ALGORITMOS Y PROGRAMACIÓN

Julio Cesar Rodríguez Casas

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

Algoritmos y Programación
Julio Cesar Rodríguez Casas
Bogotá D.C.

Fundación Universitaria del Área Andina. 2018

Catalogación en la fuente Fundación Universitaria del Área Andina (Bogotá).

Algoritmos y Programación

ISBN (impreso): **978-958-5462-95-3**

Fundación Universitaria del Área Andina
Calle 70 No. 12-55, Bogotá, Colombia
Tel: +57 (1) 7424218 Ext. 1231
Correo electrónico: publicaciones@areandina.edu.co

Director editorial: Eduardo Mora Bejarano
Coordinador editorial: Camilo Andrés Cuéllar Mejía
Corrección de estilo y diagramación: Dirección Nacional de Operaciones Virtuales
Conversión de módulos virtuales: Katherine Medina

BANDERA INSTITUCIONAL

Pablo Oliveros Marmolejo †
Gustavo Eastman Vélez
Miembros Fundadores

Diego Molano Vega
Presidente del Consejo Superior y Asamblea General

José Leonardo Valencia Molano
Rector Nacional
Representante Legal

Martha Patricia Castellanos Saavedra
Vicerrectora Nacional Académica

Jorge Andrés Rubio Peña
Vicerrector Nacional de Crecimiento y Desarrollo

Tatiana Guzmán Granados
Vicerrectora Nacional de Experiencia Areandina

Edgar Orlando Cote Rojas
Rector – Seccional Pereira

Gelca Patricia Gutiérrez Barranco
Rectora – Sede Valledupar

María Angélica Pacheco Chica
Secretaria General

Eduardo Mora Bejarano
Director Nacional de Investigación

Camilo Andrés Cuéllar Mejía
Subdirector Nacional de Publicaciones

ALGORITMOS Y PROGRAMACIÓN

Julio Cesar Rodríguez Casas

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

ÍNDICE

ALGORITMOS Y PROGRAMACIÓN

Julio César Rodríguez Casas

EJE 1

Conceptualicemos

AREANDINA
Fundación Universitaria del Área Andina
MIEMBRO DE LA RED ILUMNO

INTRODUCCIÓN

En la actualidad las computadoras, se utilizan ampliamente en muchas áreas de los negocios, la industria, la ciencia, el hogar y la educación.

Las computadoras se han desarrollado y mejorado, según las necesidades del hombre, para realizar trabajos y cálculos más rápidos y precisos.

En este referente de pensamiento se pretende mostrar al estudiante cómo, a partir de unas especificaciones de un problema del mundo real, se diseña una solución para dicho problema (algoritmo) susceptible de ser codificada en un lenguaje de programación.

Con este objetivo en mente, se describirán las propiedades básicas de cualquier algoritmo, un conjunto de bloques básicos que permiten la construcción de algoritmos y diversas formas de representación de los algoritmos. En el tema también se mostrarán las distintas fases que se deben de seguir para buscar una solución a un problema del mundo real. En este tema no se aborda ningún lenguaje de programación particular; sino que se muestra cómo diseñar soluciones a problemas que sean fáciles de implementar en cualquier lenguaje de programación.

Antes de continuar, vamos a ver en la página principal del eje, la videocápsula 1 *Introducción a la computación* para contextualizar de manera general.

Evolución y arquitectura del computador

Para comenzar, veamos una línea de tiempo al respecto de la evolución y arquitectura del computador:

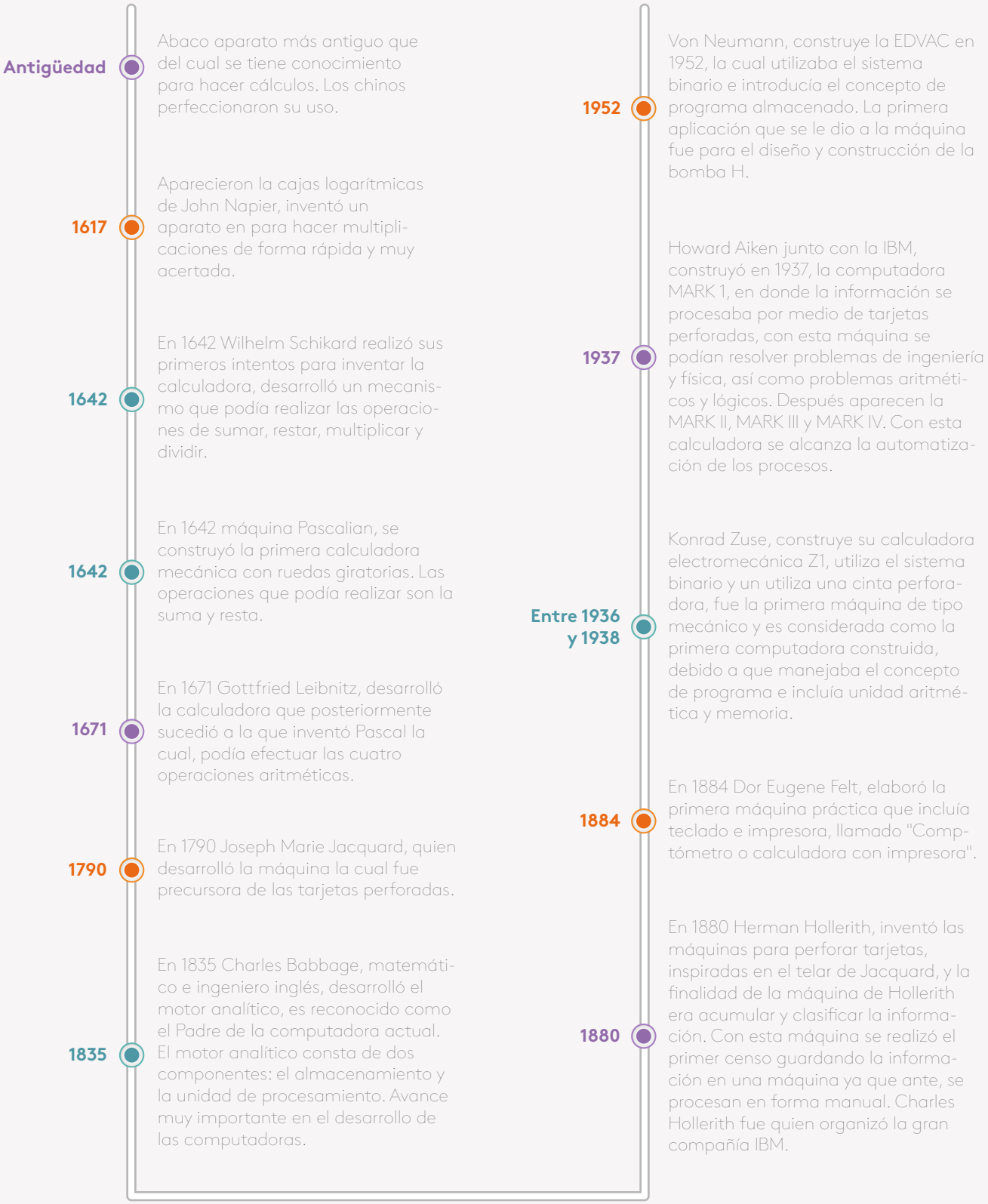

Figura 1. Línea de tiempo
Fuente: propia

Arquitectura del computador

Introducción a la arquitectura de computadores

La arquitectura de computadoras se refiere al diseño conceptual y la estructura operacional fundamental de un sistema que conforma una computadora, es decir, la disciplina dedicada a la construcción, estudio y aplicación de los computadores recibe el nombre de arquitectura de computadores y puede ser dividida en cinco partes fundamentales:

1. Entrada y salida.
2. Comunicaciones.
3. Control.
4. Procesamiento.
5. Almacenamiento.

Esta es de interés tanto para los ingenieros en electrónica y computación, dedicados al diseño de hardware, como para los científicos en computación e ingenieros de software, dedicados al diseño de programas.

Asimismo, la arquitectura de computadores es un concepto que integra software, hardware, algoritmos y lenguajes de programación para el procesamiento de datos y la generación de información.

Conceptos iniciales de la arquitectura de computadores

Un computador es un sistema secuencial síncrono complejo que procesa información, esta se trata de información binaria, utilizando solamente los dígitos de valores lógicos '1' y '0'. Estos valores lógicos binarios se corresponden con valores de tensión eléctrica, de manera que un '1' lógico corresponde a un nivel alto a 5 voltios y un '0' lógico corresponde a un nivel bajo de tensión cercano a 0 voltios; estos voltajes dependen de la tecnología que utilicen los dispositivos del computador.

Síncrono
Los circuitos secuenciales síncronos solo permiten un cambio de estado en los instantes marcados o autorizados por una señal de sincronismo de tipo oscilatorio denominada reloj.

Procesador

Figura 2. Procesador.
Fuente: http://bit.ly/2hBO4TS

Es el cerebro del sistema, encargado de procesar todos los datos e informaciones. A pesar de que es un dispositivo muy sofisticado no puede llegar a hacer nada por sí solo. Para hacer funcionar a este necesitamos algunos componentes más como lo son memorias, unidades de disco, dispositivos de entrada/salida y los programas. El procesador o núcleo central está formado por millones de transistores y componentes electrónicos de un tamaño microscópico. El procesamiento de las tareas o eventos que este realiza va en función de los nanosegundos, haciendo que los miles de transistores que contiene este trabajen en el orden de los MHz. La información binaria se introduce mediante dispositivos periféricos que sirven de interfaz entre el mundo exterior con el usuario. Estos periféricos lo que van a hacer será traducir la información que el usuario introduce en señales eléctricas, que serán interpretadas como unos y ceros, los cuales son interpretados de una manera más rápida por la computadora, ya que el lenguaje máquina utiliza el código binario para ser interpretado por el computador.

Arquitectura clásica de un computador modelo Von Neumann

La máquina Von Neumann, como todos los computadores modernos de uso general, consta de cuatro componentes principales:

1. **Dispositivo de operación (DO)**, que ejecuta instrucciones de un conjunto especificado, llamado sistema (conjunto) de instrucciones, sobre porciones de información almacenada, separada de la memoria del dispositivo operativo (aunque en la arquitectura moderna el dispositivo operativo consume más memoria -generalmente del banco de registros-), en la que los operandos son almacenados directamente en el proceso de cálculo, en un tiempo relativamente corto.

2. **Unidad de control (UC)**, que organiza la implementación consistente de algoritmos de decodificación de instrucciones que provienen de la memoria del dispositivo, responde a situaciones de emergencia y realiza funciones de dirección general de todos los nodos de computación. Por lo general, el DO y la UC conforman una estructura llamada CPU. Cabe señalar que el requisito es consistente, el orden de la memoria (el orden del cambio de dirección en el contador de programa) es fundamental a la hora de la ejecución de la instrucción. Por lo general, la arquitectura que no se adhiere a este principio no se considera von Neumann.

3. **Memoria del dispositivo,** un conjunto de celdas con identificadores únicos (direcciones), que contienen instrucciones y datos.

4. **Dispositivo de E/S (DES)**, que permite la comunicación con el mundo exterior de los computadores, son otros dispositivos que reciben los resultados y que le transmiten la información al computador para su procesamiento.

En el esquema de la figura 3, se muestra la estructura básica propuesta por Von Neumann que debe llevar una computadora para su correcta operación:

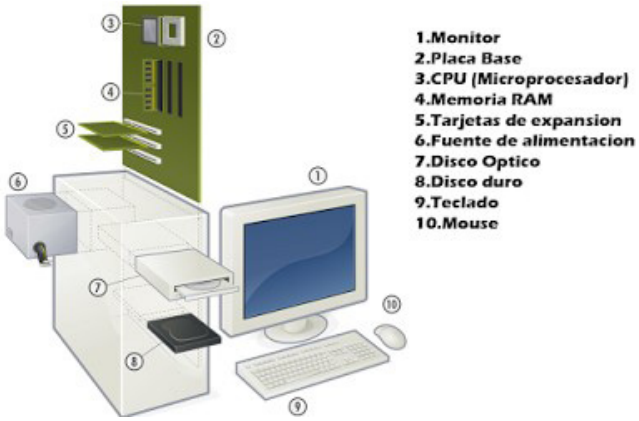

Figura 3. Estructura básica de una computadora.
Fuente: http://bit.ly/2kloZeY

La **unidad central de procesamiento** o **unidad de procesamiento central** (conocida por las siglas **CPU**, del inglés: Central Processing Unit), es el hardware dentro del computador u otros dispositivos programables, que interpreta las instrucciones de un programa mediante la realización de las operaciones básicas aritméticas, lógicas y de entrada/salida del sistema. El término, y su acrónimo, ha estado en uso en la industria de la Informática por lo menos desde el principio de los años 1960. La forma, el diseño de CPU y la implementación de las CPU ha cambiado drásticamente desde los primeros ejemplos, pero su operación fundamental sigue siendo la misma.

Memoria: es la responsable del almacenamiento de datos.

