PROGRAMACIÓN I - UNIDAD 1: INTRODUCCIÓN A LA PROGRAMACIÓN

Autor: Ing. Anderson, Florencia Marina

Contenido

Conceptos generales

Para poder comprender al mundo de la programación, primero necesitamos conocer acerca de ciertos conceptos:

Software:

Sistema formal de un sistema informático, que comprende el conjunto de componentes lógicos que hacen posible la realización de tareas específicas.

El software es la parte lógica e intangible que poseen los sistemas informáticos.

El software utiliza el hardware para poder operar (el hardware es la parte física y tangible del sistema informático, por ejemplo: CPU, memoria RAM, monitor, mouse, teclado, cámara de video, micrófono, etc.)

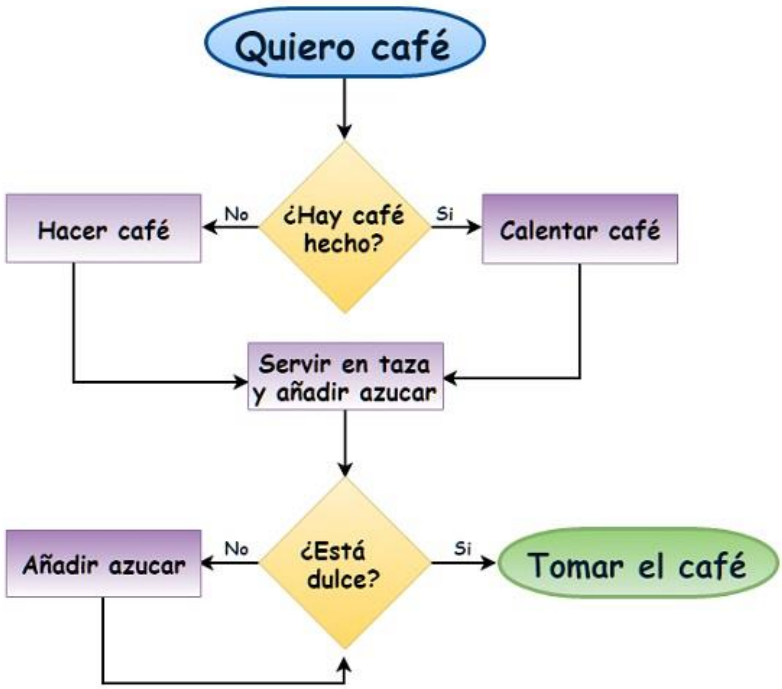
Ejemplos de software: Un videojuego, un editor de texto, la aplicación que utilizamos para pedir un taxi, aplicaciones de mensajería como Whatsapp, Telegram, etc

Algoritmo: Conjunto ordenado de operaciones sistemáticas que permite hacer un cálculo y hallar la solución de un tipo de problema.

Los algoritmos describen la solución a un problema en términos de los datos requeridos para representar el caso del problema y el conjunto de pasos necesarios para producir el resultado pretendido.

Vamos a utilizar algoritmos para la programación, pero en la vida cotidiana ya aplicamos algoritmos sin darnos cuenta. Algunos ejemplos son los manuales de usuario, que muestran algoritmos para usar un aparato, las instrucciones que recibe un trabajador de su patrón o una receta de un plato.

Los algoritmos pueden representarse de distintas maneras, a través de pseudocódigo, diagrama de flujo, diagrama de chapín, etc. En ésta materia nos vamos a enfocar en la representación a través de diagrama de flujo.



Lenguaje de programación: Un lenguaje de programación es un lenguaje formal (o artificial, es decir, un lenguaje con reglas gramaticales bien definidas) que le proporciona a una persona, en este caso el programador, la capacidad de escribir (o programar) una serie de instrucciones o secuencias de órdenes en forma de algoritmos con el fin de controlar el comportamiento físico o lógico de un sistema informático, de manera que se puedan obtener diversas clases de datos o ejecutar determinadas tareas.

Los lenguajes de programación deben ser precisos, deben poder traducirse sin ambigüedad en lenguaje máquina para que sean ejecutados por computadoras. Pero deben ser utilizados (leídos, comentados, probados, etc.) por personas.

Pseudocódigo: Mezcla de lenguaje de programación y léxico habitual en el que estamos trabajando, sin conocer lenguajes específicos.

```
subproceso funcion bizzbuzz
para (i <- 1; i<=100; i++) {
    establecer print_number a verdadero;
    Si i es divisible por 3
        escribir "Bizz";
        establecer print_number a falso;
    Si i es divisible por 5
        escribir "Buzz";
        establecer print_number a falso;
    Si print_number, escribir i;
    escribir una nueva línea;
}
```

¿Qué es la programación?

