UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

LICENCIATURAS A DISTANCIA

INFORMÁTICA

PROGRAMACIÓN

PROGRAMACIÓN		Clave: 1369
Plan: 2006		Créditos: 8
Licenciatura: Informática		Semestre: 3°
Área: Informática		Hrs. Asesoría: 4
Requisitos: Ninguno		Hrs. Por semana: 4
Tipo de asignatura:	Obligatoria (x)	Optativa ( )

INTRODUCCIÓN GENERAL A LA ASIGNATURA

Todas las computadoras necesitan ser programadas, es decir, almacenar en memoria la información sobre las tareas que van a ejecutar.

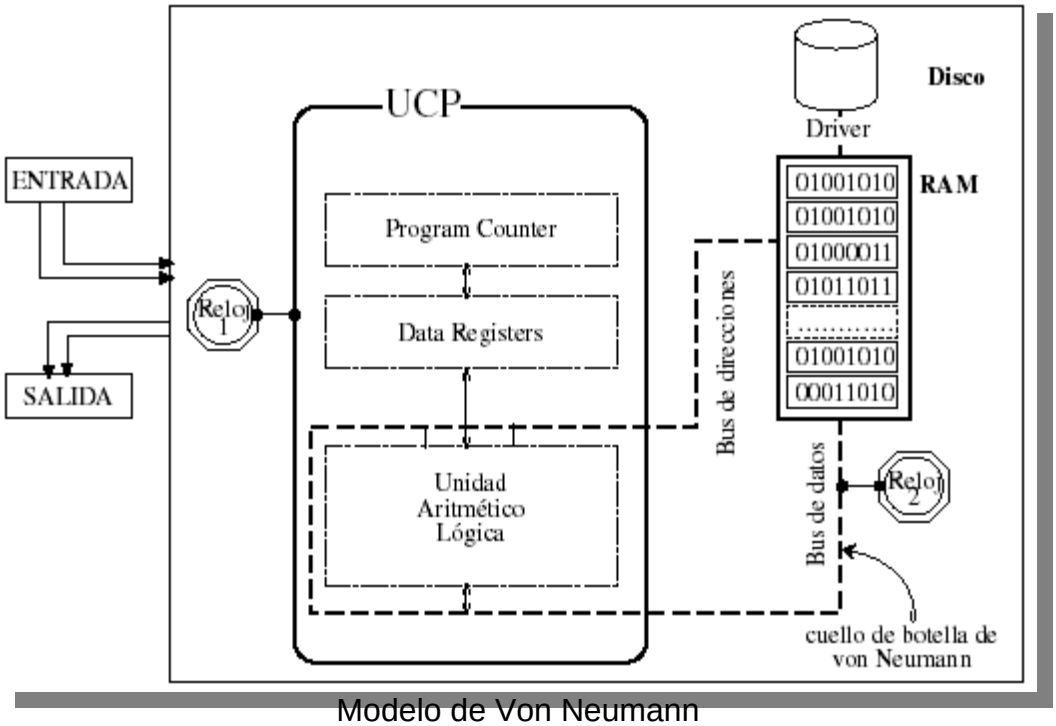
"Una de las definiciones comúnmente presentadas de lenguaje de programación es *notación para comunicarle a una computadora lo que deseamos que haga*. Esta definición es inadecuada, ya que antes de los años 40 las computadoras se programaban mediante cableado e interruptores"[1] y el cablear es todo menos una notación.

"En los 40 se dio un adelanto importante en el diseño de las computadoras, cuando John von Neumann (se pronuncia *"Noiman"*) tuvo la idea de que una computadora no debería estar "cableada" para ejecutar algo en particular, sino

[1] Kenneth C. Louden, *Programming Languages: Principles and Practice*, 2a ed., Brooks Cole, 2003.

que podría lograrse que una serie de códigos almacenados como datos determinarán las acciones ejecutadas por una unidad de procesamiento central"[2]

Desarrolló un modelo que lleva su nombre, para describir el concepto de *"programa almacenado en memoria"*.[3]



Modelo de Von Neumann

Carga horaria / Tiempo estimado de estudio: 68 Horas.

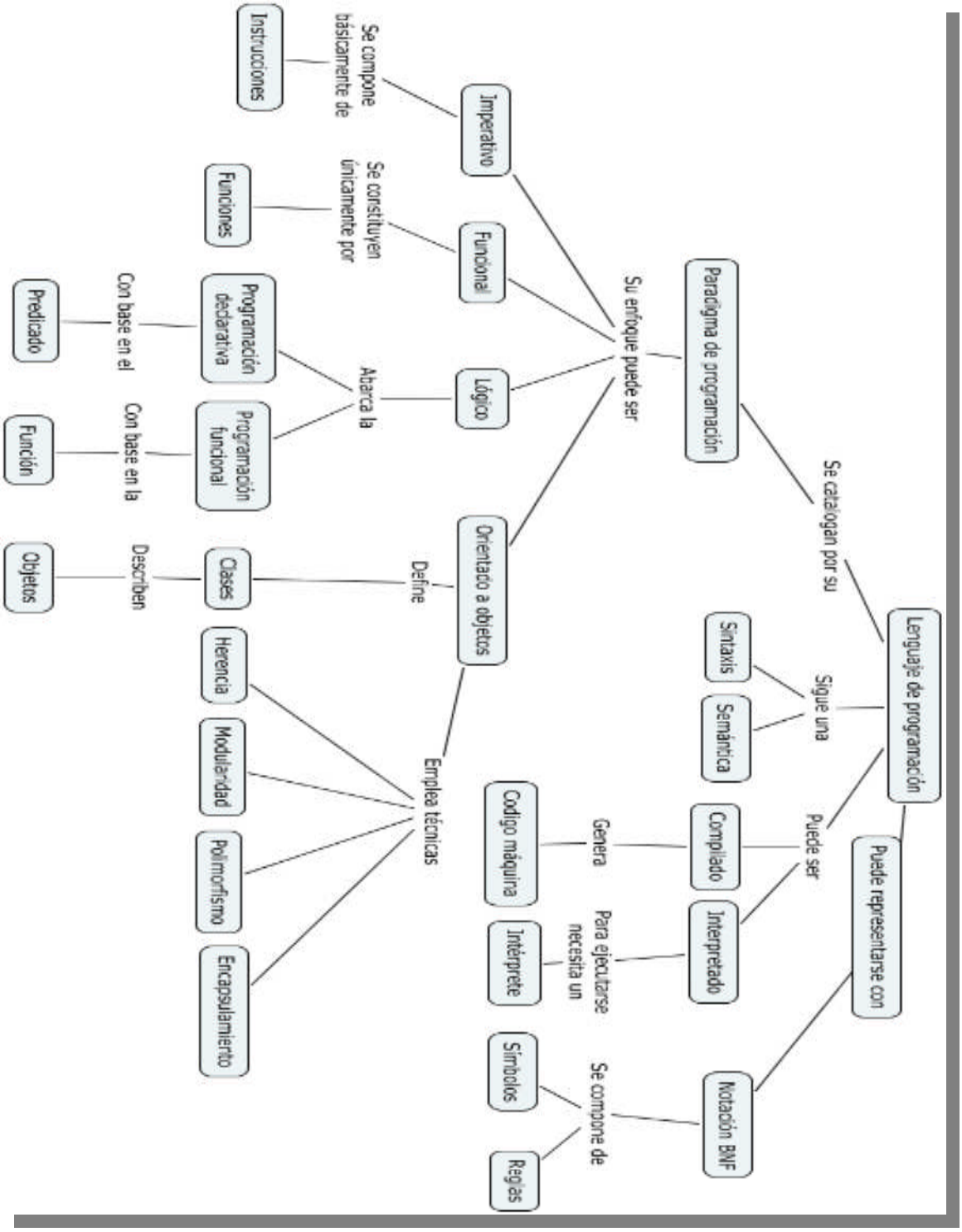
Objetivo general de la asignatura:

Al finalizar el curso el alumno conocerá la evolución de los lenguajes de programación, así como las diferentes filosofías (paradigmas) que emplean para describir modelos de la realidad.

[2] Kenneth C. Louden, *Programming Languages: Principles and Practice*, ed. cit.

