|  |
| --- |
| **Algoritmos y Resolución de Problemas** |
| Resumen |
| **Resumen General** |

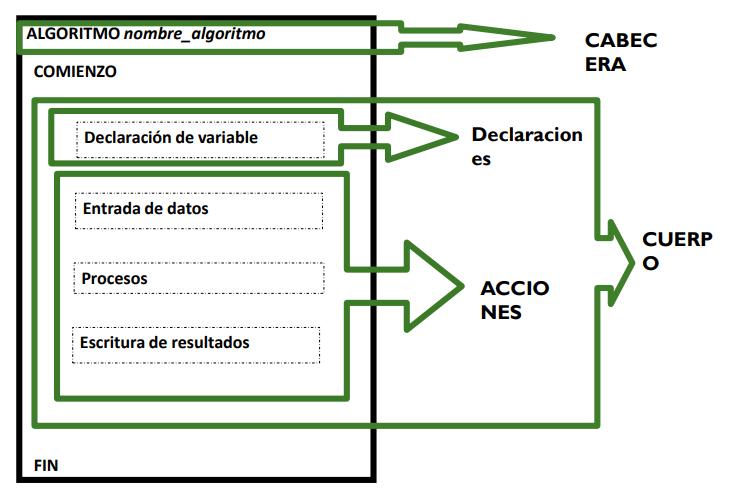
|  |
| --- |
| Facultad de Ciencias Exactas, Físicas y Naturales  Universidad Nacional de San Juan |

Eje N°1

# Algoritmo

Secuencia ordenada y finita de instrucciones para resolver un problema.

## Estructura



## Elementos

### Datos

Representación simbólica de una característica de una entidad (Sujeto, Objeto o Evento).

### Expresiones

Descripción formal de un cálculo.

### Acciones

Instrucciones pertenecientes a un algoritmo que permiten resolver un problema.

# Datos

## Atributos

### Nombre

Identificador del dato

* Debe empezar con una letra
* No debe contener espacios
* Debe ser representativo a los datos asociados

### Valor

Valor del dato. Puede variar o no durante la ejecución del programa.

### Tipo

Determina el tipo de valor que se puede almacenar y las operaciones que pueden realizarse.

### Dirección de Memoria

Identifica el espacio de memoria donde se almacena el dato.

## Clasificación

### Constante

Nombre simbólico de una dirección de memoria donde se almacena un dato cuyo valor no se modifica en la ejecución del programa.

Constante<identificador> = <expresión>

### Variable

Nombre simbólico donde se almacena un dato cuyo valor se modifica durante la ejecución del programa.

<tipo> <identificador> = <expresión>

## Tipos

### Simples

Dato indivisible

#### Numérico

Representación de cantidades. Se pueden utilizar operadores aritméticos y relacionales.

* Entero: Representación de los números enteros.

1,2,3…

* Real: Representación de números con decimales.

-0,5 ,7,78 ,8,2…

#### Carácter

Representación de un carácter de la tabla ASCII.

A,1,&…

* Alfabéticos: Desde la A a la Z, sin incluir la Ñ.
* Numéricos: Desde el 0 al 9. No se pueden realizar operaciones ya que no son numéricos.
* Especiales: Símbolos, Signos de Puntuación, etc.

#### Lógico

Representación de valores binarios.

Verdadero,Falso

### Estructurados

Colección de datos.

#### Cadena

Conjunto de caracteres.

"Hola","Mundo","!"…

#### Arreglo

Conjunto finito y ordenado de datos homogéneos identificados con un único nombre.

#### Registro

Conjunto finito y ordenado de datos heterogéneos identificados con un único nombre.

#### Archivo

# Expresiones

## Elementos

### Operadores

Indican el tipo de cálculo a realizar.

### Operandos

Constantes, variables u otras expresiones.

