

**Primera Tarea Introducción a tecnologías de la información**

**Investigación de lenguajes y paradigmas de programación**

**Estudiante: Gabriel León Castro**

# Pseudocódigo

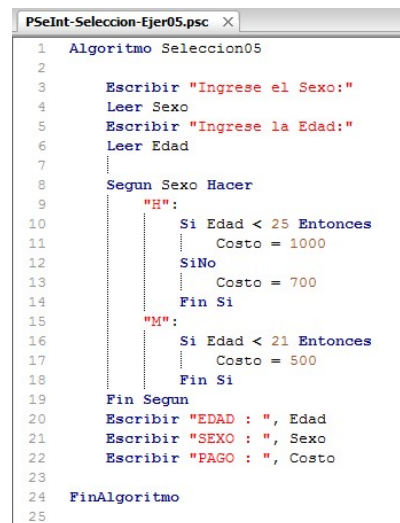
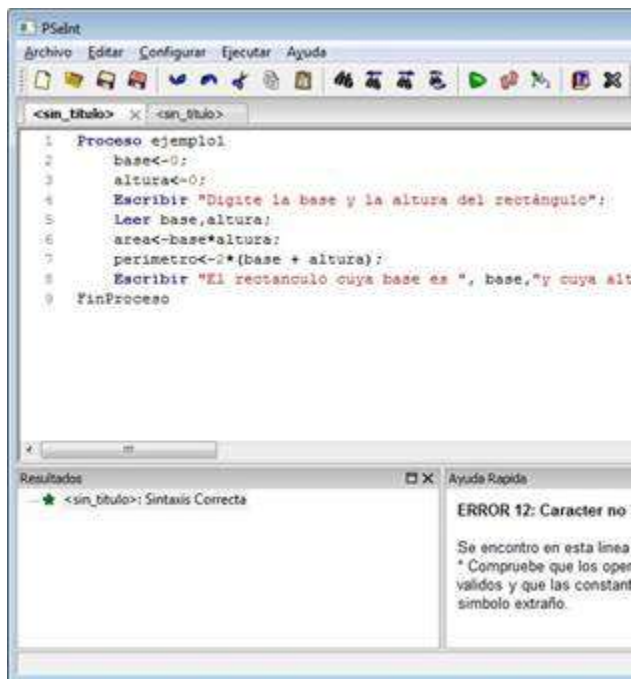
## Historia y ¿por qué se crea?

El Pseudocódigo fue empezado a crear en diciembre de 1950 por Patrick Naughton, ingeniero de Sun Microsystems, junto a otros compañeros de él. Y fue creada con el propósito de ser leído por un humano y no por una máquina y con independencia de cualquier otro lenguaje de programación. Normalmente, el pseudocódigo omite detalles que no son esenciales para la comprensión humana del algoritmo, tales como declaraciones de variables, código específico del sistema y algunas subrutinas.

## ¿Para qué sirve?

Su principal función es el representar un algoritmo de la forma más detallada posible. También se usa mucho en el ambiente didáctico para enseñar a las personas a programar debido a que la sintaxis de este es más sencilla de comprender que la de un lenguaje de programación.

## Ejemplos de la investigación



## Ejemplo propio

```
Algoritmo Ordenamiento_burbuja
  Definir temp, vector Como Entero
  Dimension vector(5)
  vector(1) = 5
  vector(2) = 3
  vector(3) = 1
  vector(4) = 4
  vector(5) = 2

  para x = 2 hasta 5 hacer
    para a=1 hasta 4 hacer
      si vector(a) > vector(a+1)
        temp = vector(a)
        vector(a) = vector(a+1)
        vector(a+1) = temp
      FinSi
    FinPara
  FinPara

  para z = 1 hasta 5 Hacer
    Escribir vector(z)
  FinPara
FinAlgoritmo
```

# Lenguaje compilado

## Historia

Al comienzo de la década de los 50, John Backus estaba trabajando con SSEC, uno de los primeros ordenadores de IBM, y desarrolló el programa SPEEDCODING. Tomando éste como base, se emprendió, en 1954, la creación de un lenguaje para añadirle más prestaciones al modelo IBM 704, que iba a salir pronto al mercado y en 1956 se terminó el compilador Fortran (FORMula TRANslator) y se incluyó en la IBM 704.

