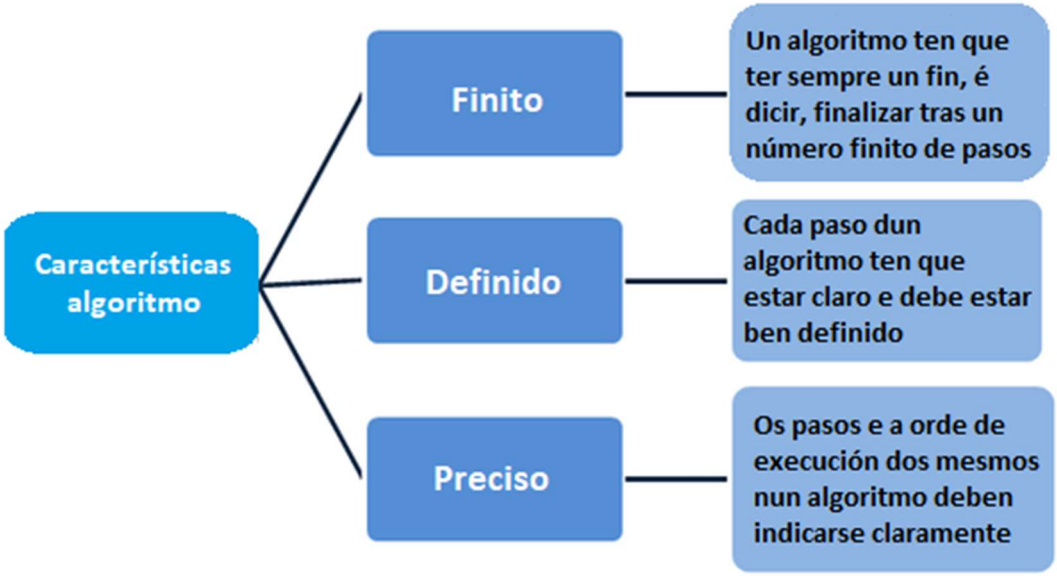
INTRODUCIÓN Á PROGRAMACIÓN

Algoritmos e Programas

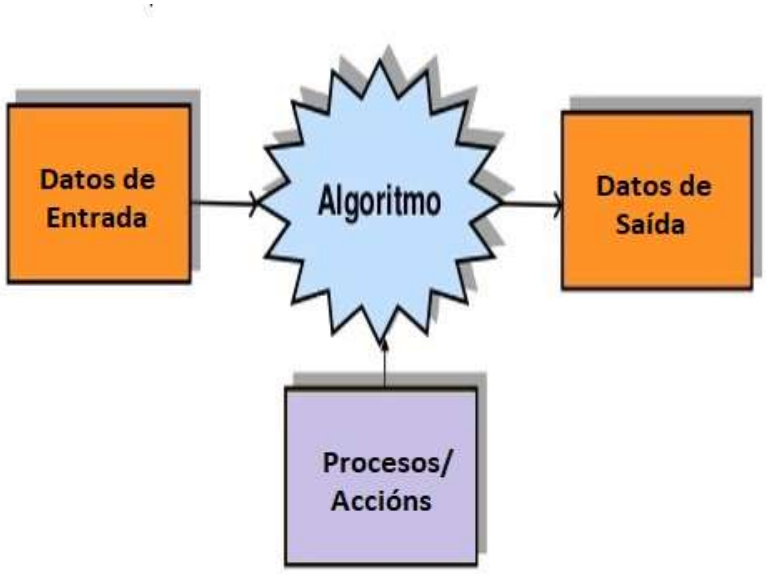
Un **algoritmo** é unha secuencia ordenada de operacións tal que a súa realización resolve un determinado problema.

As características fundamentais que debe cumprir todo algoritmo son:

- **Un algoritmo debe ser preciso** e indicar a orde de realización de cada paso.
- **Un algoritmo debe estar definido**: Se seguimos o algoritmo para a mesma entrada varias veces os resultados obtidos deben ser os mesmos.
- **Un algoritmo debe ser finito**: O algoritmo debe contar cun número finito de pasos.



Na definición dun algoritmo debemos distinguir tres partes: a entrada de datos, o proceso dos mesmos e a devolución de resultados.



Na nosa vida diaria facemos uso de algoritmos continuamente para unha multitude de cousas, desde conducir a operacións matemáticas de uso común. Gran parte da nosa aprendizaxe realizámola mediante a memorización dos algoritmos que nos permiten resolver tipos de problema concretos (sumar, restar, multiplicar, dividir, sacar raíces, elaborar unha tortilla de patacas, etc). Se nos fixamos ben, en todos estes casos aprendemos unha serie de pasos que aplicamos mecanicamente para chegar ao resultado.