## Clasificación

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Expresión** | **Operandos** | **Operadores** | | **Tipo de Resultado** | |
| **Aritmética** | Variables o Constantes Numéricas | | + Suma  - Resta  \* Multiplicación  / Cociente  div División Entera  raíz (Operando)  resto (Operando)  %  potencia (Base, Exponente)  ^ | | Numérico | |
| **Relacional** | Datos del mismo tipo | | > Mayor  < Menor  == Igual  != Desigual  >= Mayor o igual  <= Menor o igual | | Lógico | |
| **Lógica** | Expresiones  Variables o Constantes Numéricas | | NO Negación  Y Conjunción  O Disyunción | | Lógico | |

# Acciones

## Simples

Instrucción que no puede ser subdividida.

### Asignación

Instrucción en la que se determina el valor de un dato variable o constante.

<identificador> = <valor>

### Lectura

Instrucción que permite recibir un valor por parte del usuario.

Leer <variable>

### Escritura

Instrucción que permite entregar un valor al usuario.

Escribir <Valor,Variable,Constante o Expresión>

## Estructuradas

Instrucción compuesta por acciones simples.

-

Eje N°2

# Conceptos Generales

## Especificación de un Problema

Para comprender un problema se debe determinar:

* Datos de Entrada: Datos conocidos con los que se trabajará para obtener los resultados.
* Instrucciones: Conjunto de reglas que vinculan los datos de entrada con los de salida.
* Datos de Salida: Resultados que se desean obtener.

## Técnica DYV – Divide y Vencerás

Método que reduce un problema complejo en otros más sencillos para facilitar su comprensión y desarrollo de una solución.

Se divide el programa principal en subprogramas, repitiendo hasta que los subprogramas obtenidos sean lo suficientemente pequeños como para resolverlos directamente.

## Otros conceptos

### Datos relevantes e irrelevantes

* Relevantes: Datos de gran importancia en un contexto determinado.
* Irrelevantes: Datos de poca o nula importancia en un contexto determinado.

### Datos explícitos e implícitos

* Explícitos: Dato claramente identificable en un contexto determinado.
* Implícitos: Dato que no es expresado en forma directa, sino, que surge de la inferencia de otros. Debe ser transformado en explícito.

# Construcción de Algoritmos

## Algoritmos Simples

1. Lectura del Enunciado
   1. Lectura Superficial: Plantea una idea general del problema
   2. Lectura Comprensiva: Permite identificar el significado preciso del enunciado
2. Identificación
   1. Datos de Salida
   2. Datos Explícitos
      1. Determinar si son relevantes o no
   3. Datos Implícitos
      1. Determinar si son relevantes o no
      2. Transformarlos en datos explícitos
3. Detección de imprecisiones y ambigüedades