[3] Fuente: http://informatica.utem.cl/~mcast/CCOMPUTACION/Introduccion/ModelosComputacionales.pdf [Consulta: 30 de enero de 2009]

ESTRUCTURA CONCEPTUAL



Temario oficial de la asignatura

Unidad 1. Definición de un lenguaje de programación

Unidad 2. Paradigma Imperativo

Unidad 3. Paradigma orientado a objetos

Unidad 4. Paradigma funcional

Unidad 5. Paradigma lógico

Introducción a la unidad

Un lenguaje de programación es el resultado de querer hacer que una computadora o dispositivo programable ejecute cosas que nosotros le indiquemos. Pero al tener su propio lenguaje (ceros y unos), inicialmente no era tan sencillo programar debido a que una sola operación de suma implicaba varias líneas de código máquina, lo que hacía lento el proceso de construcción de programas. La solución: crear programas a partir de abstracciones que nosotros aplicamos en la vida real y que se han ido implementando en los lenguajes posteriores al lenguaje máquina. Así surgió el lenguaje ensamblador como el primer intento de cambiar los 0s y 1s por nemotecnias como ADD, MOV, LOAD, etcétera, las cuales son más fáciles de utilizar. Ya en la tercer y cuarta generación las nemotecnias se transforman en estructuras muy semejantes al lenguaje natural, con una sintaxis y una semántica que nos permite trasladar nuestra lógica de resolver los problemas a la lógica de una computadora. Veamos a continuación cuáles son estas abstracciones y sus tipos, además de los dos elementos que nos permiten definir a un lenguaje de programación como tal: la sintaxis y la semántica.

Objetivo particular de la unidad

Definirá qué es un lenguaje de programación y podrá explicar las clasificaciones que se hacen de éste, sus niveles de abstracción, los paradigmas que los implementan, su evolución e historia, su definición formal (a través de la sintaxis y semántica) y el proceso de traducción.

LO QUE SÉ

Antes de comenzar el estudio de la unidad, te pido contestar las siguientes preguntas. No es necesario que investigues, contéstalas de acuerdo a lo que sepas:

- ¿Qué es la arquitectura Von Neumann?
- ¿Qué es un lenguaje de programación?
- ¿Cuán tipos de lenguajes conoces?
- ¿Para qué se emplean los lenguajes de programación?

Para enviar tu respuesta, pulsa el botón **Editar mi envío**; se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios**

Temas de la unidad I

Resumen de la unidad

En la presente Unidad se abordarán los conceptos básicos que rigen a los lenguajes de programación en general.

Partiendo de su definición formal, se presentará una breve historia de los lenguajes de programación resaltando tanto su evolución como su impacto en las técnicas de programación que se han dado a lo largo del tiempo. Así mismo, se presentarán los conceptos básicos sobre el manejo de datos y la estructura de ejecución

Se introducirán los conceptos sobre la generación de programas tanto compilados como interpretados, así como de las diferencias técnicas y conceptuales que dichos enfoques.

Tema 1. ¿Qué es un lenguaje de programación?

Objetivo del tema

Identificar el concepto de lenguaje de programación.

Desarrollo

"Un lenguaje de programación es un sistema notacional que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora, y que describe operaciones computacionales en una forma legible tanto para la computadora como para el ser humano".[4] Es un conjunto de símbolos (lexemas) que se agrupan en categorías (tokens), contiene reglas sintácticas y semánticas que le dan estructura y significado a sus elementos y expresiones.

No sólo existen lenguajes que sirven para programar, también hay otro tipo de lenguajes, como los lenguajes de marcado (HTML por ejemplo) y los metalenguajes (como XML o la notación BNF), que permiten crear otros lenguajes (llamados lenguajes objeto).

Los lenguajes de programación intentan parecerse a los lenguajes naturales, pero esto lleva a un problema: nuestro lenguaje es ambiguo, y un lenguaje de programación puede ser todo, menos ambiguo.

[4] Kenneth C. Louden, *Programming Languages: Principles and Practice*, 2a ed., Brooks Cole, 2003.

ACTIVIDAD 1

1. Investiga en diversas fuentes bibliográficas distintas a las proporcionadas en este tema y en Internet, tres definiciones sobre el concepto de “Lenguaje de Programación”.

2. Después realiza un cuadro didáctico resumiendo las definiciones (mínimo 5) que proporcionan los autores. Incluye mínimo una referencia bibliográfica distinta a las presentadas y una referencia en línea.

Autor	Definición	Conceptos clave que se expresan.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 2. Clasificación de los lenguajes de programación

Objetivo del tema

Identificar las bases de catalogación aplicables a los lenguajes de programación, según los criterios de abstracción, ejecución y paradigmas.

Desarrollo

Los lenguajes de programación se clasifican según su nivel de abstracción, la forma en que se ejecutan en la computadora y por el paradigma o filosofía que implementen:

Clasificación de los lenguajes de programación

Según su nivel de abstracción

Lenguaje de bajo nivel: se programa en 0s y 1s, conocido como lenguaje máquina.

Lenguaje de medio nivel: utiliza nemotecnias para programar, se le conoce como lenguaje ensamblador.

ejemplo C: Hola Mundo!

```
#include <stdio.h>

int main()
{
  printf("Hola Mundo!\n");
  return 0;
}
```

Lenguajes de alto nivel: están formados por palabras que se usan en los idiomas o lenguajes naturales, como el inglés. Ejemplos de este tipo de lenguajes están: C, C++, Java, Basic, Pascal, Ruby y PHP.

```
-- escribe en el archivo rom.asc los valores binarios contenidos en el archivo rom.dat
-- en formato 1s y 0s ascii (8 bytes por línea y separados un espacio)
--
#include <stdio.h>
main()
{int nbl=8;
int i,j,data;
FILE *archivodat,*archivoasc;
archivodat=fopen("rom.dat","r");
archivoasc=fopen("rom.asc","w");
fprintf(archivoasc,"MSB a la izquierda, LSB a la derecha, primera dirección 0..0 \n\n");
j=0;
while ((data=getc(archivodat))!=EOF)
     {
     for (i=0x80;i>0x0;i=i/2)
          {
          if ((data&i)>0)
               fprintf(archivoasc,"1");
          else
               fprintf(archivoasc,"0");
          }
     j++;
     if ((j%nbl)==0)
          fprintf(archivoasc,"\n");
     else
          fprintf(archivoasc," ");
     }
fclose(archivodat);
fclose(archivoasc);
}
```

Según la forma de ejecución

Con base en lo anterior, las computadoras trabajan ya se con 0 y 1s, ensamblador o alto nivel. Cuando se usan los niveles medio y alto de abstracción, los programas se ejecutan de dos maneras:

- Un programa llamado intérprete que va ejecutando cada una de las instrucciones que va leyendo de un programa. A este proceso se le llama interpretar.
-

Un programa llamado compilador que traduce cada una de las instrucciones de un programa a su equivalente en lenguaje de 0 y 1s (lenguaje máquina). A ese proceso se le llama compilación.

Según el paradigma de programación

Un paradigma de programación es un conjunto de reglas y conceptos que dirigen la elaboración de programas que a su vez constituyen software o programas de aplicación.

Existen cuatro paradigmas principales de programación:

- Paradigma imperativo
- Paradigma orientado a objetos
- Paradigma funcional
- Paradigma lógico

ACTIVIDAD 1

Lee el capítulo 6: los años 50 y 60, del libro de Peña Marí *Los primeros lenguajes de alto nivel* **(ANEXO 1)** De Euclides a Java, Historia de los algoritmos y de los lenguajes de programación.1ª Edición, 2006 y realiza un cuadro sinóptico que incluya la siguiente información.

Lenguaje de programación	Ejemplos de datos que maneja	Ejemplos de sentencias permitidas.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar**. Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 2

Investiga en mínimo dos fuentes bibliográficas distintas el concepto de paradigma y paradigma de programación. Con la información obtenida elabora en media cuartilla un texto que explique en qué consiste cada uno de ellos. No olvides citar tus fuentes.