Programación es el proceso de tomar un algoritmo y codificarlo en un lenguaje de programación, de modo que pueda ser ejecutado por una computadora. Aunque existen muchos lenguajes de programación y muchos tipos diferentes de computadoras, el primer paso es la necesidad de tener una solución. Sin un algoritmo no puede haber un programa.

La programación es una actividad humana que requiere de mucha creatividad, no solo basta comprender los conceptos abstractos, herramientas y técnicas; todas esas teorías representan solo una pequeña parte de lo que necesita una persona para convertirse en un programador de software.

Se conoce como programación al proceso de diseñar, codificar, depurar y mantener el código fuente de un programa (software).

Es el arte que nos permite representar por medio de código las distintas soluciones.

La programación es una actividad colaborativa y debe basarse en la comunicación.

Fases de la resolución de un problema

La teoría de la resolución de un problema de programación nos indica que existen 6 fases fundamentales en éste proceso, y éstas son:

1. **Análisis del problema:** Estudiamos y analizamos el problema hasta comprenderlo por completo y seguimos las especificaciones otorgadas por el cliente, empresa o persona que encarga el programa.
2. **Diseño del algoritmo**.
3. **Codificación**.
4. **Ejecución**, **verificación y depuración**: El programa se ejecuta y se revisa rigurosamente en busca de algún error.
5. **Mantenimiento:** Una vez que nuestro programa funciona perfectamente, éste se actualiza y se modifica según sea necesario.
6. **Documentación**: Se refiere a que el programa se documenta por completo desde el momento del análisis hasta la parte de mantenimiento.

Tipos de lenguajes de programación

Por el nivel:

Existen tres tipos de lenguajes claramente diferenciados por nivel; el lenguaje máquina y los lenguajes de bajo nivel y los de alto nivel.

- Lenguaje de Máquina: es el lenguaje de programación que entiende directamente la máquina (computadora). Este lenguaje de programación utiliza el alfabeto binario, es decir, el 0 y el 1.
- Lenguaje de bajo nivel: Son mucho más fáciles de utilizar que el lenguaje máquina, pero dependen mucho de la máquina o computadora como sucede con el lenguaje máquina. Un código escrito en un lenguaje de bajo nivel interactúa directamente con el procesador de la computadora o CPU y es capaz de ejecutar comandos muy básicos que son generalmente difíciles de leer por una persona. Como un ejemplo de lenguaje de bajo nivel podemos mencionar el Assembly (lenguaje ensamblador), que utiliza palabras clave para ejecutar comandos básicos como leer, mover y almacenar datos.
- Lenguaje de alto nivel: Los lenguajes de programación de alto nivel son más fáciles de aprender porque se usan palabras o comandos del lenguaje natural, generalmente del inglés. Ejemplo: C, Pascal, Python

Por la forma como se ejecutan:

El programa escrito en un lenguaje de programación de alto nivel **(programa fuente)** no se puede ejecutar directamente en una computadora.

La opción más común es compilar el programa obteniendo un **código objeto**, aunque también, si el lenguaje lo soporta, puede ejecutarse en forma directa pero solo a través de un intérprete. A partir de esto entonces se diferencian:

- **Lenguajes compilados:** necesitan de un programa especial que lea el código fuente y cree un archivo binario ejecutable para una plataforma específica. Ejm: C, Pascal.
- **Lenguajes interpretados:** necesitan de un programa que traduzca en directo el código fuente escrito a instrucciones de la plataforma en la que se ejecutan.

Los primeros son más rápidos debido a que al compilar un programa las ordenes son más entendibles para la computadora, mientras que al interpretarlo la máquina primero debe leer el código y convertir al paso las instrucciones a instrucciones de máquina entendibles para ella.

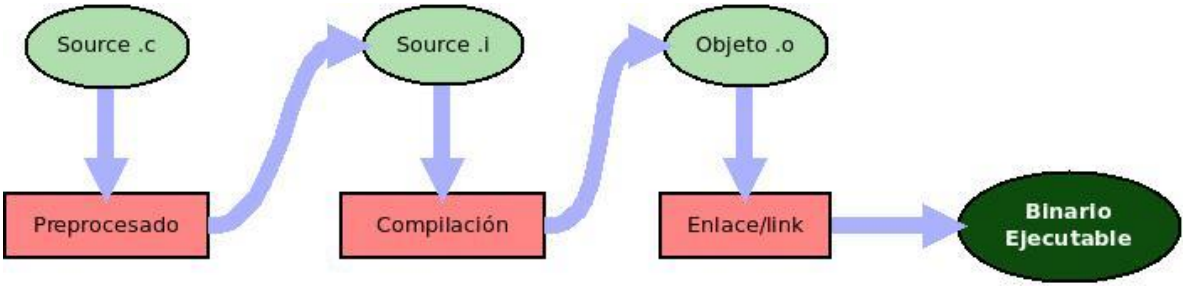
Algunos lenguajes, tal como BASIC, disponen de ambas formas de ejecución.

