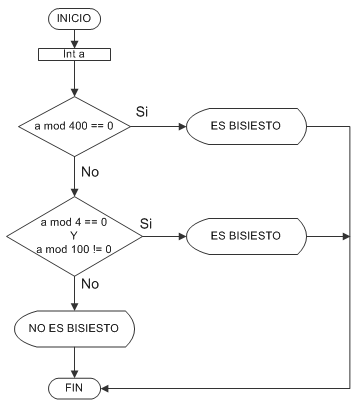
**EJERCICIOS ESTRUCTURAS DE CONTROL**

1. Escribir un programa que tenga dos variables de tipo int con los siguientes nombres: primera y segunda. Inicializar las variables con cualquier número. Chequear si la primera es mayor, menor o igual que la segunda. En cada caso lanzar un mensaje informativo por la consola.
2. Escribir un programa que tenga una variable (inicializarla con cualquier valor) y que sea capaz de decir si el valor de esa variable es par o impar (si es divisible por 2, es decir, si el resto de dividir por 2 es cero).
3. Escribir un programa que tenga una variable (inicializarla con cualquier valor) y que sea capaz de decir si el valor de esa variable es positivo o negativo.
4. Escribir un programa que sea capaz de saber si un año es bisiesto o no. Para ello inicializar una variable de tipo int con un valor que represente a un año. El código que tenéis que representar es el siguiente:



1. Escribir un programa que sea capaz de categorizar a una persona en función de su altura. Para ello se seguirá la siguiente regla en función de si la persona es un hombre o una mujer:
   * Si es mujer:
     + Si la altura es menor de 1,35 la mujer es enana.
     + Si la altura está entre 1,35 (no incluido) y 1,65 la mujer es de estatura media
     + Si la altura está entre 1,65 (no incluido) y 1,90 la mujer es alta
     + Si la altura es mayor de 1,90 (no incluido) la mujer es gigante.
   1. Si es hombre:
      * Si la altura es menor de 1,40 el hombre es enano.
      * Si la altura está entre 1,40 (no incluido) y 1,65 el hombre es de estatura media.
      * Si la altura está entre 1,65 (no incluido) y 2,00 el hombre es alto.
      * Si la altura es mayor de 2,00 (no incluido) el hombre es gigante.

Para probar el programa crear dos variables: una de tipo float y otra de tipo boolean que indicará si es mujer o no.

Pista: hay que anidar los ifs.

1. Escribir un programa que dado un carácter pueda decir si está en mayúsculas o minúsculas. Para resolver este programa hay que tener en cuenta que dadas las letras: a, b, c…..z Java tiene en cuenta el orden de tal forma que pueden compararse y a sería menor que b o c.

De igual forma sucede con las mayúsculas se pueden comparar A y B dando como resultado que A es menor que B.

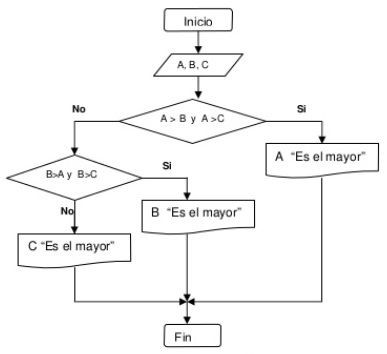
Asimimo, las mayúsculas son menores que las minúsculas.

Para verlo más claro podéis echar un vistazo a la [tabla ASCII](https://www.asciitable.com/) ahí podéis ver cómo a cada símbolo se le asigna un número (un byte).

Por tanto, para resolver este problema tenemos que comprobar lo siguiente:

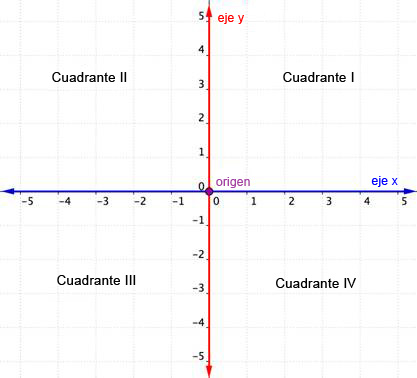
* + - Si el carácter dado está entre ‘a’ y ‘z’ (incluidas). Entonces la letra es minúscula.
    - Si el carácter dado está entre ‘A’ y ‘Z’ (incluidas). Entonces la letra es mayúscula.

