Introducción a la Programación y los Lenguajes

Introducción a la Informática

Ing. Soldiamar Matamoros Encalada

Conceptos Básicos

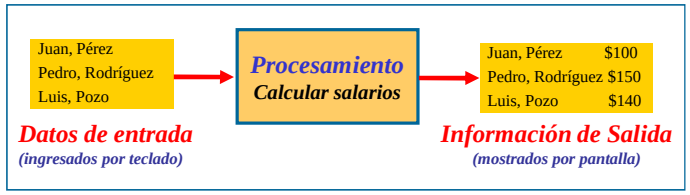
- **Programa**
  - Grupo de instrucciones que indican a la computadora que debe hacer.
  - Es un conjunto de instrucciones que el computador puede "*entender*" y ejecutar.
    - Para poder "*entender*" el significado de una instrucción, estas deben ser escritas en el lenguaje que el computador maneja (0's y 1's).
- **Programación**
  - Es el proceso de escribir estas instrucciones.
  - La programación abarca también un conjunto de técnicas formales e informales que nos permiten escribir programas eficientes y rápidos.

Conceptos Básicos

- **Datos**
  - Representación de hechos y/o eventos, sin ningún significado
  - Sirven de materia prima para los Sistemas de Información
- **Información**
  - Es el resultado del procesamiento de datos
  - Tiene significado para el usuario
  - Puede ser utilizada como "dato" para otro proceso

Conceptos Básicos

- **Procesamiento**
  - Se refiere al grupo de operaciones que se aplican sobre los datos para conseguir la información.
  - La computadora es una máquina que efectúa operaciones para "recibir" datos, "procesarlos" y "mostrar" la información obtenida a quien corresponde.



¿Qué es un Algoritmo?

- **Concepto:** Conjunto de instrucciones que especifican la secuencia ordenada de operaciones a realizar para resolver un problema.
  - En otras palabras, un algoritmo es un método o fórmula para la resolución de un problema.
  - El término algoritmo es muy anterior a la era informática
  - Proviene de Mohammed al-Khowârizmî, matemático persa del siglo IX que enunció paso a paso las reglas para sumar, restar, multiplicar y dividir números decimales.
  - Su apellido se tradujo al latín en la palabra **algoritmus**

Cont…

- El computador sólo puede ejecutar instrucciones sencillas
  - Es necesario descomponer cada acción en un subconjunto de operaciones mas pequeñas.
- El computador ejecuta solo una instrucción a la vez
  - Es necesario establecer un orden lógico para la ejecución de las mismas.
- Un algoritmo es como una receta de cocina, en la cual se debe declarar que ingredientes -datos- son necesarios, como son cocinados -procesados- y cuales son los resultados -información-

Características de un Algoritmo

- Un algoritmo debe ser:
  - **Comprensible o Claro y preciso:** Sin ambigüedades e indicar el orden de realización de cada paso.
  - **Definido y Predecible:** Si se sigue el algoritmo dos veces, partiendo de la misma situación inicial, se debe obtener el mismo resultado cada vez.
  - **Efectivo:** para que todos los pasos puedan llevarse a cabo. Por ejemplo:
    - Suba al piso inferior y camine 3 pasos a la derecha (seria imposible de llevar a cabo).

Cont…

- **Finito**. (Principio y fin) El algoritmo debe terminar en algún momento; o sea, debe tener un número finito de pasos.
  - Un paso mal planteado puede provocar que el algoritmo no encuentre un fin. Por ejemplo:
    - Espere hasta que aparezcan tres hermanos gemelos.

Un algoritmo con las mismas entradas siempre debe producir el mismo resultado.

Cont…

- **Un algoritmo debe describir tres partes:**
  - **Entradas**
    - **Ingredientes y utensilios usados**
  - **Procesos**
    - **Elaboración de la receta en la cocina**
  - **Salidas**
    - **Terminación del plato**

Ejemplo:

- Buscar la palabra PSICOSOMÁTICO en el diccionario.
  - Defina y describa en sus palabras los pasos que debería seguir para buscar la palabra.
  - Generalice los pasos y escriba un conjunto de instrucciones que sirvan para encontrar **cualquier palabra** en un diccionario.

