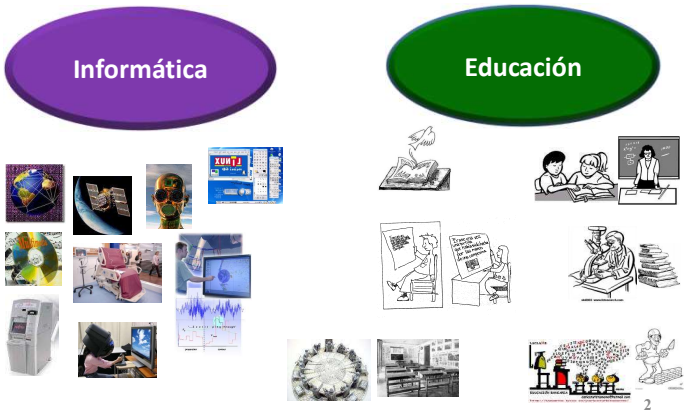
Tecnologías en Educación Matemática

MODULO 10

Dpto. de Ciencias e Ingeniería de la Computación
UNIVERSIDAD NACIONAL DEL SUR
Año 2019

1

Tecnología Computacional y Educación



2

Tecnología Computacional y Educación

¿Qué son las TICs?

- Son tecnologías que constituyen nuevos canales de comunicación
- La denominación de TIC es utilizada para referirse a una serie de nuevos medios como hipertextos, multimedia, Internet, realidad virtual o televisión por satélite
- El paradigma actual de las TIC son las redes informáticas que permiten en la interacción de las computadoras ampliar la potencia y funcionalidad que tienen en forma individual, permitiendo no sólo procesar información almacenada en soportes físicos, sino también acceder a recursos y servicios prestados por computadoras situadas en lugares remotos

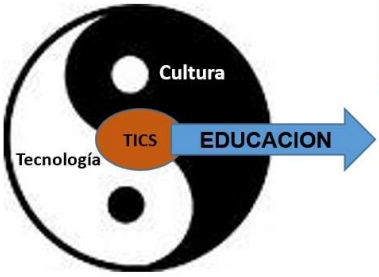
3

Tecnología Computacional y Educación

Niveles de integración TICs: paradigmas educativos

Niveles de integración TICs: paradigmas educativos

Sociedad de la Información



- Marco Neoliberal Globalizador
- Brecha Digital

TECNOCRATA
- Alfabetización digital de estudiantes para que utilicen las TIC como instrumento para mejorar la productividad en el proceso de la información (aprender sobre las TICs)
- Utilización las TIC como fuente de información y proveedor de materiales didácticos (aprender de las TICs)

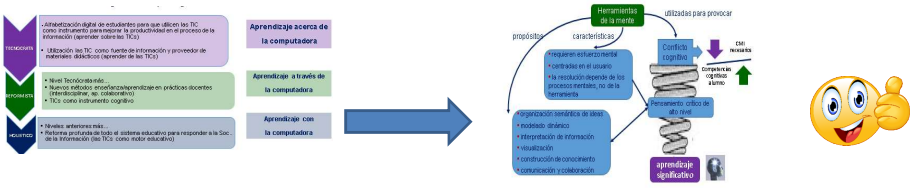
REFORMISTA
- Nivel Tecnócrata más…
- Nuevos métodos enseñanza/aprendizaje en prácticas docentes (interdisciplinar, ap. colaborativo)
- TICs como instrumento cognitivo

HOLISTICO
- Niveles anteriores más…
- Reforma profunda de todo el sistema educativo para responder a la Soc.. de la Información (las TICs como motor educativo)

Tecnología Computacional y Educación

Visión HOLISTICA DE TICS en Educación Matemática

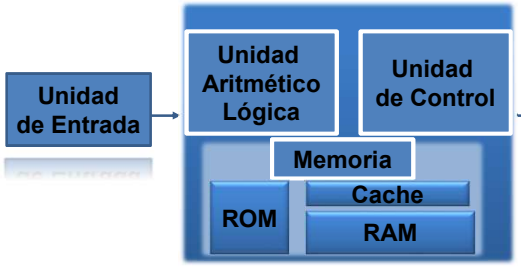
¿Por qué aprender a programar en la escuela secundaria?
La computadora como herramienta de la mente…



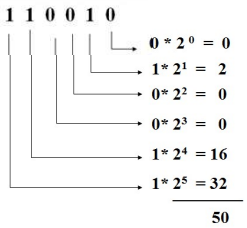
PENSAMIENTO LOGICO-MATEMATICO

- Capacidad de resolución de problemas
- Estrategias para favorecer pensamiento abstracto
- Procesos de modelado del mundo que nos rodea
- Etc

¿Qué es una Computadora?
Hardware: Arquitectura von Neumann



Representación de la información
En la computadora se usa el SISTEMA BINARIO.
Su base es el número 2.

