CONCEPTOS BÁSICOS De PROGRAMACIÓN INFORMÁTICA

1 – PROGRAMACIÓN ESTRUTURADA

- A programación informática consiste basicamente en indicar a SECUENCIA de OPERACIÓNS que queremos aplicarlle a uns DATOS DE ENTRADA para conseguir uns determinados RESULTADOS DE SAÍDA. Seguramente pode facerse de varias formas, pero sempre hai unhas que son mellores que outras e que calculan os resultados con máis exactitude e máis rapidamente.
- A PROGRAMACIÓN ESTRUTURADA céntrase na secuencia e conxunto de operacións que hai que realizar, sen preocuparse demasiado da natureza dos datos que se van manexar nin dos resultados que se van obter.
- Crear un programa de ordenador é bastante parecido a elaborar unha receita de cociña. Hai que seguir uns pasos determinados, e hai que facelos nunha determinada orde, porque se non é así, o prato non sairá perfectamente. Esa secuencia de pasos recibe o nome de ALGORITMO.

1 – PROGRAMACIÓN ESTRUTURADA

Basicamente só se poden utilizar TRES TIPOS de estruturas de operacións:

- SECUENCIA: as instrucións vanse realizando unha tras outra, na orde que nós decidamos. Pero obviamente é moi importante elixir a secuencia correcta, seguramente a única que nos pode levar ao resultado final.
- SELECCIÓN, ALTERNATIVA ou CONDICIONAL: Nun determinado punto, e en función de determinados criterios (matemáticos ou lóxicos principalmente), o programa pode realizar diferentes operacións.
- ITERACIÓN ou REPETICIÓN: Nun determinado punto, unha instrución ou un conxunto delas poden repetirse en múltiples ocasións para conseguir os obxectivos perseguidos.

1 – PROGRAMACIÓN ESTRUTURADA

SECUENCIA

Este é un sinxelo exemplo de secuencia de instrucións ou operacións. Queremos calcular a media aritmética de tres números:

- O primeiro que facemos é pedir ou averiguar eses números dos que queremos facer a media, e almacenamos cada un deles nunha variable diferente, para non confundilos nen mesturalos.
- Só unha vez que xa coñecemos os tres números e os temos almacenados por separado (non antes), xa podemos sumalos.
- E unha vez que realizamos a suma, podemos facer a división para obter a media aritmética, o resultado que estabamos buscando.
- Como vedes, a operación é sinxela pero hai que facer as operacións nesa orde, porque senón sería imposible conseguir o resultado buscado.