Exemplo 1:

Algoritmo que permita obter un refresco dunha máquina automática expendedora de bebidas embotelladas.

1. Inicio
2. Localizar a bebida desexada e mirar o seu custo
3. Introducir no sitio correspondente unha cantidade maior ou igual ao custo, preferiblemente igual
4. Pulsar o botón correspondente coa bebida
5. Si a máquina non ten produto
    a) No panel visualizarase produto esgotado
    b) Retornará o importe introducido

   SiNon

    Si a cantidade introducida é menor que o custo
        a) No panel visualizarase importe insuficiente
        b) Retornará o importe introducido

    SiNon
        a) Na bandexa de saída sairá o produto seleccionado
        Si a cantidade é maior que o custo
            a) Retornarase a diferenza de custo na bandexa de cambio
6. Fin

Os ordenadores basicamente non son máis que un conxunto de circuítos que son capaces de levar a cabo operacións moi básicas (suma, resta, multiplicación, división, comparación, salto, copia de información) de modo moi rápido. A programación non é máis que o procedemento polo que indicamos as operacións a realizar para resolver un problema. Estas operacións deben indicarse de modo moi preciso, xa que un ordenador levará a cabo as instrucións indicadas de xeito literal, sen ningunha interpretación pola súa parte, isto contrasta co xeito en que podemos indicar os algoritmos a outros seres humanos, na maior parte dos casos cunha descrición non demasiado precisa é suficiente.

Para poder levar a cabo as instrucións dos algoritmos nos ordenadores de uso común, estas deben estar codificadas en **formato binario**, xa que é moi simple construír sistemas electrónicos que realicen as operacións básicas neste formato. O formato binario ou ***base 2*** emprega 2 símbolos para codificar información. Como contraste, para representar números habitualmente utilizamos o formato **base 10** que emprega 10 símbolos.

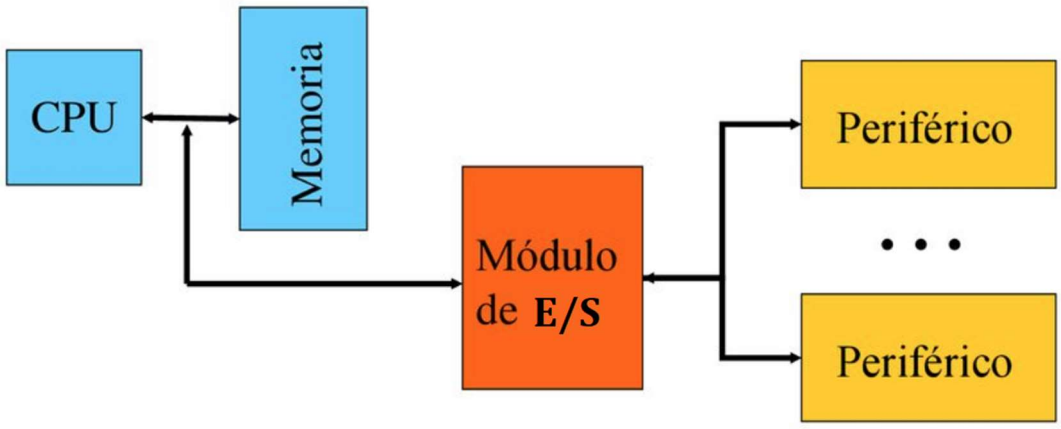
Para poder representar información non numérica, é necesario establecer un código que faga corresponder certas combinacións binarias con símbolos concretos. No momento de descifrar a información é necesario saber si se trata de un número ou de un símbolo non numérico, para poder interpretalo do xeito correcto. O código para representar símbolos máis habitual historicamente é o código ASCII, aínda que hoxe se utiliza practicamente sempre o código UNICODE.

Exercicio 1

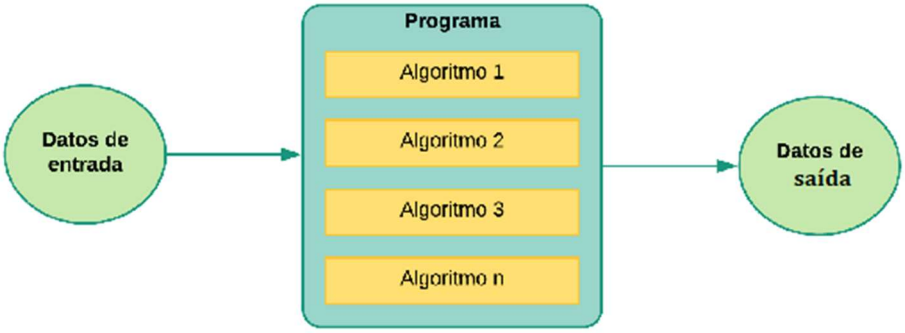
Describe un algoritmo para cociñar unha tortilla de patacas.