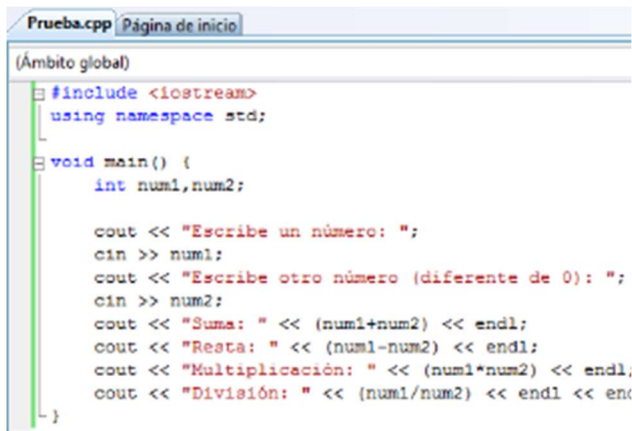
## ¿Por qué se crea?

Se crea con el fin de facilitar al usuario la programación, ya que es más complicado programar en lenguaje binario que con un lenguaje de programación compilado.

## ¿Para qué sirve?

Los lenguajes compilados sirven para convertir código fuente en lenguaje binario y que así el CPU pueda comprenderlo.

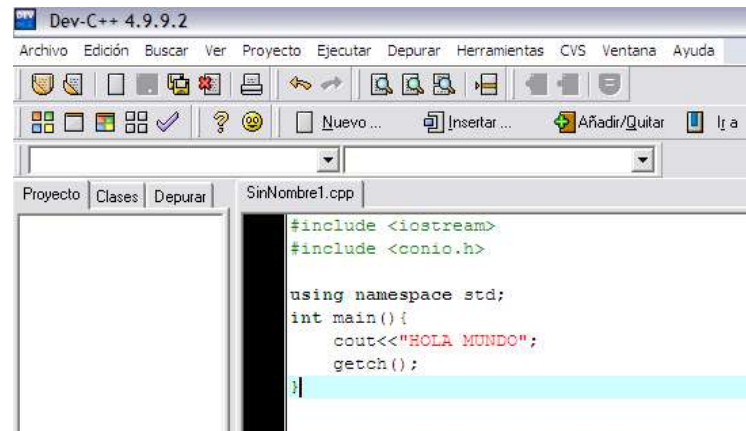
## Ejemplos de la investigación



```
#include <iostream>
using namespace std;

void main() {
    int num1, num2;

    cout << "Escribe un número: ";
    cin >> num1;
    cout << "Escribe otro número (diferente de 0): ";
    cin >> num2;
    cout << "Suma: " << (num1+num2) << endl;
    cout << "Resta: " << (num1-num2) << endl;
    cout << "Multiplicación: " << (num1*num2) << endl;
    cout << "División: " << (num1/num2) << endl << endl;
}
```



```
Dev-C++ 4.9.9.2
Archivo Edición Buscar Ver Proyecto Ejecutar Depurar Herramientas CVS Ventana Ayuda

#include <iostream>
#include <conio.h>

using namespace std;
int main() {
    cout << "HOLA MUNDO";
    getch();
}
```

## Ejemplo propio

```
#include <iostream>

using namespace std;

int main() {

    int A[] = {6,5,9,3,0,1,8,7,4,2};

    for(int i = 0; i < 10; i++){

        for(int e = 0; e < 9; e++){

            if(A[e] > A[e+1]){

                int aux = A[e];
                A[e] = A[e+1];
                A[e+1] = aux;

            }

        }

    }

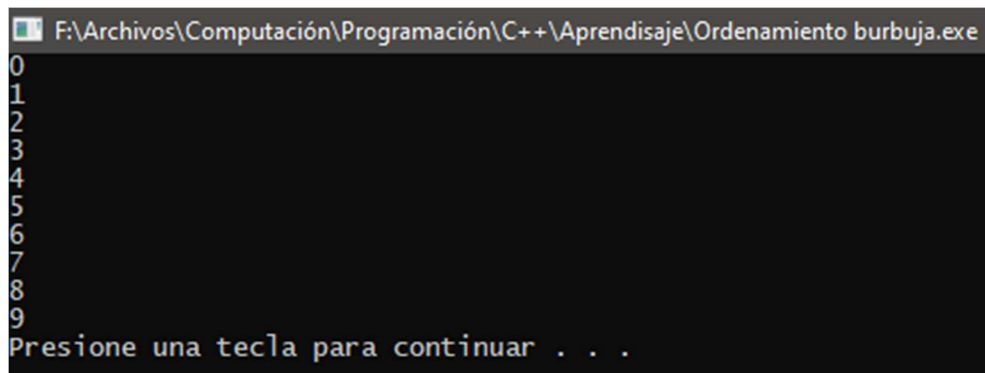
    for(int e = 0; e < 10; e++){

        cout << A[e] << "  \n";

    }

    system ("pause");
    return 0;

}
```



```
F:\Archivos\Computación\Programación\C++\Aprendisaje\Ordenamiento burbuja.exe
0
1
2
3
4
5
6
7
8
9
Presione una tecla para continuar . . .
```