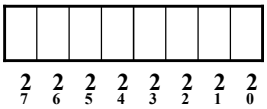


¿Qué es una Computadora?
Hardware: Almacenamiento de Información

Bit (Binary Digit) es la unidad mínima de información, puede ser representada en el sistema binario por los dígitos 0 y 1, los cuales están asociados a los estados "on" y "off".

Byte Son ocho bits colocados uno al lado del otro.

Cada casilla es un bit



Byte

¿Qué es una Computadora?
Hardware: Almacenamiento de Información

Unidad	Cant. Bits (como pot. de 2)	Equivale a
1 Bit	$1 = 2^0$	
1 Byte	$8 = 2^3$	8 bits
1 Kilobyte (KB)	2^{10}	1024 bytes
1 Megabyte (MB)	2^{20}	1024 KB = 1.048.576 bytes
1 Gigabyte (GB)	2^{30}	1024 MB = 1.073.741.824 bytes
1 Terabyte (TB)	2^{40}	1024 GB = 1.099.511.627.776 bytes
1 Petabyte (PB)	2^{50}	1024 TB = 1.125.899.906.842.624 bytes
1 Exabyte (EB)	2^{60}	1024 PB = 1.152.921.504.606.846.976 bytes
1 Zettabyte (ZB)	2^{70}	1024 EB = 1.180.591.620.717.411.303.424 bytes
1 Yottabyte (YB)	2^{80}	1024 ZB = 1.208.925.819.614.629.174.706.176 bytes

¿Qué es una Computadora?
Hardware: Almacenamiento de Información

Almacenamiento en la nube:

- **está basado en redes donde los datos están alojados por terceros. Ejemplos: dropbox, google docs, google drive.**
- **se almacena información (datos) de diferente tipo (documentos escritos, música, fotos, gráficos, planillas, etc)**

¿Qué es una Computadora? Software

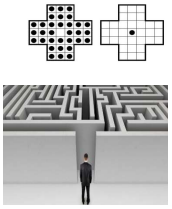
De Sistema: Programas que controlan y organizan el funcionamiento de los programas que se ejecutan, y gestionan internamente los recursos físicos de la computadora.
Ej: sistema operativo Windows, Sistema Operativo Linu Kernel de un celular (ej Android)

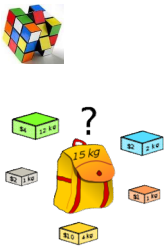
De Aplicación: Programas que controlan y aprovechan el funcionamiento de la computadora con el fin de realizar una tarea específica.
Ej: paquete Microsoft Office, Winamp, Navegador Crome, Buscador Google

Problemas Posibles de Resolver con una Computadora

Un problema es un hecho, situación o cuestión que precisa una solución

Ejemplos:
- Encontrar el camino más corto desde la UNS hasta mi casa.
- Aprobar una materia.
- Armar un rompecabezas.
- Encontrar el valor de una variable en una ecuación.
- Hacer una torta
- Pronosticar el clima





Problemas Posibles de Resolver con una Computadora

Un problema puede ser

Convergente: la resolución llega a ser una, o varias pero conocidas. Se caracterizan por ser lógicos estructurados

- Con una solución: ej. calcular un perímetro
- Con varias soluciones: ej. formas de recorrer un mapa
- Sin solución: ej dividir por cero
- Con infinitas soluciones: ej 2x-1 = 3x + 3 - x - 4

Divergente: Las soluciones pueden ser muy contradictorias y variadas, llegando a tener una cantidad infinita de posibles decisiones a tomar al respecto.
Ej: problemas sociales y filosóficos

Problemas y Algoritmos

En un problema convergente se pueden distinguir

- Los datos.
- La incógnita.
- Reglas que los vinculan.