Pedir 1º dato e gardalo en X

Pedir 2º dato e gardalo en Y

Pedir 3º dato e gardalo en Z

Sumar os tres datos e

gardar o resultado en S

Dividir S entre 3 e amosar

o resultado na pantalla

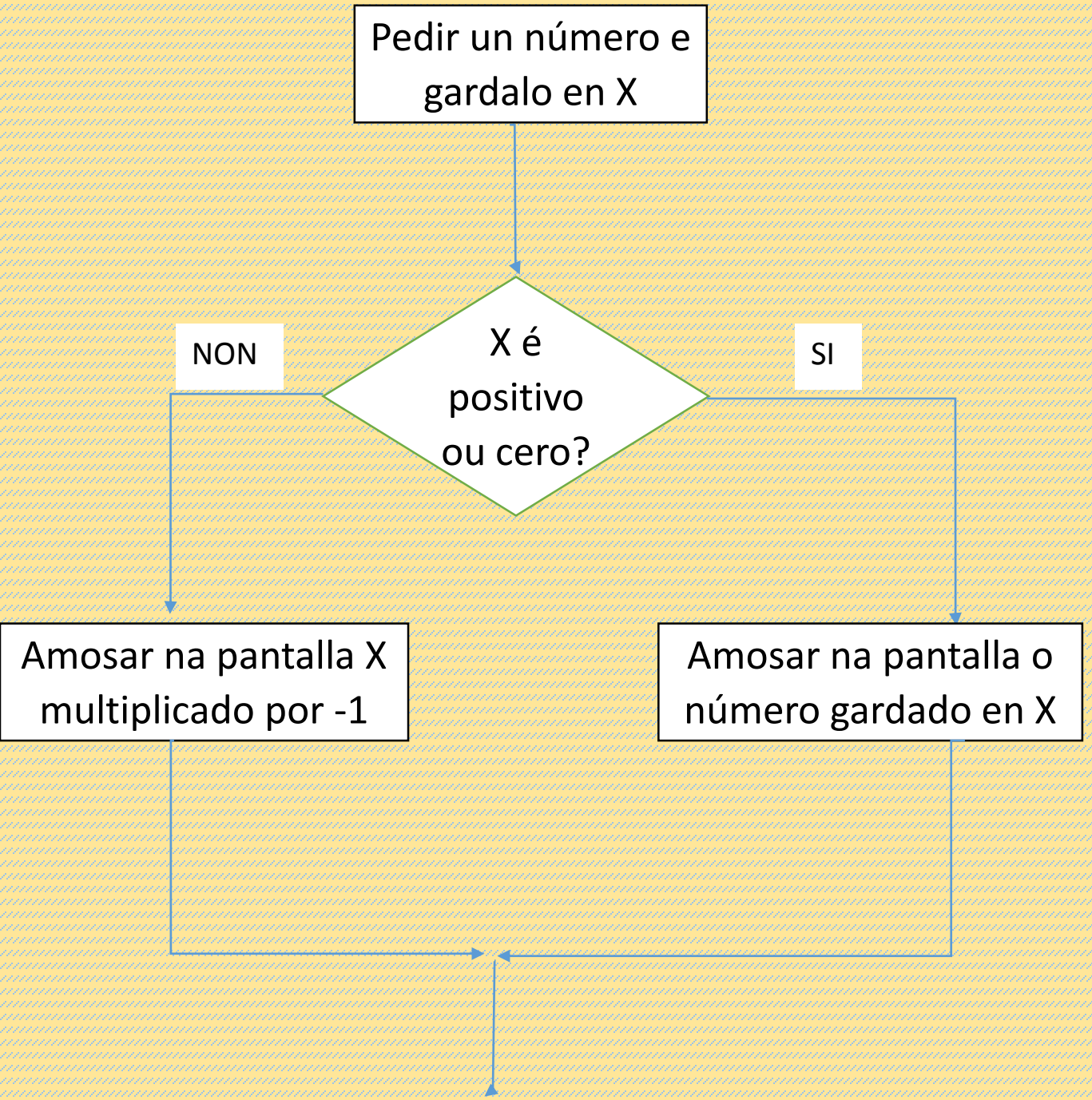
1 – PROGRAMACIÓN ESTRUTURADA

- Este sería o diagrama de fluxo (esquema gráfico) que indica en que orde se farían as operación necesarias para calcular a media aritmética de 3 números.
- Como vedes, a secuencia é ben simple, pero hai que decidila con antelación e non confundirse, para que a operación saia correctamente.

1 – PROGRAMACIÓN ESTRUTURADA

SELECCIÓN, ALTERNATIVA ou CONDICIONAL

- Imos calcular o valor absoluto dun número. Unha vez almacenado o número en X, comprobamos se o número é positivo ou cero, ou ben se é negativo, porque a forma de calcular o valor absoluto é diferente neses dous casos. Unha vez realizada unha das dúas operacións (pero non as dúas, só unha delas), o programa continúa.
- Na maioría de linguaxes de programación (que case sempre utilizan palabras do idioma inglés), a estrutura de Selección, Alternativa ou Condicional utiliza palabras como IF (o “si” condicional en inglés), THEN, ELSE, CASE e outras parecidas.
- Na seguinte diapositiva veremos o diagrama de fluxo para este tipo de estrutura de programación.