1. Escribir un programa que nos diga si un triángulo es **equilátero** (tiene sus tres lados iguales), **isósceles** (tiene dos lados iguales) o **escaleno** (tiene los tres lados diferentes). Para ello:
   1. crear tres variables de tipo int: primerLado, segundoLado, tercerLado.
   2. Chequear si los tres lados son iguales. Si lo son, son equiláteros.
   3. Si no es equilátero, chequear si es el isósceles. Es decir, si el primerLado es igual al segundoLado o el primerLado es igual al tercerLado o el segundoLado es igual al tercerLado. Si se cumple está condición entoces es isósceles.
   4. Si no se cumplen las dos condiciones anteriores entonces es escaleno.
2. Crear un programa que averigüe el mayor de tres números. Para ello usar el siguiente pseudocódigo:



Para probar el programa crear tres variables de tipo int e incializarlas con cualquier valor.

1. Crear un programa que dado unas coordenadas (dos variables x e y de tipo int) sea capaz de decir a qué cuadrante pertenece el punto dado:



El programa tiene que dar la siguiente salida por consola. Ejemplos:

* + - El punto x = 3, y = 5 pertenece al cuadrante I.
    - El punto x = -2 y = 4 pertenece al cuadrante II.
    - El punto x = -1 y = -3 pertenece al cuadrante III.
    - El punto x = 8 y = -5 pertenece al cuadrante IV.
    - El punto x = 0 y = 0 pertenece al origen.

1. Nos piden crear un programa que diga si un alumno puede estudiar unos estudios o no. Las condiciones para que el alumno pueda entrar a estudiar son las siguientes:
   * 1. Matemáticas >= 6,5 Y Física >= 6 Y Química >= 5 O que la media entre ellas sea mayor que 7.

Para probar el programa crear tres variables de tipo float que representen las notas de cada una de las asignaturas.

El programa tiene que decir si el candidato es elegible o no.

1. Escribir un programa que calcule el precio que tiene que pagar un consumidor por el consumo eléctrico. Para ello nos guiaremos de la siguiente tabla:
   1. Hasta 199. 1,20 la unidad
   2. Entre 200 y 400 (incluido). 1,5 la unidad
   3. Entre 400 y 600. 2,0 la unidad.
   4. Más de 600 (no incluido). 3,0 la unidad

Para ello inicializar una variable de tipo float que represente al consumo. Cada uno de los umbrales o límites (199, 200, 400, etc.) representarlos en una clase de constantes.

El programa debe de dar una salida por consola del tipo:

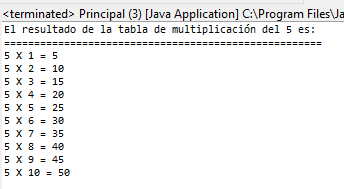
El precio de la unidad correspondiente al consumo XX es YY.

1. Escribir un programa que calcule la potencia de un número. Para ello crearemos tres variables de tipo int:
2. La primera variable contendrá la potencia al cual queremos elevar el número.
3. La segunda variable contendrá el número del cual queremos obtener el resultado de la potencia.
4. La tercera variable la usaremos para ir almacenando el resultado.

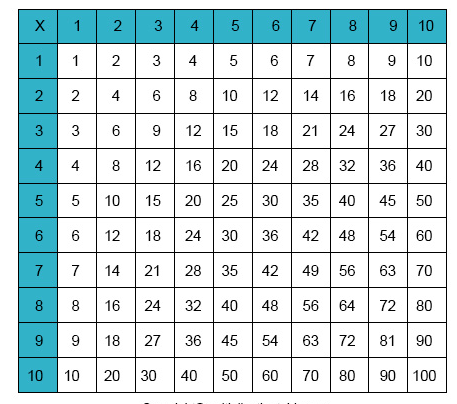
El programa debe de dar una salida por consola del tipo:

El resultado de elevar XX a YY es ZZ

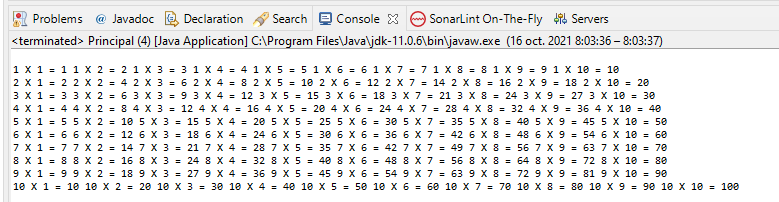
1. Escribir un programa que calcule la tabla de multiplicación de un número entero determinado. Para ello crear una variable de tipo int e inicializarla con cualquier valor. El programa debe dar como salida la tabla de multiplicación de ese número de la siguiente forma:



1. Escribir un programa que calcule la tabla de multiplicación del 1 al 10:



La salida del programa tiene que ser la siguiente



Pistas:

* Usar 2 bucles for anidados.
* Si queremos imprimir algo por consola, pero no saltar de línea usaremos el comando:
  + System.out.print en lugar de System.out.println

1. La secuencia de Fibonacci es una secuencia infinita de números naturales que empieza en 0 y 1. Cada uno de los números siguientes de la secuencia se obtiene sumando los dos pares anteriores. Por ejemplo, la secuencia de Fibonacci:

0, 1, 1, 2, 3, 5, 8, 13 ….

Se calcula de la siguiente forma:

* Se empieza con 0 y 1.
* El tercer número de la secuencia (1) es el resultado de sumar los dos anteriores (0+1).
* El cuarto número de la secuencia (2) es el resultado de sumar los dos anteriores (1+1).
* El quinto número de la secuencia (3) es el resultado de sumar los dos anteriores (1+2).
* Y así sucesivamente.

Esta secuencia aparece en numerosas manifestaciones de la naturaleza (flores, ramas de árboles, frutas, etc.) así como en los brazos espirales de las galaxias. Además, tiene aplicaciones en los análisis técnicos de bolsa, en la música o en matemáticas.

Se pide calcular la secuencia de Fibonacci hasta un límite. Para ello debemos declarar una variable de tipo int que llamaremos limite y que nos marcará el límite de la secuencia.

Además, declararemos dos variables más que nos indicarán los dos primeros números por los que empezará la secuencia (la semilla).

Podéis usar una variable más para acumular el resultado intermedio.

Posteriormente y con un bucle for crearemos la secuencia usando como límite el establecido en la variable.

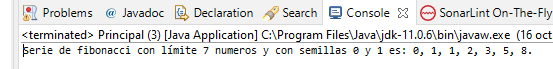
Por ejemplo, si las variables de inicio son:

int limite = 7;

int semilla1 = 0;

int semilla2 = 1;

La secuencia debe de ser:



Requisito:

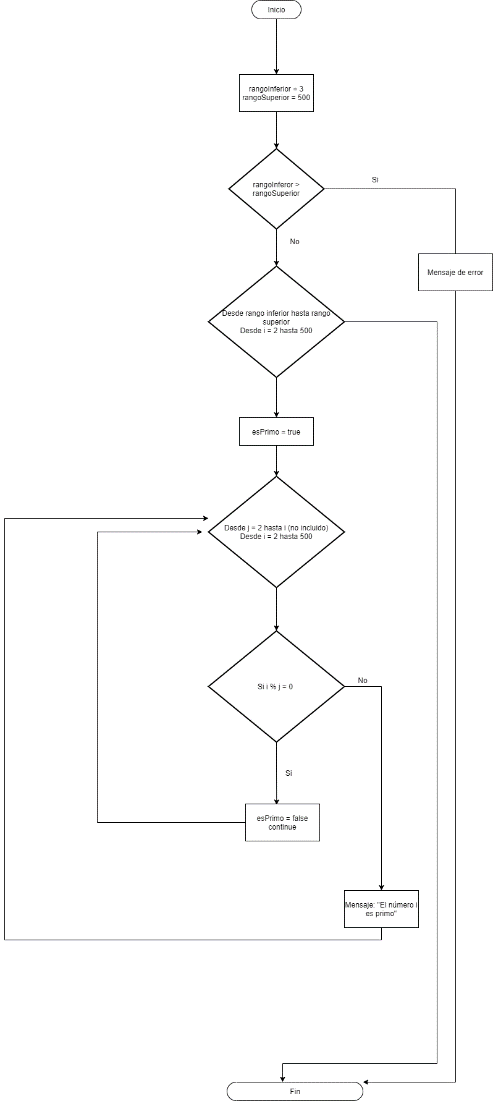
El mensaje debe ser exactamente igual que el que sale en la imagen (incluidas las comas y el punto final).