Cont…

- Resolviendo el Problema
  - Para buscar la palabra PSICOSOMATICO:
    - Buscar un diccionario Español
    - Buscar la sección que contiene la letra P
    - Dentro de la sección P,
    - Buscar la siguiente letra S,
    - Buscar la siguiente letra I
    - En la parte de Psico, buscar Psicosomático
    - Escribir el significado en algún lugar

Instrucciones

- **Si tuviese que indicarle a alguien que nunca ha visto un diccionario, como debe usarlo para encontrar palabras, ¿cómo lo haría?**
  - Buscar el diccionario español
  - Buscar entre las secciones alfabéticas del diccionario **hasta** encontrar la sección que coincida con la primera letra de la palabra que buscamos.
  - Buscar sucesivamente letra tras letra de la palabra en la sección, **hasta** encontrar la palabra.
  - Leer el significado.

Formas de Representar un Algoritmo

- Un algoritmo puede ser representado en papel utilizando cualquiera de las siguientes formas:
  - **Lenguaje natural** (en nuestro caso español)
    - Un lenguaje expresado de esta forma, corre el riesgo de no ser suficientemente claro.
  - **Gráficos (Diagramas de Flujo)**
    - Un método que tiene mucha acogida, cada paso, se especifica a través de un gráfico.
  - **Pseudocódigo**
    - Un lenguaje mas formal que el natural, pero suficientemente flexible y fácil para ser comprendido por alguien sin mucha experiencia.

De Algoritmo a Programa

- También podríamos llevar la solución de un problema, al computador, pero ninguna de las técnicas anteriores funcionarían.
  - La computadora solo entiende 0s y 1s.
  - Para comunicarnos con el computador e indicarle órdenes, necesitamos algo más formal, que permita convertir un algoritmo en un programa: un lenguaje de programación.

De Algoritmo a Programa

- Usualmente:
  - Un algoritmo se escribe en papel, y se revisa.
  - Cuando hay seguridad, se lleva a la computadora, donde se escribe el programa (implementación).
  - Se prueba que las órdenes dadas al computador, a través del programa, hagan exactamente lo que deseamos.

Lenguajes de Programación

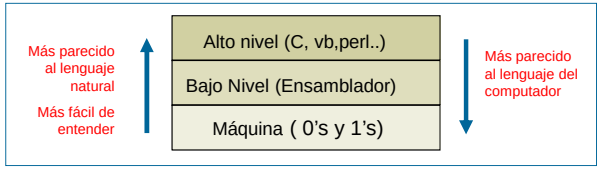
- Un **lenguaje** se puede definir como un conjunto de palabras y formas de expresión por medio de las cuales se comunican y relacionan miembros de una comunidad determinada.
- **Lenguaje de programación**
  - Es el lenguaje a través del cual los seres humanos se comunican con la computadora.
  - Es un conjunto de símbolos y de reglas, que se usan para expresar algoritmos
- Al igual que los lenguajes que usamos para comunicarnos, poseen un:
  - Un **léxico:** vocabulario o conjunto de símbolos permitidos
  - Una **sintaxis:** que indica cómo realizar construcciones del lenguaje y
  - Una **semántica**, que determina el significado de cada construcción correcta.

Cont…

- **Lenguajes de propósito general**
  - **Imperativos** o **procedurales**
    - Se basan en la asignación. Se especifica la secuencia de instrucciones.
    - **De guiones** (***scripts***). Más simples. Suelen ser interpretados.
    - **Internet**: Hipertexto (HTML) acceso a Internet y base de datos (PHP, ASP)
  - **Orientados al objeto**
    - Se basan en la asignación y son más próximos al mundo real (objetos).
  - **Declarativos**
    - Basados en reglas de deducción (**lógicos**) o funciones (**funcionales**).
    - Se especifica el resultado requerido.
  - **Visuales**
    - Entornos que generan código automáticamente.
- **Lenguajes de propósito específico**
  - De robótica (VAL II), bases de datos (SQL), hardware (Step-5).