Para enviar tu respuesta, pulsa el botón **Editar mi envío;** se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios**

ACTIVIDAD 3

Utilizando la lectura de la Dra. Oktaba **(ANEXO 2)** y el capítulo 7 *Hacia la estructuración: El paradigma imperativo* **(ANEXO 3)** de Peña Marí, elabora un ensayo de no más de1 cuartilla, que aborde los siguientes puntos:

- Paradigmas de programación.
- Los lenguajes de programación y el procesamiento inherente a ellos.
- Representación de datos y operaciones.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

Autoevaluación

Con base en la lista de lenguajes presentado en la página Anexo: Historia de los lenguajes de programación de Wikipedia **(http://es.wikipedia.org/wiki/Historia_de_los_lenguajes_de_programaci%C3%B3n),** relaciona los lenguajes de programación que a continuación se mencionan con sus respectivas características. Arrastra el nombre del lenguaje al lugar correspondiente.

☐ Imperativo, uso académico semántica formal. ☐ Alto nivel, declarativo, basado en listas. ☐ 1er. Lenguaje orientado a objetos. ☐ Académico, propósito general, emplea la programación estructurada. ☐ Lenguaje declarativo, permite la manipulación de información de una Base de Datos, se basa en el manejo de relaciones ☐ Orientado a objetos, define una máquina virtual, independiente de plataforma (Framework). ☐ Orientado a objetos, define una máquina virtual, independiente de plataforma.	1 ALGOL 2 LISP 3 SIMULA 4 PASCAL 5 SQL 6 JAVA 7 C#

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 3. Abstracciones en los lenguajes de programación

Objetivo del tema

Reconocer los niveles de abstracción que presentan los lenguajes de programación.

Desarrollo

Existen dos tipos de abstracciones que se implementan en los lenguajes de programación:

Tipos de abstracciones en los lenguajes de programación.
De datos: Resumen las propiedades de los datos como cadenas de caracteres, números o árboles de búsqueda.
De control: Resumen propiedades de la transferencia de control, o sea, de la modificación de la trayectoria de ejecución de un programa con base en una situación determinada. Ejemplos: bucles, sentencias condicionadas, llamadas de procedimiento, etcétera.

Todas las abstracciones tienen niveles, que miden la cantidad de información contenida en la abstracción.

Existen tres niveles de las abstracciones
Básicas: Reúnen la información de máquina más localizada.
Estructuradas: Reúnen información más global sobre la estructura del programa.
Unitarias: Reúnen información sobre una parte completamente funcional de un programa.

3.1 Abstracciones de datos

Básicas: resumen la representación interna de valores de datos comunes en una computadora. Ejemplo: a menudo los valores enteros de datos se almacenan en una computadora utilizando una representación de complemento a dos. Las localizaciones en la memoria que contienen valores de datos se abstraen dándoles un nombre: se conocen como identificadores o variables. El tipo de valor de datos también recibe un nombre y se conoce como tipo. A las variables se les dan nombres y tipos de datos mediante una declaración.

```
var x: integer;
int x;
```

Estructuradas: La estructura de los datos es el método principal para la abstracción de colecciones de valores de datos relacionados entre sí. Una estructura típica es el arreglo: reúne datos en una secuencia de elementos de indización individual. A las variables se les puede dar una estructura de datos.

```
int a[10];
INTEGER a(10);
```

Las estructuras de datos también pueden ser nuevos tipos de datos:

```
typedef int intArray[10];
```

Unitarias: Es la reunión de códigos relacionados entre sí en localizaciones específicas dentro de un programa, ya sea en forma de archivos por separado o como estructuras de lenguaje por separado dentro de un archivo. Incluyen acuerdos convencionales y restricciones de acceso, que se conocen como encapsulado de datos y ocultamiento de la información. Ejemplos: los módulos, los paquetes, las clases (intermedia entre el nivel estructurado y unitario), componentes, librerías (bibliotecas), etcétera.

3.2 Abstracciones de control

Básicas: Son aquellos enunciados o sentencias que combinan unas cuantas instrucciones de máquina en una sentencia abstracta más comprensible.
Ejemplo: el enunciado de asignación, que resume el cómputo y almacenamiento de un valor en la localización dada por una variable.

```
x = x + 3; // enunciado de asignación
```

Este enunciado de asignación representa la recuperación del valor de la variable x, agregando el entero 3 a la misma y almacenando el resultado en la localización de x.

Estructuradas: Dividen un programa en grupos de instrucciones que están anidadas dentro de pruebas que gobiernan su ejecución. Ejemplo: Enunciados de selección: if, case, switch. Un mecanismo adicional para estructurar el control es el procedimiento (subprograma o subrutina).

```
procedure gcd (u, v: in integer; x: out integer) is
…
end gcd;
```

Este procedimiento puede ser llamado simplemente con nombrarlo y proporcionarle los parámetros apropiados o argumentos reales:

```
gcd (8, 18, d);
```

Unitarias: Sirven para incluir una colección de procedimientos que proporcionan servicios relacionados lógicamente con otras partes del programa y que forman una parte unitaria o independiente. Es esencialmente lo mismo que una abstracción unitaria de datos, y se implementa de igual forma a través de módulos, paquetes, clases, etcétera. La diferencia consiste en el enfoque: en las abstracciones de control unitarias el enfoque está en las operaciones y en los servicios que proporcione, más que en los datos, pero las metas de reutilización y formación de bibliotecas se conservan.

ACTIVIDAD 1

Investiga para los lenguajes Pascal, Fortran, Basic, C y Java las consideraciones específicas para el uso de los siguientes conceptos:

- Variables
- Arreglos
- Estructuras o registros

Elabora una tabla donde indiques el nombre del lenguaje y las consideraciones específicas que se requieren para cada uno. No olvides citar las fuentes que consultaste.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar**. Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 2

Redacta en no más de una cuartilla, el uso de las estructuras de control clásicas: IF, SWITCH o CASE, FOR, WHILE y DO.. WHILE.

Para enviar tu respuesta, pulsa el botón **Editar mi envío;** se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios**

Autoevaluación

Elige la opción que conteste correctamente cada pregunta. Al final obtendrás tu calificación de manera automática.

1. En Pascal la forma de declarar una variable es:

- a) <nombre variable> : <tipo>;
- b) <tipo> : <nombre variable>;
- c) <nombre variable> = <tipo>;

2. En C la forma en que se puede declarar un arreglo es:

- a) <tipo> <variable>;
- b) <tipo> <variable>[];
- c) <variable> <tipo>[];

3. La estructura de control que garantiza que al menos una vez se ejecutará es:

- a) IF
- b) FOR
- c) DO..WHILE

4. Una clase es el ejemplo de una:

- a) Abstracción de control unitaria.
- b) Abstracción de control básica.
- c) Abstracción de datos estructurada.

5. Los arreglos son un ejemplo de una:

- a) Abstracción de datos básica.
- b) Abstracción de datos estructurada.
- c) Abstracción de datos unitaria.

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 4. Historia y evolución de los leguajes de programación

Objetivo del tema

Identificar la evolución de los lenguajes y su desarrollo a lo largo del tiempo.

Desarrollo

¿Cuál fue el primer lenguaje de programación? Hasta la fecha, ¿cuántos lenguajes de programación han existido? ¿Cómo han evolucionado?

Para responder estas preguntas, qué mejor que una *timeline* [5] **(ANEXO 4)** de los lenguajes de programación, la cual se anexa en formato, En este sitio encontrarás un trabajo de investigación muy completo sobre la historia y evolución de los lenguajes de programación desde 1954 con el surgimiento de Fortran, hasta 2008 y Phyton (y hasta donde se acumule). Otros lenguajes que se muestran son: BASIC, C, Perl, Pascal, PHP, C++, C#, VB.NET, Delphi, Eiffel, Java, JavaScript, Oz, PHP, Python, Ruby, Smalltalk, Haskell, Miranda, Lisp, Scheme, Ocaml, ML, Prolog, etcétera.

[5] Fuente: Éric Lévénez: “Computer Languages History”, en línea, disponible en: http://www.levenez.com/lang/ [consulta: 15 de agosto de 2008].