Os ordenadores modernos están construídos de xeito que dispoñen de:

- Unha **memoria** capaz de almacenar tanto os datos a procesar como as instrucións dos algoritmos.
- Unha serie de **dispositivos que nos permiten introducir datos** (que serán codificados a secuencias binarias).
- Unha serie de **dispositivos que permiten visualizar os resultados** (convertendo as secuencias binarias representado os resultados nos caracteres que esteamos habituados a ler).



Un **programa** implementa un ou varios algoritmos especificados mediante unha **linguaxe de programación**.

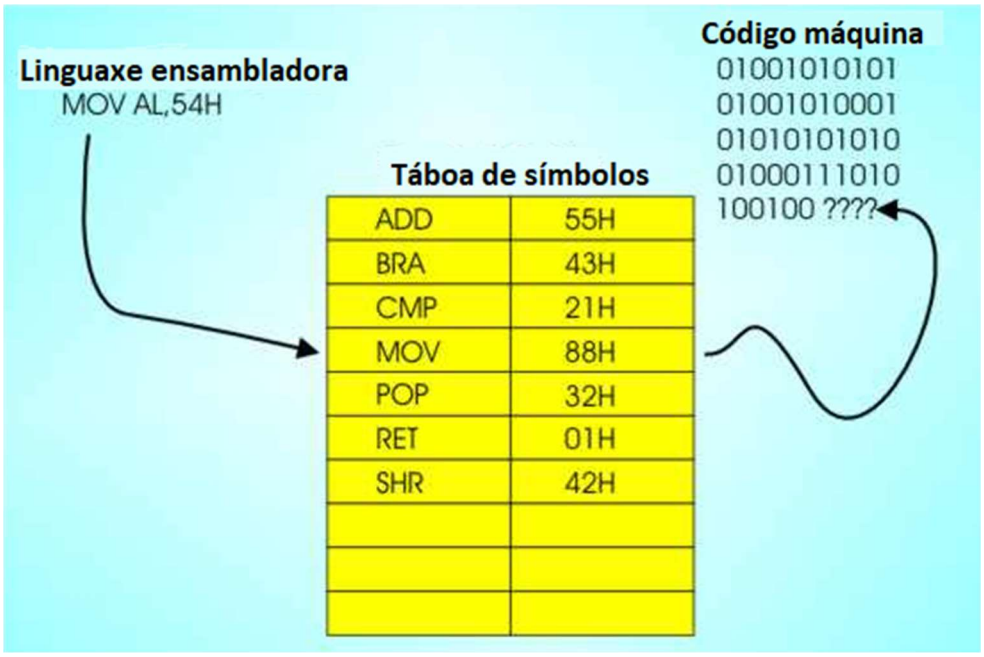


Inicialmente, os algoritmos programábanse introducindo no ordenador directamente o código binario das instrucións que desexábamos levar a cabo (do xogo de instrucións que era capaz de executar o ordenador). Este código recibe o nome de **código máquina**, xa que é o código que executa realmente a máquina.

A programación directamente en código máquina era moi tediosa, longa e propensa a erros polo que se decidiu utilizar **nemónicos** para cada unha das instrucións das que dispoñía cada ordenador e logo escribir os programas utilizando eses nomes. Para que o ordenador puidera levar a cabo as instrucións era necesario traducir eses nomes ao código binario correspondente. O proceso de tradución denominouse "ensamblado" e a os nomes e as regras empregadas "**Linguaxe Ensambladora**".

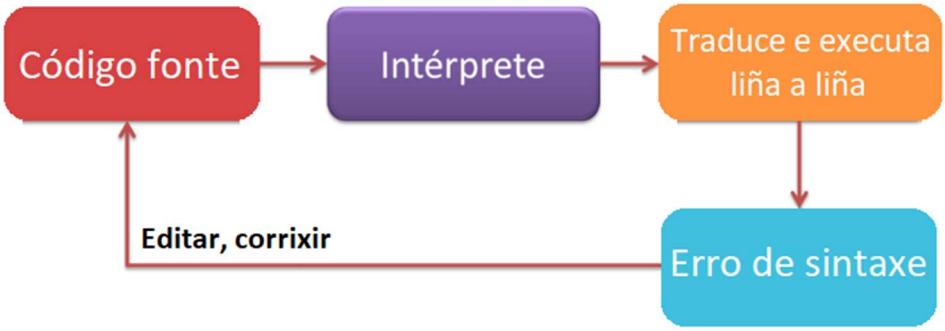
O **ensamblador** encárgase de traducir un ficheiro fonte escrito en linguaxe ensambladora a un ficheiro obxecto en código máquina. A linguaxe ensambladora é directamente traducible a código máquina. A linguaxe ensambladora depende do hardware.