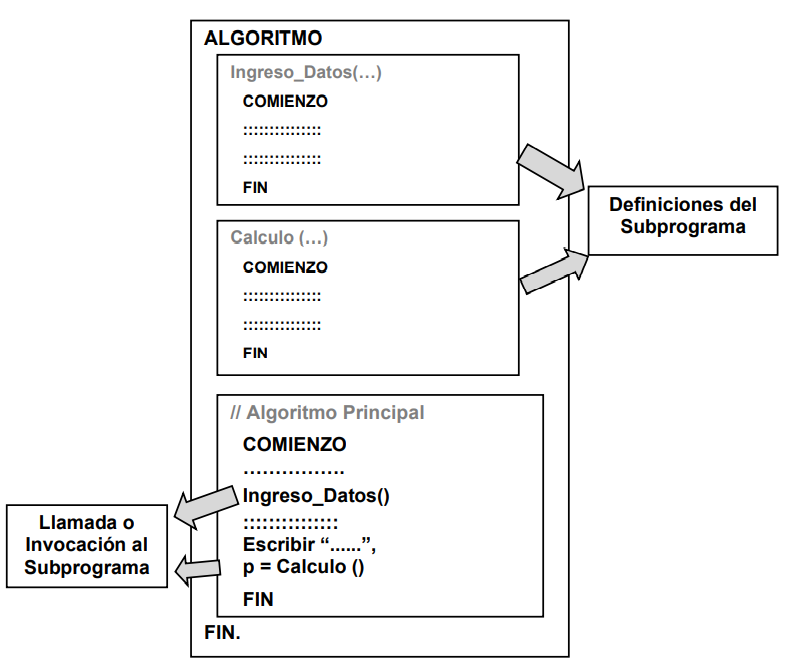
## Algoritmos con Subprogramas

1. Especificación del Problema
   1. Identificación
      1. Datos de Salida
      2. Datos de Entrada
      3. Restricciones
   2. Interpretación general del Problema
   3. División del Problema en partes
2. Identificación de Subprogramas
   1. Nombre
   2. Descripción
   3. Datos de Entrada
   4. Datos de Salida
3. Identificación de Programa Principal
   1. Nombre
   2. Datos de Entrada
   3. Datos de Salida
4. Codificación
   1. Subprograma
   2. Programa Principal

# Subprograma

Algoritmo que resuelve una tarea específica, permitiendo el retorno de un valor.

## Estructura



## Definición e Invocación

### Definición

Desarrollo de un algoritmo que establezca el ambiente del subprograma, sus variables e instrucciones.

### Invocación

Llamada del subprograma desde cualquier parte del programa, para ejecutar sus instrucciones.

## Parámetros

### Reales

Valores enviados desde la invocación del subprograma.

### Formales

Valores recibidos en el encabezado del subprograma

***Nota: Ambos deben tener las mismas características en orden, cantidad y tipo.***

# Seguimiento de Algoritmos

Ejecución paso a paso de las instrucciones de un algoritmo con datos concretos, para determinar si cumple o no con el objetivo planteado.

## Algoritmo Simple

|  |  |  |  |
| --- | --- | --- | --- |
| **Algoritmo Principal** | | | |
| **Variable N°1** | **Variable N°2** | **Variable n** | **Salida** |
|  |  |  |  |

## Algoritmo con Subprogramas

|  |  |  |  |
| --- | --- | --- | --- |
| **Algoritmo Principal** | | | |
| **Variable N°1** | **Variable N°2** | **Variable n** | **Salida** |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Subprograma N°1** | | | |
| **Variable N°1** | **Variable N°2** | **Variable n** | **Salida** |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Subprograma n** | | | |
| **Variable N°1** | **Variable N°2** | **Variable n** | **Salida** |
|  |  |  |  |

Eje N°3

# Estructuras de Control

Instrucciones que determinan el flujo de ejecución de un programa.

## Secuencia

Determina el orden en el que se ejecutan las acciones de un algoritmo. Se ejecutan de arriba a abajo.

Acción 1

Acción 2

…

Acción N

## Selección

Determina la ejecución de una acción entre otras acciones alternativas en función de una o varias condiciones, modificando el flujo del programa.

### Selección Simple

Si (<condición>) Entonces

--- Acciones ---

Fin Si

### Selección Doble

Si (<condición>) Entonces

--- Acciones ---

Sino

--- Acciones ---

Fin Si

### Selección Múltiple

Según (<variable>)

valor 1: Acción 1

valor 2: Acción 2

…

valor n: Acción n

Fin Según

## Iteración

Determina la repetición de una o varias acciones en función de una o varias condiciones.

### Para

Repite una o varias acciones una cantidad de veces determinada. Se conoce la cantidad de iteraciones que se realiza.

Para <variable> desde <valor inicial> hasta <valor final>

--- Acciones ---

Fin Para

#### Variable de Control

Variable que establece la cantidad de iteraciones a realizar, contando las iteraciones realizadas.

### Mientras

Repite una o varias acciones en función de una o varias condiciones. La condición se evalúa al inicio de la estructura. No se conoce la cantidad de iteraciones que realiza.

Mientras (<condición>)

--- Acciones ---

Fin Mientras

#### Variable Centinela

Variable que determina la cantidad de iteraciones en función de su valor.

### Hacer – Mientras

Repite una o varias acciones en función de una o varias condiciones. La condición se evalúa al final de la estructura. No se conoce la cantidad de iteraciones que realiza.