# Programación orientada a objetos (Poo)

## Historia

Surge en Noruega en 1967 con un lenguaje llamado simulada 67, desarrollado por krinsten Nygaard y Ole-johan Dahl, en el centro de cálculo noruego Simula 67 introdujo por primera vez los conceptos de clases, corutinas y subclases (conceptos muy similares a los lenguajes orientados a objetos de hoy en día).

## ¿Por qué se crea?

El problema consistía en cómo adaptar el software a nuevos requerimientos imposibles de haber sido planificados inicialmente.

En los 70's científicos del centro de investigación en Palo Alto Xerox (Xerox park) inventaron el lenguaje Small talk que dio respuesta al problema y fue el primer lenguaje Orientado a Objetos puro.

## Ejemplos de la investigación

```
class CRONOMETRO{  
  
    struct time tm; // Variable que coge la hora  
  
    int andando;  
  
    void dif_tm(time tr, time ts, time *dif);  
  
    public:  
  
    void Arranca(void);  
  
    void Muestra(void);  
  
    void Para(void);  
  
};
```

```
public class UniversityStudent {  
    int id;  
    String name;  
    String gender;  
    String university;  
    String career;  
    int numSubjects;  
    public UniversityStudent(int id, String name, String gender,  
        String university, String career, int numSubjects) {  
        this.id = id;  
        this.name = name;  
        this.gender = gender;  
        this.university = university;  
        this.career = career;  
        this.numSubjects = numSubjects;  
    }  
    void inscribeSubjects() {  
        // TODO: implement  
    }  
    void cancelSubjects() {  
        // TODO: implement  
    }  
    void consultRatings() {  
        // TODO: implement  
    }  
}
```

## Ejemplo propio

```
#include <iostream>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

class Persona{
private:
    string nombre;
    int edad;

public:
    Persona();

    void leer();
    void correr();
};

Persona::Persona(int _edad,string _nombre){
    edad =_edad;
    nombre =_nombre;
}

void Persona::leer() {

    cout<<"soy"<<nombre<<" y  estoy leyendo un libro"<<endl;

}

void Persona::correr() {
    cout<<"Soy"<nombre<<" y estoy corriendo"<<endl;
}

int main() {
    Persona P1=(20,"Alejandro");
    Persona P2=(19,"Maria");
    Persona p3=(21,"Juan");
    p1.leer();
    p2.correr();

    p3.leer();
    p3.correr();

    getch();
    return 0;
}
```

# Programación funcional

## Historia

El paradigma funcional se empezó a desarrollar por el matemático John McCarthy en 1956, para programar los primeros proyectos de inteligencia artificial sobre un computador IBM 704 durante su desarrollo este crea el lenguaje de programación Lisp en 1958.

## ¿Por qué se crea?

Se crea con el fin de investigar la definición de función, la aplicación de las funciones y la recursión.

## Ejemplos de la investigación

```
1  # include < stdio .h >
2
3  int contador = 0;
4
5  int aumentar_contador ( int incremento ) {
6      contador += incremento ;
7      return contador ;
8  }
9  void mostrar_contador ( void ) {
10     printf ( " El valor del contador es : %d \n " , contador ) ;
11 }
12 int main ( void ) {
13     mostrar_contador () ;
14     aumentar_contador (5) ;
15     mostrar_contador () ;
16     aumentar_contador (10) ;
17     mostrar_contador () ;
18     return 0;
19 }
```

```
function saludar () {
    console.log('Mi primera función en Javascript');
}

saludar(); // Llamamos a la función

// → Mi primera función en Javascript
```