Compilación en C: Desde un fuente hasta un binario ejecutable

Pre-procesamiento: El código fuente escrito en C es pre-procesado por el compilador que creará un objeto de extensión<<.i>>. En éste paso lo que hace el compilador es interpretar todas las directivas de pre-procesamiento que hayamos utilizado, como **#define, #include, #ifdef,** etc… y además, eliminará todos los comentarios que hayamos escrito.

Compilación: El siguiente paso es compilar el código y el resultado es un código objeto no ejecutable de extensión <<.o>>.

Enlace: Para que el código objeto se vuelva ejecutable, se realiza un proceso de enlace del código con las librerías del sistema que se utilizan.



IDE

Un entorno de desarrollo integrado (IDE), es un software que proporciona servicios integrales para facilitarle al programador el desarrollo de software.

Normalmente, un IDE consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador (para ayudar a buscar errores en el software). La mayoría de los IDE tienen auto-completado inteligente de código (IntelliSense). Algunos IDE contienen un compilador, un intérprete, o ambos

Un ejemplo de un IDE es “Visual Sudio Code” que es compatible con varios lenguajes de programación como C, C#, Java, PHP, Visual Basic, HTML entre otros.

Paradigmas de programación

Un paradigma de programación es una manera o estilo de programación de software. Los lenguajes de programación adoptan uno o varios paradigmas en función del tipo de órdenes que permiten implementar como, por ejemplo, Python o JavaScript, que son multiparadigmas.

Paradigma imperativo

Los programas consisten en una sucesión de instrucciones o conjunto de sentencias, como si el programador diera órdenes concretas. El desarrollador describe en el código paso por paso todo lo que hará su programa.

Algunos lenguajes: **Pascal, COBOL, FORTRAN, C, C++**, etc.

Dentro del paradigma imperativo encontramos a

- Programación estructurada: es un tipo de programación imperativa donde el flujo de control se define mediante bucles anidados, condicionales y subrutinas, en lugar de a través de GOTO.
- Programación procedimental: Este paradigma de programación consiste en basarse en un número muy bajo de expresiones repetidas, englobarlas todas en un procedimiento o función y llamarlo cada vez que tenga que ejecutarse.
- Programación modular: consiste en dividir un programa en módulos o subprogramas con el fin de hacerlo más manejable y legible. Se trata de una evolución de la programación estructurada para resolver problemas de programación más complejos.

Paradigma declarativo

Este paradigma no necesita definir algoritmos puesto que describe el problema en lugar de encontrar una solución al mismo. Este paradigma utiliza el principio del razonamiento lógico para responder a las preguntas o cuestiones consultadas.

Este paradigma a su vez se divide en dos:

- Programación Lógica: El problema se modela con enunciados de lógica de primer orden. Ejemplo: Prolog
- Programación funcional: Los programas se componen de funciones, es decir, implementaciones de comportamiento que reciben un conjunto de datos de entrada y devuelven un valor de salida. Ejemplos: Lisp, Scala, R

Programación orientada a objetos

En este modelo de paradigma se construyen modelos de objetos que representan elementos (objetos) del problema a resolver, que tienen características y funciones. Permite separar los diferentes componentes de un programa, simplificando así su creación, depuración y posteriores mejoras. La programación orientada a objetos disminuye los errores y promociona la reutilización del código. Es una manera especial de programar, que se acerca de alguna manera a cómo expresaríamos las cosas en la vida real. Ejemplos de lenguajes de programación orientados a objetos serían Java, Python o C#

Programación reactiva

Este paradigma se basa en **escuchar lo que emite un evento o cambios en el flujo de datos**, en donde los objetos reaccionan a los valores que reciben de dicho cambio. Las librerías más conocidas son Project Reactor, y RxJava. React/Angular usan RxJs para hacer uso de la programación reactiva.