Lenguajes de Programación

- Las computadoras "hablan" su propio lenguaje (numérico) 1s y 0s.
- Para poder comunicarnos con ellas, existen varias alternativas:
  - Usar el mismo lenguaje de ellas: un lenguaje de máquina
  - Usar un lenguaje parecido al de las computadoras: un lenguaje de bajo nivel
  - Usar un lenguaje parecido al nuestro: un lenguaje de alto nivel



Beneficios de los lenguajes de alto nivel

- Son independientes de la arquitectura física del computador.
- Sin saber ensamblador se pueden crear paquetes de programas.
- Transportabilidad (portabilidad): permiten utilizar los mismos programas en computadores diferentes, con distinto lenguaje máquina.
- Fácil de usar: Las operaciones se expresan con sentencias o frases muy parecidas al lenguajes matemático o al lenguaje natural.
- Legibilidad: Los versiones de programas escritas en lenguajes de alto nivel son mucho más fáciles de leer, modificar y mejorar que las versiones escritas en lenguaje ensamblador.

Traducciones

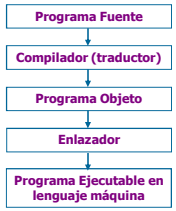
- La máquina no puede entender el lenguaje de programación, entonces de debe "traducir" al lenguaje de máquina.
  - Los lenguajes que la computadora puede entender directamente son máquina y y ensamblador
- La traducción de:
  - lenguaje ensamblador (el código fuente) a un código máquina (o código objeto) no es un proceso muy complicado y se realiza normalmente por un programa especial llamado compilador.
  - lenguajes de alto nivel (código fuente) a un código máquina también se realiza con un:
    - Compilador, en este caso más complejo, o mediante un
    - Intérprete

Traducciones

- **Intérpretes**
  - Recorren el código fuente una línea cada vez.
  - Cada línea se traduce a código máquina y se ejecuta.
  - Son más lentos que los compiladores (no producen un código objeto)
  - Cuando la línea se lee por segunda vez, como en el caso de los programas en que se reutilizan partes del código, debe compilarse de nuevo.
  - Aunque este proceso es más lento, es menos susceptible de provocar fallos en la computadora.
  - Ejemplo: Basic, Qbasic, Smalltalk, Java (necesita un interprete para entender el código en bytes)

Cont...

- **Compilador**
  - Traduce de un lenguaje de programación a código de máquina o ensamblador.
  - Al traducir el código fuente generan un código objeto, creando una lista de instrucciones en código máquina.
  - El código objeto resultante es un programa rápido y listo para funcionar, pero que puede hacer que falle el ordenador si no está bien diseñado.

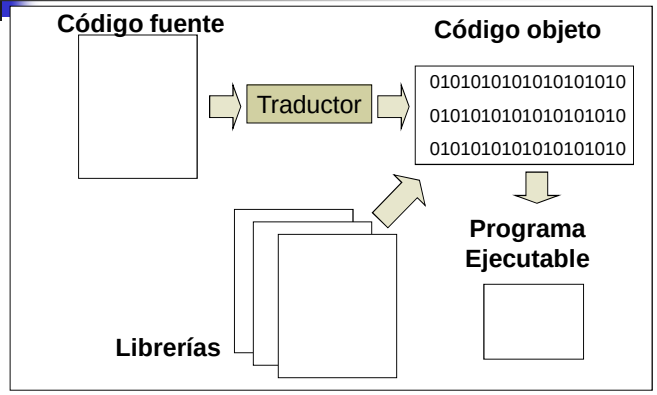


  - **Código fuente:** Son las instrucciones que escribimos en lenguaje de alto nivel.

Enlace (Linking)

- El código objeto es enlazado con otros códigos objetos para producir un **código ejecutable**.
- Estos otros códigos objetos incluyen objetos predefinidos llamados **librerías.** Las librerías son conjuntos de operaciones comúnmente solicitadas por los programas. (Esto lo veremos más adelante).
- El proceso de combinar todos los códigos objetos en un ejecutable es llamado **enlace (linking).**

Proceso de Compilación



Errores de Programación

- Los lenguajes de programación tienen su propio vocabulario y sus propias reglas.
- Estas reglas permiten determinar si las instrucciones están o no construidas apropiadamente y son conocidas como **Reglas de Sintaxis.**
- Los errores producidos al romper estas reglas son conocidos como **Errores de Sintaxis.**
- En caso de existir un error de sintaxis, el compilador nos lo indica para ser corregido.
- Existe otro tipo de error, que en la mayoría de las ocasiones el compilador no lo detecta. Estos son errores en la lógica del programa y son conocidos como "**bugs**".
- El proceso de encontrar y corregir estos errores se llama **depuración (debug).**

Mantenimiento del Software

- El software requiere mantenimiento, principalmente, por 2 razones:
  - Las necesidades de los clientes están en constante cambio, lo que en muchas ocasiones implica cambio en los programas.
  - Los "bugs" pueden sobrevivir incluso luego de realizar pruebas minuciosas.

