

Instituto Tecnológico de Costa Rica

Escuela de ingeniería en computación



Curso: Lenguajes de Programación

Profesor: Allan Rodríguez Dávila

Investigación

Elaborado por:

Gadyr Calderón Diaz

Carné: 2022327328

Índice

| | |
|--|----|
| Lenguajes de Programación. | 3 |
| Lenguajes de Programación en el tiempo. | 4 |
| Conceptos Importantes. | 11 |
| Referencias. | 14 |

Lenguajes de Programación.

Se le denomina programa al conjunto de órdenes e instrucciones que se le dan al ordenador para que logre resolver un problema. Entendiendo que es un programa, podemos ver hacia atrás en el tiempo y entender como fueron fundamentados los primeros lenguajes de programación.

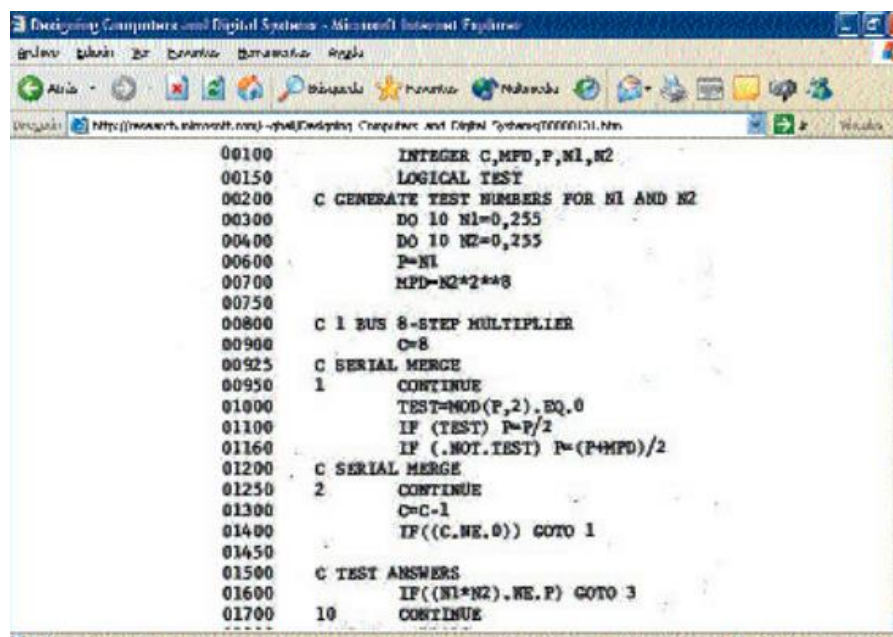
En los primeros avances de la informática se efectuaba un único lenguaje el cual entiende el microprocesador, su propio código binario, también como lo conocemos “lenguaje máquina”, pero, como bien sabemos el programar en lenguaje maquina es muy tedioso y complicado ya que los datos de entrada tienen que mandarse en sistema binario. De esta manera en los años 50s se dio la creación de una nueva notación simbólica la cual conocemos como “Código de Ensamble”, el cual utiliza una serie de abreviaciones mnemónicas para representar ciertas operaciones funcionales. En los inicios de este nuevo lenguaje la traducción a lenguaje máquina se hacía de manera manual, pero de manera rápida se observó que el ordenador podía hacer esta traducción, desarrollando así el “Ensamblador”.

Conforme las tecnologías fueron avanzando y los ordenadores fueron introduciéndose al mundo de lo empresarial y académico, los lenguajes primitivos fueron sustituidos por lenguajes más sencillos de entender y emplear, estos son conocidos como lenguajes de alto nivel, los cuales presentan una estructura que logra un mayor entendimiento para los seres humanos en comparación con la del ordenador.

Lenguajes de Programación en el tiempo.

Los lenguajes de programación fueron creciendo en cuanto a entendimientos al usuario, los cuales, mencionados anteriormente, son los lenguajes de alto nivel, el inicio de los lenguajes de alto nivel fue adaptándose según avanzaba la tecnología y el uso de ellos, como también fueron surgiendo más y más lenguajes. Con esto claro, procederemos a ver los lenguajes de programación en su evolución y línea de tiempo.