ACTIVIDAD 1

Empleando como referencias el poste History of Programming Languages de Oreilly, que se encuentra en la página http://www.eecs.ucf.edu/~leavens/ComS541Fall97/hw-pages/history/ y Computer Languages History, http://oreilly.com/news/graphics/prog_lang_poster.pdf

Elige tres programas por cada década y realiza un cuadro sinóptico que incluya las características, tipos de datos que manejan, estructuras de control que tienen y el paradigma que representan cada uno de ellos.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 2

Realiza un cómic sobre la historia y evolución de los lenguajes de programación. Busca antecedentes y creadores de los siguientes lenguajes: autocódigos, FORTRAN, COBOL, Algol 60, LISP, APL, BASIC, APL, Simula, Algol 68, Pascal, PROLOG, Small Talk, C, Modula-2, ADA, C++, Delphi, Python, Visual Basic, Perl, Java, JavaScript, PHP, C#, VB.NET y Ruby. Puedes utilizar para su elaboración animación digital, video, etc

Ejemplo de cómic elaborado por alumnos de la carrera de informática.

Realiza tu actividad y guárdala en tu computadora, una vez concluida, presiona el botón **Examinar**. Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

Autoevaluación

Elige la opción que conteste correctamente cada una de las siguientes preguntas. Al Final obtendrás tu calificación de manera automática.

1. El año de aparición del lenguaje Ruby es:

- a) 1991
- b) 1993
- c) 1995

2. Java es un lenguaje que aparece en la década de:

- a) 1980
- b) 1990
- c) 2000

3. Un antecesor directo de C# es:

- a) FORTRAN IV
- b) RUBY
- c) Java

4. Un antecesor de Action Script es:

- a) ECMAScript
- b) LiveScript
- c) Ruby

5. El lenguaje Perl influencia en:

- a) PHP
- b) C++
- c) Java

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 5. Definición de un lenguaje de programación

Objetivo del tema

Identificar los elementos centrales de la definición de lenguajes de programación.

Desarrollo

Antes, los lenguajes de programación eran solo descripciones informales en inglés. Un lenguaje de programación necesita una descripción precisa y completa que lo defina formalmente. Esta definición se da con:

- **La sintaxis**
- **La semántica**

Sintaxis

Es la estructura de un lenguaje. Son las reglas que indican cómo realizar las construcciones del lenguaje de programación. Es como la *gramática* de un lenguaje natural: describe las maneras en que las diferentes partes del lenguaje pueden ser combinadas para formar otras partes. Representa la estructura superficial del lenguaje. Por ejemplo, en C:

```
x = x + 3; // secuencia de símbolos válida
x 3 x + =; // secuencia no válida
```

La sintaxis proporciona la información significativa necesaria para entender un programa y proporciona mucha información para la traducción del programa fuente en un programa objeto. Ejemplo:

La expresión 2 + 3 * 4 = 14 es interpretada como:

2 + (3 * 4) = 14

y no como:

(2 + 3) * 4 = 20

El léxico

Ayuda a la especificación de la sintaxis y estructura de un lenguaje de programación. Es similar a la ortografía de un lenguaje natural. Constituye el conjunto de símbolos permitidos o vocabulario. El léxico se conforma de:

- Lexemas
- Tokens
- Sentencias

Componentes del léxico
Lexema: Son las unidades sintácticas de más bajo nivel. Incluyen: identificadores, operadores y palabras especiales.
Token: Es una categoría de lexemas (el lexema es un atributo del token).
Sentencia: Los lenguajes de programación utilizan conjuntos de cadenas de caracteres pertenecientes a algún alfabeto. Una sentencia es cada una de estas cadenas. Dicho de otra forma, son cadenas de lexemas.

Algunas categorías de lexemas (o tokens) son:

• Conjunto de caracteres ASCII (abcdefgABCDEFG123456[]{}/+-*)
• Identificadores (cualquier nombre de una variable, constante, tipo de dato)
• Símbolos operadores (+*/-=<>!)
• Palabras clave y palabras reservadas (if, else, case, this, include)
• Literales y constantes ($_GET, $_POST, cualquier valor de un tipo de dato)
• Comentarios (// Este es un comentario, */ Este es otro comentario*/)
• Espacios y delimitadores (|,_,-)
• Formatos de libre campo y campo fijo (posición en columnas de una instrucción)
• Expresiones (cualquier expresión regular) • Sentencias o enunciados (declaraciones o asignaciones)

Los tokens de un lenguaje de programación pueden describirse empleando expresiones regulares:

Expresión regular	Token
letra (letra | dígito | '_')*	Identificador
('a'..'z') U ('A'..'Z')	letra
'0'..'9'	dígito

Metalenguajes y notación BNF

Un metalenguaje es usado para definir a otros lenguajes. A estos se les llama lenguajes objeto. En lingüística la sintaxis que describe la gramática de un lenguaje es un ejemplo de metalenguaje.

Por ejemplo, *bisílaba* es toda aquella palabra que tiene dos sílabas. Pero en sí misma la palabra *"bisílaba"* no es bisílaba por definición.

Otro ejemplo del uso de los metalenguajes es en las matemáticas: los metalenguajes sirven para resolver paradojas matemáticas. ¿Cuál de las siguientes proposiciones dice la verdad?

A: El enunciado B dice la verdad.
B: El enunciado A dice mentiras.

Necesitamos una proposición escrita en metalenguaje matemático para poder calificar a las otras dos proposiciones, las cuales estarán definidas en lenguaje objeto.

En informática, XML es un ejemplo de metalenguaje de marcado, ya que a partir de él se describen otros lenguajes de marcado objeto como XLink, XMath, XSL, etcétera. Otro ejemplo de metalenguaje informático es la notación BNF.

La notación BNF

La notación BNF se usa para describir lenguajes de programación. Es una nomenclatura que sirve para describir la sintaxis de un lenguaje usando ciertos símbolos y reglas.

Se manejan dos tipos de elementos: Los elementos o **símbolos terminales**, que pertenecen al vocabulario y que se escriben tal cual, y los elementos o **símbolos no terminales** que se escriben entre los símbolos <>.

Los símbolos BNF son:

Sintaxis	Significado
::=	se define como
<>	se usa para delimitar el nombre de una categoría
|	se usa para separar opciones de una categoría

Ejemplo:

<digito>::= 0|1|2|3|4|5|6|7|8|9
<operador>::= =|-|*|+|/
<letra> ::= a|A|b|B|c|C|…|z|Z

Semántica

Son las reglas que determinan el significado de un lenguaje.
En C la sintaxis del if:

<enunciado if>::= if (<expresión>) <enunciado>
[else <enunciado>]

En C la semántica de if:

Un enunciado if es ejecutado, primero, evaluando su expresión, misma que debe tener tipo aritmético o apuntador, incluyendo todos los efectos colaterales, y si se compara diferente de 0, el enunciado que sigue a la expresión es ejecutado. Si existe un parte else, y la expresión es 0, el enunciado que sigue al “else” es ejecutado.

ACTIVIDAD 1

Investiga la biografía de los creadores de la notación BNF: John Backus y Peter Naur.

Realiza un reporte no mayor a dos cuartillas y no olvides citar las fuentes que consultaste.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 2

Considerando los lenguajes de programación Pascal, C y Java encuentra la sintaxis de cada uno de los siguientes elementos:

- Arreglos
- Condicional IF .. ELSE
- Estructura CASE o SWITCH
- Ciclo FOR
- Ciclo WHILE
- Ciclo DO .. WHILE

Elabora un cuadro comparativo entre las sintaxis de dichos lenguajes y sus principales características.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

Autoevaluación

Lee cuidadosamente lo que se te pide y elige la opción que conteste correctamente cada pregunta.

1. for(i = 0; i < 20; i++)

- a) 19
- b) 20
- c) 21

2. for(j = 12; j > 5; j--)

- a) 7
- b) 8
- c) 6

3. for(h = 15; h < 20; h++)

- a) 3
- b) 4
- c) 5

4. for(l = 20; l > 10; l -= 2)

- a) 5
- b) 6
- c) 7

5. for(a = 45; a < 10; a -= 4)

- a) 9
- b) 5
- c) 0

6. for(b = 15; b < 100; b += 4)

- a) 23
- b) 22
- c) 21

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 6. Traducción de los lenguajes de programación

Objetivo del tema

Analizar el proceso de compilación a partir de un código fuente hasta la generación de un programa ejecutable.