La importancia de la Ingeniería de Software

- La disciplina de escribir programas que puedan ser de fácil comprensión y mantenimiento se conoce como Ingeniería del Software.
- Algunos consejos:
  - Cuando escribas un programa trata de imaginar lo que alguien puede sentir cuando lo lea dos años después.
    - ¿Lo entenderá?
    - ¿El programa le indicará lo que está tratando de hacer?
    - ¿Será fácil de modificar? .. Entre otras.

Pasos para Crear un Programa

- **El Proceso de Programar**
  - Escribir un programa es un proceso lento.
  - Los programas son escritos para resolver problemas.
  - El proceso de resolver un problema es complejo y se han desarrollado metodologías formales.
  - Para crear un programa, podemos definir los siguientes pasos:
    - Identificación del problema (Leer bien el problema)
    - Análisis y resolución del problema (Diseñar la Solución)
    - Ejecutar o Implementar la solución

Identificar el problema

- **¿Qué hago?**
  - Para identificar el problema se realizan dos actividades:
    - Dada una situación determinada, descubrir cuál es el problema que esta involucra.
    - Crear un enunciado (no mayor de 4 líneas) que exprese con precisión y de manera clara el problema a resolver

Análisis y resolución del problema

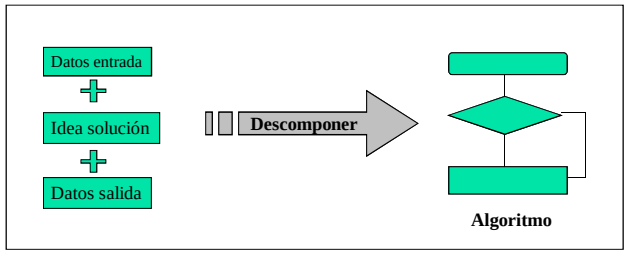
- Luego de comprender bien un problema lo primero que se debe que hacer es:
  - Identificar que resultado se desea obtener (salida)
  - Identificar que datos se necesitan para generar el resultado (entrada)
  - Describir los pasos que nos llevaran al resultado (**algoritmo**)
- **Ejercicios:**
  - Frente a los siguientes problemas, identificar entradas y salidas
    - Calcular el volumen de un cilindro.
    - Identificar el mayor de tres números.
    - Buscar una palabra en el diccionario.

Resolución del problema: ¿Cómo lo hago?

- **Diseño de la Solución**
  - Para diseñar una solución es necesario:
    - Entender a cabalidad el problema
    - Identificar las áreas del conocimiento que abarca el problema.
    - Determinar las actividades que se pueden ejecutar con ayuda de un computador
    - Determinar como el computador va a realizar las actividades a él encomendadas (entradas, procesos y salidas)
    - Plantear y contrastar soluciones alternativas y escoger la mejor.

Implementación de la solución

- **¿Cómo hago que el computador lo haga?**
  - Una vez que se tiene una idea concreta de cómo resolver un problema hay que hacer que el computador realice las instrucciones deseadas.



Análisis y Resolución de un Programa Sencillo

- A continuación vamos a escribir nuestro primer programa. Consideremos el siguiente planteamiento:
  - Se desea ordenarle a la computadora que muestre un mensaje en el monitor: **Hola Mundo**

Nuestro primer algoritmo

En Pseudocódigo

```
/*Programa para
 mostrar Hola Mundo en la pantalla
*/
Begin
{
TAB /*Operación de Salida*/
    Write ("Hola Mundo");
}
End
```

Comentarios, van encerrados entre /* y */

Un algoritmo siempre debe indicar donde comienza

Las ordenes van encerradas entre llaves

Las instrucciones siempre terminan con un ;

Nuestro Primer Programa en MATLAB

```
%Programa para mostrar Hola Mundo
%Operación de salida

fprintf('Hola Mundo \n');
```

Comentarios:
-No son tomados en cuenta al traducir el programa fuente.
-Son útiles, porque ayudan a explicar lo que se está haciendo en el programa.