Os nemónicos da linguaxe ensambladora almacénanse nunha táboa chamada **táboa de símbolos** que o programa ensamblador consulta para converter unha determinada instrución escrita en linguaxe ensambladora a código máquina.

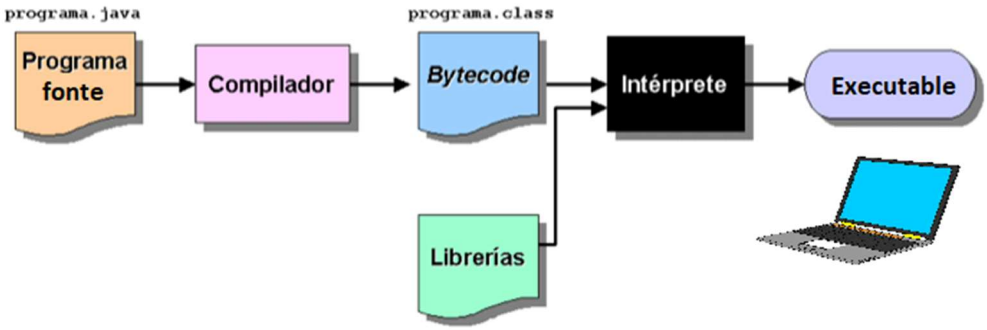


Os algoritmos escritos en linguaxe ensambladora eran moi longos de escribir e difíciles de entender, xa que as instrucións empregadas eran demasiado simples, polo que se desenvolveron linguaxes que permitían empregar estruturas máis complexas como a execución condicional, a estrutura repetitiva, o uso de **variables** ou o agrupamento de instrucións en **funcións**. Estas instrucións tamén era necesario traducilas ao código binario específico do ordenador, mediante un proceso denominado "**compilación**" realizado por un programa chamado "**Compilador**", específico da linguaxe que se utilizara.

Outro xeito de conseguir que o ordenador levara a cabo os programas realizados con este tipo de linguaxes era que en lugar de traducir todo o programa para producir un programa "binario", se traducía e executaba no mesmo momento, "interpretando" a linguaxe ao código máquina necesario sobre a marcha. Este proceso chámase "**interpretación**", e o programa encargado de facer este traballo "**Intérprete**".



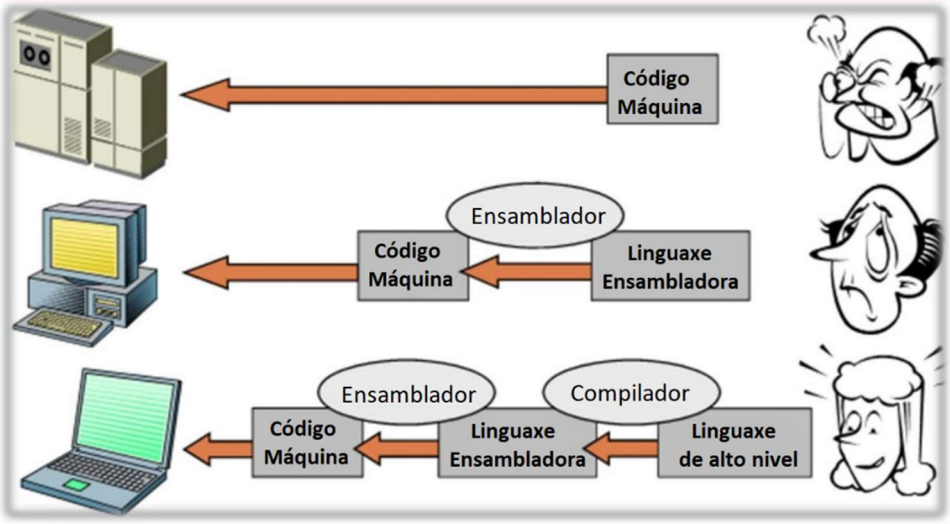
Un concepto intermedio entre o compilador e o intérprete popularizado pola linguaxe de programación Java é o da **máquina virtual**. Neste caso os programas escritos en linguaxe de alto nivel compílanse a un código intermedio en vez de a código máquina. Este código intermedio é executado por un intérprete coñecido como Máquina Virtual.