Entrada/Salida: transfiere datos entre el entorno exterior y el computador. En él se encuentran los controladores de periféricos que forman la interfaz entre los periféricos, la memoria y el procesador.

Sistema de interconexión: buses; es el mecanismo que permite el flujo de datos entre la CPU, la memoria y los módulos de entrada/salida. Aquí se propagan las señales eléctricas que son interpretadas como unos y ceros lógicos.

Periféricos: estos dispositivos son los que permiten la entrada de datos al computador, y la salida de información una vez procesada. Un grupo de periféricos puede entenderse como un conjunto de transductores entre la información física externa y la información binaria interpretable por el computador. Ejemplos de estos dispositivos son el teclado, el monitor, el ratón, el disco duro y las tarjetas de red.

Unidad central de procesamiento

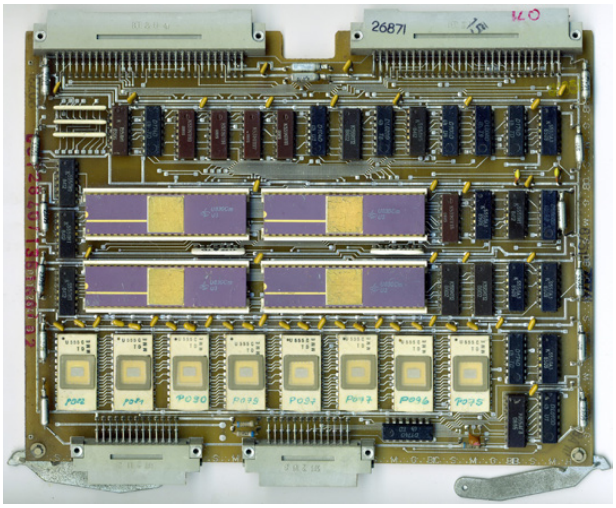

Figura 4. Unidad central de procesamiento.
Fuente: http://bit.ly/2gwSOKV

Controla el funcionamiento de los elementos de un computador. Desde que el sistema es alimentado por una corriente, este no deja de procesar información hasta que se corta dicha alimentación. La CPU es la parte más importante del procesador, debido a que es utilizado para realizar todas las operaciones y cálculos del computador.

Unidad de Control (UC): la unidad de control se encarga de leer de la memoria las instrucciones que debe de ejecutar y de secuenciar el acceso a los datos y operaciones a realizar por la unidad de proceso. La UC genera las señales de control que establecen el flujo de datos en todo el computador e interno en la CPU. Una instrucción no es más que una combinación de unos y ceros. Consta de un código de operaciones binarias para ejecutar la instrucción, la UC la almacena en un registro especial, interpreta su código de operación y ejecuta la secuencia de acciones adecuada, en pocas palabras decodifica la instrucción.

Operaciones binarias
Son aquellas en las cuales el resultado de verdadero o falso, 1 o 0.

Unidad Aritmética Lógica o ALU (por su sigla en inglés Arithmetic Logic Unit): es la parte de la CPU encargada de realizar las transformaciones de los datos. Gobernada por la UC, la ALU consta de una serie de módulos que realizan operaciones aritméticas y lógicas. La UC se encarga de seleccionar la operación a realizar habilitando los caminos de datos entre los diversos operadores de la ALU y entre los registros internos.

Registros internos: el almacenamiento de los resultados a la ejecución de las instrucciones en la memoria principal podría ser lento y excesivamente tendría muchos datos en el sistema de interconexión con la memoria, con lo que el rendimiento bajaría. De la misma manera, también se almacenan en registros internos la configuración interna del CPU o la información durante la última operación de la ALU. Los principales registros de una CPU son:

1. **Contador de programa:** se encarga de almacenar la dirección de la siguiente instrucción a ejecutar.

2. **Registro de instrucción:** se almacena la instrucción capturado en memoria y la que se está ejecutando.

3. **Registro de estado:** compuesto por una serie de bits que informan el resultado obtenido en la última operación de la ALU.

4. **Registro acumulador:** algunas CPU realizan operaciones aritméticas en un registro llamado acumulador, su función es la de almacenar los resultados de las operaciones aritméticas y lógicas.

Memoria

En la memoria se almacena el programa y los datos que va a ejecutar el CPU. Las instrucciones son códigos binarios interpretados por la unidad de control, los datos de igual manera se almacenan de forma binaria.

Las diversas tecnologías de almacenamiento dependen del tiempo de acceso a los datos; por lo tanto, se realiza un diseño jerárquico de la memoria del sistema para que esta pueda acceder rápidamente a los datos. El principio de que sea más rápida la memoria haciendo que tenga velocidades similares al CPU, sirve para diseñar el sistema de memoria. La memoria principal de los computadores tiene una estructura similar a la mostrada en el esquema de la Figura 4. Se considera como una matriz de celdas en la que la memoria puede acceder a los datos aleatoriamente.

Entrada/Salida

Como sabemos una computadora tiene dispositivos de entrada y salida como son los que contiene el gabinete, disco duro, placa madre, unidades de CD o DVD, etc. El problema principal que existe entre ellos es su tecnología y que tienen características diferentes a los del CPU, estos también necesitan una interfaz de cómo se van a entender con el CPU, al igual que el procesador y el controlador periférico para intercambiar datos entre la computadora.

Sistema de interconexión: buses

La conexión de los diversos componentes de una computadora, tales como discos duros, tarjetas madres, unidades de CD, teclados, ratones, etc. se efectúan a través de los buses. Un bus se define como un enlace de comunicación compartido que usa múltiples cables para conectar subsistemas. Cada línea es capaz de transmitir una tensión eléctrica que representa un '1' o un '0'. Cuando hay varios dispositivos en el mismo bus, habrá uno que podrá enviar una señal que será procesada por los demás módulos. Si se mandan los datos al mismo tiempo marcará un error o una contención del bus, por lo que el acceso estará denegado. Según su criterio de funcionalidad los buses se dividen en:

Buses de datos: es el que se utiliza para transmitir datos entre los diferentes dispositivos del computador.

Buses de direcciones: sirve para indicar la posición del dato que se requiere acceder.

Bus de control: sirven para seleccionar al emisor y al receptor en una transacción del bus.

Bus de alimentación: sirve para proporcionar a los dispositivos voltajes distintos.

Periféricos: se entenderán todos aquellos dispositivos que son necesarios para suministrar datos a la computadora o visualizar los resultados. Los periféricos se conectan mediante un bus especial a su controlador o al módulo de E/S.

Entre los periféricos de entrada tenemos al teclado, ratones, pantallas, digitalizadoras y más. Otros dispositivos periféricos fundamentales para la interacción del hombre con la computadora son las terminales de vídeo y las tarjetas gráficas.