La resolución de un problema comienza con la correcta identificación de cada uno de estos elementos

Un algoritmo es un conjunto finito de pasos (una secuencia de operaciones) que indican como se resuelve un determinado problema convergente en un tiempo finito

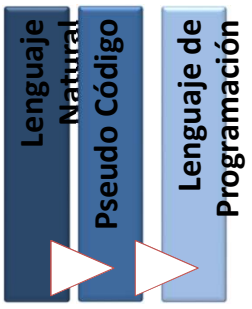
Problemas y Algoritmos

Lenguaje Algoritmico o Pseudocódigo: serie de símbolos y reglas que se utilizan para describir de manera explícita un proceso.

• Lenguaje coloquial en español ó inglés.

•Consta de un conjunto de frases con restricciones que pueden utilizarse en la especificación de programas.

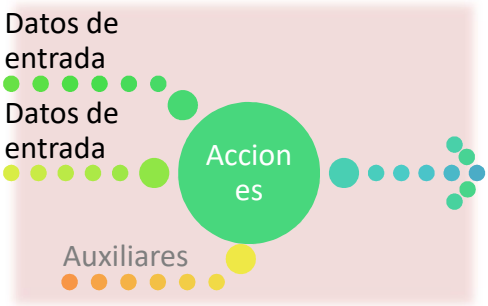
•Provee un balance entre la precisión formal de un lenguaje de programación y la informalidad y legibilidad del lenguaje natural.



Algoritmos – Datos y Operadores

Los algoritmos combinan datos con acciones.

- **Los datos de entrada son utilizados y/o transformados mediante las acciones para así obtener los datos de salida, y en estos datos de salida se encuentra representada la solución buscada.**



Algoritmos – Datos y Operadores

Los algoritmos combinan datos con acciones.

- **Los datos de entrada son utilizados y/o transformados mediante las acciones para así obtener los datos de salida, y en estos datos de salida se encuentra representada la solución buscada.**

DATOS

Distintos Dominios (letras,números, booleanos, etc)
Combinados en expresiones mediante operadores

Operadores aritméticos
Operadores relacionales
Operadores lógicos
Operadores de asignación

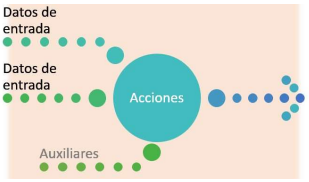
Algoritmos – Datos y Operadores

Los algoritmos combinan datos con acciones.

Todo dato tendrá asociado:
- **un nombre: FIJO**
- **un valor: puede CAMBIAR mientras ejecutamos el algoritmo.**

Ejemplos:

Nombre ← "Juan"
…
Nombre ← "Silvina"
….
Resultado ← 6
Resultado ← 6 / Valor 1



Algoritmos – Datos y Operadores

Los algoritmos combinan datos con acciones.

Todo dato tendrá asociado:
- **un nombre: FIJO**
- **un valor: puede CAMBIAR mientras ejecutamos el algoritmo.**

Asignación: Acción mediante la cual se establece el valor de un dato.

<nombre de dato> ← <expresión>

¿Los datos en matemática tienen esta forma o son valores????

Algoritmos – Operadores de Asignación

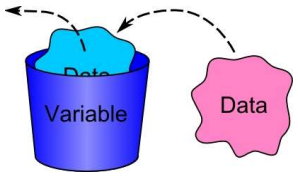
Para asignarle o atribuirle un valor a una variable se utiliza el operador de asignación

< nombre de dato> ← <expresión>

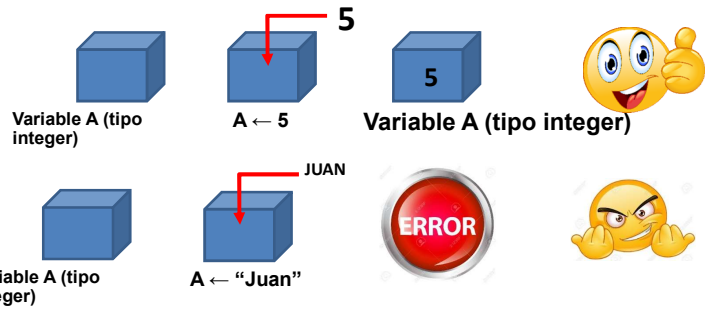
Luego de realizada esta acción, el dato que aparece a la izquierda contendrá el valor de la expresión.
Si el dato contenía algún otro valor antes de la asignación, ese valor se perderá.

