Material descargable

Fundamentos de la Programación

¿QUÉ ES LA PROGRAMACIÓN?

La programación es el arte de dar instrucciones precisas y ordenadas para realizar una tarea específica. Este concepto trasciende diversos ámbitos de la vida cotidiana, desde planificar una salida con amigos hasta organizar unas vacaciones. En el contexto de la informática, la programación se centra en la creación de programas mediante la elaboración de algoritmos, que son secuencias de pasos definidos para que una máquina ejecute las acciones deseadas. Los componentes fundamentales de un programa son el lenguaje de programación y los algoritmos, los cuales se combinan para lograr el funcionamiento deseado.

¿QUÉ ES UN LENGUAJE DE PROGRAMACIÓN?

Un lenguaje de programación es un sistema formal que **permite** a los programadores **escribir una serie de instrucciones, acciones consecutivas, datos y algoritmos para resolver problemas**. Estas instrucciones son posteriormente traducidas a un lenguaje de máquina comprensible por el hardware de la computadora.

Existen diferentes tipos de lenguajes de programación:

- **Lenguaje máquina:** Es el más primitivo, compuesto por una secuencia de dígitos binarios (0 y 1) que la computadora interpreta directamente. Por ejemplo: 10110000 01100001.
- **Lenguajes de alto nivel:** Estos lenguajes tienen como objetivo simplificar el trabajo del programador al utilizar instrucciones más comprensibles. Además, permiten escribir código utilizando estructuras similares a los idiomas humanos como español o inglés. Para que estos códigos sean ejecutables, se traducen al lenguaje máquina mediante traductores o compiladores. Por ejemplo: Java, Python, Ruby, C++, JavaScript, entre otros.

¿QUÉ ES UN ALGORITMO?P

En la programación, los algoritmos son fundamentales. Te permiten dar instrucciones claras a tu programa para resolver problemas específicos. **Un algoritmo es un método para dar instrucciones y resolver un problema**. Consiste en una secuencia de pasos lógicamente ordenados que, partiendo de la información que proporcionas, te permite obtener resultados concretos que forman la solución del problema.

Los algoritmos son independientes tanto del lenguaje de programación que utilices como de la computadora en la que los ejecutes. En cada problema, el algoritmo puede expresarse en un lenguaje de programación diferente y ejecutarse en una computadora distinta; sin embargo, el algoritmo siempre será el mismo. Por ejemplo, en una analogía con la vida diaria, una receta de cocina se puede expresar en español, inglés o francés, pero los pasos para elaborar el plato se siguen sin importar el idioma del cocinero.

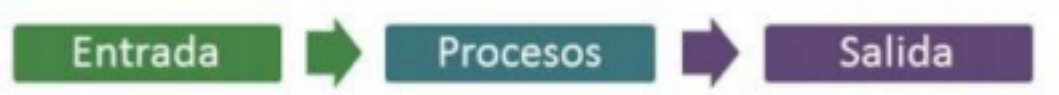
Los algoritmos son más importantes que los lenguajes de programación o las computadoras. **Un lenguaje de programación es solo un medio para expresar un algoritmo y una computadora es simplemente un procesador para ejecutarlo.** Tanto el lenguaje de programación como la computadora son los medios para obtener un fin: hacer que el algoritmo se ejecute y se efectúe el proceso correspondiente.

CARACTERÍSTICAS FUNDAMENTALES DE LOS ALGORITMOS

1. **Precisión**: Un algoritmo debe ser preciso y especificar el orden de realización de cada paso de manera clara y sin ambigüedades.
2. **Definición específica:** Un algoritmo debe estar específicamente definido. Esto significa que si se ejecuta el mismo algoritmo dos veces con los mismos datos de entrada, se debe obtener el mismo resultado cada vez.
3. **Finitud**: Un algoritmo debe ser finito, es decir, debe tener un número finito de pasos y debe terminar en algún momento. Debe tener un inicio y un final definidos.
4. **Corrección**: El resultado del algoritmo debe ser el resultado esperado. Un algoritmo debe producir resultados correctos para los datos de entrada proporcionados.
5. **Independencia**: Un algoritmo es independiente tanto del lenguaje de programación en el que se expresa como de la computadora que lo ejecuta. Debe funcionar de manera consistente independientemente del entorno de ejecución.