Desarrollo

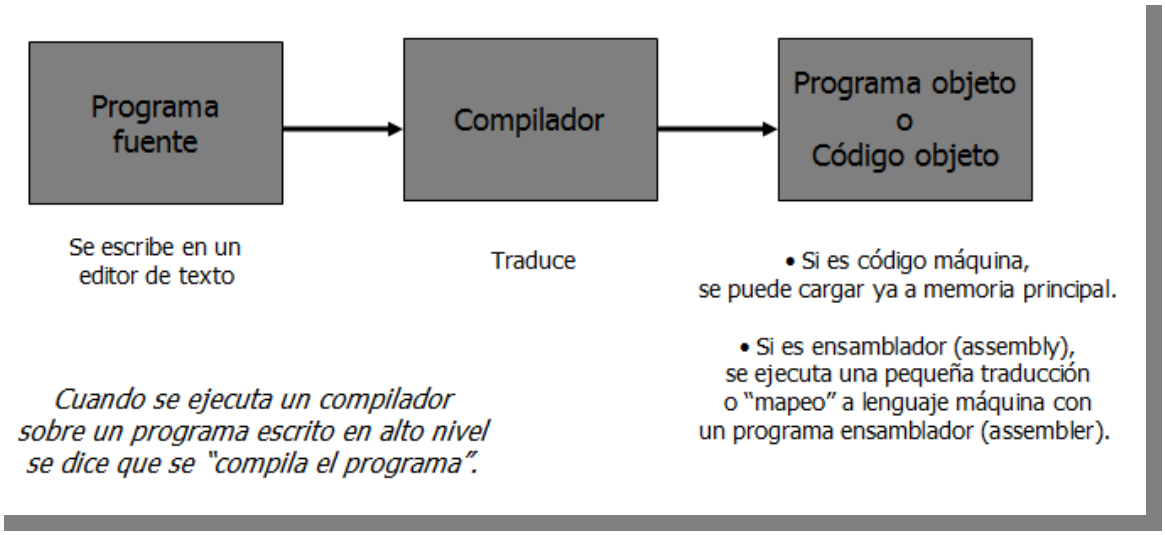
Cualquier código fuente debe ser traducido a código binario para que las instrucciones que pusimos en él puedan ser entendidas y ejecutadas por la máquina (las computadoras no entienden directamente los lexemas “if (valor==0){instrucción 1} else {instrucción 2}”, sino que éstos deben pasarse a 0y1s).

Hay programas encargados de realizar esta traducción, se llaman traductores. Hay dos tipos: compiladores e intérpretes.

Compilación

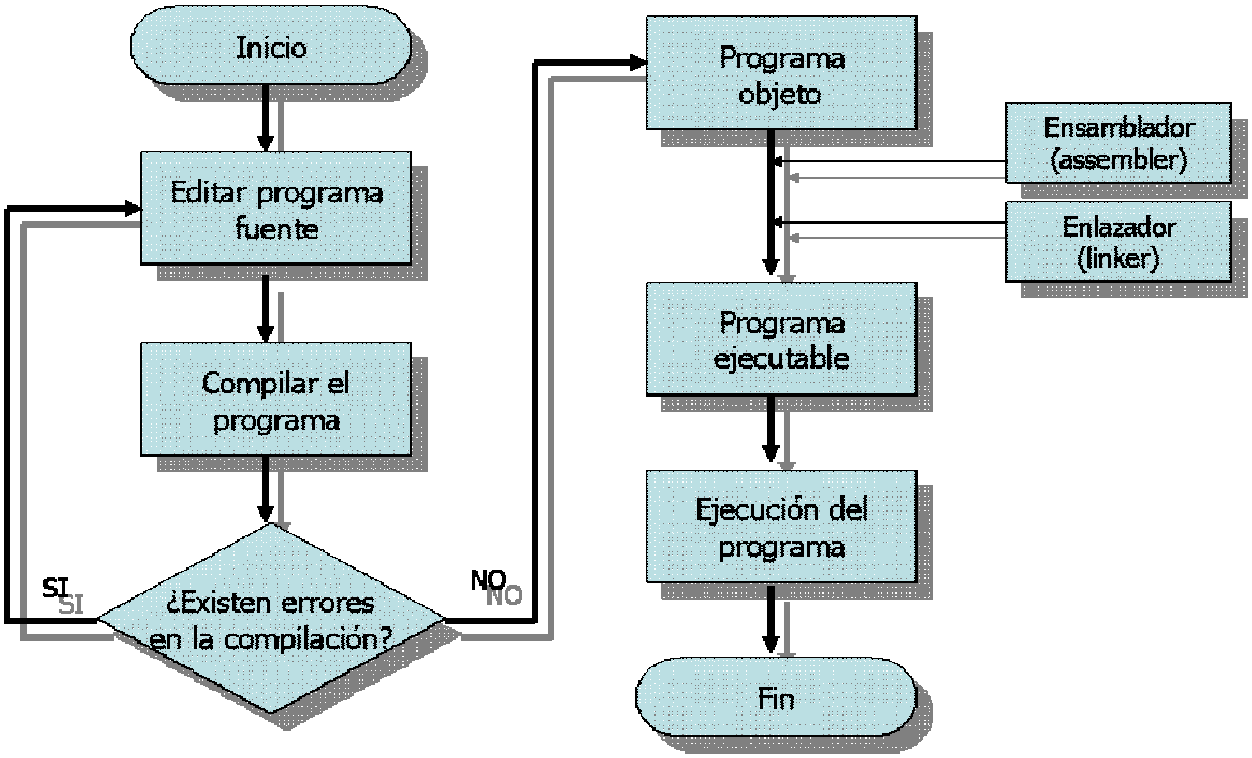
Se atribuye el término compilador a Grace Murray Hopper. Los primeros compiladores se escribieron en los años 50. FORTRAN es el primer lenguaje compilado con éxito.

Un compilador es un programa que traduce un programa fuente escrito en lenguaje fuente y da como resultado un programa objeto en lenguaje objeto.



Entrada y salida en el proceso de compilación[6]

El proceso de compilación tiene el siguiente flujo:



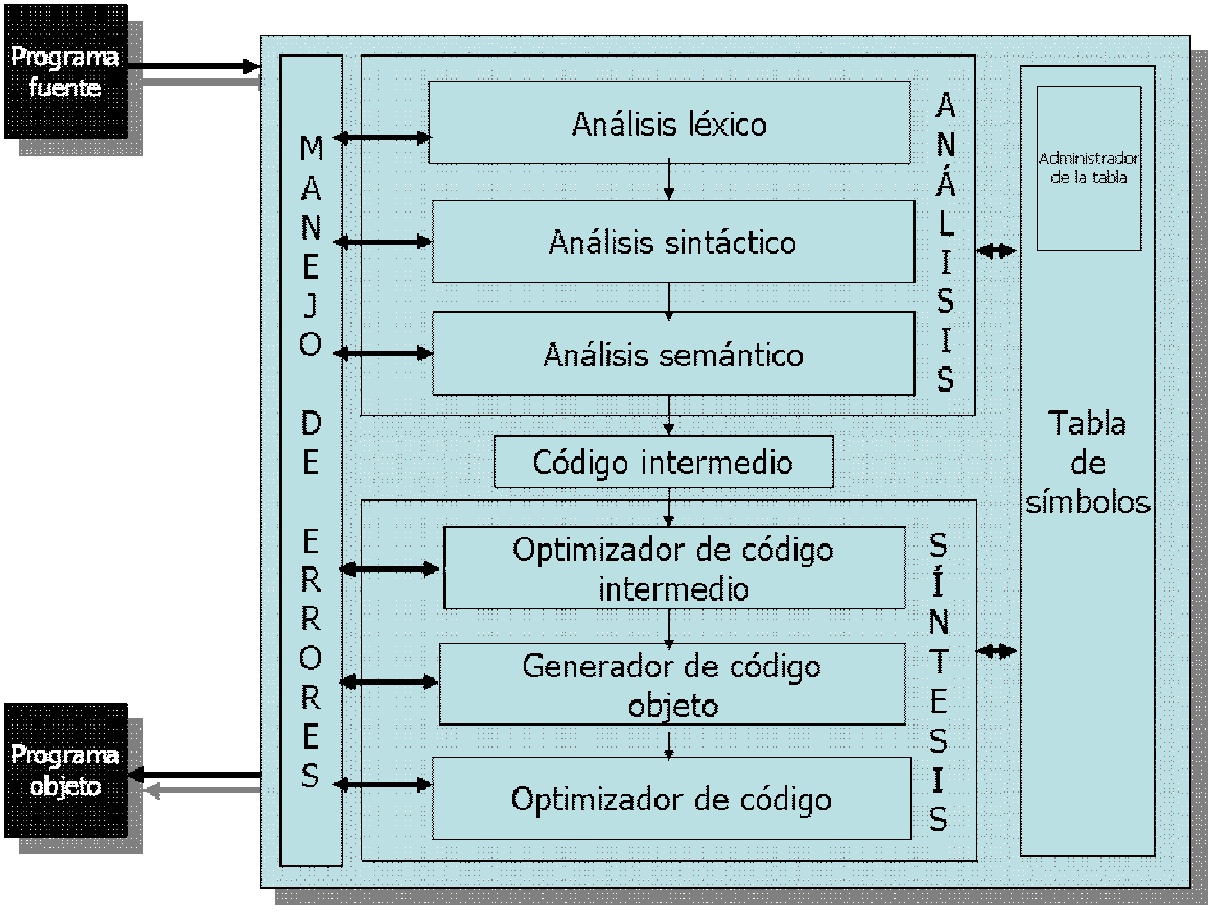
Proceso de compilación en flujo de datos[7]