Ejemplos:

Precio ← 5
Suma ← a+ 3.5
…
Precio ← 30
Suma ← 45 + a



Algoritmos – Operadores de Asignación



Variable A (tipo integer)

A ← 5

Variable A (tipo integer)

Variable A (tipo integer)

A ← "Juan"

Ejemplos:

Nombre ← "Pedro"
OBS: Nombre vale "Pedro"

a ← 4,
Duplicar_a ← a
OBS: Duplicar_a tiene valor 4

Dividendo ← 8
Divisor ← 2
Resultado ← Dividendo / Divisor
OBS: Resultado vale 4

Algoritmos – Estructuras de Control

Algoritmos: Los pasos se llevan a cabo siguiendo un orden llamado flujo

Estructura de un algoritmo

FLUJO DEL PROCESO

- **Secuenciales**
  - Entrada de datos
  - Asignación
  - Salida de Datos
- **Condicionales (selección)**
  - Simples
  - Dobles
  - Múltiples
- **Cíclicas o Repetitivas o Iterativas**
  - Repetir x veces
  - Repetir numero desconocido de veces
    - Repetir mientras
    - Repetir hasta

Algoritmos – Estructuras de Control - Secuencia

SECUENCIA: una acción sigue a la otra en secuencia La salida de la acción i es la entrada de la acción i+1

Algoritmo Levantarse para Clases
DE: TEM
Comienzo
Suena el despertador
Me aseo
Me cambio
Desayuno
Viajo hasta la UNS
Entro al laboratorio de TEM
Comienza clase TEM
Fin

Algoritmos – Estructuras de Control

SECUENCIA: una acción sigue a la otra en secuencia La salida de la acción i es la entrada de la acción i+1

Pero ¿ Y si quiero repetir acciones?

Algoritmo Subir Escalera
Comienzo
Subir escalon 1
Subir escalon 2
Subir escalon 3
…
Subir escalon 1345
FIN

NO PUEDO GENERALIZAR SOLUCION

NECESITO UN ALGORITMO PARA CADA CANTIDAD DIFERENTE DE ESCALONES

Algoritmos – Estructuras de Control - Iteración

ITERACION: repito n veces un grupo de acciones

Número natural

Repetir N veces
ACCIONES

PARA i desde: 1 hasta: N HACER
ACCIONES

Repetir 3 veces
Acción 1
Acción 2

Ejecución:

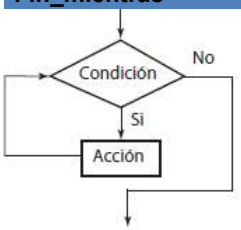
Acción 1
Acción 2
Acción 1
Acción 2
Acción 1
Acción 2

Algoritmos – Estructuras de Control - Cíclicas

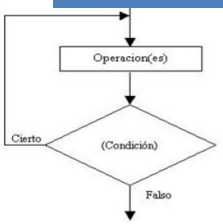
Si el número de repeticiones es desconocido, y las acciones se repetirán de acuerdo a cierta condición (expresión lógica).

Dos alternativas:

Mientras <condición> hacer
Acciones
Fin_mientras



Repetir
Acciones
Hasta <condición>



Algoritmos – Estructuras de Control - Cíclicas

Si el número de repeticiones es desconocido, y las acciones se repetirán de acuerdo a cierta condición (expresión lógica). Dos alternativas:

Mientras <condición> hacer
Acciones
Fin_mientras

Repetir
Acciones
Hasta <condición>

- **La condición se comprueba al inicio, antes de entrar al ciclo**
- **Las acciones se ejecutan mientras que la condición es verdadera**
- **Las acciones del ciclo se pueden ejecutar 0 o más veces.**

- **La condición se comprueba al final, luego de ejecutar una vez el ciclo.**
- **Las acciones se ejecutan mientras que la condición es falsa.**
- **Las acciones del ciclo se ejecutan al menos una vez.**

OJO CON LOS BUCLES INFINITOS

Algoritmos – Estructuras de Control

Pero ¿ Y si quiero tomar decisiones?