Además, el proceso de programación implica resolver constantemente problemas de manera algorítmica. Esto implica plantear el problema de manera que se indiquen los pasos necesarios para obtener los resultados deseados a partir de

los datos conocidos. Un algoritmo básicamente consta de tres elementos: datos de entrada o información de entrada, procesos y la información de salida. La estructura de un programa sigue esta misma lógica: datos de entrada, proceso y salida.

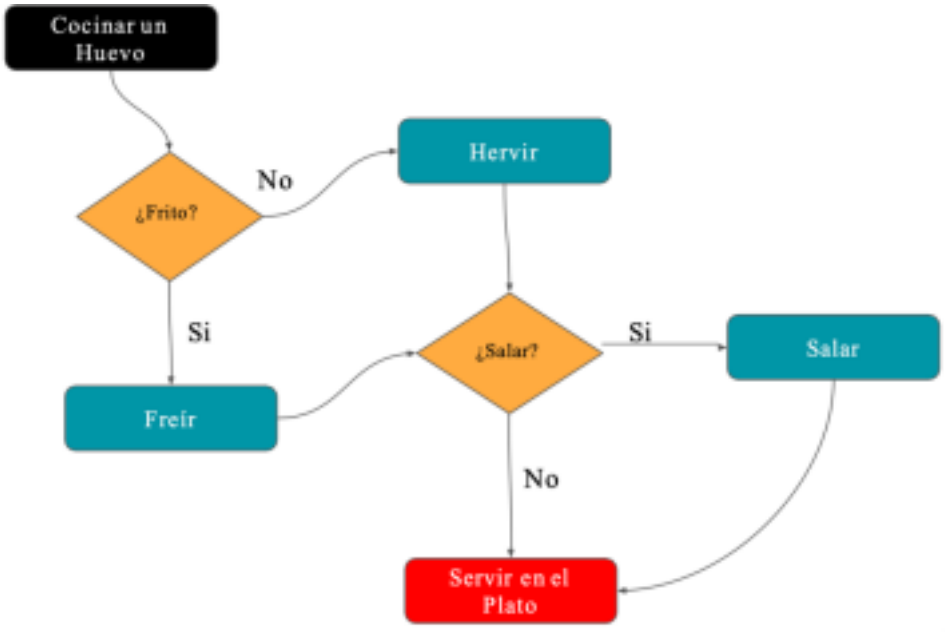


¿CÓMO REPRESENTAR UN ALGORITMO?

La mejor manera de representar un algoritmo es a través de un diagrama de flujo.

Un diagrama de flujo es una esquematización gráfica de un algoritmo, que muestra de forma visual los pasos o procesos necesarios para alcanzar la solución de un problema. Estos diagramas se basan en el uso de diferentes símbolos para representar operaciones específicas, lo que permite visualizar las distintas acciones que deben llevarse a cabo para resolver un problema, indicando el orden lógico en que deben realizarse.

Veamos un ejemplo de cómo sería un diagrama de flujo para los pasos de cocinar un huevo:



Como podemos observar, el diagrama muestra paso a paso cómo cocinar un huevo, incluyendo todas las opciones disponibles a la hora de prepararlo.