A este punto, los invitamos a revisar la presentación de Jorge Chico sobre *Introducción a los computadores*.

¡Muy bien! Desarrollemos el crucigrama en la página principal del eje para afianzar los términos requeridos para la conceptualización de algoritmos.

Algoritmo, lenguaje y programa

Adicional a lo anterior, es importante tener presente tres conceptos:

Algoritmo: es una sucesión finita de pasos exento de ambigüedades que se pueden ejecutar en tiempo finito cuya razón de ser, es la resolución de problemas, cuya solución es expresable mediante un algoritmo.

Lenguaje: de programación es el responsable de que la computadora siga paso a paso las órdenes que el programador ha diseñado en el algoritmo. Con esto se entiende que el lenguaje de programación es una especie de intermediario entre el computador y el usuario, para que este último pueda darles respuesta a los problemas mediante la computadora y haciendo uso de palabras (funciones), que le interpretan dicho programa al computador para la realización de este trabajo.

Veamos en la página principal del eje, el video sobre *Lenguajes de programación* de Juan Francisco Díaz.

Programa: indicar a la computadora qué es lo que tiene que hacer. En otras palabras:

- Secuencia de instrucciones.
- Instrucciones que entiende la computadora.
- Y que persiguen un objetivo: ¡Resolver un problema!

Antes de continuar, vamos a consultar la nube de palabras en la página principal del eje para conceptualizar con algunos términos adicionales que será importante tener presentes.

Continuemos con el desglose de lenguajes.

Lenguajes de programación de alto nivel

Los lenguajes de programación de alto nivel se caracterizan por expresar los algoritmos de una manera adecuada a la capacidad de una persona para procesar información, en lugar de estar orientados a su ejecución en las máquinas.

Ventajas

- Genera un código más sencillo y comprensible.
- Escribir un código válido para diversas máquinas o sistemas operativos.
- Permite crear programas complejos en relativamente menos líneas de código.
- Más cercanos a los lenguajes natural y matemático.

resultado = dato1 + dato2;

- Mayor legibilidad, mayor facilidad de codificación
- Estructuración de datos / abstracción procedimental

Traducción
 	
Compiladores:
Compilan y enlazan
Programas completos
	Código fuente
	
	COMPILADOR
		#include <iostream>
		using namespace std;
		int main()
		{
		cout << "Hola Mundo!" << endl;
		return 0;
		}

Intérpretes: Compilan, enlazan y ejecutan instrucción a instrucción	Código objeto Enlazador	0100010100111010011100... Código objeto de biblioteca
-	Programa ejecutable	Para una arquitectura concreta y un sistema operativo

Tabla 1. Lenguajes de programación de alto nivel.
Fuente: propia

Genealogía de los lenguajes

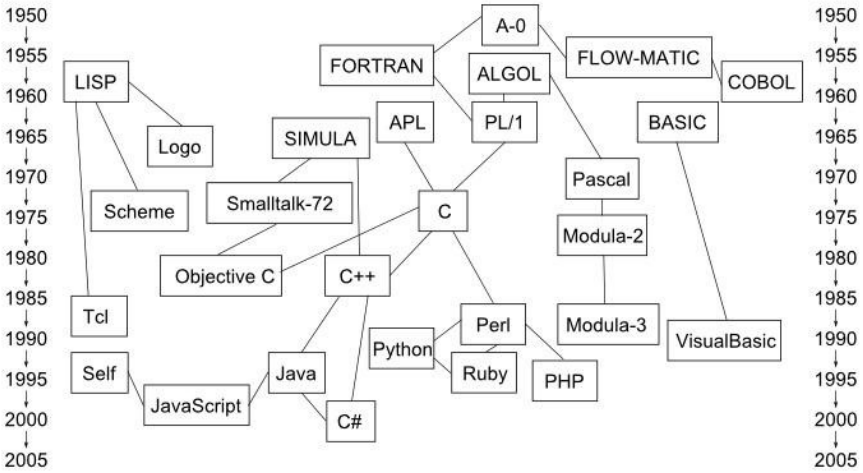

Figura 5. Genealogía de los lenguajes
Fuente: Sintes (2016).

Ahora examinaremos brevemente los grupos principales de lenguajes.

Lenguajes de programación imperativos y funcionales

Los lenguajes de programación generalmente se dividen en dos grupos principales con base al procesamiento de sus comandos: lenguajes imperativos y lenguajes funcionales.

Lenguaje de programación imperativo: un lenguaje imperativo programa mediante una serie de comandos agrupados en bloques y compuestos de órdenes condicionales, que permiten al programa retornar a un bloque de comandos si se cumple la condición. Estos fueron los primeros lenguajes de programación en uso y aún hoy muchos lenguajes modernos usan este principio.

No obstante, los lenguajes imperativos estructurados carecen de flexibilidad debido a la secuencialidad de las instrucciones.

Lenguaje de programación funcional: un lenguaje de programación funcional (a menudo llamado lenguaje procedimental) es un lenguaje que crea programas mediante funciones, devuelve un nuevo estado de resultado y recibe como entrada el resultado de otras funciones. Cuando una función se invoca a sí misma, hablamos de recursividad.

El programa: codificación del algoritmo en un lenguaje de programación. El código fuente es escrito en un lenguaje de programación.

```
1   ' Globales -----------
2   Var Variable0:Booleano
3   Var Variable1:Cadena
4   ' Fin Globales -------
5   Proc Procedimiento ' <- Procedimiento sin retorno.
6      Var Variable2:Entero ' Locales
7      Var Variable3:Real
8
9      Si Variable0 = Falso Entonces ' Condición "If"
10        Contar Variable2 = 0 a 9 ' Bucle "For"
11           Variable1 = Variable1 + "1"
12        Seguir ' "End For"
13     FinSi ' "End If"
14
15     Variable3 = 5.13
16  FinProc
```

Figura 6. Lenguaje de programación
Fuente: http://bit.ly/2yZTNKp

Instrucciones en una computadora

Una instrucción es cada paso de un algoritmo, pero que lo ejecuta la computadora.

Tipos de instrucciones

- Entrada y Salida. E/S: pasar información del exterior al interior del ordenador y al revés.
- Aritmético-lógicas: aritméticas: ^,/,mod,+,-,*; Lógicas: or, and, not. Comparación <, <=, >, >=, =, <>.
- Selectivas: permiten la selección de una alternativa en función de una condición.
- Repetitivas: repetición de un número de instrucciones un número finito de veces.

¡Recordemos que!

Un programa es un conjunto de instrucciones que ejecutadas ordenadamente resuelven un problema.