Tipos de datos

- Un **dato** es aquella información relativa a un objeto y manipulable por la máquina.
- Los datos tendrán diferente naturaleza según la magnitud a la que hagan referencia.
- Ejemplo:
  - Si el dato que quiera informatizarse es la velocidad de un móvil, habrá que pensar en un dato numérico (reales o enteros)
- Un **tipo de dato** se define como:
  - **Un conjunto de valores,** aquellos que puede tomar cualquier dato de dicho tipo.
  - **Un conjunto de operaciones,** definidas sobre dichos valores, que permiten operar adecuadamente con ellos.

Cont…

- Podemos dividir los datos en:
  - **Datos elementales:** Se consideran indivisibles en unidades más simples
    - Pueden ser **definidos por el usuario** y **predefinidos** (entero, real, lógico y carácter)
  - **Datos Estructurados o Estructuras de Datos:** consisten en una agrupación lógica de elementos individuales, cada uno de los cuales, es a su vez, o bien un dato simple u otra estructura de datos.
    - Ejemplo: Vectores, Matrices, Registros, etc.

Datos Elementales

- **Datos de tipo ENTERO**
  - Posibles valores: números sin decimales
  - Operaciones:

Aritméticas
- Suma: 2 + 3 = 5
- Resta: 2 – 3 = – 1
- Multiplicación: 2 * 3 = 6
- División entera: 9 DIV 2 = 4
- Módulo: 9 MOD 2 = 1

Operaciones relacionales: 3 > 2 es Verdadero

- Se debe tener en cuenta las indeterminaciones propias de los números binarios.
- Ejemplo: si el máximo número positivo que se puede representar es 32767 (n=15) y queremos sumarle 1, el resultado 32768 no tiene representación. A este tipo de indeterminación se le denomina *overflow (desbordamiento).*

Cont…

- **Datos de tipo REAL**
  - Posibles valores: números (con) decimales
  - Operaciones:

Aritméticas
- Suma: 2.3 + 3.2 = 5.5
- Resta: 2.3 – 3.2 = – 0.9
- Multiplicación: 2.3 * 3.2 = 7.36
- División: 2.3 / 3.2 = 0.71875

Operaciones relacionales:
3.2 > 2.3 es Verdadero

- Consiste, básicamente, en expresar el número de la forma **N = m * Be**
  - **N** es el número real a representar
  - **B** es la base utilizada (prefijada para cada computadora)
  - **e** es el exponente del número
  - **m** es la mantisa.
- En la computadora, el número se almacena uniendo el signo, el exponente y la mantisa, cada uno con un número de bits prefijado.

Cont…

- Cont…
  - Aspectos a tener en cuenta:
    - Desbordamiento (overflow) en tipos numéricos:
      - Determinado por el almacenamiento
      - **Ejemplo:** $11111111_{(2}$ + $00000001_{(2}$ = ??? (harían falta 9 bits para el resultado)
    - Se producen errores de precisión (redondeo)
      - El número de bits con que se representa la mantisa determina la precisión
      - **Ejemplo:** Con tres decimales de precisión los números 15.3243 y 15.3244 se representan como 15.324
    - Por la falta de precisión las operaciones de suma y multiplicación no cumplen siempre las propiedades distributiva y asociativa, debido a los errores de redondeo que se acumulan durante todo el cálculo.

Cont…

- **Datos de tipo LÓGICO**
  - Posibles valores: {Verdadero, Falso}
  - Operaciones lógicas:
    - Y: V Y F = F Y V = F Y F = F; V Y V = V
    - O: V O V = V O F = F O V = V; F O F = F
    - NO: NO Verdadero = Falso; NO Falso = Verdadero
  - Operadores lógicos: AND, OR, NOT, NAND, NOR y XOR.

operadores	AND	OR	NAND	NOR	XOR
V op V	V	V	F	F	F
V op F	F	V	V	F	V
F op V	F	V	V	F	V
F op F	F	F	V	V	F

Cont…

- **Cont…**
  - Operaciones de relación:
    - 5 < 6, con resultado externo Verdadero.
    - 10.5 = 10.58, con resultado externo Falso.
    - "B">"F", con resultado externo Falso.
    - (5 + 7) ≠ (2 - 6), con resultado externo Verdadero.
  - Operación de relación: expresión con dos operandos del mismo tipo (entero, real, carácter, …) *y* un *operador de relación* (<, >, =, , <=, >=).