Programación de programas

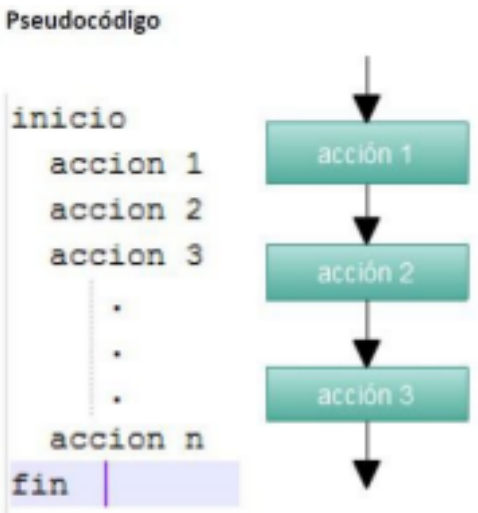
Hemos hablado sobre la programación y sus dos componentes principales: los lenguajes de programación y los algoritmos. Ahora, ¿cómo se reflejan estos conceptos en un programa?

Un programa es esencialmente una serie de algoritmos escritos en algún lenguaje de programación para computadoras. Consiste en un conjunto de instrucciones, es decir, órdenes dadas a la computadora, que llevarán a cabo una tarea específica. En resumen, un programa es un medio para alcanzar un fin, y ese fin es usualmente definido como la información necesaria para resolver un problema.

Diseño del programa: Para diseñar un programa, podemos utilizar herramientas de representación de algoritmos, también conocidas como lenguajes de programación. Estas nos permiten definir la secuencia de pasos necesarios para obtener el resultado deseado.

Especificaciones del programa: Una vez que decidimos desarrollar un programa, necesitamos establecer un conjunto de especificaciones que debe cumplir. Esto incluye la entrada, la salida y los algoritmos de resolución, que indican cómo obtener las salidas a partir de las entradas proporcionadas.

Estructura de un programa secuencial: Un programa puede ser lineal (secuencial) o no lineal. Un programa lineal ejecuta las instrucciones secuencialmente, sin bifurcaciones, decisiones o comparaciones.



Estructura de un programa secuencial

Codificación: Una vez que tenemos las especificaciones, pasamos a la codificación del programa. Esto implica escribir la solución del problema en una serie de instrucciones detalladas en un lenguaje de programación reconocible por la computadora.

Escritura de algoritmos/programas: Un algoritmo consta de dos partes: una cabecera de programa y un bloque de algoritmo. La cabecera del programa

comienza con la palabra reservada "algoritmo" seguida por el nombre asignado al programa. El bloque de algoritmo incluye las acciones de declaración y las acciones ejecutables.

Elementos de un programa: Los elementos básicos de un programa son los componentes que conforman las instrucciones para crear y resolver problemas. Estos elementos incluyen **identificadores, variables, constantes, operadores y palabras reservadas.**

IDENTIFICADORES

Los identificadores son conjuntos de caracteres alfanuméricos de cualquier longitud que se utilizan para identificar las entidades del programa, como el nombre del programa, nombres de variables, constantes, subprogramas, entre otros. En PSeInt, los identificadores deben constar solo de letras, números y/o guiones bajos (_), comenzando siempre con una letra y escritos preferiblemente en minúsculas. Es importante destacar que no pueden contener tildes ni la letra "Ñ", ya que esto podría generar errores en el programa.

Además, es crucial asignar nombres claros a los identificadores. Por ejemplo, si queremos representar una frase, el identificador podría ser simplemente "frase", y si deseamos representar una suma, podríamos llamar al identificador "suma".

¿Qué pasa si necesitamos un identificador de más de una palabra?

Cuando necesitamos crear un identificador compuesto por más de una palabra, utilizamos una convención llamada camelCase. Esta convención consiste en escribir la primera palabra en minúscula y comenzar cada palabra subsiguiente con mayúscula. Por ejemplo, si queremos que el identificador refleje que estamos sumando números, lo llamaríamos "sumaNumeros", donde la primera letra de la segunda palabra está en mayúscula. De esta manera, mantenemos la regla de escribir siempre en minúsculas.

Esta convención no se limita solo a la segunda palabra; si necesitamos agregar una tercera o cuarta palabra, seguimos el mismo principio. Por ejemplo, podríamos tener el identificador "sumaNumerosEnteros" para una representación aún más clara del concepto.

