|  |  |  |  |
| --- | --- | --- | --- |
| **ACTIVDAD #1** | | | |
| **PARADIGMAS** | **DESCRIPCION** | **SIMBOLO O NOMBRE** | **EJEMPLO** |
| **IMPERATIVO** | sencillez y cercanía a la arquitectura de las computadoras convencionales, memoria principal y la unidad central de proceso. | **C:**  memoria principal y la unidad central de proceso. | **.  Inicio**  2.  Mostrar “Introduzca el diámetro, en metros” : Pedir D  3.  Mostrar “Introduzca la altura, en metros” : Pedir H  4.  R = D/2 : Pi = 3,141593  5.  V = Pi \* (R ^ 2) \* H  6.  Mostrar “El volumen del cilindro es de”, V, “metros cúbicos”  **7.  Fin [aprenderaprogramar.com]** |
| **FUNCIONAL** | funciones matemáticas puras, características más significativas del paradigma funcional. | **Haskell :** | filterPositivos  :: [Integer] -> [Integer]  filterPositivos [] = []  filterPositivos (x:xs) | x>=0 = x : filterPositivos xs                          |otherwise = filterPositivos xs |
| **LOGICO** | el programa nos retorna un valor verdadero, consiste en realizar una pregunta, para obtener por inferencia los resultados, los cuales deduce a partir de la relación entre las premisas y las reglas. | **Prolog** | ?- esHombre(juan).  yes ?- esHombre(pedro).  yes ?- esHombre(juanCarlos).  no ?- esMamifero(leon).  No  gusta(alguien\_maria,maria). /\* ∃x (gusta(x,maria)) \*/ gusta(alguien\_paula,paula). /\* ∃x (gusta(x,paula)) \*/ |
| **ESTRUCTURADA** | La programación estructurada es un paradigma de programación basado en utilizar [funciones](https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion5/funciones.html#python-funciones) o subrutinas, y únicamente tres estructuras de control | [if](https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion4/condicional_if.html" \l "python-condi-if)  , [while](https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion4/bucle_while.html" \l "python-bucle-while) y [for](https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion4/bucle_for.html" \l "python-bucle-for).  **PHYTON** | while ( condición\_a\_evaluar ) {  acción/es a realizar ;  acción/es a realizar ;  };  {  // Crear variables auxiliares  int contador=0;  // Inicializar el contador  contador=1;  // repetir mientras que el contador tenga  // un valor inferior a 10  while (contador<11) {  // visualizar el valor de contador  printf("\n%d",contador);  // incrementar contador en una unidad  contador=contador+1;  }; |
| **PROCEDIMENTAL** | Esta técnica consiste en basarse de un número muy bajo de expresiones repetidas, englobarlas todas en un procedimiento o función y llamarlo cada vez que tenga que ejecutarse. | **Basic**  **COBOL**  **Matlab**  **Pascal** | funcion anterior\_posterior( numero ) {  out(numero-1);  out(numero);  out(numero+1);  }  for( i = 0; i < tamaño(lista); i++) {  anterior\_posterior( lista[i] );  } |
| **MODULAR** | El método más conocido para resolver un problema es dividirlo en problemas más pequeños, llamados subproblemas metodología del divide y vencerás o programación top-down. | **Ada, PL/I, Erlang, Pascal, Algol, COBOL** |  |
| **DECLARATIVA** | al escribir el código, no es necesario determinar el procedimiento según el cual se alcanza el resultado | **Prolog,Lisp,Haskell,**  **Miranda,Erlang,SQL (en un sentido amplio)** | $nombres = array\_values($listaparticipantes); |