[6] Fuente: Perea, Ismael: *Apuntes de programación para tercer semestre de la licenciatura en informática*, 2008.

Este proceso se puede dividir en una serie de fases, que pueden llevarse a cabo simultáneamente o consecutivamente y cada una de las cuales transforma el programa fuente de una representación a otra. Se pueden agrupar las fases en etapas[8] :

- Etapa de análisis o front-end
- Etapa intermedia o middle-end
- Etapa de síntesis o back-end

Veamos el detalle del proceso y sus fases:



El proceso de compilación, sus etapas y fases[9]

[7] Fuente: Perea, Ismael: *Apuntes de programación para tercer semestre de la licenciatura en informática*, 2008.
[8] César Ignacio García Osorio: *Introducción a los compiladores*, material en línea, disponible en: http://pisuerga.inf.ubu.es/cgosorio/ALeF/UD1/compilacion.pdf, consultado el 28/01/09.
[9] César Ignacio García Osorio: *Introducción a los compiladores*, material en línea, disponible en: http://pisuerga.inf.ubu.es/cgosorio/ALeF/UD1/compilacion.pdf, consultado el 28/01/09

Programa fuente

Un programa fuente o código fuente o simplemente "fuente" es el conjunto de instrucciones válidas para un lenguaje de programación en particular que indican a una computadora lo que debe realizar. Se crean en cualquier editor de texto o en línea de comandos (ejemplo: el shell de Unix).

- La etapa de análisis (front-end o etapa inicial) agrupa aquellas fases que dependen principalmente del lenguaje fuente[10] (java, C, C++, etcétera). Las fases son:

Etapas de la fase de análisis
Análisis léxico Analizador léxico (también llamado scanner): Agrupa los caracteres individuales en entidades lógicas léxicas[11] (lexemas) e identifica el token con el que se corresponden. Realiza las entradas oportunas en la tabla de símbolos. La salida es una secuencia de tokens, provoca una simplificación de la entrada.
Análisis sintáctico Analizador sintáctico (también llamado parser): Analiza la estructura general de todo el programa, agrupando las entidades simples identificadas por el scanner en construcciones mayores, como sentencias, bucles, rutinas, etcétera, que componen el programa completo. Normalmente se utiliza la representación de árboles sintácticos para reflejar dicha estructura.[12] Aplica una gramática para construir el árbol sintáctico para un programa a partir de la secuencia de tokens.

[10] César Ignacio García Osorio: *Introducción a los compiladores*, material en línea, disponible en: http://pisuerga.inf.ubu.es/cgosorio/ALeF/UD1/compilacion.pdf, consultado el 28/01/09

[11] César Ignacio García Osorio: *Introducción a los compiladores*, material en línea, disponible en: http://pisuerga.inf.ubu.es/cgosorio/ALeF/UD1/compilacion.pdf, consultado el 28/01/09.

[12] Véase, César Ignacio García Osorio: *Introducción a los compiladores*, material en línea, disponible en: http://pisuerga.inf.ubu.es/cgosorio/ALeF/UD1/compilacion.pdf, consultado el 28/01/09.

Análisis semántico

Una vez determinada la estructura del programa se puede analizar su significado (semántica) mediante un análisis en el que se determina qué variables almacenarán enteros, cuáles números de punto flotante, si el acceso a los arrays cae dentro del rango fijado en su definición, etcétera.[13] También, realiza un estudio de los aspectos contextuales del lenguaje:

- Comprobación de tipos
- Comprobación de unicidad (un objeto se define una sola vez)
- Comprobación de nombres (todas las variables que se usan deben estar declaradas)
- Se completa la información almacenada en la tabla de símbolos a la vez que se consulta para realizar las comprobaciones oportunas

Generación de código intermedio

Es el código generado por la etapa de análisis que recibe como entrada la etapa de síntesis. Un tipo de código intermedio es el AST (Abstract Syntax Trees) que es una forma condensada de árboles de análisis, con sólo nodos semánticos y sin nodos para símbolos terminales (se supone que el programa es sintácticamente correcto). Otros tipos de códigos intermedios son: DAG (Directed Acyclic Graphs), TAC (Three Address Code) y RTL (Register Transfer Language).

[13] Véase, César Ignacio García Osorio: *Introducción a los compiladores*, material en línea, disponible en: http://pisuerga.inf.ubu.es/cgosorio/ALeF/UD1/compilacion.pdf, consultado el 28/01/09.

La etapa de síntesis (back-end o etapa final) agrupa aquellas fases que dependen principalmente[14] de la arquitectura de la computadora (i860, 88000, OpenRISC, SPARC, 8086, 8088, x86-64, etcétera). Sus fases son:

Etapas de la fase de síntesis
Optimizador de código intermedio: Transforma la representación intermedia a otra equivalente pero más eficiente[15]
Programa objeto Es el código generado por el proceso de compilación en lenguaje máquina o bytecode (en el caso de Java) y se distribuye en varios archivos que corresponden a cada código fuente compilado. Después se obtiene un programa ejecutable para lo cual se han enlazado todos los archivos de código fuente con un programa llamado enlazador o linker.
Generador de código objeto Genera un programa binario equivalente al código fuente para su ejecución en la computadora de arquitectura específica, añadiéndole posiblemente rutinas de bibliotecas y códigos de inicialización.[16]
Optimizador de código Finalmente puede haber un optimizador de código para mejorar aún más el código generado haciéndolo de menor tamaño y más rápido.[17]

[14] Véase, César Ignacio García Osorio: *Introducción a los compiladores*, material en línea, disponible en: http://pisuerga.inf.ubu.es/cgosorio/ALeF/UD1/compilacion.pdf, consultado el 28/01/09.
[15] Véase, César Ignacio García Osorio: *Introducción a los compiladores*, material en línea, disponible en: http://pisuerga.inf.ubu.es/cgosorio/ALeF/UD1/compilacion.pdf, consultado el 28/01/09.
[16] Véase, César Ignacio García Osorio: *Introducción a los compiladores*, material en línea, disponible en: http://pisuerga.inf.ubu.es/cgosorio/ALeF/UD1/compilacion.pdf, consultado el 28/01/09.
[17] Véase, César Ignacio García Osorio: *Introducción a los compiladores*, material en línea, disponible en: http://pisuerga.inf.ubu.es/cgosorio/ALeF/UD1/compilacion.pdf, consultado el 28/01/09.

Tabla de símbolos

Un compilador debe tener un determinado comportamiento ante programas erróneos. Este proceso se agrupa en la fase de manejo de errores. Cada una de las fases anteriores interactúa con este manejador.[18]

Otro elemento de la compilación es la tabla de símbolos a la que accede cada una de las fases a través del administrador de la tabla de símbolos. En esta tabla se almacena información sobre variables, constantes, funciones y procedimientos, tipos de datos, memoria asignada, ámbito, alcance, etcétera.[19] Es una estructura de datos que contiene un registro por cada identificador con sus atributos. Es como un diccionario en donde las fases anteriores introducen esta información sobre los identificadores y después la utilizan de varias formas.