Tipos de lenguajes

Lenguaje máquina: todo se programa con 1 y 0, que es lo único que entiende el ordenador.

A continuación, encontrará una breve lista de los lenguajes de programación actuales:

Lenguaje	Principal área de aplicación	Compilado/interpretado
ADA	Tiempo real	Lenguaje compilado
BASIC	Programación para fines educativos	Lenguaje interpretado
C	Programación de sistema	Lenguaje compilado
C++	Programación de sistema orientado a objeto	Lenguaje compilado
Cobol	Administración	Lenguaje compilado
Fortran	Cálculo	Lenguaje compilado
Java	Programación orientada a Internet	Lenguaje intermediario
MATLAB	Cálculos matemáticos	Lenguaje interpretado
Cálculos matemáticos	Cálculos matemáticos	Lenguaje interpretado
LISP	Inteligencia artificial	Lenguaje intermediario
Pascal	Educación	Lenguaje compilado
PHP	Desarrollo de sitios web dinámicos	Lenguaje interpretado
Inteligencia artificial	Inteligencia artificial	Lenguaje interpretado
Perl	Procesamiento de cadenas de caracteres	Lenguaje interpretado

Tabla 2. Lenguajes de programación usados en la actualidad
Fuente: propia

Compilación del lenguaje

En un lenguaje compilado el código fuente antes de ser ejecutado es convertido a lenguaje máquina (C, C++) aunque también puede ser convertido a representación intermedia que posteriormente es interpretada y convertida a lenguaje máquina JIT (Java, C#). El compilador puede detectar una gran cantidad de errores que en un lenguaje interpretado o de tipado dinámico se descubrirán en tiempo de ejecución.

Lectura recomendada

Se invita al estudiante a realizar la lectura recomendada de *algoritmos*.

Algoritmos

Les invitamos a revisar las instrucciones de la actividad de repaso 1 e ir contrastando con el conocimiento sobre algoritmos que desarrollaremos a continuación.

Para empezar, veamos el video.

Video

¿Qué es un algoritmo?

Un algoritmo es una secuencia de pasos organizados entre sí que describe el proceso que se debe seguir, sin ambigüedades para dar solución a un problema específico.

Algoritmo

La palabra algoritmo se deriva de la traducción al latín de la palabra árabe alkhowarizmi, nombre de un matemático y astrónomo árabe que escribió un tratado sobre manipulación de números y ecuaciones en el siglo IX, Mohámed ben Musa, que alcanzó gran reputación al enunciar paso a paso las reglas para sumar, restar, multiplicar y dividir números con decimales.

La resolución de un problema mediante una computadora consiste en la especificación del problema, construir un programa que lo resuelva.

Los procesos necesarios para la creación de un programa son:

Figura 7. Proceso de creación de un programa
Fuente: propia

Algunos ejemplos de algoritmos son:

- Una receta de cocina.
- Un manual de uso.
- Las instrucciones para realizar un trámite.

Ahora bien, las características fundamentales que debe tener todo algoritmo son:

- Debe ser preciso, es decir, indicar el orden de realización de cada paso.
- Debe estar definido, esto es, si se ejecuta varias veces partiendo de las mismas condiciones iniciales debe obtenerse siempre el mismo resultado.
- Debe ser finito (debe tener un número finito de pasos).
- Debe ser independiente del lenguaje de programación que se emplee para implementarlo.

Metodología y construcción de algoritmos

En cualquier algoritmo se pueden distinguir tres partes:

1. La entrada de datos (la información sobre la cual se va a efectuar operaciones).
2. Procesamiento.
3. Salida del resultado (la información que debe proporcionar).

Veamos un ejemplo clásico de algoritmos: determinar el volumen de una caja de dimensiones A, B y C.

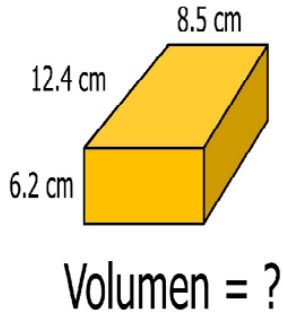

Figura 8. Volumen de una caja
Fuente: propia

Se puede establecer de la siguiente manera:

1. Inicio.
2. Leer las medidas A, B y C.
3. Realizar el producto de A * B * C y guardarlo en V. (V = A * B * C).
4. Escribir el resultado V.
5. Fin.

Como se puede apreciar, se establece de forma precisa la secuencia de los pasos por realizar; además, si se les proporciona siempre los mismos valores a las variables A, B y C, el resultado del volumen será el mismo y, por consiguiente, se cuenta con un final.

Diagramas de flujo

Los diagramas de flujo son una herramienta que permite representar visualmente qué operaciones se requieren y en qué secuencia se deben efectuar para solucionar un problema dado.

Diagramas de flujo
Un diagrama de flujo es la representación gráfica mediante símbolos especiales, de los pasos o procedimientos de manera secuencial y lógica que se deben realizar para solucionar un problema dado.