Si es un usuario frecuente dale el descuento del 20%
Si es estudiante dale el descuento del 25%
Si es un jubilado dale el descuento del 30%

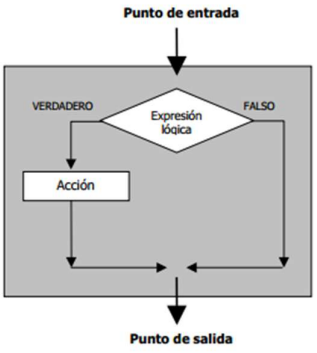
Algoritmo Calcular Descuento
DE: monto, tipo-cliente
DS: descuento
Comienzo
Si tipo-cliente es UsuarioFrecuente entonces descuento ← Monto-20%
Sino
….

Algoritmos – Estructuras de Control - Condicionales

CONDICIONAL SIMPLE: Es el tipo de selección donde se tiene un bloque de instrucciones, cuya ejecución sólo debe darse en caso de que una determinada condición se cumpla.

Si <condicion>
entonces
Sentencias
Fin_si

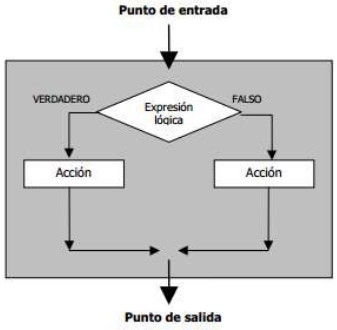
Ejemplo:
Si (Edad > 16) entonces
MostrarCartel "Puede votar"
Fin_si



Algoritmos – Estructuras de Control - Condicionales

CONDICIONAL DOBLE: Es similar a la anterior con la salvedad de que en este tipo de estructura se indican acciones no sólo para la rama "verdadera" sino también para la "falsa"; es decir, en caso de la expresión lógica evaluada sea cierta se ejecutan una acción o grupo de acciones y en caso de que sea falsa se ejecuta un grupo diferente.

```
Si <condicion>
entonces
    Sentencias 1
Sino
    Sentencias 2
Fin_si
```



Ejemplo:

```
Si (temperatura < 20)
entonces
    PrenderCalefaccion
sino
    PrenderCalefacción
    CerrarPuertas ← si
Fin_si_entonces_sino
```

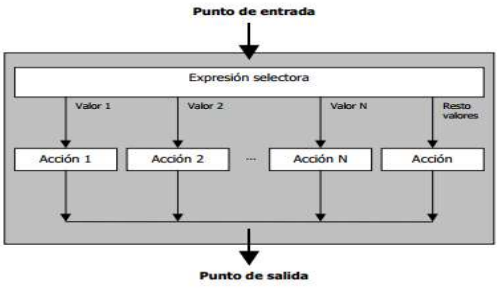
Algoritmos – Estructuras de Control - Condicionales

CONDICIONAL MULTIPLE: Esta estructura evalúa una expresión que pueda tomar n valores y ejecuta una acción o grupo de acciones diferente en función del valor tomado por la expresión selectora.

```
    En caso de
    DATO
        Valor1:
    accion1
        Valor2:
    accion2
    …
    ValorN:
    accionN
    Fin_encaso
```

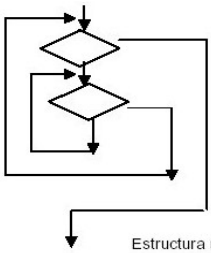
Ejemplo:

```
En caso de EstadoCivil
    "soltero": solteros ← solteros + 1
    "casado": casados ← casados +1
    "divorciado": divs ← divs + 1
    "viudo": viudos ← viudos + 1
    "otro": otro ← otro + 1
Fin_encaso
```



Algoritmos – Estructuras de Control

PUEDEN ANIDARSE y/0 COMBINARSE



Estructura repetitiva Anidada

```
ALGORITMO RESERVAR SILLAS PARA FAMILIARES
....
Alumno ← 1
QuedanSillas ← 200
Repetir
    GrupoFamiliar ← 0
    Reservar ← V
    Mientras (QuedanSillas = V) y GrupoFamiliar < n) y    (Reservar=V)
    hacer
        Reservar ← Preguntar si quiere para otro familiar
        Si Reservar = V entonces
            Reservar Lugar
            QuedanSillas ← QuedanSillas -1
            GrupoFamiliar ← GrupoFamiliar
```

Algoritmos Ciclos Anidados

Existen diversos problemas en los cuales es necesario utilizar ciclos repetitivos anidados, es decir, uno "dentro" del otro.