Cont…

- **Tipo de datos CARÁCTER**
  - Posibles valores: Conjunto finito y ordenado de caracteres ({a,..,z,A,..,Z,0,..,9,!,",·,$...})
  - Operaciones:
    - Dentro de ASCII se pueden usar las funciones para convertir el dato
    - **Ordinal** (posición en el conjunto, de caracter a entero):
      - ORD ("A") = 65
    - **Carácter** (dada la posición, de entero a caracter):
      - CHR (65) = "A"
      - CHR(ORD("B")) es B
    - **Operaciones relacionales:** "A" = "B" es Falso

Estructuras de datos

- Podemos crearlos a partir de los tipos de datos elementales.
- **Ejemplos:**
  - tipo de datos **complejo** formado por una pareja de datos reales.
  - tipo de datos **fecha** compuesto por tres enteros.
  - tipo de datos **dirección** formado por cadenas de caracteres (calle, población,...), y por enteros y caracteres (portal, piso y letra, ...).

Cont...

- Pueden ser:
  - Según el **tipo** de los elementos:
    - **Homogéneas** (datos del mimo tipo)
    - **Heterogéneas** (datos de tipos diferentes)
  - Según el **uso de memoria**:
    - **Estáticas** (número fijo de elementos: arrays y registros – acceso rápido)
    - **Dinámicas** (número de elementos puede variar durante la ejecución - lento acceso)
  - Según el **acceso**:
    - **Por nombre** (ejemplo registros)
    - **Posición** (estructuras matrices, pilas y colas)
    - **Clave** (es uno de sus campos llamado clave: como los árboles)

Cont...

- **Arrays**
  - Agrupación de un número fijo de elementos del mismo tipo
  - Es homogénea, estática y de acceso por posición
  - 1 o más dimensiones
  - Única operación: acceso utilizando índices

Array unidimensional (vector):

A (3) = 2

1	2	3	4	5
1	9	2	8	3

Permiten el acceso directo o aleatorio a sus elementos.

Las operaciones básicas son la lectura y escritura de datos en las distintas posiciones

	1	2	3	4	5
1	1	13	9	15	5
2	11	2	14	4	6
3	10	12	3	7	8

Array bidimensional: B (3, 2) = 12

Cont...

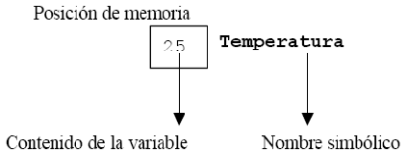
- **Cadenas de Caracteres (string)**
  - Secuencia de caracteres
  - Es homogénea
  - En una variable de tipo string se puede almacenar una frase, un nombre, una matricula de un coche, etc.
  - Operaciones:
    - **Concatenación:**
      - "hola" + " amigo" = "hola amigo"
    - **Comparación** - Relacionales (usando orden alfabético):
      - "hola" es mayor que "adios" (h > a)
    - **Extracción de Subcadena:**
      - Cadena1 ¬ "Pepe Pérez López"
      - Cadena1 (6:10) = "Pérez"
    - **Longitud:**
      - Longitud ("hola") = 4

Cont...

- **Otras estructuras**
  - Registros, punteros, listas, pilas, colas, árboles...
  - Permiten almacenar la información como queramos en memoria y/o ficheros

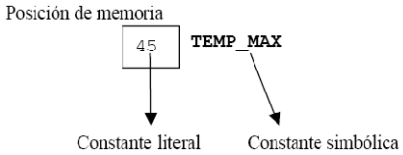
Variables

- Posición de memoria que almacena un dato cuyo valor puede cambiar durante la ejecución.



Constantes

- Posición de memoria que almacena un dato cuyo valor **NO** puede cambiar durante la ejecución.