Hacer

--- Acciones ---

Mientras (<condición>)

Eje N°4

# Operaciones Básicas

## Contador

Variable entera que almacena la frecuencia de un evento. Debe ser modificada en una estructura iterativa. Se inicializa en 0.

contador = 0

<estructura iterativa>

contador = contador + 1

<fin estructura iterativa>

escribir contador

## Acumulador

Variable entera o real que almacena el resultado de una serie de sumas sucesivas. Debe ser modificada en una estructura iterativa. Se inicializa en 0.

acumulador = 0

<estructura iterativa>

acumulador = acumulador + variable

<fin estructura iterativa>

escribir acumulador

## Máximo

Variable entera o real que se compara con un conjunto de datos y almacena el valor máximo de estos. Puede ser modificada en función de una condición que los compara. Se inicializa en un valor más bajo que el lote de datos que se van a procesar.

maximo = 0

si (maximo < variable> entonces

maximo = variable

fin si

escribir maximo

## Mínimo

Variable entera o real que se compara con un conjunto de datos y almacena el valor mínimo de estos. Puede ser modificada en función de una condición que los compara. Se inicializa en un valor más alto que el lote de datos que se van a procesar.

minimo = 99999

si (minimo < variable> entonces

minimo = variable

fin si

escribir minimo

## Bandera

Variable lógica que almacena un estado. Puede ser modificada en función de una o varias condiciones. Se utiliza para determinar la ocurrencia o no de un evento.

bandera = falso

si ( <condicion> ) entonces

bandera = verdadero

fin si

si ( bandera == verdadero) entonces

--- evento ocurrido ---

sino

--- evento no ocurrido ---

fin si

# Cuadro Comparativo entre Operaciones Básicas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operación Básica** | **Tipo de Variable** | Inicio | **Proceso** | **Uso** |
| **Contador** | Entero | c = 0 | c = c + 1 | Se usa cuando la iteración finaliza. |
| **Acumulador** | Entero o Real | a = 0 | a = a + <variable> | Se usa cuando la iteración finaliza. |
| **Máximo** | Entero o Real | max = 0 | si (max < <variable>)  max = variable  fin si | Se almacena el valor máximo cuando la iteración finaliza |
| **Mínimo** | Entero o Real | min = 99 | si (min > <variable>)  min = variable  fin si | Se almacena el valor mínimo cuando la iteración finaliza |
| **Bandera** | Lógico | b = falso | si (<condicion>)  b = verdadero  fin si | Se realizan acciones en función de su valor |

Eje N°5

# Arreglo

Conjunto finito y ordenado de datos homogéneos identificados con un único nombre.

<tipo de dato> <identificador> [<tamaño>]

## Elementos

### Componente

Dato único de un arreglo que está almacenado en una posición de memoria.

### Índice

Valor que indica la posición de una componente dentro de un arreglo.

## Características

* Cada dato se almacena en posiciones contiguas de memoria.
* Es una estructura de datos estática, ya que el espacio que ocupa en la memoria es fijo.
* Todos sus componentes deben ser del mismo tipo de dato.

# Operaciones con Arreglos

## Carga

Colocar información en un arreglo.

### Carga Secuencial

Colocar la información en orden lineal en posiciones contiguas de memoria, partiendo de la posición con índice cero.

#### Método N°1 - Lectura

Para i Desde 0 Hasta <tamaño de arreglo> – 1

Leer <identificador de arreglo> [i]

Fin Para

#### Método N°2 - Asignación

Para i Desde 0 Hasta <tamaño de arreglo> – 1

<identificador de arreglo> [i] = <valor>

Fin Para

### Carga Aleatoria

Colocar la información sin un orden determinado.

Entero posicion, valor

Escribir “Ingrese la posicion de la componente”

Leer posicion

Escribir “Ingrese el valor de la componente”

Leer valor

<identificador de arreglo> [posicion] = valor

## Recorrido

Mostrar o procesar las componentes de un arreglo en el orden en el que se encuentran almacenadas en memoria.