Supongamos el siguiente fragmento de código:

```
PARA i desde 1 hasta 5 hacer
    ...
    PARA j desde 1 hasta 3 hacer
        <acción1>
    ...
```

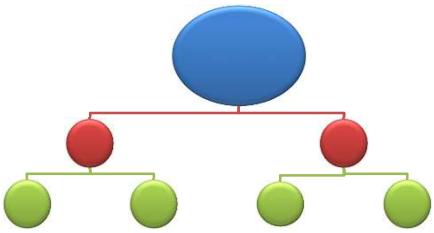
¿Cuántas veces se repite la acción1?

Algoritmos – Primitivas

Descomposición de Problemas en Sub-Problemas:

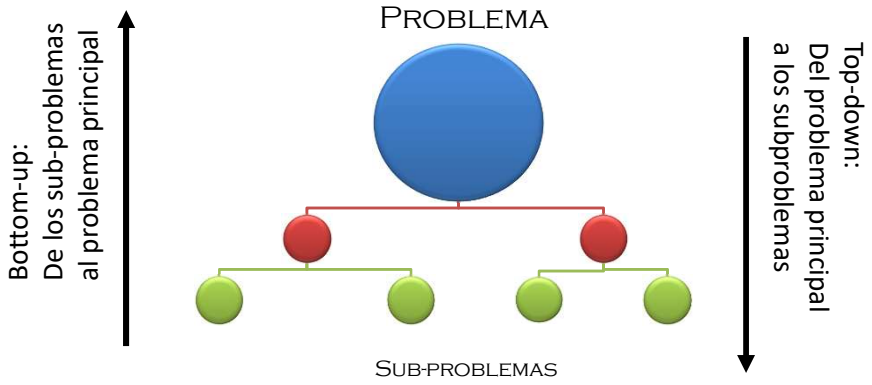
Cuando la complejidad de los problemas aumenta, la tarea de hallar una solución se torna más difícil.

Una metodología para reducir la complejidad consiste en plantear la solución del problema a partir de la solución de una serie de sub-problemas más sencillos que forman parte del problema original.



Algoritmos – Primitivas

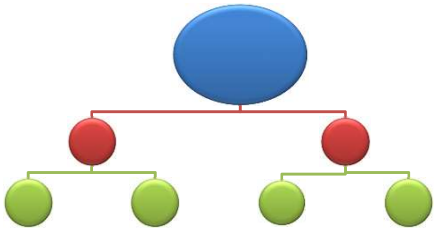
"Bottom-up vs. Top-down"



Algoritmos – Primitivas

"Bottom-up vs. Top-down"

- La técnica *bottom-up* consiste en ir resolviendo en primer lugar los problemas más elementales, usando esas soluciones para resolver problemas mayores.
- La técnica *top-down* resuelve primeramente el problema mayor, dejando pendiente la solución de los problemas más elementales para más adelante.



Algoritmos – Primitivas

Algoritmos como Primitivas

Un algoritmo se identifica por su **nombre**,
sus **datos de entrada** y sus **datos de salida**.

Un algoritmo A puede usar otro algoritmo B como primitiva, y para ello debe indicar:

- el nombre de B,
- los datos de A que serán los datos de entrada para B,
- en qué datos el algoritmo A recibirá los datos de salida del algoritmo B (NO SIEMPRE).

Al usar un algoritmo como primitiva,
decimos que lo estamos "invocando" o "llamando".

Algoritmos – Primitivas

Invocación

La invocación de un algoritmo debe coincidir con la definición del algoritmo en los siguientes aspectos:

- el nombre,
- la cantidad de datos de entrada,
- la cantidad de datos de salida,
- el orden de los datos de entrada,
- el orden de los datos de salida y
- el tipo de los datos (numérico o lógico).

Algoritmos – Primitivas

Invocación a FUNCION

Cuando un algoritmo tiene **un sólo dato de salida**, puede ser invocado **desde una expresión**.