Estructuras de control de flujo

- Necesidad de herramientas para alterar el orden lógico de las sentencias
  - Ejecutar unas sentencias u otras
    - **Estructura selectiva**
  - Ejecutar un número de veces unas acciones
    - **Estructura iterativa**
  - Capacidad de realizar preguntas sobre objetos del programa
    - **Expresiones lógicas**

Cont...

- Composición secuencial de sentencias

- Recurso insuficiente por sí sólo.
- No permite resolver problemas que exijan una toma de decisión.
- No permite la ejecución de un conjunto de acciones un número determinado de veces.

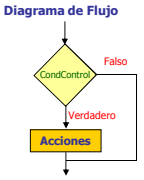
Cont…

- Estructuras Selectivas
  - Control de selección, alternativas.
  - Ejecutar un bloque de acciones dependiendo de la evaluación de una condición.
  - 3 tipos de sentencias alternativas:
    - *Sentencias de selección simple*
    - *Sentencias de selección binaria*
    - *Sentencias de selección múltiple*

Cont…

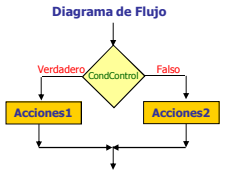
- Estructuras Selectivas

Diagrama de Flujo



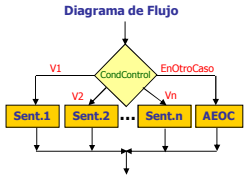
Selección Simple
SI CondControl ENTONCES
{Acción}
FINSI

Diagrama de Flujo



Selección Binaria
SI CondControl ENTONCES
{Acción}
SINO
{Acción}
FINSI

Diagrama de Flujo

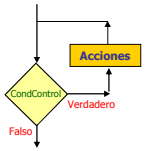


Selección Multiple
CASO expresión SEA
valores1: sentencias1
valores2: sentencias2
…… ……
valoresn: sentenciasn
SINO
accionesEOC
FINCASO

Cont…

- Estructuras Repetitivas
  - **Número de veces indeterminado a priori**
    - Estructura MIENTRAS

Diagrama de Flujo



MIENTRAS CondControl HACER
{Acción}
FINMIENTRAS

Bucle controlado por contador
Se ejecuta un número determinado de veces.
Variable de control del bucle

Bucle controlado por centinela
Centinela= valor especial que controla el final del bucle
Es necesario actualizar el centinela en cada iteración

Bucle contador
Útil cuando se quiere contar el número de veces que se ejecuta el bucle.
La expresión lógica no depende del contador

Cont…

- Estructuras Repetitivas
  - **Número de veces indeterminado a priori**
    - Estructura REPETIR

Diagrama de Flujo

REPETIR
{Acciones}
HASTA QUE CondControl

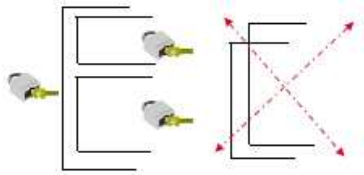
Cont…

- Estructuras Repetitivas
  - **Número de veces Determinado a priori**
    - Estructura PARA

```
PARA vcb=vi HASTA vf ( PASO p) HACER
acciones
FINPARA
```

Anidamientos

- Al igual que en las estructuras selectivas, no hay restricciones en las sentencias del cuerpo del bucle
- La estructura interna debe de estar incluida en la externa totalmente



```
Ejemplo:
PARA i= 1 HASTA 20 HACER
    PARA j= 1 HASTA 10 HACER
        acciones
    FINPARA
FINPARA
```