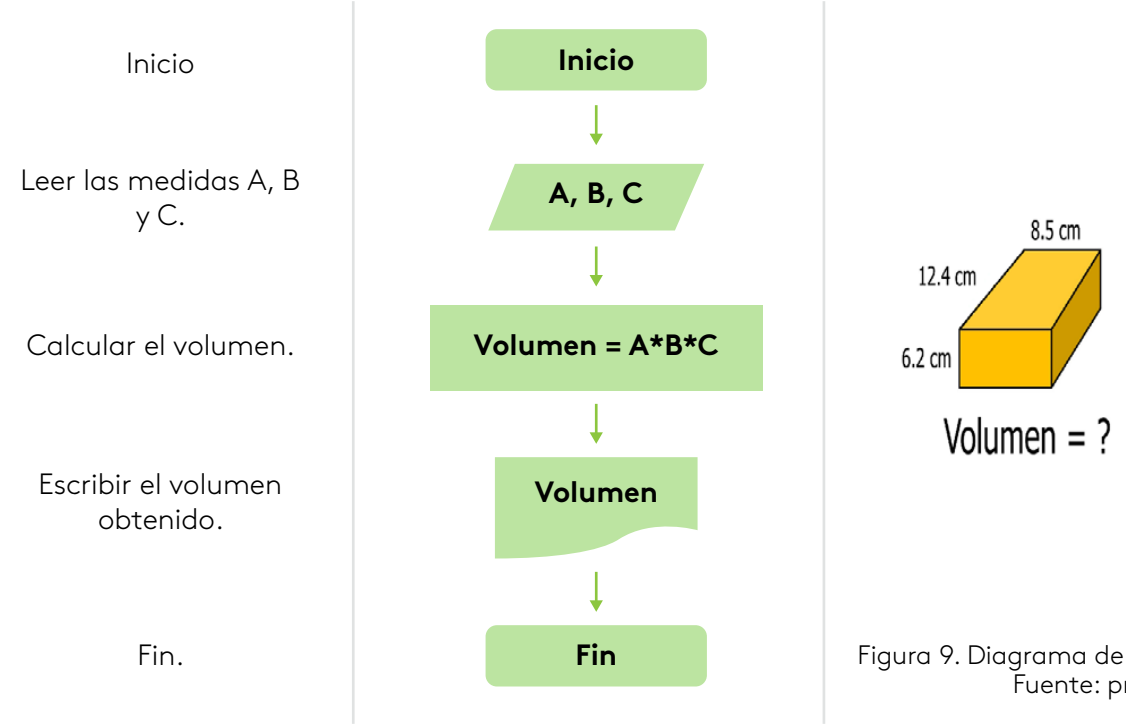

Figura 9. Diagrama de flujo
Fuente: propia

En principio, el flujo de ejecución de un programa será el orden en el cual se escriban las instrucciones (ejecución secuencial). Sin embargo, este esquema de ejecución impone serias limitaciones: a menudo hay ciertas porciones de código que sólo se deben ejecutar si se cumplen ciertas condiciones y tareas repetitivas que hay que ejecutar un gran número de veces. Por ello, se han definido una serie de estructuras de programación, independientes del lenguaje concreto que se está utilizando, que permiten crear programas cuyo flujo de ejecución sea más versátil que una ejecución secuencial.

Variables y tipos de datos

Veamos a continuación, los elementos que hacen parte de un algoritmo de programación.

Variables

Una variable es un objeto cuyo valor puede cambiar durante el desarrollo del algoritmo. Se identifica por su nombre y por su tipo, que podrá ser cualquiera, y es el que determina el conjunto de valores que podrá tomar la variable. En los algoritmos se pueden declarar variables. Cuando se traduce el algoritmo a un lenguaje de programación y se ejecuta el programa resultante, la declaración de cada una de las variables originará que se reserve un determinado espacio de memoria etiquetado con el correspondiente identificador.

Tipos de datos

Un tipo de datos es la propiedad de un valor que determina su dominio (qué valores puede tomar), qué operaciones se le pueden aplicar y cómo es representado internamente por el computador.

Todos los valores que aparecen en un programa tienen un tipo.

A continuación, revisaremos los tipos de datos elementales de Python. Además de éstos, existen muchos otros, y más adelante aprenderemos a crear nuestros propios tipos de datos.

Números enteros

El tipo int (del inglés *integer, que significa «entero») permite representar números enteros.*

Los valores que puede tomar un int son todos los números enteros: ... -3, -2, -1, 0, 1, 2, 3, ...

```
1570
+4591
-12
```

Los números enteros literales se escriben con un signo opcional seguido por una secuencia de dígitos:

Números reales

El tipo float permite representar números reales.

El nombre float viene del término punto flotante, que es la manera en que el computador representa internamente los números reales.

Hay que tener mucho cuidado, porque los números reales no se pueden representar de manera exacta en un computador. Por ejemplo, el número decimal 0.7 es representado internamente por el computador mediante la aproximación 0.69999999999999996. Todas las operaciones entre valores float son aproximaciones. Esto puede conducir a resultados algo sorpresivos:

```
>>> 1/7 + 1/7 + 1/7 + 1/7 + 1/7 + 1/7 + 1/7
0.9999999999999998
```

Los números reales literales se escriben separando la parte entera de la decimal con un punto. Las partes entera y decimal pueden ser omitidas si alguna de ellas es cero:

```
>>> 881.9843000
881.9843
>>> -3.14159
-3.14159
>>> 1024.
1024.0
```

Otra representación es la notación científica, en la que se escribe un factor y una potencia de diez separados por una letra e. Por ejemplo:

```
>>> -2.45E4
-24500.0
>>> 7e-2
0.07
```

Los dos últimos valores del ejemplo son iguales, respectivamente, a 6.02×10236.02 ×1023 y 9.1094×10−319.1094×10−31.

Texto

A los valores que representan texto se les llama strings, y tienen el tipo str. Los strings literales pueden ser representados con texto entre comillas simples o comillas dobles:

```
"ejemplo 1"
'ejemplo 2'
```

La ventaja de tener dos tipos de comillas es que se puede usar uno de ellos cuando el otro aparece como parte del texto:

```
"ALGORITMOS Y PROGRAMACIÓN"
'Ella dijo "hola"'
```

Es importante entender que los strings no son lo mismo que los valores que en él pueden estar representados:

```
>>> 5 == '5'
False
>>> True == 'True'
False
```

Los strings que difieren en mayúsculas y minúsculas, o en espacios también son distintos:

```
>>> 'mesa' == 'Mesa'
False
>>> ' mesa' == 'mesa '
False
```

Nulo

Existe un valor llamado None (en inglés, «ninguno») que es utilizado para representar casos en que ningún valor es válido, o para indicar que una variable todavía no tiene un valor que tenga sentido.

El valor None tiene su propio tipo, llamado NoneType, que es diferente al de todos los demás valores.

Revise en la página principal del eje el video.

Video

Cómo se realiza un algoritmo.

Constantes

Son datos cuyo valor no cambia durante todo el desarrollo del algoritmo. Las constantes podrán ser literales o con nombres.

Ahora bien, ya que identificamos los tipos de datos que se utilizan en el diseño de un algoritmo, a continuación, vamos a ver aquellas estructuras de tipo repetitivo que son vitales para aquellos problemas típicos que requieren ejecutar una instrucción a partir de una determinada condición que se debe cumplir. Veamos:

La estructura de control if

La sentencia if permite a un programa tomar una decisión para ejecutar una acción u otra. Diariamente tomamos decisiones, por ejemplo, evaluamos el estado del tiempo para decidir si llevamos sombrilla, verificamos si la tarjeta de Transmilenio tiene cargado dinero para viajar, sino hay que cargarle dinero.

A La sentencia if se le conoce como estructura de selección simple y su función es realizar o no una determinada acción o sentencia, basándose en el resultado de la evaluación de una expresión (verdadero o falso), en caso de ser verdadero se ejecuta la sentencia.

```
if ( expresión(es ) )
      { sen tencias  }
else
      { sentencias }
```

Por ejemplo**,** si dada la edad de una persona quiero dar un mensaje de que es o no mayor de edad, suponiendo que una persona mayor de edad tiene por lo menos 21 años, el procedimiento será el siguiente.