Uno de los primeros lenguajes en surgir fue “**FORTRAN**”, este tuvo su inicio en la década de los 50s, siendo más específicos, en el año de 1954 John Backus desarrollo el programa “**SPEEDCODING**” para uno de los primeros ordenadores de IBM, este lenguaje de programación fue diseñado para la resolución de problemas científico-técnicos, resultando ligeramente sencillo de aprender si se denomina de manera anticipada la notificación matemática. Este lenguaje, aunque se fue mejorando con el tiempo, fue superado por otros lenguajes ya que sus programas carecen de estructuración y son difíciles de seguir.



```
00100      INTEGER C,MFD,P,M1,M2
00150      LOGICAL TEST
00200      C GENERATE TEST NUMBERS FOR M1 AND M2
00300      DO 10 M1=0,255
00400      DO 10 M2=0,255
00600      P=M1
00700      MFD=M2*2**8
00750
00800      C I BUS 8-STEP MULTIPLIER
00900      C=8
00925      C SERIAL MERGE
00950      1      CONTINUE
01000      TEST=MOD(P,2).EQ.0
01100      IF (TEST) P=P/2
01160      IF (.NOT.TEST) P=(P+MFD)/2
01200      C SERIAL MERGE
01250      2      CONTINUE
01300      C=C-1
01400      IF((C.NE.0)) GOTO 1
01450
01500      C TEST ANSWERS
01600      IF((M1*M2).NE.P) GOTO 3
01700      10      CONTINUE
-----
```

“Figura 1. Lenguaje **FORTRAN**”

Prosiguiendo con el siguiente lenguaje, a finales de la década de los 50s, en el año de 1959 tuvo lugar una conferencia en la cual se dieron las especificaciones para desarrollar el lenguaje de **COBOL** (COmmon Business Oriented Language), un lenguaje orientado hacia funcionalidades administrativas, de gran portabilidad y legibilidad, la primera aparición se dio un año después.

[illegible]

Los programas se estructuran en cuatro divisiones, las cuales son, Identification, Environment, Data and Procedure, estas a su vez se dividen en subsecciones las cuales también se dividen en párrafos que contienen frases e instrucciones. En tiempos modernos COBOL se utiliza exclusivamente en algunos grandes sistemas informáticos más que todo en sistemas

bancarios, esto es más bien para mantener el código existente que para desarrollar nuevas aplicaciones.

En **1964**, los profesores John G. Kemeny y Thomas E. Kurtz diseñaron un nuevo lenguaje con el propósito que los estudiantes pudieran introducirse a los sistemas de tiempo compartido, a este nuevo lenguaje le llamaron **BASIC** dado a su sencillez, este es el más difundido, aplicándose en tareas de gestión como también en aplicaciones científicas.

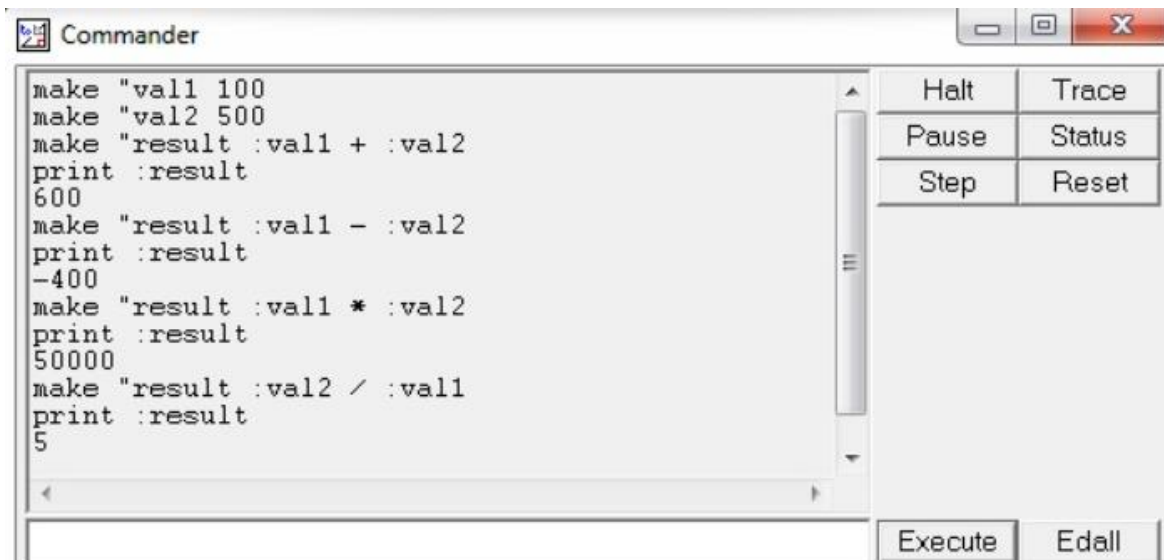
```
10 'This will draw 5 spheres
20 GOTO 160
50 IF VERT GOTO 100
60 CIRCLE (X,Y),R,C,,,0.7
70 FOR I = 1 TO 5
80 CIRCLE (X,Y),R,C,,,I*.2:NEXT I
90 IF VERT THEN RETURN
100 CIRCLE (X,Y),R,C,,,1.3
110 CIRCLE (X,Y),R,C,,,1.9
120 CIRCLE (X,Y),R,C,,,3.6
130 CIRCLE (X,Y),R,C,,,9.8
140 IF VERT GOTO 60
150 RETURN
160 CLS:SCREEN 1:COLOR 0,1:KEY OFF:VERT
170 X=160:Y=100:C=1:R=50:GOSUB 50
180 X=30:Y=30:C=2:R=30:GOSUB 50
190 X=30:Y=169:GOSUB 50
200 X=289:Y=30:GOSUB 50
210 X=289:Y=169:GOSUB 50
220 LINE (30,30)-(289,169),1
230 LINE (30,169)-(289,30),1
240 LINE (30,169)-(289,30),1,B
```