Para i Desde 0 Hasta <tamaño de arreglo> – 1

Escribir <identificador de arreglo> [i]

Fin Para

# Operaciones Básicas con Arreglos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operación Básica** | **Tipo** | Inicio | **Proceso** | **Uso** |
| **Arreglo**  **Contador** | Entero | Poner 0 todas las componentes | Para i Desde 0 Hasta N – 1  Arreglo [i] = Arreglo [i] + 1  Fin Para | Se usa cuando la iteración finaliza. |
| **Arreglo Acumulador** | Entero o Real | Poner 0 todas las componentes | Para i Desde 0 Hasta N – 1  Arreglo [i] = Arreglo [i] + <variable>  Fin Para | Se usa cuando la iteración finaliza. |
| **Máximo** | Entero o Real | max = 0 | Si (max < Arreglo [i])  max = Arreglo [i]  Fin Si | Se almacena el valor máximo cuando la iteración finaliza |
| **Mínimo** | Entero o Real | min = 9999 | Si (min > Arreglo [i])  min = variable  Fin Si | Se almacena el valor mínimo cuando la iteración finaliza |
| **Bandera** | Lógico | b = falso | Si (<condicion>)  b = verdadero  Sino  i = i + 1  Fin Si | Se realizan acciones en función de su valor |

# Seguimiento de Algoritmos con Arreglos

Para realizar el seguimiento de un algoritmo con arreglos, debe crearse un espacio por cada componente del arreglo, para luego, colocar la dirección de memoria (100h) al final de su seguimiento.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Algoritmo Principal** | | | | | | |
| **Arreglo N°1** | | | **Variable N°1** | **Variable N°2** | **Variable n** | **Salida** |
|  |  |  |  |  |  |  |
| 100h | | |  |  |  |  |

En los subprogramas, cada vez que se realice una modificación, se invoca al arreglo a través de su dirección de memoria y se cambian sus valores en el algoritmo principal.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Subprograma N°1** | | | | |
| **Arreglo N°1** | **Variable N°1** | **Variable N°2** | **Variable n** | **Salida** |
| 100h |  |  |  |  |

Eje N°6

## Subarreglo

Generar un arreglo a partir de los valores de otro arreglo en función de un criterio determinado. Debe mantener el mismo tamaño y tipo que el arreglo de origen,

### Generación

Tipo cargarSubarreglo(Tipo arreglo[N], Tipo subarreglo[N])

Comienzo

Entero i, indiceMaximo

indiceMaximo = 0

Para i Desde 0 Hasta N – 1

Si ( <condicion> ) Entonces

subarreglo[indiceMaximo] = arreglo[i]

indiceMaximo = indiceMaximo + 1

Fin Si

Fin Para

Retorna (indiceMaximo)

Fin

### Muestra

void mostrarSubarreglo(Tipo subarreglo[N], Entero indiceMaximo)

Comienzo

Entero i

Para i Desde 0 Hasta indiceMaximo – 1

Escribir arreglo[i]

Fin Para

Fin

## Búsqueda

Recuperar datos de un arreglo en función de un criterio de búsqueda determinado. Una vez finalizada la búsqueda, si existe el elemento buscado, se obtiene su posición dentro del arreglo.

### Búsqueda Secuencial

Recomendado para arreglos con tamaño pequeño.

#### Método N°1 – Con Bandera

/\* Subprograma \*/

Logico buscarSecuencialConBandera(Tipo arreglo[N], Tipo elementoBuscado)

Comienzo

Entero i

Logico bandera

i = 0

bandera = Falso

Mientras ((i < N) y (bandera == Falso))

Si (arreglo[i] == elementoBuscado) Entonces

bandera = Verdadero

Sino

i = i + 1

Fin Si

Fin Mientras

Retorna (bandera)

Fin

/\* Algortimo Principal \*/

Comienzo

Tipo arreglo[N]

Tipo elementoBuscado

...

Logico bandera

...