En estos casos, la invocación sólo contendrá el nombre de la primitiva y los datos de entrada.

Nombre_Funcion (dato-ent1, dato-ent2,...,)

En cada llamada, los datos de entrada deben tener un valor ya asignado, y ese valor es enviado al algoritmo llamado. El dato de salida será un único valor devuelto al lugar donde se realizó la invocación.

Se usa cuando tenemos 1 UNICO DATO DE SALIDA

Algoritmos – Primitivas

Invocación a PROCEDIMIENTO

Se escribe el nombre seguido de los datos de entrada y los datos de salida subrayados; todos los datos van encerrados entre paréntesis.

Nombre(dato-ent1, dato-ent2,..., dato-sal1, dato-sal2, ...)

Al momento de realizar la llamada, los datos de entrada deben tener un valor ya asignado, y ese valor es enviado al algoritmo llamado. Los datos de salida toman el valor devuelto por el algoritmo que fue llamado.

Se usa cuando tenemos CERO o MAS DE 1 DATO DE SALIDA

Algoritmos y Trazas

Traza de un algoritmo: ¿COMO SÉ SI DISEÑÉ BIEN MI ALGORITMO o si un algoritmo funciona bien?

- **Ejecución manual de forma secuencial de las sentencias que lo componen.**
- **La traza de un algoritmo (o programa) indica la secuencia de acciones (instrucciones) de su ejecución, así como, el valor de las variables del algoritmo (o programa) después de cada acción (instrucción).**

La función principal que posee realizar la traza de un algoritmo es la de comprobar cómo funciona o para realizar la etapa de depuración en la que se intenta corregir errores, simplificar el algoritmo al máximo e incrementar su eficacia y velocidad.

Algoritmos y Trazas

Visiones de un Algoritmo

Especificación del algoritmo

- La especificación comprende al texto con información sobre los datos involucrados, y el conjunto de acciones que permiten resolver una clase de problemas.
- Desde este punto de vista un algoritmo puede pensarse como una entidad estática.

Algoritmos y Trazas

Visiones de un Algoritmo

Ejecución del algoritmo

- Cuando se llevan a cabo las acciones indicadas por el algoritmo para distintos valores de el o los datos de entrada.
- Durante la ejecución estamos hallando la solución de un problema específico o de una instancia de la clase de problemas.
- La ejecución es un proceso dinámico.

Algoritmos y Trazas

```
Inicio
Entero NUM
(1)   NUM ← 0
(2)   Mientras (NUM < 5) hacer
(3)       NUM ← NUM + 1
(4)       Escribir NUM
      Fin mientras
Fin algoritmo
```

TRAZA

PASO	NUM	Comentario
1	0	
2		0 < 5 entra al lazo
3	1	Incrementa NUM
4		Escribe NUM va a paso (2)
2		1 < 5 entra al lazo
3	2	Incrementa NUM
4		Escribe NUM va a paso (2)
2		2 < 5 entra al lazo
3	3	Incrementa NUM
4		Escribe NUM va a paso (2)
2		3 < 5 entra al lazo
3	4	Incrementa NUM
4		Escribe NUM va a paso (2)
2		4 < 5 entra al lazo
3	5	Incrementa NUM
4		Escribe NUM va a paso (2)
2		5 < 5 falso, sale del lazo y termina

Algoritmos ALGUNOS PROBLEMAS

Problema:

Dos números son equivalentes en dígitos si están formados exactamente por los mismos dígitos en iguales cantidades, sin importar el orden.

Ejemplo:

- 3245 es equivalente en dígitos 2453
- 898 es equivalente en dígitos 889
- 898 NO es equivalente en dígitos 9889
- 7673 NO es equivalente en dígitos 3676
- 7673 NO es equivalente en dígitos 367

Algoritmos – Algunos Problemas

Primera aproximación

Podríamos contar cuántas veces aparece cada dígito (del 0 al 9) en ambos números, y si estas cantidades coinciden, los números son equivalentes en dígitos.