En pseudocódigo el anterior ejemplo se vería de la siguiente forma:

```
Inicio
Mostrar "¿Qué edad tienes?"

Leer edad

Si Edad >20 Entonces
Mostrar "Eres mayor de Edad"
Fin Si
Fin
```

La sentencia for es una sentencia que implementa un bucle, es decir, que es capaz de repetir un grupo de sentencias un número determinado de veces.

La sintaxis de un ciclo for es simple en los lenguajes de alto nivel es incluso muy similar, de hecho, con tan solo tener bien claros los 3

componentes del ciclo **for** (inicio, final y tamaño de paso)

```
for (int i = valor inicial; i <= valor final; i = i + paso)
{
        ....
        ....
Bloque de Instrucciones....
        ....
        ....
}
```

Pseudocódigo
El pseudocódigo consta de un falso lenguaje. Se considera que, es un código informal que se utiliza como inicio de un algoritmo.

Sentencia While

Veamos la definición de esta sentencia, tomada de Ceballos (2007).

La sentencia while ejecuta una secuencia, simple o compuesta, cero o más veces, dependiendo de una condición:

Su sintaxis es:

```
while (condición)
        Bloque de instrucciones;
```

Si se requiere realizar más de un estatuto se deben utilizar llaves.

```
while ( condición )
    {
        Bloque de Instrucciones;
    }
```

Donde condición es cualquier expresión numérica, relacional o lógica y bloque de instrucciones es una sentencia simple o compuesta.

La ejecución de la sentencia while funciona de la siguiente forma:

1. Se evalúa la condición. El resultado es falso o verdadero.
2. Si el resultado de la evaluación es falso (0), no se ejecuta y se pasa el control a la siguiente sentencia en el programa.
3. Si el resultado de la evaluación es verdadero (1), se ejecuta la sentencia y el proceso descrito se repite desde el punto 1.
4. Un ciclo while tiene una condición del ciclo, una expresión lógica que controla la secuencia de repetición. (Ceballos, 2007).

Implementación de algoritmos

Diagramas de flujo (DFD)

¿Qué es DFD?: los diagramas de flujo de datos (DFD en inglés) son un tipo de herramienta de modelado, permiten modelar todo tipo de sistemas, concentrándose en las funciones que realiza, y los datos de entrada y salida de esas funciones.

Componentes de los DFD

PROCESOS (burbujas): representan la parte del sistema que transforma ciertas entradas en ciertas salidas.

FLUJOS: representan los datos en movimiento. Pueden ser flujos de entrada o flujos de salida. Los flujos conectan procesos entre sí y también almacenes con procesos.

ALMACENES: representan datos almacenados. Pueden ser una base de datos, un archivo físico.

TERMINADORES: representan entidades externas que se comunican con el sistema. Esas entidades pueden ser personas, organizaciones u otros sistemas, pero no pertenecen al sistema que se está modelando.

Flujos de control

Los flujos de control son señales o interrupciones, en tanto los procesos de control son burbujas que coordinan y sincronizan otros procesos; los procesos de control solo se conectan con flujos de control.
Los flujos de control de salida "despiertan" otras burbujas, en tanto los flujos de control de entrada, especifican que una tarea terminó o se presentó un evento extraordinario.

¡Importante!

Existen procesos y flujos especiales llamados procesos de control y flujos de control. Se emplean para modelar sistemas en tiempo real.

Representación de un sistema en DFD

Un sistema puede representarse empleando varios diagramas de flujos de datos, cada flujo de datos puede representar una parte "más pequeña" del sistema.

Los DFD permiten una partición por niveles del sistema. El nivel más general se representa con un DFD global llamado diagrama de contexto.

El diagrama de contexto DFD representa a todo el sistema con una simple burbuja o proceso, las entradas y salidas de todo el sistema, y las interacciones con los terminadores.

Complementos del DFD

Los DFD suelen servir para comprender fácilmente el funcionamiento de un sistema. De todas maneras, no es la única herramienta para diagramar sistemas, es más, se debe complementar con otras herramientas para agregar comprensión y exactitud al DFD.

Para la implementación de algoritmos es ideal instalar en el computador programas como PseInt.

¿Qué es PseInt?: es una herramienta para asistir a los estudiantes en sus primeros pasos en programación. Mediante un simple e intuitivo pseudolenguaje en español (complementado con un editor de diagramas de flujo), le permite centrar su atención en los conceptos fundamentales de la algoritmia computacional, minimizando las dificultades propias de un lenguaje y proporcionando un entorno de trabajo con numerosas ayudas y recursos didácticos.

Se puede descargar de forma libre ingresando a Google:

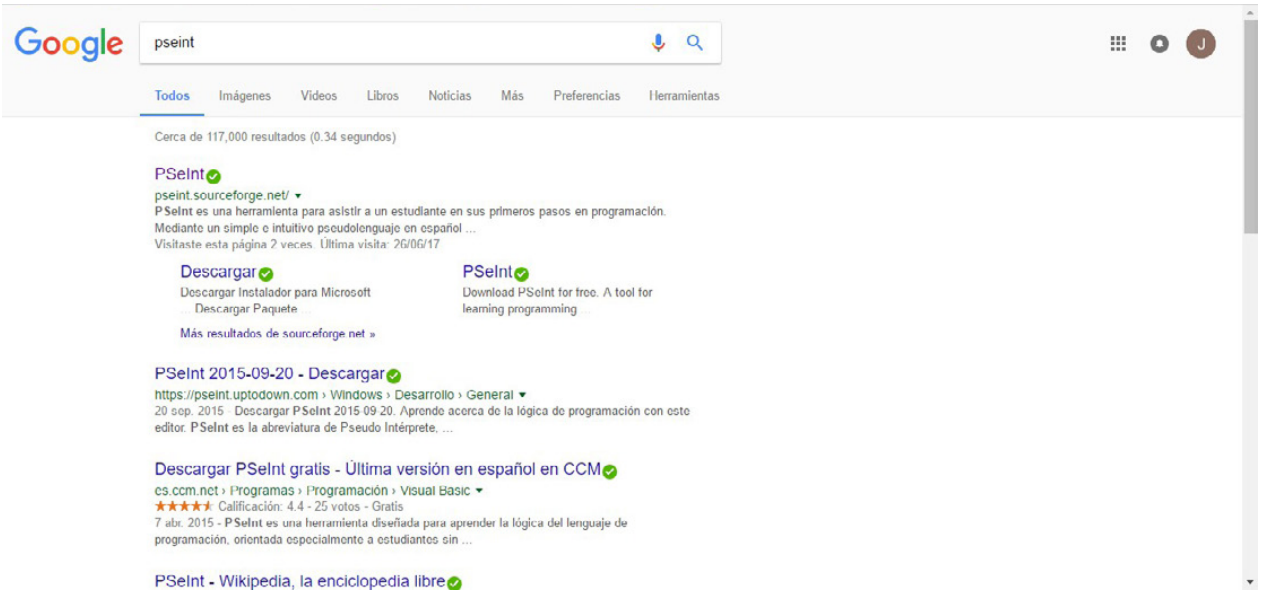

Figura 10. Búsqueda en google de descarga de PseInt
Fuente: captura de pantalla "Búsqueda de Google"

¡Importante!