Escribir "Ingrese el elemento que desea buscar"

Leer elementoBuscado

bandera = buscarSecuencialConBandera(arreglo[N], elementoBuscado)

Si (bandera == Verdadero) Entonces

Escribir "El elemento se encuentra en el arreglo"

Sino

Escribir "El elemento no se encuentra en el arreglo"

Fin Si

...

Fin

#### Método N°2 – Sin Bandera

/\* Subprograma \*/

Entero buscarSecuencialSinBandera(Tipo arreglo, Tipo elementoBuscado)

Comienzo

Entero i

i = 0

Mientras ((i < N) y (arreglo[i] != elementoBuscado))

i = i + 1

Fin Mientras

Retorna (i)

Fin

/\* Algortimo Principal \*/

Comienzo

Tipo arreglo[N]

Tipo elementoBuscado

...

Entero indiceBuscado

...

Escribir "Ingrese el elemento que desea buscar"

Leer elementoBuscado

indiceBuscado = buscarSecuencialSinBandera(arreglo, elementoBuscado)

Si (indiceBuscado < N) Entonces

Escribir "El elemento se encuentra en la posición N°", indiceBuscado

Sino

Escribir "El elemento no se encuentra en el arreglo"

Fin Si

...

Fin

#### Método N°3 – Variable Centinela

/\* Subprograma \*/

Entero buscarSecuencialCentinela(Tipo arreglo[N + 1], Tipo elementoBuscado)

Comienzo

Entero i

i = 0

arreglo[N] = elementoBuscado

Mientras (arreglo[i] != elementoBuscado)

i = i + 1

Fin Mientras

Retorna (i)

Fin

/\* Algortimo Principal \*/

Comienzo

Tipo arreglo[N + 1]

Tipo elementoBuscado

...

Entero indiceBuscado

...

Escribir "Ingrese el elemento que desea buscar"

Leer elementoBuscado

indiceBuscado = buscarSecuencialCentinela(arreglo, elementoBuscado)

Si (indiceBuscado == N) Entonces

Escribir "El elemento no se encuentra en el arreglo"

Sino

Escribir "El elemento se encuentra en la posición N°", indiceBuscado

Fin Si

...

Fin

### Búsqueda Binaria

Recomendado para arreglos ordenados de gran tamaño.

/\* Subprograma \*/

Entero buscarBinaria(Tipo arreglo[N], Tipo elementoBuscado)

Comienzo

Entero inferior, superior, medio

inferior = 0

superior = N - 1

medio = (inferior + superior) div 2

Mientras ( (inferior <= superior) y (arreglo[medio] != elementoBuscado) )

Si (elementoBuscado < arreglo[medio]) Entonces

superior = medio - 1

Sino

inferior = medio + 1

Fin Si

medio = (inferior + superior) div 2

Fin Mientras

Si (inferior <= superior) Entonces

Retorna (medio)

Sino

Retorna (-1)

Fin Si

Fin

/\* Algortimo Principal \*/

Comienzo

Tipo arreglo[N]

Tipo elementoBuscado

...

Entero indiceBuscado

...

Escribir "Ingrese el elemento que desea buscar"

Leer elementoBuscado

indiceBuscado = buscarBinaria(arreglo, elementoBuscado)

Si (indiceBuscado != -1) Entonces

Escribir "El elemento se encuentra en la posición N°", indiceBuscado

Sino

Escribir "El elemento no se encuentra en el arreglo"

Fin Si

...

Fin

Eje N°7

# Registro

Conjunto finito y ordenado de datos heterogéneos identificados con un único nombre.

### Declaración de Tipo

Registro <identificador de registro>

{

<tipo> <identificador de campo 1>

<tipo> <identificador de campo 2>

…

<tipo> <identificador de campo n>

}

### Declaración de Variable

<identificador de registro> <identificador de variable>

## Elementos

### Campo

Dato único de un registro que está almacenado en una o varias posiciones de memoria consecutivas.

<tipo de dato> <identificador de campo>

#### Acceso a un Campo

<identificador de registro>.<identificador de campo>

#### Anidamiento de Registros

Los campos de registro pueden ser, a su vez, un registro.