Interpretación

Hay dos formas de ejecutar un programa escrito en alto nivel:

- Compilación: traducir todo el programa fuente a otro programa equivalente en código máquina. Entonces se ejecuta el programa obtenido.
- Interpretación: interpretar las instrucciones del programa fuente y ejecutarlas una por una.

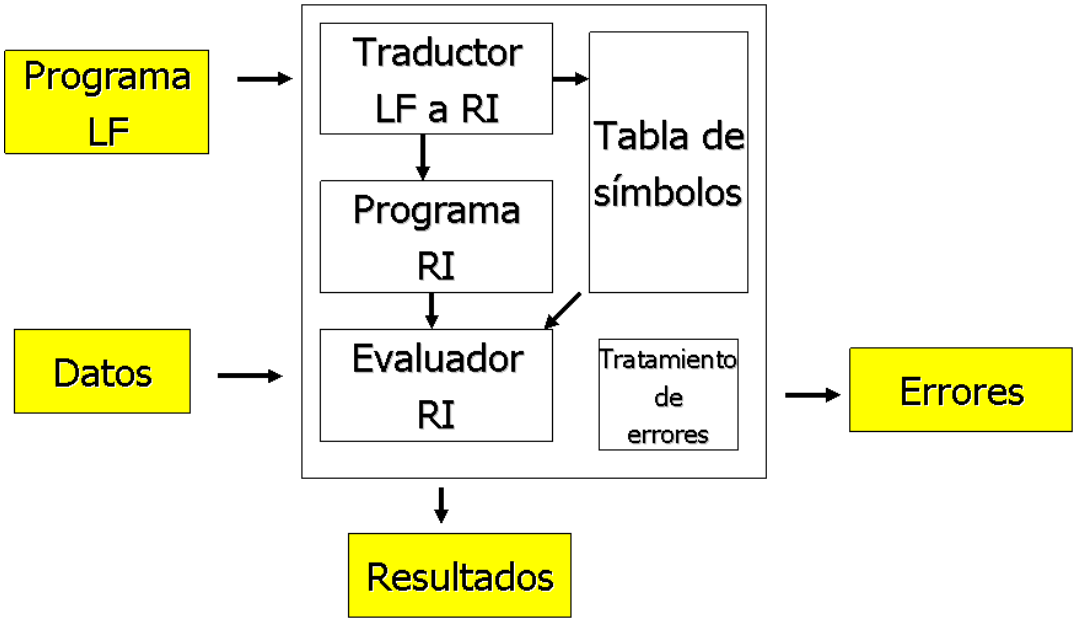
Hay lenguajes de programación que utilizan un programa intérprete traductor, el cual analiza directamente la descripción simbólica del programa fuente y realiza las instrucciones dadas.[20]

[18] Véase, César Ignacio García Osorio: *Introducción a los compiladores*, material en línea, disponible en: http://pisuerga.inf.ubu.es/cgosorio/ALeF/UD1/compilacion.pdf, consultado el 28/01/09.
[19] Véase, César Ignacio García Osorio: *Introducción a los compiladores*, material en línea, disponible en: http://pisuerga.inf.ubu.es/cgosorio/ALeF/UD1/compilacion.pdf, consultado el 28/01/09.
[20] Cf. Lenguajes de programación: "Lenguajes de programación", material en línea, disponible en: http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml, consultado el 28/01/09.

El intérprete en los lenguajes de programación simula una máquina virtual, donde el lenguaje de máquina es similar al lenguaje fuente.[21]

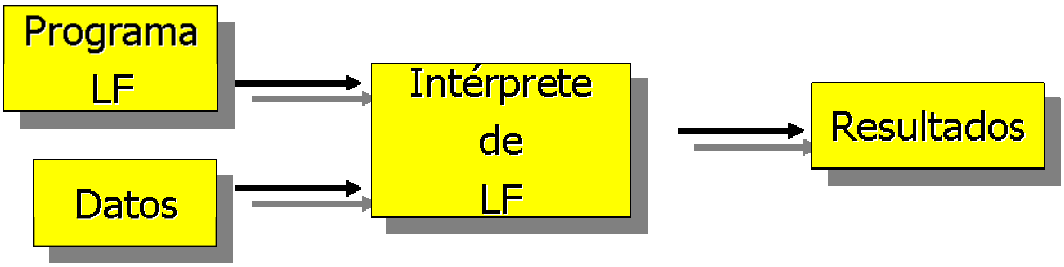
La ventaja del proceso intérprete es que no necesita de dos fases para ejecutar el programa (la fase de compilación y la fase de carga de datos que son propias de los lenguajes compilados), sin embargo su inconveniente es que la velocidad de ejecución es más lenta ya que debe analizar e interpretar las instrucciones contenidas en el programa fuente.[22]

Los intérpretes realizan la traducción y ejecución de forma simultánea, es decir, un intérprete lee el código fuente y lo va ejecutando al mismo tiempo.



Entradas y salidas del proceso de interpretación[23]

[21] Cf. Lenguajes de programación: "Lenguajes de programación", material en línea, disponible en: http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml, consultado el 28/01/09.
[22] Cf. Lenguajes de programación: "Lenguajes de programación", material en línea, disponible en: http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml, consultado el 28/01/09
[23] Fuente: Ismael Perea: *Apuntes de programación para tercer semestre de la licenciatura en informática*, 2008.



El proceso de interpretación en detalle[24]

Las diferencias entre un compilador y un intérprete básicamente son:

Diferencias entre compilador e intérprete

• Un programa compilado puede funcionar por sí solo mientras que un código traducido por un intérprete no puede funcionar sin éste.
• Un programa traducido por un intérprete puede ser ejecutado en cualquier máquina ya que, cada vez que se ejecuta el intérprete, tiene que compilarlo.
• Un archivo compilado es mucho más rápido que uno interpretado.[25]

[24] Fuente: Perea, Ismael: Apuntes de programación para tercer semestre de la licenciatura en informática, 2008

[25] Sara Álvarez: "Proceso de traducción de los lenguajes de programación", 24/02/06, material en línea, disponible en: http://www.desarrolloweb.com/articulos/2387.php, recuperado el 28/01/09.

ACTIVIDAD 1

En los sistemas operativos Unix o Linux existe un compilador de C que se llama gcc. Utiliza una cuenta en un servidor Linux o Unix y contesta las siguientes preguntas

a) La instrucción $gcc holaMundo.c genera:

a) holaMundo
b) holaMundo.s
c) a.out
d) holaMundo.o

b) La instrucción $gcc holaMundo.c –o holaMundo genera:

a) holaMundo.s
b) holaMundo
c) a.out
d) holaMundo.exe

c) La instrucción $gcc holaMundo.c –c genera:

a) a.out
b) holaMundo.o
c) holaMundo.s
d) holaMundo

d) La instrucción $gcc holaMundo.c –S genera:

a) código binario
b) código *assembly*
c) código *assembler*
d) bytecode

Copia las preguntas en un documento en Word y contéstalas. Sube tus respuestas a la plataforma.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

Autoevaluación

Relaciona los siguientes conceptos con sus respectivas definiciones. Arrastra el concepto a la línea correcta. Al final obtendrás tu calificación de manera automática.

☐ Transforma la representación intermedia a otra equivalente pero más eficiente. _____.	1. Programa Objeto
☐ Es el código generado por el proceso de compilación en lenguaje máquina o bytecode (en el caso de Java) y se distribuye en varios archivos que corresponden a cada código fuente compilado. _____	2. Interpretación
☐ Finalmente puede haber un optimizador de código para mejorar aún más el código generado haciéndolo de menor tamaño y más rápido._____	3. Compilación
☐ Traducir todo el programa fuente a otro programa equivalente en código máquina.____ _____-	4. Optimizador de código intermedio
☐ Interpretar las instrucciones del programa fuente y ejecutarlas una por una. _____	5. Optimizador de código

Cuestionario de autoevaluación

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar**. Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

1. ¿Qué es un lenguaje de programación?
2. ¿Qué es un lenguaje traductor?
3. ¿Qué es un compilador?
4. ¿Cuál es la diferencia entre un intérprete y un compilador?
5. ¿Cuáles son las etapas de la compilación?
6. ¿Qué es un analizador sintáctico?
7. ¿Qué es un analizador semántico?
8. ¿Para qué sirve una tabla de símbolos?
9. ¿Qué es un metalenguaje?