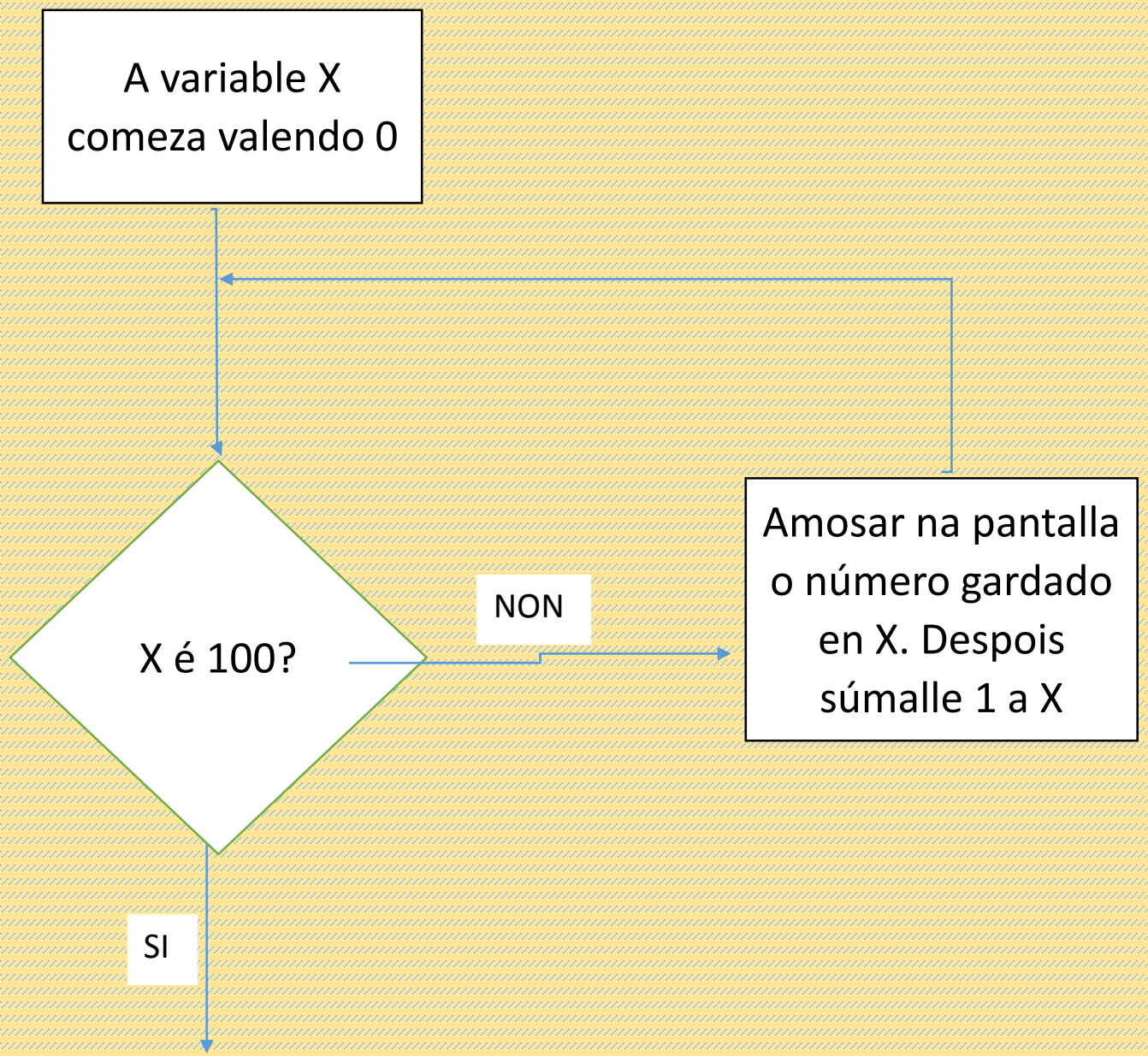
1 – PROGRAMACIÓN ESTRUTURADA

- Como vedes neste diagrama de fluxo, dependendo do signo do número gardado en X, irá pola parte dereita ou pola esquerda, pero non polas dúas ao mesmo tempo.
- Isto débese a que o valor absoluto non se calcula da mesma forma para os positivos e os negativos, o programa ten que coller o camiño axeitado para cada caso.