VARIABLES Y CONSTANTES

Los programas de computadora necesitan información para resolver problemas. Esta información puede ser un número, un nombre, entre otros. Para organizar esta información y asegurarnos de no perderla ni acceder a ella cuando sea

necesario, utilizamos variables y constantes. Podemos imaginar las variables y constantes como pequeñas cajas que almacenan información en su interior.

Cada variable y constante tiene un identificador, que es como su nombre y nos ayuda a distinguirlas entre sí. Este identificador nos indica qué información contiene cada una y nos facilita su manipulación dentro del programa.

Entre la información que manejamos, a veces necesitaremos valores que no cambian, como por ejemplo el valor de Pi. Estos valores se llaman constantes. Por otro lado, hay valores que necesitamos que cambien durante la ejecución del programa, como por ejemplo el resultado de un cálculo. Estos valores se almacenan en variables.

Una variable es un objeto o tipo de dato cuyo valor puede cambiar durante la ejecución del programa. Dependiendo del lenguaje de programación, existen diferentes tipos de variables, como enteras, reales, caracteres, lógicas y de cadena. Cada tipo de variable puede almacenar solo un tipo específico de dato. Por ejemplo, una variable de carácter solo puede almacenar caracteres, mientras que una variable entera solo puede almacenar números enteros.

Por otro lado, una constante también puede ser de diferentes tipos, como enteras, reales, caracteres, lógicas y de cadena. Sin embargo, a diferencia de las variables, las constantes permanecen con el mismo valor durante toda la ejecución del programa. Por ejemplo, el valor de Pi siempre será el mismo y no cambiará mientras se ejecute el programa.

TIPOS DE DATOS EN PSEINT

En PSeInt, las variables y constantes almacenan información según el tipo de dato que definamos para ellas. Por ejemplo, si indicamos que una variable guardará números enteros, significa que su tipo de dato es entero.

Los tipos de datos disponibles en PSeInt son:

- **Entero**: Solo permite almacenar números enteros.
- **Real**: Permite almacenar números con decimales. El punto se utiliza como separador decimal. Por ejemplo, 3.14.
- **Carácter**: Utilizado para almacenar un solo carácter. Los caracteres se encierran entre comillas simples. Por ejemplo, 'a' o 'A'.
- **Lógico**: Se utiliza para almacenar expresiones lógicas que pueden ser verdaderas o falsas. Se representa como encendido/apagado o 0/1. Este tipo de dato se utiliza principalmente para condiciones en el programa.
- **Cadena**: Permite almacenar cadenas de caracteres. Las cadenas se encierran entre comillas dobles. Por ejemplo, "esto es una cadena" o "hola mundo".

Es importante tener en cuenta:

- En PSeInt, los términos "cadena" y "carácter" son equivalentes y pueden usarse indistintamente.
- El plural de "carácter" puede ser "caracteres" o "cadena".

¿CÓMO CREO UNA VARIABLE?

Cuando creamos una variable, es importante asignarle un identificador siguiendo las reglas que hemos aprendido en la sección de identificadores, y especificar el tipo de dato que queremos que almacene. Para hacer esto, utilizamos la palabra reservada "Definir". Esta instrucción nos permite especificar el tipo de una o más variables.

La sintaxis para definir una variable es la siguiente:

```
Definir <nombre_variable> como <tipo_de_dato>
```

Primero escribimos la palabra "Definir", luego el nombre que queremos asignar a nuestra variable y finalmente el tipo de dato que queremos que almacene.

Por ejemplo, si queremos declarar una variable de tipo entero, escribimos:

```
Definir varNumero Como Entero
```

De esta manera, "varNumero" se convierte en una variable capaz de almacenar valores enteros.

¿CÓMO ASIGNAMOS VALORES A LAS VARIABLES?

La instrucción de asignación nos permite almacenar un valor en una variable previamente definida. Es nuestra forma de guardar información en una variable para utilizarla más adelante. Podemos realizar esta asignación de dos maneras diferentes, utilizando el signo igual "=" o una flecha "<-".