1. Crear un programa que dado un número entero nos diga el día de la semana al que corresponde. Es decir, si el número es 1 el programa deberá decir: “Lunes”. Si el número es 2 el programa deberá decir: “Martes”. Si el número no está entre 1 y 7 el programa deberá decir: “El número no es válido, debe estar entre 1 y 7”.

Para ello debemos declarar una variable de tipo int que inicializaremos con cualquier número y usando la estructura de control “switch” implementaremos la lógica del programa.

1. Se pide crear una calculadora que haga sumas, restas, multiplicaciones, divisiones y restos. Para ello tendremos cinco variables:
2. Dos variables de tipo int que almacenarán cada uno de los números con los que haremos la operación aritmética.
3. Una variable de tipo int que almacenará el resultado (la llamaremos “resultado”).
4. Una variable de tipo char que almacenará el tipo de operación a realizar. Podrá tener los siguientes valores en función de la operación (‘+’, ‘-’, ‘x’, ‘X’, ‘/’, ‘%’). La llamaremos operación.
5. Una variable de tipo boolean que llamaremos “operacionCorrecta”. Esta variable estará a true si la operación elegida es la correcta (‘+’, ‘-’, ‘x’, ‘X’ ‘/’, ‘%’). Si no es la correcta estará a false.
6. Usando el operador switch haremos la operación que corresponda y almacenaremos el resultado en la variable “resultado”.
7. Si la variable es ‘x’ o ‘X’ se hará la operación de multiplicación (pista: jugad con la sentencia break).
8. Si la variable operación no corresponde con alguno de los símbolos mencionados en el apartado c tendremos que emitir un mensaje avisando que la operación elegida no es la correcta. Además, pondremos la variable “operacionCorrecta” a false.
9. Fuera de la estructura switch tendremos que informar del resultado solamente si la operación elegida es la correcta. De lo contrario no emitiremos ningún mensaje.
10. Un número primo es aquel número que solo es divisible por sí mismo y por 1. Se pide hacer un programa que dado un rango entre dos números encuentre los números primos. Para ello:
11. Crear dos variables de tipo int. Se llamarán *“rangoInferior”* y *“rangoSuperior”.* Estas variables representarán el rango de números entre los que tendremos que buscar los números primos.
12. Crear un control que compruebe que la variable *“rangoInferior”* es menor que lavariable *“rangoSuperior”.* Si esto no se cumple el programa debe terminar y no debe hacer más cálculos. Además, emitirá un mensaje advirtiendo del error.
13. Si la variable *“rangoInferior”* es menor que la variable *“rangoSuperior”* el programa debe buscar todos los números primos entre ese intervalo.

El programa debe resolverse según este algoritmo:



Explicación:

Tenemos un rango superior e inferior el cual usaremos para buscar los números primos que hay entre ambos. La primera comprobación que tenemos que hacer es si el rango inferior es menor que el rango superior, si no se da esa condición debemos de lanzar un mensaje de error y detener el programa. Si no se cumple esta condición podemos continuar con la lógica del programa.

La lógica para encontrar los números primos será la siguiente:

Recorremos todos los números desde rangoInferior hasta rangoSuperior (incluido). Con cada uno de los números haremos lo siguiente: suponemos que es primo (ponemos la variable esPrimo a true). Comprobaremos si es divisible por algún número desde 2 hasta él mismo menos 1 (excluimos el 1 y el propio número). Si con algún número de estos la condición es true pondremos la variable esPrimo a false y usaremos el mandato continue para que no siga comprobando.