“Figura 3. Programa en BASIC”

Este nuevo lenguaje tuvo bastante popularidad ya que por su forma sencilla y rápida de aprender tuvo un impacto significativo en los usuarios que lo utilizaron, asimismo, este tenía un interprete que necesitaba poca memoria.

Para este mismo año de **1964**, Seymour Papert se incorporó al MIT. Luego de tres años, Papert comenzó con el desarrollo de un nuevo lenguaje, el cual serviría para introducir al mundo

de la programación a los de menor edad, con esto surgió el lenguaje **LOGO**, el cual tuvo mayor reconocimiento luego de que Papert sacara su libro “Mindstorms: Children Computers and Powerful Ideas”, este tuvo un buen recibimiento por centros educativos, especialmente en enseñanza primaria y secundaria. En el periodo actual, **LOGO** ha dejado de enseñarse en primaria y secundaria ya que este lenguaje se presenta un poco difícil de aprender, ya que este cuenta con utilización continua de listas, procedimientos de recursión, los cuales no son sencillos de manejar.



“Figura 4. LOGO PROGRAM”

En **1972** se dio a conocer el lenguaje **PROLOG** (PROgramation LOGique), el cual es un tipo de lenguaje que es declarativo, esto significa que no está basado en ordenes sino en declaraciones, más en detalle, PROLOG le proporciona al ordenador una serie de conocimientos sobre un tema, con un conjunto de reglas y el programa nos responderá cada una de las preguntas que deseemos hacerle sobre el tema siempre y cuando las respuestas puedan deducirse del

conocimiento proporcionado, PROLOG no está destinado al cálculo científico. Su aplicación en el campo de la inteligencia artificial, definiendo objetos y estableciendo relaciones, permite resolver problemas lógicos, desarrollar sistemas expertos, investigar en la comprensión del lenguaje humano, etc.

```
% neutrality(+Matrix,+Exprs,-Exprs): the function  $\mathcal{N}(X)$ 
neutrality(AttM, X, Y) :-
    mv_mult(AttM, X, Z),      %  $\mathcal{R}^+(X)$ 
    maplist(bnot, Z, Y).

% innocuousity(+Matrix,+Exprs,-Exprs): the function  $\mathcal{I}(X)$ 
innocuousity(AttM, X, Y) :-
    transpose(AttM, AttM_t), % transpose operation
    mv_mult(AttM_t, X, Z).   %  $\mathcal{R}^-(X)$ 
    maplist(bnot, Z, Y).

% defense(+Matrix,+Exprs,-Exprs): the function  $\mathcal{F}(X)$ 
defense(AttM, X, Y) :-
```

“Figura 5. PROLOG PROGRAM”

En los laboratorios de Bell, New Jersey trabajaron dos investigadores, Kenneth Thompson y Dennis Ritchie, los creadores del sistema operativo **UNIX**. En **1970**, Thompson desarrolló un lenguaje experimental al que llamó “B”. Pasando dos años, Ritchie se basó en B para desarrollar un nuevo lenguaje de propósito general el cual llamó “C”. En el que C, no depende de la arquitectura del hardware, C es uno de los lenguajes más portables del mercado.