La estructura de la asignación es la siguiente: <variable> <- <expresión> o <variable> = <expresión> , donde la expresión puede ser una expresión matemática o lógica, otra variable o una constante.

Cuando se ejecuta la asignación, primero se evalúa la expresión de la derecha y luego se asigna el resultado a la variable de la izquierda. Es importante tener en cuenta que el tipo de dato de la variable y el de la expresión deben coincidir.

TIPOS DE INSTRUCCIONES

Las instrucciones —acciones— básicas que se pueden implementar de modo general en un algoritmo y que esencialmente soportan todos los lenguajes son las siguientes:

INSTRUCCIONES DE INICIO/FIN

Son utilizadas para delimitar bloques de código. Por ejemplo, Algoritmo y FinAlgoritmo.

INSTRUCCIONES DE ESCRITURA O SALIDA

Se utilizan para escribir o mostrar mensajes o contenidos de las variables en un dispositivo de salida. La salida puede aparecer en un dispositivo de salida (pantalla, impresora, etc.).

La operación de salida se denomina escritura (escribir). La instrucción Escribir permite mostrar información y valores de variables en la interfaz grafica o ambiente.

```
Escribir <exprl> , <expr2> , ... , <exprN>
```

Con el escribir también podemos mostrar variables o valores con un mensaje previo, para esto vamos a concatenar nuestra variable, usando una coma o un espacio en blanco, con un mensaje entre comillas.

```
Escribir "Mensaje entre comillas" variable
Escribir "La suma de los números es:" suma
```

INSTRUCCIONES DE LECTURA

Para que las computadoras realicen cálculos útiles, es necesario proporcionarles los datos necesarios para ejecutar las operaciones que generarán resultados. Estos datos se introducen mediante operaciones de entrada, que leen la información desde un dispositivo de entrada y la asignan a variables específicas.
La operación de entrada, comúnmente conocida como "leer", permite ingresar datos desde dispositivos como el teclado, tarjetas perforadas o unidades de disco al procesador de la computadora.
En PSeInt, la instrucción Leer se utiliza para ingresar información desde el teclado a través de la interfaz gráfica. Su sintaxis es la siguiente:

```
Leer <variable1>, <variable2>, ..., <variableN>
```

Por ejemplo, podemos definir una variable entera llamada "num" y asignarle un valor

mediante la instrucción Leer.

OPERADORES ALGEBRAICOS

Los operadores algebraicos o también conocidos como operadores aritméticos. Realizan operaciones aritméticas básicas: suma, resta, multiplicación, división, potenciación y modulo para datos de tipo numérico tanto enteros como reales. Estas operaciones son binarias porque admiten dos operandos.

Operador Algebraicos	Significado	Resultado
+	Suma	suma = 2 + 2
-	Resta	resta = 10 - 4
*	Multiplicación	multiplicación = 10 * 2
/	División	división = 9 / 3
^	Potenciación	potencia = 10 ^ 2
% o MOD	Módulo (resto de la división entera)	resto = 4 % 2

Reglas de prioridad:
Cuando una expresión contiene dos o más operadores, es necesario seguir reglas matemáticas para determinar el orden de las operaciones. Estas reglas, conocidas como reglas de prioridad, son las siguientes:

1. Las operaciones dentro de paréntesis se evalúan primero. Si hay paréntesis anidados (uno dentro de otro), se evalúan primero las expresiones más internas.
2. Las operaciones aritméticas siguen un orden de prioridad generalmente aceptado:

- Operadores dentro de paréntesis. ()
- Operadores unitarios (potenciación).
- Operadores de multiplicación (*), división (/), y módulo (%).
- Operadores de suma (+) y resta (-).

En caso de que haya varios operadores con la misma prioridad en una expresión o subexpresión entre paréntesis, el orden de evaluación es de izquierda a derecha, lo que se conoce como asociatividad.