Curso de Programación

Orientado a videojuegos

Empezamos en minutos...

Maestro Jedi: Daniel Delgado
Duración: al menos 10 horas.
El resto depende de ti...

Dos últimos ejercicios de algoritmos lineales:

1 Una agencia de viaje requiere que un programa haga el cálculo de tu moneda local a dólares.

2 Tres personas han invertido dinero para formar una empresa, de acuerdo al capital invertido por cada uno, ¿cuánto % de acciones le toca a cada uno?

1) Datos de entrada: valor del dólar con respecto a tu moneda, y cantidad de dinero a cambiar.
Proceso: cantidad de dinero / precio de un dólar en moneda local de otro país.
Salida: Conversión de tu moneda a dólares.

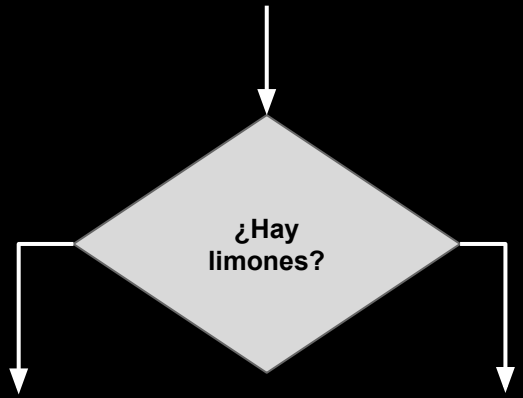
2) Datos de entrada: Dinero invertido por la persona 1, 2 y 3. Suma del dinero invertido.
Proceso: Regla de 3 para calcular porcentaje para cada persona.
Datos de salida: % de acciones por persona.

```
Begin CambiaRapido
        Var Number: valorDolar, cantidadDinero = 0;
        Input “Introduzca el valor de 1 dólar en tu país”, valorDolar;
        Input “Introduzca la cantidad de dinero a cambiar”,
cantidadDinero;
        Print cantidadDinero & “ son ” & cantidadDinero / valorDolar
& “dólares”;
End CambiaRapido
```

```
Begin Acciones
        Var Number: montoPersona1, montoPersona2,
montoPersona3, total = 0;
        Input “Introduzca inversión de la Persona 1”, montoPersona1;
        Input “Introduzca inversión de la Persona 2”,
montoPersona2;
        Input “Introduzca inversión de la Persona 3”,
montoPersona3;
        total = montoPersona1 + montoPersona2  + montoPersona3;
        Print “A la persona 1 le tocan ” & montoPersona1 * 100 / total;
        Print “A la persona 2 le tocan ” & montoPersona2 * 100 /
total;
        Print “A la persona 3 le tocan ” & montoPersona3 * 100 /
total;
End Acciones
```

Nuevo nivel desbloqueado

Estructura de Programación de Decisión



Así como en el ejemplo de la primera clase, en el que había que verificar si había limones, al programar nos vamos a encontrar a menudo, con problemas que no podremos resolver sin revisar si ciertas condiciones se cumplen o no.

Así debes reconocer estas situaciones y decirle a la computadora qué hacer en cada caso. Esto hace que las secuencias no sean lineales ya, sino estructuras con bifurcaciones.

Un ejemplo básico y muy usado en clases es por ejemplo la división de dos números, en la que debes saber si el divisor es distinto de cero, para poder llevarla a cabo. Si el divisor es = 0 entonces no se puede resolver y se debe mostrar un mensaje al usuario.

Las decisiones a tomar implican estudiar si una o más variables cumplen una o más condiciones, y establecer qué instrucciones se deben ejecutar o ignorar, sabiendo que pueden ser 1 o más instrucciones.

Condiciones esenciales para la felicidad... de tu programa

Todo depende de las condiciones, pero ¿qué es una condición en programación?

Una condición es una expresión de relación o lógica entre dos o más variables y que puede cumplirse o no, según los valores almacenados en las variables involucradas.

Una condición de relación es la comparación entre dos variables del mismo tipo, o entre una variable y una constante del mismo tipo. Estas condiciones están sujetas a la comparación de las variables por medio de los operadores de comparación vistos hace dos clases.

Una condición lógica son varias condiciones de relación unidas por los operadores lógicos and, or, not y cuyo resultado depende de la tabla de verdad de dichos operadores.

Las respuestas a estas condiciones lógicas o de relación siempre es un booleano, siempre sera un 0 o un 1, un true or false.

Ejemplos:

Supongan que tenemos dos variables a y b.

a = 4
b = 3

Comparaciones de Relación
Si preguntamos es a = 9? La respuesta obtenida es falso ya que es igual a 4.
Si preguntamos es b <= 10? La respuesta obtenida es verdadero ya que 3 es menor que 10.

Comparaciones Lógicas (dos o más comparaciones de relación)
Si preguntamos ahora:

a > 0 AND a < 10: La computadora arrojaría un true (verdadero) de esta comparación ya que al mismo tiempo se cumple que 4 es mayor que cero y 4 es menor que 10.
a == 2 OR b == 3: La computadora arrojaría un true (verdadero) ya que aunque a no es igual a 2, b si es igual a 3.
NO(a < b): La computadora arrojaría un true (verdadero) ya que el NO está negando que a sea menor que b, lo cual es cierto.

Podemos decir entonces que cuando se usa un operador lógico AND, las dos condiciones se deben cumplir para que la computadora devuelva un TRUE. Si alguna condición es falsa, la computadora devolverá un FALSE.
Cuando se usa un OR, las dos o alguna de las dos condiciones debe ser verdadera para que la computadora devuelva un TRUE. Si las dos son falsas, la computadora devolverá un FALSE.
Cuando se usa un operador NO, se niega la condición expresada dentro del NO. Así la condición dentro del NO deberá ser falsa, para que la computadora devuelva un TRUE, si la condición dentro del NO es verdadera, entonces la computadora devolverá un FALSE.

Ejercicios

Según los valores de a, b y c responda si la computadora daría un TRUE or FALSE en cada caso.

Ejercicio 1:
a = 0;
b = 4;
c = 3;

a < b TRUE
c == a + b FALSE
c >= b / 2 TRUE
d == a + b + c ERROR

Ejercicio 2:

a = true;
b = 10;
c = “hola”;

b != c ERROR
a == NOT(b < 20 AND c == “mundo”)
TRUE
a <> b & c ERROR

Escribir comparaciones de relación y lógicas

a = 5, b = 6, c = 4, d = “diez”

a <= b + c TRUE
a - c < b TRUE
a < b AND b > c TRUE
a == b OR b >= c TRUE
NOT(a == b) TRUE
a != b AND b > c TRUE
A !=b AND b == c FALSE

A >= c AND (c == b OR a < 10) TRUE