## Ejemplo propio

```
const bubbleSort = arr => {  
  const l = arr.length;  
  for (let i = 0; i < l; i++) {  
    for (let j = 0; j < l - 1 - i; j++) {  
      if ( arr[ j ] > arr[ j + 1 ] ) {  
        [ arr[ j ], arr[ j + 1 ] ] = [ arr[ j + 1 ], arr[ j ] ];  
      }  
    }  
  }  
  
  return arr;  
};  
  
const arr = [10, 4, 40, 32, 67, 12, 43, 31, 65, 1];  
const result = bubbleSort(arr);  
  
result;
```

```
Array [  
  1,  
  4,  
  10,  
  12,  
  31,  
  32,  
  40,  
  43,  
  65,  
  67,  
]
```

# Programación lógica

## Historia

Desde su aparición, las computadoras se han programado utilizando lenguajes muy cercanos al de la propia máquina: operaciones aritméticas simples, instrucciones de acceso a memoria, etc. Un programa escrito de esta manera puede ser muy difícil de comprender por el un ser humano, incluso uno entrenado. Actualmente, estos lenguajes pertenecientes al paradigma de la Programación imperativa han evolucionado de manera que ya no son tan crípticos. Sin embargo, la lógica matemática es la manera más sencilla de expresar formalmente problemas complejos y de resolverlos mediante la aplicación de reglas, hipótesis y teoremas, para el intelecto humano. De ahí que el concepto de "programación lógica".

## Ejemplos de la investigación

```
es_padre(terach, abraham).
es_padre(terach, nachor).
es_padre(terach, haran).
es_padre(abraham, isaac).
es_padre(haran, lot).
es_padre(haran, milcah).
es_padre(haran, yiscah).

es_madre(sarah, isaac).

es_hombre(terach).
es_hombre(abraham).
es_hombre(nachor).
es_hombre(haran).
es_hombre(isaac).
es_hombre(lot).

es_mujer(sarah).
es_mujer(milcah).
es_mujer(yiscah).

es_hijo(X,Y):- es_padre(Y,X), es_hombre(X).

es_hija(X,Y):- es_padre(Y,X), es_mujer(X).

es_abuelo(X,Z):- es_padre(X,Y), es_padre(Y,Z).
```

```
>>> from kanren import Relation, facts
>>> parent = Relation()
>>> facts(parent, ("Homer", "Bart"),
...              ("Marge", "Bart"),
...              ("Homer", "Lisa"),
...              ("Marge", "Lisa"),
...              ("Homer", "Maggie"),
...              ("Marge", "Maggie"),
...              ("Abe", "Homer"))
```

## Lenguaje ensamblador

### Historia

Se creó en 1950 por Mauricio V. Wilkes de la universidad de Cambridge.

### ¿Por qué se crea?

Se crea con el fin de facilitar la tarea de programar las computadoras debido a que con el lenguaje de ensamblador, se programaba con un conjunto de instrucciones mientras que sin este se hacía con códigos binarios y esto resultaba mucho más complicado para los programadores.

## Ejemplos de la investigación

```
1 section .data
2     msg db "Hello World!", 0xA, 0xD
3     len equ $ - msg
4 section .text
5     global _start
6 _start:
7     mov eax, 4
8     mov ebx, 1
9     mov edx, msg
10    mov ecx, len
11    int 0x80
12
13    mov eax, 1
14    int 0x80
```

```
.include "macros.s"
N = 5
.data
RES: .word 0
V: .word -1, -2, -3, -4, -5
.text
main:
MOVI R1, 0 ; Contador
MOVI R2, N ; Límite
MOVI R3, 0 ; Suma parcial
$MOVEI R4, V ; @ inicial de V
bucle:
CMPLT R5, R1, R2
BZ R5, fibucle ; Acaba en 5
ADD R5, R1, R1 ; Cada dato
;ocupa 2 bytes
ADD R5, R4, R5
LD R6, 0(R5) ; Leemos un dato
ADD R3, R3, R6 ; Acumulamos
ADDI R1, R1, 1 ; Actualizamos
; el índice
BNZ R1, bucle ; Salta siempre
fibucle:
$MOVEI R4, RES; Resultado
ST 0(R4), R3
HALT ; Acaba
```

