es un programa que trabaja como un módulo independiente y que puede o no transferir valores de la función principal hacia éste y regresar o no un valor de la función a la función principal (main( )).

TIPOS DE FUNCIONES

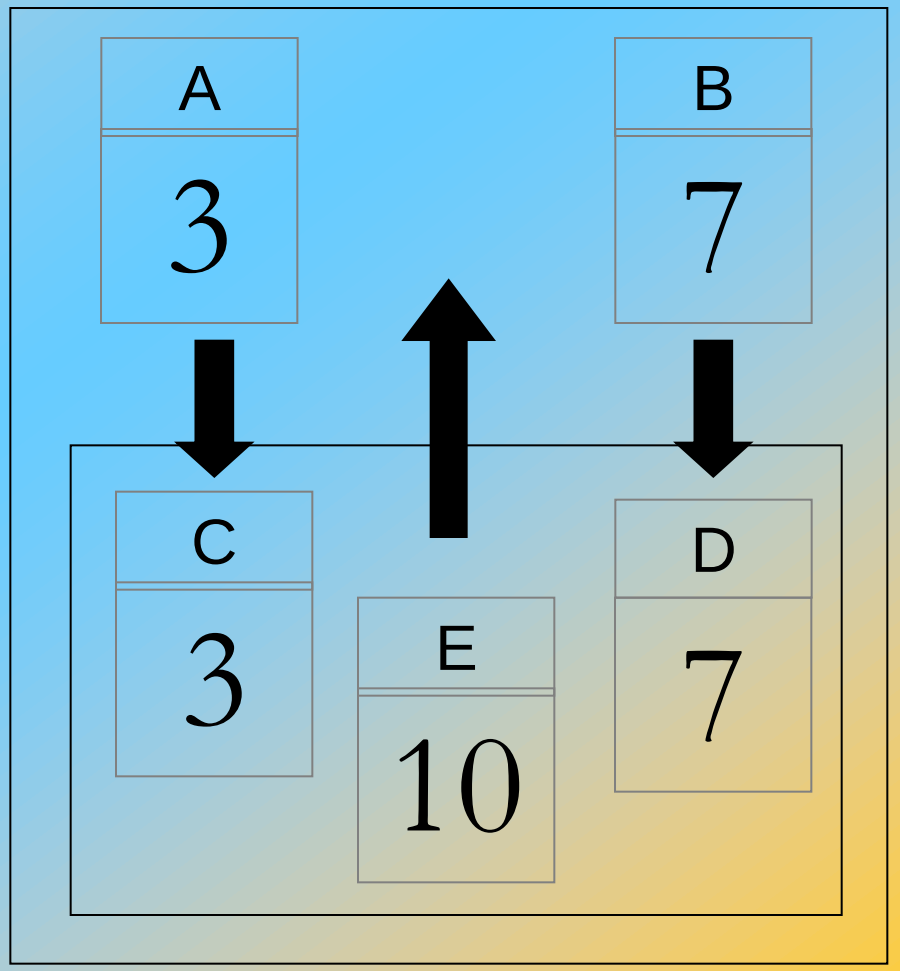
```
FUNCIÓN main ( )
{
   mensaje ( );
}

void mensaje (     void   )
    {
    Despliega“HOLA”;
    }
```

NO TIENE ARGUMENTOS

```
FUNCIÓN main ( )
{
 Entero A, B;
 Leer &A, &B;
 Desplegar A, B, suma (A,B); }
Ent suma ( Ent C, Ent D)
{
    Entero E;
    E = C + D;
    Regresa ( E );
}
```

VARIABLES LOCALES

```
FUNCIÓN main ( )
{
Entero A, B;
Leer &A, &B;
Desplegar A, B, suma(A,B);
}
  Ent suma ( Ent C, Ent D)
 { Entero E;
    E = C + D;
    Regresa ( E );
 }
```



FUNCIÓN SUMA

```
FUNCIÓN main ( )
{
Entero A, B;
Leer &A, &B;
Suma ( A, B,);
}
Void suma ( Ent C, Ent D)
{
Entero E;
E = C + D;
Desplegar C, D, E;
}
```

```
FUNCIÓN main ( )
{
Suma ( );
}
void  suma ( void)
{
Entero A, B, D;
Leer &A, &B;
C = A + B
Desplegar  A, B, C;
}
```

FUNCIÓN FACTORIAL

```
Entera factorial (   Entera D)

 {Entera C = 1, fac = 1;
  Mientras (C< =D)
  {
  fac = fac *C; C++;
  }
  regresa (fac);
 }
void main ( )
{
Entero K;
Leer &K;
Desplegar  K, factorial ( K);
}
```

RECURSIVIDAD

```
Entera factorial (   Entera N)
 { Entera f =1;
   Si (N > 1 ) entonces
            f = factorial ( N-1)*N;
   caso contrario
            f = 1;
   finsi
 }

void main ( )
{
Entera: K;
Leer &K;
Desplegar K, factorial ( K);
}
```

ARREGLOS

- Un arreglo es un segmento de la memoria principal (RAM), que al asignarle un nombre, junto con su dimensión (número de elementos), se puede almacenar o leer información de ella, haciendo referencia a su posición mediante uno o más indices.

ARREGLOS

- Tipos de arreglos
  - Una dimensión → Lineales (vector)
  - Dos dimensiones → Matriciales
  - Tres dimensiones

ARREGLOS LINEALES

- LEER Y DESPLEGAR UN VECTOR

```
INICIO
    Entero L, M;
     Real A[7];
     Para ( L = 0 ; L < 7; L++)
     {
          lee & A[L];
     }
     Para (M = 0; M < 7; M++)
     {
          desplegar A[M];
     }
Fin
```