1 – PROGRAMACIÓN ESTRUTURADA

ITERACIÓN ou REPETICIÓN

- Queremos escribir todos os números do 0 ao 99, nesa orde. A variable comeza valendo 0. Compróbase se o valor da variable xa chegou a 100. Se non é así, amósase ese valor na pantalla e increméntase o seu valor en 1. E volve facerse a comparación, unha e outra vez, ata que finalmente X xa chegou ao valor 100, e nese intre deixa de repetirse o proceso. Como resultado desas múltiples repeticións, todos os números do 0 ao 99 foron escritos na pantalla na orde correcta.
- Na maioría de linguaxes de programación (que case sempre utilizan palabras do idioma inglés), a estrutura de Iteración ou Repetición utiliza palabras como DO, WHILE, UNTIL, FOR, NEXT, REPEAT e outras parecidas.



1 – PROGRAMACIÓN ESTRUTURADA

- Como vedes, aquí está dando voltas continuamente polo cadro da dereita, ata que o valor contido na variable finalmente chega a 100.
- Nese momento, o programa xa continúa a súa marcha normal.
- Iso permite realizar algo moitas veces, que é algo moi necesario na programación informática.

2 – VARIABLES E TIPOS DE DATOS

Unha VARIABLE representa un pequeno anaco da memoria no que se almacena un determinado dato.

A persoa que está creando un novo programa debe decidir cantas variables necesita para levar a cabo o obxectivo que persigue ese programa. Nalgunhas linguaxes tamén é necesario decidir que TIPO DE DATOS se almacena nesa variable, porque non todas son iguais, e unhas ocupan máis espazo en memoria que outras.

Por sorte, as linguaxes que damos nós (Logo, Scratch e Basic) son “pouco tipadas”, o cal significa que non é obrigatorio declarar previamente o tipo de datos que conterá unha variable, os programas poden funcionar igual sen facer ese trámite (en linguaxes máis profesionais, iso é impensable).

Durante a execución do programa, o valor que contén unha variable pode cambiar todas as veces que sexa necesario (por iso se lles chama “variables”).

2 – VARIABLES E TIPOS DE DATOS

- As variables necesitan ter un nome recoñecible durante toda a execución do programa (o nome si que non pode cambiar) para distinguilas perfectamente entre sí.
- Non é obrigatorio, pero é moi recomendable que o seu nome faga referencia ao concepto real que representa o seu valor. Por exemplo, se nunha variable vas almacenar a idade dunha persoa, ten moita lóxica que a esa variable lle chames “idade”, pero nada impide que lle poidas chamar “z”, se ti o decides así.
- As normas léxicas para poñer nomes varían dunha linguaxe de programación a outra, pero en xeral son bastante parecidas, e inclúen regras como que o nome dunha variable non pode coincidir co de ningunha instrución desa linguaxe, ou que poden conten números no seu nome, pero non poden ter só números nin comezar cunha cifra.
- Por exemplo, en case todas as linguaxes poderás crear unha variable chamada “num1”, pero nunca poderás chamarlle “1num” nin “1”.