Luego de esto, a principios de los 80, Bjarne Stroustrup diseñó una ampliación de C y, en **1984**, la convirtió en un compilador que fue llamado “C++”, el cual está enfocado a la programación orientada a objetos.


```

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
File Edit Search Run Compile Debug Project
NONAME00.CPP
#include<stdio.h>
int main()
{
printf("\n Welcome to JMD Tutorial");
return 0; _
}

```

“Figura 6. C PROGRAM”

A principios de los 70 se empezó con el desarrollo del lenguaje de programación **PASCAL**, por parte del suizo Niklaus Wirth, este buscaba la creación de un lenguaje que fuera fácil, pero a la vez potente y que siguiera las pautas estructurales. **PASCAL** es el lenguaje más sencillo que posibilita el acceso a la informática teórica, como lo es la descomposición modular, recursividad, punteros y más. El lenguaje fue sino hasta el año de 1980 que sufrió la primera formalización y tuvo su estandarización en 1983.

```


Borland Pascal 7.0
File Edit Search Run Compile Debug Tools Options Window Help
SOL4.PAS
var F,L:real;
i,j,n:integer;
x:array[1..10] of real;
y:array[1..10] of real;

begin
write('n=');readln(n);
FOR i:=1 TO n DO
begin
write('x[',i,']=');readln(x[i]);
write('y[',i,']=');readln(y[i]);
end;
begin
write('x[',n+1,']=');readln(x[n+1]);
end;
y[n+1]:=0;
F:=0;
FOR j:=1 TO n DO begin
L:=1;
FOR i:=1 TO n DO
begin
IF i<>j THEN
begin
L:=L*(x[n+1]-x[i])/(x[j]-x[i]);
end;
end;
y[n+1]:=y[n+1]+y[j]*L;end;
writeln('y[',n+1,']=',y[n+1]:1:0);
FOR i:=1 TO n DO
begin
writeln('x[',i,']=',x[i]:10:10,' y[',i,']=',y[i]:10:10);
end;
begin
writeln('x[',n+1,']=',x[n+1]:10:10,' y[',n+1,']=',y[n+1]:10:10);
end;
end;

```

“Figura 7. PASCAL PROGRAM”

En **1990** se desarrolló por parte de James Gosling de Sun Microsystems basándose en C y C++ el lenguaje de Java. El objetivo de este lenguaje era crear un interfaz atractivo e intuitivo para la electrónica de consumo. En agosto de **1995**, ya con el nombre de JAVA, se presentó en sociedad.



```
C:\Windows\system32\cmd.exe
29/11/2012 07:25 PM <DIR> .
29/11/2012 07:25 PM <DIR> ..
29/11/2012 07:25 PM 116 hello.java
1 File(s) 116 bytes
2 Dir(s) 135,240,376,000 bytes free

C:\Java Program>java hello.java

C:\Java Program>dir
Volume in drive C is OS
Volume Serial Number is F49D-E1A9

Directory of C:\Java Program

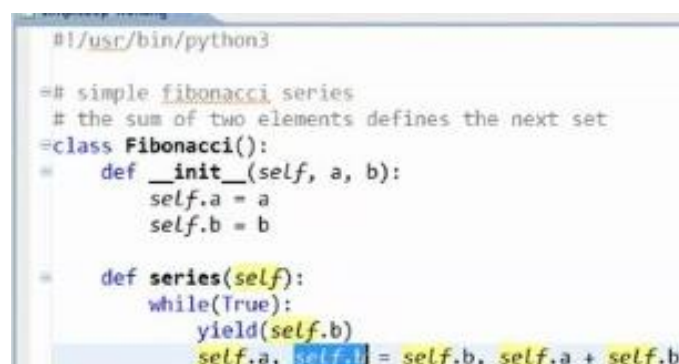
29/11/2012 07:28 PM <DIR> .
29/11/2012 07:28 PM <DIR> ..
29/11/2012 07:25 PM 116 hello.java
29/11/2012 07:28 PM 419 index.class
2 File(s) 535 bytes
2 Dir(s) 135,240,310,272 bytes free

C:\Java Program>java index
Hello Voutuh!!!

C:\Java Program>
```

