

Algoritmos computacionales

Sitio:

[Agencia de Aprendizaje a lo largo de la Vida](#)

Curso:

Técnicas de Programación - Turno mañana

Libro:

Algoritmos computacionales

Imprimido por:

Tomas Friz

Día:

viernes, 17 de septiembre de 2021, 12:45

Tabla de contenidos

- 1. Secciones de algoritmos
- 2. Herramientas y técnicas de representación de Algoritmos
- 3. Variables y constantes
- 4. Operadores
- 5. Estructuras
- 6. Estructuras Condicionales
- 7. Estructuras repetitivas

1. Secciones de algoritmos



Todo algoritmo consta de tres secciones principales.

Entrada: Es la introducción de datos para ser transformados por medio del proceso. La introducción de estos datos puede realizarse por medio de dispositivos de entrada tales como:

- El teclado
- El Mouse
- El disco rígido
- Discos externos
- Pendrives
- Lector de código de barra
- Lector de tarjetas magnéticas/ de cajeros automáticos.

Proceso: Es el conjunto de operaciones a realizar para dar solución al problema utilizando los datos de entrada. Este proceso se ejecuta internamente en la CPU.

Salida: Son los resultados obtenidos/deseados a través del proceso, y en conjunto a los datos de entrada obtendremos como respuesta la solución a nuestro problema.

Las salidas se producen en dispositivos de salida estos son:

- Pantalla
- Impresoras
- Parlantes
- Discos rígidos
- Pendrives

Esto genera un círculo virtuoso que puede repetirse una y otra vez

Ejemplos de algoritmos?

1.
2.
3.
4. ...
5. ...
6.

2. Herramientas y técnicas de representación de Algoritmos

[Pseudocódigo](#). (más prezi)

<https://prezi.com/whlgeiu4vmet/pasos-de-un-algoritmo-con-pseudocodigo/>

El [pseudocódigo](#) es una forma de expresar los distintos pasos que va a realizar un algoritmo/programa, de la forma más parecida a un lenguaje de programación, pero utilizando términos que pueden ser entendidos sin dificultad. Su principal función es la de representar los pasos la solución a un problema o algoritmo, de la forma más detallada posible, utilizando un lenguaje cercano al de programación. El [pseudocódigo](#) no puede ejecutarse en un ordenador, es un código escrito para que lo entienda el ser humano y no la máquina.

Otra característica del [pseudocódigo](#) es que si bien existen convenciones en muchos de los términos que se utilizan, podemos incluir nuevos términos o reemplazar términos ya existentes siempre y cuando sean claros y específicos.

Diagramas de flujo:

Los diagramas de flujo son una manera de representar gráficamente un algoritmo o un proceso de cualquier naturaleza. La representación gráfica de estos procesos emplea una serie determinada de figuras geométricas que representan cada paso puntual del proceso. Estas formas definidas se conectan entre sí a través de flechas y líneas que marcan la dirección del flujo de nuestros algoritmos y establecen el recorrido del proceso, como si fuera un mapa.

Ver si Ejemplo

Diagramas de Nassi-Shneiderman (NSD)

Estos diagramas combinan la utilización de bloques gráficos (también llamados cajas) y el pseudo código. Cada una de los bloques utilizados representan una estructura específica.

3. Variables y constantes

Variables

Una variable es un espacio en memoria donde guardaremos diferentes tipos de valores y a los cuales podremos acceder mediante un nombre simbólico. Los tipos de variables y la cantidad de tipos que podamos utilizar dependen en realidad de cada [lenguaje de programación](#).

Constante

Una constante es un espacio de memoria donde podemos almacenar valores de distintos tipos, pero que a diferencia de las variables, estos valores son inmodificables, es decir, cuando una constante toma un valor este no podrá ser modificado.

Los tipos de valores que pueden almacenar las constantes son los mismos que las variables

La forma habitual de definir una variable es especificando el tipo, seguidamente del nombre simbólico. Pero como dijimos anteriormente esto dependerá del lenguaje utilizado.

4. Operadores

Operadores matemáticos

Un operador, es un elemento que indica el tipo de operación que se le va a aplicar a uno o más datos. Por ejemplo, por medio de estos operadores podemos realizar operaciones aritméticas, hacer comparaciones, asignar valores a nuestras variables

Operadores lógicos/aritméticos (pendiente)

Ejemplos.Acomuladores

Ejemplos.Contadores

Ejemplos.Condicionales

Ejemplos.Iteradores

5. Estructuras

Estructuras: A la hora de programar, podemos decir que existen 2 tipos de estructuras

- Las estructuras de control
- Las estructuras repetitivas (pendiente)

6. Estructuras Condicionales

Las estructuras condicionales permiten ejecutar una o más líneas de nuestro código dependiendo de una condición.

Supongamos que queremos hacer un programa que le solicite al usuario su edad y que como salida nos informe por pantalla si una persona es mayor de edad o no.

Para hacer esto necesitamos de una estructura que por medio de una condición evalúe la edad que el usuario no ingresó y según el resultado de esta evaluación informaremos si el usuario es mayor de edad o no. La condición para que el usuario sea mayor de edad, es que su edad sea mayor o igual a 18.

(El código necesario para resolver esto podría ser en JAVA/// PASARLO A PSEUDOCODIGO)

```
// creamos una variable donde almacenaremos la edad que ingrese nuestro usuario.
```

```
Int Edad;
```

```
// Mostramos por pantalla un mensaje solicitando la edad del usuario
```

```
Cout << "Ingrese su edad";
```

```
// Almacenamos el valor ingresado por el usuario en la variable Edad
```

```
Cin >> Edad;
```

```
// Ahora evaluamos si la edad es mayor o igual 18
```

```
Si (Edad >= 18) entonces
```

```
{
```

```
    // si la condición se cumple mostramos el siguiente mensaje
```

```
    Cout << "Ud es mayor de edad";
```

```
}
```

```
Else
```

```
{
```

```
    // Sino se cumple la condición mostramos el siguiente mensaje
```

```
    Cout << "Ud es menor de edad";
```

```
}
```

Como verán para resolver este problema utilizamos la estructura condicional Si (If en ingles). Esta estructura esta formada por una condición la cual estará entre () – en este caso (Edad >= 18), en caso que esta condición sea verdadera mostrara el mensaje: “Ud es mayor de edad” y en caso que no se cumpla la condición, es decir que la condición es falsa, se ejecutara el código que esta después de la clausula else (sino) emitiendo el mensaje: “Ud es menor de edad”

El uso de la clausula “else” es opcional esto quiere decir que podemos no incluirla si no es necesario. No siempre necesitamos que se ejecute código en caso que una condición sea Falsa.

Existe una forma alternativa de utilizar la estructura condicional If, es decir, no solamente utilizando esta forma opcional, podríamos hacer nuevas evaluaciones de una condición en caso que la primera no se cumpla.

Veamos un ejemplo:

Supongamos que queremos realiza un programa para un Club deportivo, donde los socios de este club son divididos en 3 categorías de acuerdo a su edad. Estas categorías son “Infantil”, “Juvenil” y “Adulto”.

Nuestro programa le solicitara al usuario que ingrese su edad y mostrara un mensaje informando la categoría que le corresponde.

7. Estructuras repetitivas

Pendiente