2 – VARIABLES E TIPOS DE DATOS

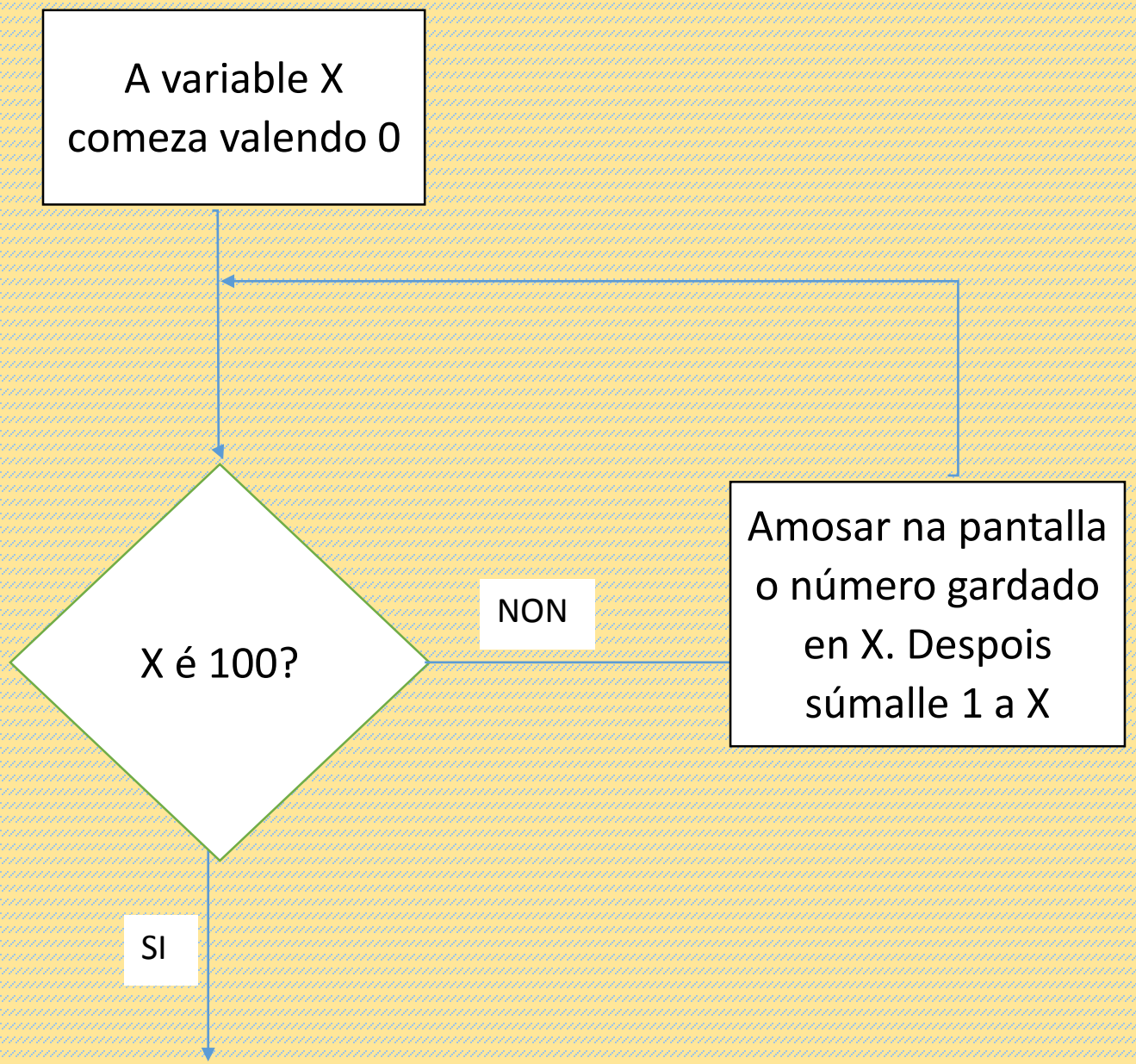
- Hai linguaxes nas que é obrigatorio (noutras é opcional ou non é necesario) declarar o TIPO DE DATOS que vai conter unha variable. Polo tanto, hai instrucións nas que non se fai ningunha operación relevante, senón que simplemente “presentas en sociedade” as variables, indicando cal é o seu nome e o tipo de datos que vai conter.
- Obviamente, iso debe ser ao principio do programa. Se posteriormente, intentas meter nesa variable un dato que ten unha natureza diferente, o programa falla e non o acepta. Por exemplo, se declaras que unha variable só vai recibir letras ou texto, e intentas almacenar nela un número, non o acepta. Ou viceversa.
- Hai moitísimas linguaxes de programación e cada unha declara os tipos de datos dun xeito diferente, pero en xeral hai algunhas palabras ou conceptos comúns. Imos poñer algúns exemplos (coas palabras en inglés, aínda que hai algunhas linguaxes que teñen versión en castelán):