#### Arreglo como campo de un registro

Los campos de registro también pueden ser arreglos.

## Características

* Cada dato se almacena en posiciones contiguas de memoria.
* Es una estructura de datos estática, ya que el espacio que ocupa en la memoria es fijo.
* Sus elementos pueden ser de tipos de dato diferentes.

# Operaciones con Registros

## Lectura

Leer <identificador de registro>.<identificador de campo>

## Escritura

Escribir <identificador de registro>.<identificador de campo>

## Asignación

<identificador de registro>.<identificador de campo> = <valor asignado>

# Arreglo de Registros

Arreglo donde sus componentes son registros.

<identificador de registro> <identificador de arreglo> [<tamaño>]

# Métodos de Ordenamiento

Clasificación de los componentes de un arreglo en función de un criterio determinado.

## Clasificación según la memoria utilizada

### Ordenación Interna

Ordenamiento que se realiza en la memoria principal. Se puede acceder a cualquier elemento del arreglo sin recorrer los anteriores. El tiempo de acceso es igual para todas las componentes.

### Ordenación Externa

Ordenamiento que se realiza fuera de la memoria principal. El tiempo de acceso depende de la última posición de memoria que fue accedida.

## Clasificación según el método de ordenamiento

### Algoritmos de Intercambio

Los elementos se consideran de dos en dos, se comparan, y se intercambian en el caso de no estar en el orden requerido. Se repite este proceso hasta que se hayan analizado todos los elementos.

### Algoritmos de Inserción

Los elementos se consideran de uno a la vez. Cada elemento se inserta en la posición que le corresponde respecto al resto de elementos que ya han sido ordenados, en función de un criterio determinado.

### Algoritmos de Selección

Se selecciona un elemento para luego compararlo con el resto según un criterio determinado, colocándolo en la posición adecuada. El elemento seleccionado suele ser el mayor o el menor del conjunto de datos. Este proceso se repite hasta que se hayan analizado todos los elementos.

# Algoritmos de Ordenamiento

## Método de la Burbuja Mejorado

Void ordenarBurbujaMejorado (Tipo arreglo[N])

Comienzo

Entero ultimaPosicion, i, auxiliar, ultimoCambio

ultimoCambio = N - 1

ultimaPosicion = 1

Mientras (ultimaPosicion != -1)

Para i Desde 0 hasta ultimoCambio - 1

Si (arreglo[i] > arreglo[i + 1]) Entonces

auxiliar = arreglo[i]

arreglo[i] = arreglo[i + 1]

arreglo[i + 1] = auxiliar

ultimaPosicion = i

Fin Si

Fin Para

ultimoCambio = ultimaPosicion

Fin Mientras

Fin

## Método de Selección

Void ordenarSeleccion (Tipo arreglo[N])

Comienzo

Entero i, j, minimo, auxiliar

Para i Desde 0 Hasta N – 2

minimo = i

Para j Desde i + 1 Hasta n – 1

Si **(arreglo[j] < arreglo[minimo])** Entonces

minimo = j

Fin Si

Fin Para

**auxiliar = arreglo[i]**

**arreglo[i] = arreglo[minimo]**

**arreglo[minimo] = auxiliar**

Fin Para

Fin

## Método de Inserción directa

Void ordenarInsercionDirecta (Tipo arreglo[N])

Comienzo

Entero i, j, valor

Para i Desde 1 Hasta N – 1 //2

valor = arreglo[i]

j = i - 1

Mientras ( (j >= 0) y (valor < arreglo[j]) )

arreglo[j + 1] = arreglo[j]

j = j - 1

Fin Mientras

arreglo[j + 1] = valor

Fin Para

Fin

Eje N°8

# Eficiencia

Brindar una solución a un problema con el menor uso de recursos posible. Sus resultados deben ser correctos.