Programación estructurada. Modularidad

BOHM Y JACOPINI: 1.965
Bases de la Programación Estructurada

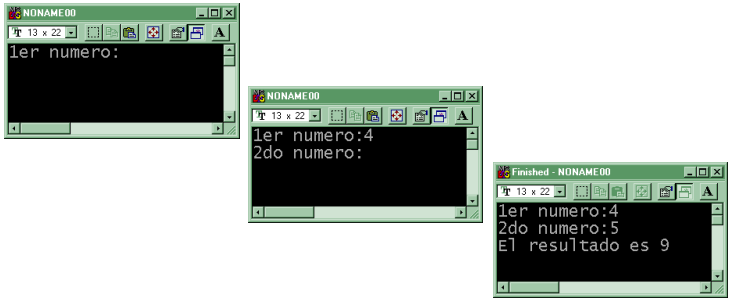
- **"Se demuestra que todo problema que pueda resolverse en un numero finito de pasos puede expresarse usando únicamente 3 tipos de estructuras o bloques fundamentales con una sola entrada y una sola salida para organizar dichos pasos:**
  - **Una caja proceso o de tratamiento secuencial.**
  - **Un mecanismo de decisión binaria.**
  - **Un mecanismo de bucle generalizado."**

Fases de un Programa

- Usualmente los programas en C, MatLab y en la mayoría de lenguajes, constan de tres fases:
  - **Fase de Entrada**: se le indica al usuario que datos debe ingresar a la computadora y se recopila esa información
  - **Fase de Computo**: los datos ingresados son procesados, se efectúan cálculos.
  - **Fase de Salida**: la información generada es mostrada, se despliegan los resultados

Calcular la Suma

- **Se desea ordenarle al computador que sume dos números enteros ingresados por teclado y muestre el resultado.**



Sumar dos Enteros

EN PSEUDOCODIGO

```
Begin
{
        a = read('1er numero');
        b = read('2do Numero');
        suma = a + b;
        write(suma);
}
```

Fase de Entrada

Fase de Computo

Fase de Salida

La Suma de Enteros

En Pseudocódigo

```
Begin
{
        int a,b;
        int suma;
        Write ("1er numero: ");
        Read (a);
        Write ("2do numero: ");
        Read (b);
        suma = a + b;
        Write ("El resultado es ", suma);
}
End
```

Las variables que se van a usar se "nombran" o declaran al inicio del programa

Fase de Entrada

Fase de Cómputo

Fase de Salida

SUMAR DOS ENTEROS

EN MATLAB

```
a = input('1er numero:');
b = input('2do numero');
suma = a + b;
fprintf('La suma %d\n', suma);
```

Fase de entrada:
-Para ordenar al computador que le pida al usuario un dato, de cualquier tipo, usamos **input**

Fase de salida:
-Para imprimir un mensaje que se usa **fprintf**.

La Suma de Enteros

EN C

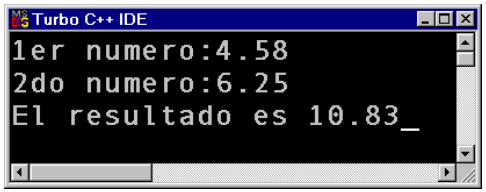
```
#include <stdio.h>
#include <genlib.h>
#include <simpio.h>
main()
{
        int a,b;
        int suma;
        printf("1er numero");
        a = GetInteger();
        printf("2do numero");
        b = GetInteger();
        suma = a + b;
        printf("El resultado es %d", suma) ;
}
```

Fase de entrada:
-Para ordenar al computador que le pida al usuario un entero, usamos la función GetInteger() de la librería **simpio.**h

Fase de salida:
-Para imprimir un mensaje que dependa de una variable entera se incluye en el mensaje **%d**, y luego se especifica la variable que se va a imprimir (suma)

SUMAR DOS REALES

EN MATLAB

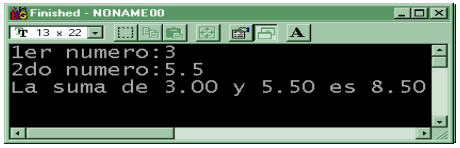



```
a = input('1er numero:');
b = input( '2do numero:');
suma = a + b;
fprintf('La suma de %f y %f: %f' , a, b, suma);
```

La Suma de Reales

EN C

```
#include <stdio.h>
#include <genlib.h>
#include <simpio.h>
main()
{
    double a,b;
    double suma;
    printf("1er numero real");
    a = GetReal();
    printf("2do numero");
    b = GetReal();
    suma =a + b;
    printf("La suma de %f y %f es  %f", a, b, suma) ;
}
```



Aquí printf imprime los 3 reales con todos sus decimales, pero se puede corregir con la siguiente línea:
printf("La suma de %.2f y %.2f es %.2f",a,b,suma);