## Ejemplo propio

ordenBurbuja:

```
mov ebx,0      ;posición
mov ecx,0      ;cambio
mov eax,0      ;i
```

bucleExterno: ; for (i=0; i<100; i++)

```
cmp eax,99
je finBuble
```

```
inc eax
mov ebx,eax
dec eax
```

bucleInterno:

```
mov edx,0
cmp ebx,100
je bucleExterno
```

```
mov edx,dword[vector+eax*4]
cmp dword[vector+ebx*4],edx
jng incrementaPosicion
```

```
mov ecx,[vector+ebx*4]
mov edx,dword[vector+eax*4]
mov dword[vector+ebx*4],edx
mov dword[vector+eax*4],ecx
```

incrementaPosicion:

```
inc ebx
```

```
cmp ebx,100
je incrementa_i
jmp bucleInterno
```

incrementa\_i:

```
inc eax
jmp bucleExterno
```

finBuble:

```
ret
```

# Lenguaje interpretado

## Historia

En 1980 se crea Smalltalk, el cual fue el primer lenguaje de programación interpretado. Fue diseñado para ser interpretado en tiempo de ejecución y permite a objetos genéricos interactuar dinámicamente entre sí.

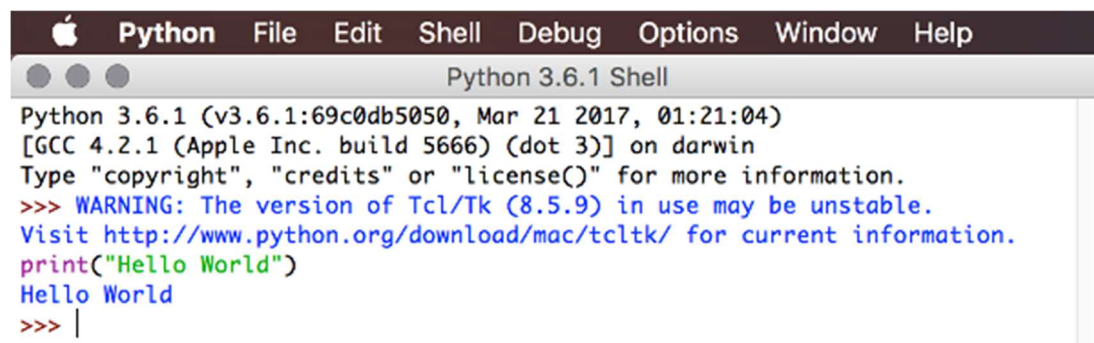
Inicialmente, los lenguajes interpretados eran compilados línea por línea, es decir, cada línea era compilada a medida que estaba a punto de ser ejecutada, y si un bucle o una subrutina hicieran que ciertas líneas se ejecutaran múltiples veces, serían recompiladas repetidamente. Esto ha llegado a ser mucho menos común. La mayoría de los lenguajes interpretados usan una representación intermedia, que combina tanto la compilación como la interpretación. En este caso, un compilador puede producir el código byte o el código enhebrado, que entonces es ejecutado por un intérprete de código byte.

## ¿Por qué se crea?

Se crea como alternativa a los lenguajes compilados, ya que estos son más flexibles, facilidad en la depuración y el tamaño del programa es más pequeño.

## Ejemplos de la investigación

```
1  import random
2
3  def buscarElemento(lista, elemento):
4      for i in range(0, len(lista)):
5          if(lista[i] == elemento):
6              return i
7
8  def imprimirLista(lista, nombre):
9      for i in range(0, len(lista)):
10         print nombre + "[" + str(i) + "]" = " + str(lista[i])
11
12 def leerLista():
13     lista=[]
14
15     i=0
16     while i < 10:
17         lista.append(int(random.randint(0, 10)))
18         i=i+1
19     return lista
20
21 A=leerLista()
22 imprimirLista(A, "A")
23 cn=int(raw_input("Numero a buscar: "))
24 print "A[" + str(buscarElemento(A, cn)) + "]"
```



```
Python 3.6.1 (v3.6.1:69c0db5050, Mar 21 2017, 01:21:04)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable.
Visit http://www.python.org/download/mac/tcltk/ for current information.
print("Hello World")
Hello World
>>> |
```

## Ejemplo propio

```
def burbuja(A):  
    for i in range(1, len(A)):  
        for j in range(0, len(A)-i):  
            if(A[j+1] < A[j]):  
                aux=A[j];  
                A[j]=A[j+1];  
                A[j+1]=aux;  
  
    print (A);
```

```
A=[6,5,3,1,8,7,2,4];  
print (A)  
burbuja(A);
```

 Python 3.8.1 Shell

File Edit Shell Debug Options Window Help

Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.

>>>

===== RESTART: C:\Users\Gaboelc\Desktop\Python.py =====

[6, 5, 3, 1, 8, 7, 2, 4]

[1, 2, 3, 4, 5, 6, 7, 8]

>>>