10 ¿Qué es el léxico de un lenguaje de programación?

Examen de Autoevaluación

Selecciona el inciso de la respuesta correcta de las siguientes preguntas. Al final obtendrás tu calificación de manera automática.

1. Un paradigma de programación es:

- a) Un conjunto de teorías generales, suposiciones, leyes y técnicas que forman una visión del mundo
- b) Una forma de representar y manipular el conocimiento
- c) Un enfoque particular o filosofía para la construcción de programas
- d) Una metodología de desarrollo de software

2. Los cuatro principales paradigmas de programación son:

- a) Imperativo, OO, funcional y procedimental
- b) OO, lógico, orientado a aspectos e imperativo
- c) Lógico, imperativo, funcional y OO
- d) Orientado a aspectos, OO, estructural y funcional

3. La expresión regular que describe los siguientes lexemas: _REQUEST, _4NUM y S_NOMBRE del token “identificador” es:

- a) <identificador>::= letra (letra | dígito | ‘_’)*, <letra> ::= (‘a’..’z’) U (‘A’..’Z’) y <dígito> ::= ‘0’..’9’
- b) <identificador>::= letra (letra | dígito | ‘_’)*, <letra> ::= ‘A’..’Z’ y <dígito> ::= ‘0’..’9’
- c) <identificador>::= (letra | ‘_’) (letra | dígito | ‘_’)*, <letra> ::= (‘a’..’z’) U (‘A’..’Z’) y <dígito> ::= ‘0’..’9’
- d) <identificador>::= (‘_’ | letra) (letra | dígito | ‘_’)*, <letra> ::= ‘A’..’Z’ y <dígito> ::= ‘0’..’9’

4. En esta estructura de datos se almacena información sobre variables, constantes, funciones y procedimientos, tipos de datos, memoria asignada, ámbito, alcance, etcétera. Es como un diccionario en donde cada fase de la compilación introduce esta información y después la utilizan de varias formas.

- a) Árbol sintáctico
- b) Tabla de errores
- c) Código intermedio
- d) Tabla de símbolos

5. A partir del modelo de Von Neumann los programadores se dieron cuenta de que sería de gran ayuda asignar símbolos nemotécnicos a los códigos de instrucción, así como a las localizaciones de memoria, y nació el:

- a) Lenguaje ensamblador
- b) Lenguaje máquina
- c) Lenguaje de marcado
- d) Lenguaje objeto

6. Un ejemplo de metalenguaje es:

- a) COBOL
- b) UML
- c) XML
- d) Java

7. Un [...] es un traductor que toma un programa fuente, lo traduce y a continuación lo ejecuta de forma secuencial.

- a) Compilador
- b) Intérprete
- c) Analizador sintáctico
- d) Optimizador de código

8. Las abstracciones de control básicas incluyen enunciados o sentencias que combinan unas cuantas instrucciones de máquina en una sentencia abstracta más comprensible. x = x + 3 es un ejemplo de este tipo de abstracción y se conoce como:

- a) Procedimiento
- b) Comparación
- c) Asignación
- d) Declaración

9. Es como la ortografía de un lenguaje natural.

- a) Sintaxis
- b) Semántica
- c) Léxico
- d) Gramática

10. El código de programación que no se compila y que es interpretado por el navegador es:

a)	b)	c)	d)
public class HolaMundo extends Applet { public void paint(Graphics g) { g.drawString("Hola mundo",0,25); } }	<HTML> <HEAD> <TITLE>Hola mundo</TITLE> </HEAD> <BODY> Hola mundo </BODY> </HTML>	class HelloWorldSwing { static public void main(String args[]) { javax.swing.JOptionPane.showMessageDialog(null,"Hola mundo"); } }	<SCRIPT language="JavaScript" type="text/javascript"> document.write("Hola Mundo"); </SCRIPT>

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

LO QUE APRENDÍ

Contesta el siguiente cuestionario:

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

1. ¿Un lenguaje que es imperativo puede ser orientado a objetos?
2. Algunos autores señalan que C++ como un lenguaje híbrido, ¿A qué se debe esta argumentación?
3. ¿Cuál es el objetivo del proceso de ensamblado?
4. Para cada paradigma de programación que se lista a continuación, indica un ejemplo de lenguaje que lo cumple:
   - Imperativo
   - Orientado a objetos
   - Funcional
   - Lógico
5. Utilizando el lenguaje C señala un ejemplo de uso de las siguientes estructuras de control:
   - FOR
   - SWITCH
   - WHILE
   - DO .. WHILE
   - IF
6. Dada las siguientes reglas de un lenguaje, genera tres cadenas válidas:
   - <cadena> ::= <digito>|<caracter>|<dígito><caracter>|<cadena><digito>
   - <caracter> ::= a|b|c|..|z
   - <digito> ::= 0|1|..|9
7. ¿Cuál es la diferencia entre el uso de las estructuras WHILE y DO .. WHILE?

Glosario de la unidad

Lenguaje de programación

Es un sistema notacional que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora, y que describe operaciones computacionales en una forma legible tanto para la computadora como para el ser humano.

Lenguajes de bajo nivel

Se programa en 0s y 1s, conocido como lenguaje máquina.

Lenguajes de medio nivel

Utiliza nemotecnias para programar, se les conoce como lenguaje ensamblador.

Lenguajes de alto nivel

Están formados por palabras que se usan en los idiomas o lenguajes naturales, como el inglés.

Paradigma de programación

Es un conjunto de reglas y conceptos que dirigen la elaboración de programas que a su vez constituyen software o programas de aplicación.

Abstracción de datos

Resumen las propiedades de los datos como cadenas de caracteres, números o árboles de búsqueda.

Abstracción de control

Resumen propiedades de la transferencia de control, o sea, de la modificación de la trayectoria de ejecución de un programa con base en una situación determinada. Ejemplos: bucles, sentencias condicionadas, llamadas de procedimiento, etcétera.

Sintaxis

Es la estructura de un lenguaje. Son las reglas que indican cómo realizar las construcciones del lenguaje de programación.

Semántica

Son las reglas que determinan el significado de un lenguaje.

Notación BNF

Se usa para describir lenguajes de programación. Es una nomenclatura que sirve para describir la sintaxis de un lenguaje usando ciertos símbolos y reglas.

Programa fuente (código fuente o simplemente, “fuente”)

Es el conjunto de instrucciones válidas para un lenguaje de programación en particular que indican a una computadora lo que debe realizar. Se crean en cualquier editor de texto o en línea de comandos.

Analizador sintáctico (también llamado parser)

Análisis de la estructura general de todo el programa, agrupando las entidades simples identificadas por el scanner en construcciones mayores, como sentencias, bucles, rutinas, etcétera, que componen el programa completo.

Análisis semántico

Análisis del significado (semántica) de un programa, en el que se determina qué variables almacenarán enteros, cuáles números de punto flotante, si el acceso a los arreglos cae dentro del rango fijado en su definición, etcétera.

Programa objeto

Es el código generado por el proceso de compilación en lenguaje máquina o bytecode (en el caso de Java) y se distribuye en varios archivos que corresponden a cada código fuente compilado.

Interpretación

Interpretar las instrucciones del programa fuente y ejecutarlas una por una.

Intérprete

Programa que va ejecutando cada una de las instrucciones que va leyendo de un programa. A este proceso se le llama interpretar.

Compilación

Traducir todo el programa fuente a otro programa equivalente en código máquina. Entonces se ejecuta el programa obtenido.

Compilador

Programa que traduce cada una de las instrucciones de un programa a su equivalente en lenguaje de 0 y 1s (lenguaje máquina). A ese proceso se le llama compilación.

MESOGRAFÍA

Referencias Bibliográficas

Sitios electrónicos

(ANEXO 1). Descarga archivo PDF de la plataforma

Los años 50 y 60: Los primeros lenguajes de alto nivel.

(ANEXO 2) Descarga la lectura de la plataforma de la Dra. Oktaba **Oktaba Lenguajes de programación**

(ANEXO 3) Descarga el archivo PDF de la plataforma (Hacia la estructuración: El paradigma imperativo) de Peña Marí.

(ANEXO 4) Descarga el archivo de la plataforma ***timeline***