Se invita al estudiante a revisar el manual PseInt del Ing. Ronald Rentería, con el fin de consolidar el concepto y desarrollo de los algoritmos. Además, será importante que se realice su instalación para diseñar los respectivos diagramas de flujo que se proponen en el módulo.

Introducción a software para interpretación e implementación (C++)

Si se desea escribir un programa en C++ se debe ejecutar como mínimo los siguientes pasos:

1. Escribir con un editor de texto plano un programa sintácticamente válido o usar un entorno de desarrollo (IDE) apropiado para tal fin.
2. Compilar el programa y asegurarse de que no ha habido errores de compilación.
3. Ejecutar el programa y comprobar que no hay errores de ejecución.

¡Importante!

Este último paso es el más costoso, porque en programas grandes, averiguar si hay o no un fallo prácticamente puede ser una tarea titánica.

Como ejemplo, si se desea escribir un archivo con el nombre hola.cpp y en él escribir un programa con emacs, por ejemplo, que es un programa de edición de textos, se puede, en GNU, ejecutar el siguiente comando:

```
$emacs hola.cpp &
```

Para otros sistemas operativos u otros entornos de desarrollo, no necesariamente se sigue este paso.

```
#include <iostream>

int main(void)

{

std::cout << "Hola Mundo" << std::endl;

std::cin.get();

//con 'std::cin.get();' lo que se hace es esperar hasta que el usuario pulse enter.

return 0;

}
```

En este punto se ha terminado este eje, por lo que es el momento de realizar los ejercicios propuestos en Actividad de repaso 2.

BIBLIOGRAFÍA

Ceballos, F. (2007). *Programación orientada a objetos con C++* (4a. ed.). Madrid, España: Editorial RA-MA. Recuperado de http://site.ebrary.com.proxy.bidig.areandina.edu.co:2048/lib/bibliotecafuaasp/detail.action?docID=11045993&p00=lenguaje+c%2B%2B

Ceballos, F. (2009). *Enciclopedia del lenguaje C++* (2a. ed.). Madrid, España: Editorial RA-MA Recuperado de: http://site.ebrary.com.proxy.bidig.areandina.edu.co:2048/lib/bibliotecafuaasp/detail.action?docID=11046572&p00=lenguaje+c%2B%2B

Gaxiola, C., y Flores, D. (2008). *Metodología de la programación con pseudocódigo enfocado al lenguaje C*. Ciudad de México, México: Editorial Plaza y Valdés, S.A. de C.V. Recuperado de http://site.ebrary.com.proxy.bidig.areandina.edu.co:2048/lib/bibliotecafuaasp/detail.action?docID=10877093&p00=algoritmos+programaci%C3%B3n

Joyanes, L. (2006). *Programación en C++: algoritmos, estructuras de datos y objetos* (2a. ed.) Madrid, España: McGraw-Hill. Recuperado de: http://site.ebrary.com.proxy.bidig.areandina.edu.co:2048/lib/bibliotecafuaasp/detail.action?docID=10491359&p00=lenguaje+c%2B%2B

Joyanes, L., Castillo, A., y Sánchez, L. (2005). *C algoritmos, programación y estructuras de datos*. Madrid, España: McGraw-Hill. Recuperado de http://site.ebrary.com.proxy.bidig.areandina.edu.co:2048/lib/bibliotecafuaasp/detail.action?docID=10491350&p00=lenguaje+c%2B%2B

Joyanes, L., Rodríguez, L., y Fernández, M. (2003). *Fundamentos de programación: libro de problemas. Algoritmos, estructuras de datos y objetos* (2a. ed.) Madrid, España: McGraw-Hill España Recuperado de http://site.ebrary.com.proxy.bidig.areandina.edu.co:2048/lib/bibliotecafuaasp/detail.action?docID=10498607&p00=algoritmos+programaci%C3%B3n

Joyanes, L., y Sánchez, L. (2006). *Programación en C++: un enfoque práctico*. Madrid, España: McGraw-Hill. Recuperado de http://site.ebrary.com.proxy.bidig.areandina.edu.co:2048/lib/bibliotecafuaasp/detail.action?docID=10491298&p00=lenguaje+c%2B%2B

Joyanes, L., y Zahonero, I. (2007). *Estructura de datos en C++*. Madrid, España: McGraw-Hill. Recuperado de http://site.ebrary.com.proxy.bidig.areandina.edu.co:2048/lib/bibliotecafuaasp/detail.action?docID=10491301&p00=lenguaje+c%2B%2B

BIBLIOGRAFÍA

Joyanes, L., Zahonero, I. (2005). *Programación en C: metodología, algoritmos y estructura de datos* (2a. ed.). Madrid, España: McGraw-Hill Recuperado de http://site.ebrary.com.proxy.bidig.areandina.edu.co:2048/lib/bibliotecafuaasp/detail.action?docID=10498360&p00=algoritmos+programaci%C3%B3n

Juganaru, M. (2014). *Introducción a la programación*. Ciudad de México, México: Grupo Editorial Patria. Recuperado de http://site.ebrary.com.proxy.bidig.areandina.edu.co:2048/lib/bibliotecafuaasp/detail.action?docID=11017472&p00=programaci%C3%B3n

Moreno, J. (2014). *Programación*. Madrid, España: Editorial RA-MA. Recuperado de http://site.ebrary.com.proxy.bidig.areandina.edu.co:2048/lib/bibliotecafuaasp/detail.action?docID=11046398&p00=lenguaje+c%2B%2B

Noguera, F., y Riera, D. (2013). *Programación*. Barcelona, España: Editorial UOC. Recuperado de http://site.ebrary.com.proxy.bidig.areandina.edu.co:2048/lib/bibliotecafuaasp/detail.action?docID=10853420&p00=lenguaje+c%2B%2B

Schildt, H. (2009). *C++: soluciones de programación*. McGraw-Hill Interamericana. Recuperado de http://site.ebrary.com.proxy.bidig.areandina.edu.co:2048/lib/bibliotecafuaasp/detail.action?docID=10433927&p00=algoritmos+programaci%C3%B3n