Unha aplicación é un conxunto de algoritmos codificados mediante unha linguaxe de programación, que, combinados entre si realizan unha actividade mais ampla. Como por exemplo un procesador de textos, un visor de vídeo.... etc. Estas aplicacións utilizan multitude de algoritmos necesarios para poder levar a cabo a súa actividade.

Exercicio 2

Indica as vantaxes e inconvenientes das linguaxes interpretadas fronte as compiladas.

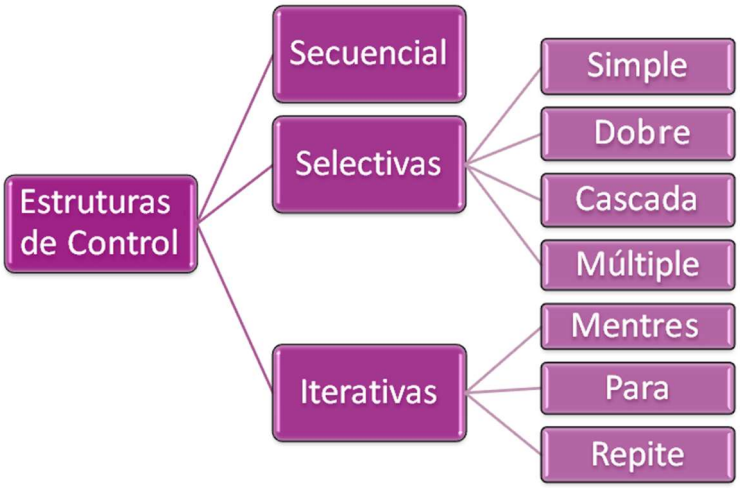


Ferramentas básicas para a creación de algoritmos

Calquera algoritmo se pode desenvolver empregando unicamente as seguintes estruturas:

- **Secuencia -** As instrucións se levan a cabo en secuencia, unha detrás de outra. Estas instrucións poden ser:
  - expresións aritmético-lóxicas que empregan operadores aritméticos e/ou operadores lóxicos. Entre os **operadores aritméticos** podemos destacar: suma (+), resta (-), multiplicación (*), división (/) e asignación (=). Entre os **operadores lóxicos** podemos destacar: igualdade (==), desigualdade (!=), maior que (>), menor que (<), maior ou igual que (>=), menor ou igual que (<=), e (&&), ou (||) e negación (!).
  - operacións de entrada ou saída de datos , como pode ser a solicitude de datos por teclado ou a visualización de información en pantalla.
- **Selección** - Permite levar a cabo un conxunto de instrucións ou outro dependendo do resultado da avaliación dunha expresión lóxica.
- **Iteración** - Permite repetir un conxunto de instrucións dependendo do resultado da avaliación dunha expresión lóxica.

O emprego de unicamente estas estruturas da lugar á **programación estruturada.**



Exemplo 2:

Realización de multiplicacións mediante sumas.

```
Pedir numero1
Pedir numero2
resultado=0
Mentres (numero2>0) Facer
  resultado=resultado+numero
numero2=numero2-1
Fin-Mentres
Visualizar resultado
```

Exemplo 3:

Indicar se un número é primo.

```
Visualiza “Introduce un número maior que cero:”
Pedir numero
Si (numero==1) Visualiza “1 é primo”
SeNon
      Conta=2
      Mentres (numero % conta !=0) conta=conta+1
      Si (conta!=numero) Visualiza numero “NON é primo”
      SeNon Visualiza numero “é primo”
Fin-Si
```

Exercicio 3

Escribir un algoritmo que nos indique se un número é par ou impar.

Exercicio 4

Escribir un algoritmo que pida un número de quilómetros e devolva a súa correspondencia en millas.

Exercicio 5

Escribir un algoritmo que realice unha división enteira de dous número enteiros mediante restas.

Exercicio 6

Escribir un algoritmo que visualice os factores primos dun número.

Exercicio 7

Escribir un algoritmo que solicite dous números enteiros e visualice o resultado do primeiro elevado ao segundo.

Exercicio 8

Escribir un algoritmo que pida un número enteiro e visualice o seu factorial. O factorial dun número (n!) é o resultado de multiplicar o número por todos os números anteriores ata chegar a 1 (n!=n*(n-1)*(n-2)*...*2*1). Por convenio o factorial de 0 é 1 (0!=1).

Exercicio 9

Escribir un algoritmo que visualice na pantalla o número de termos da serie de Fibonacci que indique o usuario. A serie de Fibonacci ten a seguinte forma: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...