### Observaciones

* Se debe seleccionar el algoritmo más rápido, aunque el código sea más complejo
* Para calcular la eficiencia se tiene en cuenta
  + Cantidad de variables
  + Número de Comparaciones
  + Número de Asignaciones

## Complejidad de un Algoritmo

Cantidad de recursos que un algoritmo necesita para su ejecución

### Complejidad Espacial

Cantidad de espacio de almacenamiento que ocupa un algoritmo

### Complejidad Temporal

Periodo de tiempo que requiere un algoritmo en su ejecución. Depende de los siguientes factores

* Factores Externos
  + Datos de Entrada
  + Calidad del Código
  + Naturaleza y rapidez de instrucciones de la máquina
* Factores internos
  + Complejidad de Algoritmo

## Tiempo de Ejecución

Lenguaje C

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipos de Datos** | | | |
| **Tipo** | **Acción** | **Pseudocódigo** | **Lenguaje C** |
| **Simple** | **Numérico Entero** | Entero <nombre> | int <nombre> |
| **Numérico Real** | Real <nombre> | float <nombre> |
| **Carácter** | Caracter <nombre> | Char <nombre> |
| **Lógico** | Logico <nombre> | - |
| **Estructurado** | **Cadena** | Cadena <nombre> | char <nombre> [ <tamaño> ] |
| **Arreglo** | <tipo> <nombre> [ <tamaño> ] | <tipo> <nombre> [ <tamaño> ] |
| **Registro** | Registro <identificador>  {  <tipo> <nombre>,  <tipo> <nombre>,  …  <tipo> <nombre>  }  <identificador> <nombre> | struct <identificador>  {  <tipo> <nombre>,  <tipo> <nombre>,  …  <tipo> <nombre>  }  struct <identificador> <nombre> |

|  |
| --- |
| **Expresiones** |

|  |  |  |
| --- | --- | --- |
| **Tipo** | **Operador** | **Símbolo** |
| **Aritméticas** | **Suma** | + |
| **Resta** | - |
| **Multiplicación** | \* |
| **Cociente** | / |
| **Resto** | % |
|  |  |  |

# Acciones

## Simples

### Asignación

<identificador> = <valor>

### Lectura

scanf(“%<tipo>”, <variable>)

### Escritura

Escribir <Valor,Variable,Constante o Expresión>

## Estructuradas

Instrucción compuesta por acciones simples.

# Estructuras de Control

Instrucciones que determinan el flujo de ejecución de un programa.

## Secuencia

Determina el orden en el que se ejecutan las acciones de un algoritmo. Se ejecutan de arriba a abajo.

Acción 1

Acción 2

…

Acción N

## Selección

Determina la ejecución de una acción entre otras acciones alternativas en función de una o varias condiciones, modificando el flujo del programa.

### Selección Simple

Si (<condición>) Entonces

--- Acciones ---

Fin Si

### Selección Doble

Si (<condición>) Entonces

--- Acciones ---

Sino

--- Acciones ---

Fin Si

### Selección Múltiple

Según (<variable>)

valor 1: Acción 1

valor 2: Acción 2

…

valor n: Acción n

Fin Según

## Iteración

Determina la repetición de una o varias acciones en función de una o varias condiciones.

### Para

Repite una o varias acciones una cantidad de veces determinada. Se conoce la cantidad de iteraciones que se realiza.

Para <variable> desde <valor inicial> hasta <valor final>

--- Acciones ---

Fin Para

#### Variable de Control

Variable que establece la cantidad de iteraciones a realizar, contando las iteraciones realizadas.

### Mientras

Repite una o varias acciones en función de una o varias condiciones. La condición se evalúa al inicio de la estructura. No se conoce la cantidad de iteraciones que realiza.

Mientras (<condición>)

--- Acciones ---

Fin Mientras

#### Variable Centinela

Variable que determina la cantidad de iteraciones en función de su valor.

### Hacer – Mientras

Repite una o varias acciones en función de una o varias condiciones. La condición se evalúa al final de la estructura. No se conoce la cantidad de iteraciones que realiza.

Hacer

--- Acciones ---

Mientras (<condición>)