“Figura 8. Java PROGRAM”

Luego, en **1991** se lanza la versión 0.9.0 de Python en alt.sources. Para 1994 Python actualiza a la versión 1.0 y en el **2000**: El equipo de desarrolladores de Python dejan CNRI y se mudan a BeOpen.com, desde donde lanzan la versión 2.0. El lenguaje Python es interpretado y es multiparadigma.



```
#!/usr/bin/python3

# simple fibonacci series
# the sum of two elements defines the next set

class Fibonacci():
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def series(self):
        while(True):
            yield(self.b)
            self.a, self.b = self.b, self.a + self.b
```

“Figura 9. Python PROGRAM”

Conceptos Importantes.

Abstracción

- La abstracción es el proceso de ocultar los detalles complejos de un sistema, permitiendo a los usuarios interactuar con funcionalidades de alto nivel sin necesidad de conocer los detalles de implementación subyacentes. En programación, la abstracción permite a los desarrolladores trabajar con objetos y conceptos en lugar de operaciones de bajo nivel.

Eficiencia

- La eficiencia en programación se refiere a la capacidad de un algoritmo o programa para utilizar los recursos de manera óptima, como el tiempo de ejecución y la memoria. Un programa eficiente realiza su tarea en el menor tiempo posible y con el menor uso de recursos.

Portabilidad

- La portabilidad es la capacidad de un programa para ser ejecutado en diferentes entornos de hardware y software sin modificación significativa. Un software portátil puede ser trasladado de un sistema a otro con facilidad.

Concisión Notacional

- La concisión notacional se refiere a la capacidad de un lenguaje de programación para expresar operaciones complejas en una forma breve y clara. Un lenguaje con alta concisión notacional permite a los desarrolladores escribir menos código para realizar las mismas tareas.

Seguridad

- La seguridad en programación se refiere a la capacidad de un lenguaje o sistema para proteger contra errores, fallos y ataques maliciosos. Esto incluye características como el manejo seguro de memoria, la prevención de desbordamientos de buffer y la protección de datos sensibles.

Ortogonalidad

- La ortogonalidad en programación se refiere a la propiedad de un sistema en el que las operaciones son independientes unas de otras. Esto significa que los cambios en una parte del sistema no afectan a otras partes, facilitando la comprensión y el mantenimiento del código.

Extensibilidad

- La extensibilidad es la capacidad de un sistema de software para ser extendido o modificado con nuevas funcionalidades sin afectar su estructura existente. Un sistema extensible permite agregar nuevas características de manera sencilla y modular.

Entorno

- En programación, el entorno se refiere al conjunto de herramientas, bibliotecas y componentes que se utilizan para desarrollar, probar y ejecutar software. Esto incluye el sistema operativo, el entorno de desarrollo integrado (IDE) y otros recursos necesarios para el desarrollo de software

Referencias.

"Abstraction is the process of reducing complexity by focusing on the main idea and ignoring specific details." - Sebesta, R. W. (2015). *Concepts of Programming Languages* (11th ed.). Pearson.

"Efficiency is a measure of how well time or effort is used for the intended task or purpose." - Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.

"Portability is the usability of the same software in different environments." - Sommerville, I. (2011). *Software Engineering* (9th ed.). Addison-Wesley.

"Conciseness in programming refers to the quality of expressing much in few words or a brief form." - Sebesta, R. W. (2015). *Concepts of Programming Languages* (11th ed.). Pearson.

"Security in programming involves measures to protect against unauthorized access, attacks, and data corruption." - Bishop, M. (2002). *Computer Security: Art and Science*. Addison-Wesley.

"Orthogonality refers to the concept of making operations independent of each other, simplifying design and use." - Sebesta, R. W. (2015). *Concepts of Programming Languages* (11th ed.). Pearson.

"Extensibility is the ability of a system to accommodate new features or changes with minimal impact on the existing system." - Sommerville, I. (2011). *Software Engineering* (9th ed.). Addison-Wesley.