2 – VARIABLES E TIPOS DE DATOS

- Int: para números enteiros, normalmente non demasiado grandes
- Char, String, Byte: letras ou textos formados por letras
- Float, Double, Real, Single, Decimal: para números decimais, de máis ou menos precisión, ou números moi grandes ou moi pequenos
- Short, Long: modificadores para algúns dos tipos numéricos, como por exemplo “int”
- Array: listas ou matrices de variables que forman unha colección
- Const: constantes, ou sexa, variables que nunca cambian de valor durante o programa
- Boolean: variables que só poden ter dous valores, como “si-non”, “certo-falso”
- Date: para almacenar datas, en diferentes formatos

3 – CONTADORES e ACUMULADORES

- Como vimos nos temas anteriores, unha variable pode tomar múltiples valores durante a execución dun programa. E hai partes dun programa que poden repetirse unha vez tras outra repetidamente. A combinación destes dous factores podería resultar caótica, pero resulta todo o contrario, é un mecanismo moi útil para resolver situacións que se producen en moitas ocasións na programación informática.
- O problema reside en entender a posible confusión que nos podería traer o parecido con expresións que coñecemos noutras disciplinas como a matemática ou a física. Nesas disciplinas científicas, o símbolo “=” serve para igualar ou comprobar se as dúas expresións que a rodean son iguais.
- En Informática ten un significado lixeiramente diferente: na variable que está á esquerda do “=” vaise almacenar o resultado de calcular o valor da expresión que está á súa dereita. De feito, hai algunha linguaxe de programación na que teñen dous operadores diferentes: o símbolo “=” para simplemente comparar dúas expresións, e o símbolo “:=” para asignar valores a variables, como indicamos antes.



3 – CONTADORES e ACUMULADORES

- Sabido isto, un CONTADOR é unha variable moi útil que nos pode permitir contar cantas veces sucede algo. E diso podemos aproveitarnos para escribir a gran velocidade unha serie de números consecutivos utilizando unha única variable, ou para asegurarnos de que algo sucede no programa xustamente a cantidade de veces que nós queríamos.
- Este é o exemplo que vimos antes para a estrutura de programación ITERATIVA ou REPETITIVA.
- Escribe na pantalla todos os números dende o 0 ata o 99.

3 – CONTADORES e ACUMULADORES

Nun programa coma o anterior, a primeira vez que pase pola instrución que amosa os números en pantalla, X valerá 0, que será escrito na pantalla. A continuación, X incrementa o seu valor en 1. Na inmensa maioría das linguaxes de programación, a instrución para facelo sería X = X + 1