Distintos lenguajes de programación en la historia

Los primeros lenguajes de programación son anteriores a la creación de la computadora moderna, eran códigos que estaban especializados en su aplicación. Entre éstos podemos nombrar:

- Máquina del telar de Jacquard: Creada en 1801, usaba los orificios de tarjetas perforadas para representar los movimientos de un brazo de la máquina de tejer. Estas "tarjetas programables" eran metidas en la máquina telar que se encargaba de leerlas como un código que indicaba las instrucciones que debía realizar de forma automática. El objetivo era crear patrones decorativos.
- En 1843 Ada Lovelace mientras traducía las memorias del matemático Luigi Menabrea acerca de la maquina analítica propuesta por Charles Babbage, creó un lenguaje de programación para calcular la serie de números de Bernoulli con esa máquina.
- En 1936 Allan Turing revolucionó la computación con su Máquina de Turing. Se trata de una máquina que demostraba que, con un algoritmo, podían resolver cualquier problema matemático. Si había un algoritmo, la máquina podía resolver el problema.

Lenguajes de programación modernos – 1957 a fines de los 60

Durante ésta etapa aparecen los primeros lenguajes de programación modernos

- En 1957 IBM crea el primer lenguaje de programación moderno, FORTRAN.
  FORTRAN es un lenguaje de programación de alto nivel de procedimental, que está especialmente adaptado al cálculo numérico y a la computación científica.
- En 1958 se crea LISP como una notación matemática práctica para los programas de computadora, basada en el cálculo Lambda y se convirtió rápidamente en uno de los lenguajes favoritos para el campo de la inteligencia artificial.
- En 1959 llegó COBOL que fue diseñado con el objetivo de poder ser usado en cualquier ordenador y que estuviera orientado principalmente a los negocios, es decir, a la llamada informática de gestión.
- En 1964 aparece la familia de lenguaje de programación Basic con su primera versión "Dartmouth BASIC", como un medio para facilitar la programación en ordenadores a estudiantes (y profesores) que no fueran de ciencias. BASIC originalmente fue desarrollado como una herramienta de enseñanza, pero ésto no impidió que se hiciera muy popular.

Lenguajes de programación modernos – fines de los 60 y década del 70

En ésta época se desarrollan la mayoría de los paradigmas más importantes. Cada uno de estos lenguajes generó toda una familia de descendientes, y los lenguajes más modernos cuentan al menos uno de ellos en su ascendencia.

- En 1967 fue lanzado Simula, un lenguaje orientado a objetos que introdujo el concepto de clase. Varios años después de su desarrollo, casi todos los lenguajes modernos comenzaron a utilizar sus principios de orientación a objetos.

- En 1970 se crea Pascal, al igual que Basic, con la finalidad de servir como una herramienta de enseñanza para la programación. Hoy en día su presencia en aplicaciones es escasa, pero se sigue utilizando en escuelas e instituciones para acercar a los alumnos a la programación.
- En 1972 llegan Smalltalk (lenguaje orientado a objetos), C (lenguaje estructurado) y Prolog (lenguaje de programación lógica)
- En 1973 aparece ML (lenguaje funcional)
- En 1979 sale C++ basado en C que añade la programación orientada a objetos. C++ es uno de los lenguajes de programación más utilizados

Lenguajes de programación modernos – década del 80

En la década de 1980 en vez de inventar nuevos paradigmas se empezó a trabajar a partir de las ideas de la década anterior. Algunos lenguajes de ésta época son: MATLAB, Perl

Lenguajes de programación modernos – década del 90

Tras años experimentando con compiladores y lenguajes de programación orientados a objetos, llegó la década de Internet. Aquí se produjo un gran crecimiento de lenguajes de programación, y se maduraron ideas del pasado.

- En 1991 surgieron Python, Visual Basic y HTML, dos lenguajes que han definido las páginas web y HTML, un lenguaje de marcado de hipertexto que es el ''código'' de Internet.
- En 1995 nacieron Java, JavaScript y PHP. Aunque se ha intentado dejar de lado a favor de nuevos lenguajes, tanto Java como JavaScript siguen siendo muy utilizados.

Lenguajes de programación modernos – desde el año 2000

- En el año 2001 llegaron C# y Visual Basic .NET
- En 2009 llegó Go, desarrollado por Google, está inspirado en C, pero es un lenguaje bastante más complicado.
- En 2014 sale Swift, lenguaje de programación multiparadigma creado por Apple enfocado en el desarrollo de aplicaciones para iOS y macOS

Índice TIOBE

El índice TIOBE es un indicador de la popularidad de los lenguajes de programación. El índice se actualiza una vez al mes. Las puntaciones se basan en estadísticas no reveladas que incluyen el número de ingenieros en todo el mundo, cursos y aplicaciones desarrolladas. También se utilizan resultados obtenidos en los motores de búsqueda más usados.

https://www.tiobe.com/tiobe-index/