Algoritmos – Algunos Problemas

```
ALGORITMO Números Equivalentes en Dígitos
DATOS DE ENTRADA: Num1, Num2 {naturales}
DATOS DE SALIDA: SonEquivalentes {lógico}
DATOS AUXILIARES: Dígito {natural}
COMIENZO
  Dígito ⟵ 0
  SonEquivalentes ⟵ Verdadero
  MIENTRAS ((Dígito <= 9) y (SonEquivalentes=Verdadero))
     SI Dígito NO aparece la misma cantidad de veces en Num1 y Num2
          ENTONCES  SonEquivalentes ⟵ falso
     FIN SI
     Dígito ⟵ Dígito + 1
Fin Mientras
FIN ALGORITMO
```

Dos datos de control: *Dígito* y *SonEquivalentes*

Algoritmos – Algunos Problemas

OJO!! Para que la solución esté completa falta especificar la resolución de los sub-problemas ESPECIFICANDO las PRIMITIVAS necesarias (podemos usar invocación a procedimientos o invocación a funciones)

```
Algoritmo ContarVeces
DE: N, Dig {Naturales}
DS: Cantidad {Natural}
Daux: -
COMIENZO
Cantidad ← 0
Mientras (N>0)
        Si (N mod 10 = Dig)
          entonces
                Cantidad ← Cantidad + 1
        FinSI
        N ← N div 10
Fin Mientras
FIN ALGORITMO
```

Algoritmos – Algunos Problemas

```
ALGORITMO Números Equivalentes en Dígitos
DE: Num1, Num2 {naturales}
DS: SonEqui {lógico}
Daux: Dig, Cant1, Cant2 {natural}
COMIENZO
  Dig ← 0
  SonEqui ← Verdadero
  Mientras ((Dig<=9) Y SonEqui)
      ContarVeces(Num1, Dig, Cant1)
      ContarVeces(Num2, Dig, Cant2)
      SI (Cant1 ≠ Cant2)
        Entonces SonEqui ← falso
      FIN SI
      Dig ← Dig + 1
  Fin Mientras
FIN ALGORITMO
```

Subproblemas

Algoritmos – Algunos Problemas

```
ALGORITMO Números Equivalentes en Dígitos
DE: Num1, Num2 {naturales}
DS: SonEqui {lógico}
Daux: Dig {natural}
COMIENZO
   Dig ← 0
   SonEqui ← Verdadero
  Mientras (Dig<=9) Y SonEqui
        Si ContarVeces(Num1, Dig) ≠ ContarVeces(Num2, Dig)
          Entonces SonEqui ← falso
        FIN SI
        Dig ← Dig + 1
  Fin Mientras
FIN ALGORITMO
```

Subproblemas

Algoritmos – Algunos Problemas

```
ALGORITMO Números Equivalentes en Dígitos
DE: Num1, Num2 {naturales}
DS: SonEqui {lógico}
Daux: Dig {natural}
COMIENZO
   Dig ← 0
   SonEqui ← Verdadero
  Mientras (Dig<=9) Y SonEqui
        Si ContarVeces(Num1, Dig) ≠ ContarVeces(Num2, Dig)
          Entonces SonEqui ← falso
        FIN SI
        Dig ← Dig + 1
  Fin Mientras
FIN ALGORITMO
```

Subproblemas

Algoritmos – Algunos Problemas

Más problemas con sub-problemas

1) A partir de una secuencia (de longitud L) de enteros ingresada por teclado, mostrar para cada uno de los enteros QUE SEAN PARES, su cantidad de dígitos.
2) A partir de una secuencia de enteros ingresada por teclado y que finaliza en 0, y un dígito D, mostrar por pantalla aquellos enteros de la secuencia que contienen al dígito D.
3) A partir de una secuencia (de longitud L) de enteros ingresada por teclado, y un dígito D, determinar si algún elemento de la secuencia contiene al dígito D.

Algoritmos – Algunos Problemas

Problemas del Modulo 8:

- ✓ Sumar los dígitos de un número N.
- ✓ Obtener el promedio de los dígitos de N.
- ✓ Sumar los dígitos de un número N que sean múltiplos de k.
- ✓ Sumar todos los números en el intervalo (a, b) que sean impares.
- ✓ Obtener el promedio de los números contenidos en el intervalo (a, b].
- ✓ Obtener el promedio de los números contenidos en el intervalo (a, b] que sean impares.
- ✓ Determinar si en el intervalo (a, b] hay algún múltiplo de k.