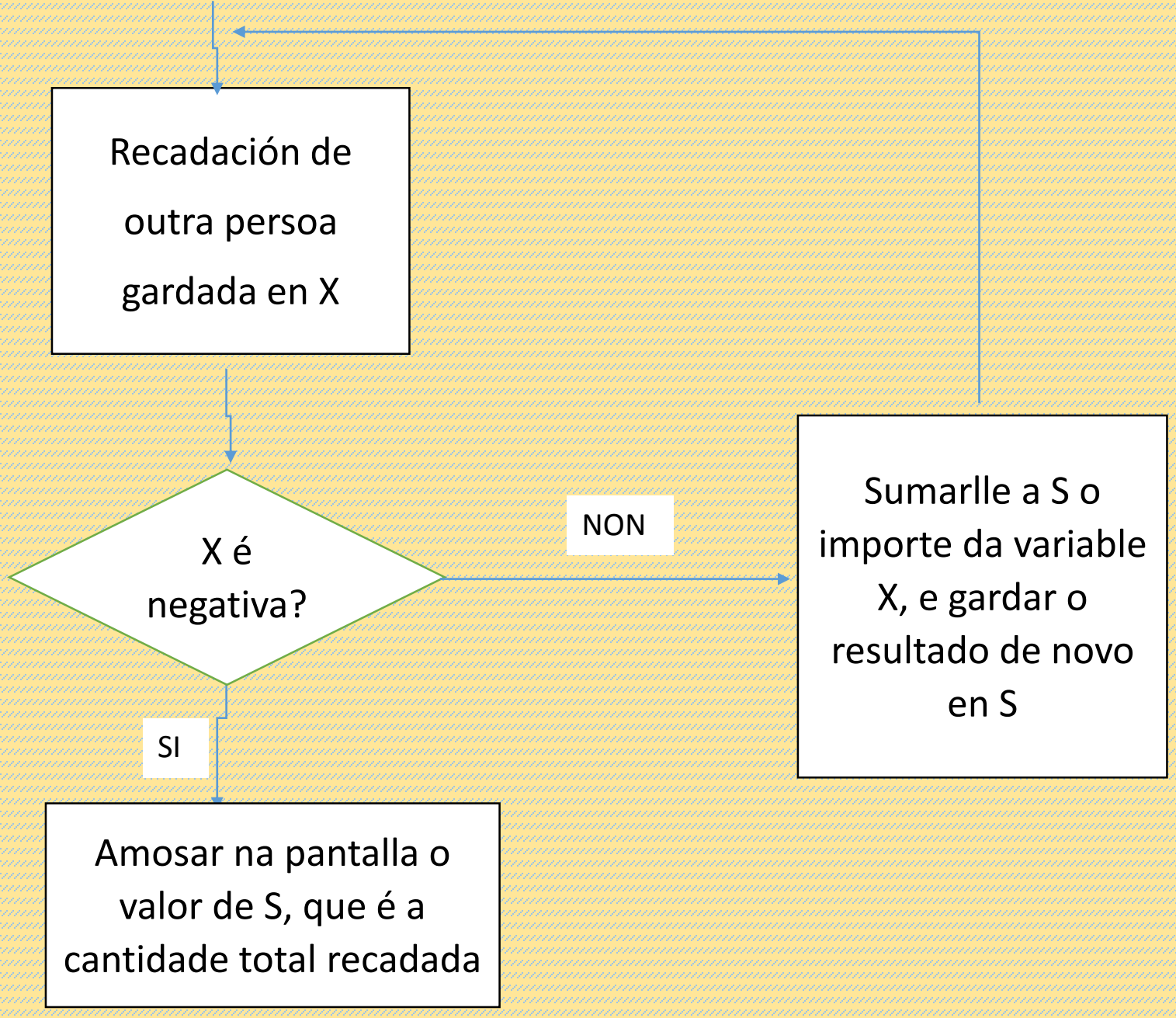
Esta expresión en Matemáticas sería unha barbaridade imposible, pero en Informática NON. Porque realmente significa “calcula o resultado de sumar o valor actual da variable X máis 1, e almacena o resultado desa operación na variable X de novo”.

Ou sexa, que despois de efectuar esta instrución, a variable X cambiou de valor, porque cando chegou a primeira vez, o valor que tiña almacenado era 0, e xusto despois de pasar por aquí, o valor almacenado xa é 1. Exactamente esa mesma operación vaina facer moitas máis veces, pero o valor é cada vez máis grande.

Polo tanto, a expresión anterior en Informática ten sentido, pero non tería ningún sentido a seguinte X + 1 = X

3 – CONTADORES e ACUMULADORES

- Pola contra, un ACUMULADOR, que funciona de forma moi semellante a un Contador, pode ter unha función algo diferente. Por exemplo, pode servir para sumar unha serie de cantidades numéricas.
- Supoñamos que imos introducindo por teclado as cantidades monetarias que recadaron varias persoas que estiveron vendendo rifas para un sorteo. Se o programa nos vai pedindo as cantidades que trouxo cada unha das persoas implicadas, un Acumulador podería dicirnos cal é a cantidade total recadada.
- Neste exemplo, daríamoslle a entender ao programa que introducir unha nova recadación negativa significa que xa non queda ningunha persoa máis por contabilizar.
- Na seguinte diapositiva imos ver o diagrama de fluxo do funcionamento deste Acumulador.



3 – CONTADORES e ACUMULADORES

- Cada vez que nos pregunte o ordenador pola recadación dunha nova persoa, gárdase en X e o programa vai ACUMULANDO a suma parcial de todas esas cantidades mediante unha instrucción como

$$S = S + X$$

Finalmente a variable S terá a recadación total.