Fuera de ese bucle interno crearemos un if para chequear si la variable esPrimo está a true, si lo está emitiremos el mensaje avisando.

1. Escribir el mismo programa que el anterior pero que además nos de la información de la cantidad de números primos que ha encontrado.
2. **Edad equivalente perros-humanos.** Tenemos la edad de un perro. Se pide calcular la equivalente a la edad de los humanos en función de esta fórmula: Para los dos primeros años del perro la edad equivalente es igual a la edad del perro por 10,5 años. Si el perro tiene más de dos años cada año perruno es el equivalente a 4 años humanos.

Para ello tenemos que crear una variable de tipo int que almacene la edad del perro (int edadPerro).

Los valores de corrección de los años (10,5 y 4) los almacenaremos en constantes que se encuentren en una clase de constantes. Como siempre los nombres deben ser significativos del valor que almacenan.

El programa debe informar por consola de la edad del perro.

1. Crear un programa que, usando las sentencias switch y case detecten los días que tiene el mes.

|  |  |
| --- | --- |
| **Mes** | **Días** |
| Enero, marzo, mayo, julio, agosto, octubre, diciembre | 31 días |
| febrero | 28 días |
| Abril, junio, septiembre, noviembre | 30 días |

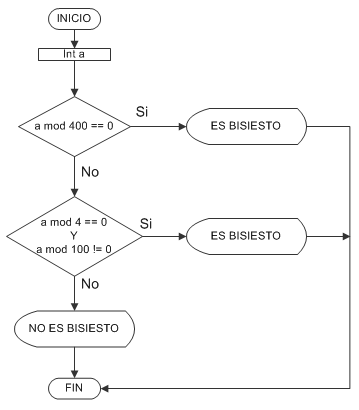
Para ello tendremos dos variables de tipo int:

* Una que guarde el número de mes.
* Otra que almacene el número de días.

Si el número de mes no se encuentra entre el rango de 1 y 12 se debe informar al usuario.

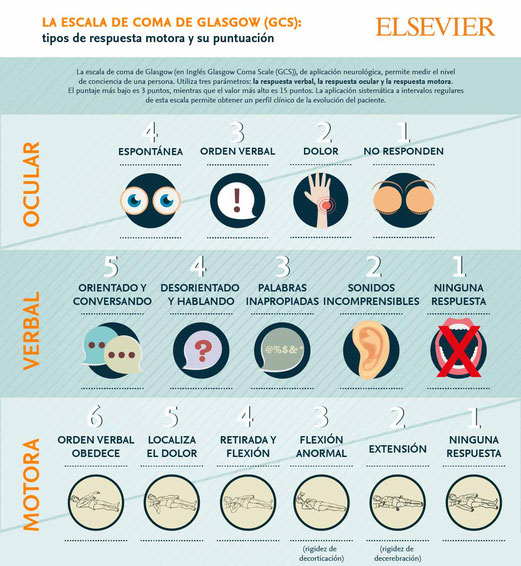
Nota: Solo debemos utilizar dos veces la sentencia System.out.println. Una para informar del número de días y otra para informar del error.

1. Mejorar el programa anterior de tal forma que sea capaz de decirnos el número de días que tiene febrero en función de si el año es bisiesto o no. Para saber si un mes es bisiesto debemos aplicar el algoritmo:



Para este nuevo programa tendremos una variable más de tipo int, que nos indicará el año sobre el que pretendemos hacer los cálculos.

1. Escribir un programa que dada una variable de tipo char nos diga si es vocal o consonante.
2. Escribir un programa que nos diga si una variable de tipo char está en el alfabeto o no. Para la resolución de este problema tenéis que tener en cuenta que las letras del alfabeto siguen una secuencia numérica en función de su posición en el alfabeto, es decir, ‘a’ > ‘b’, ‘b’ > ‘z’. Con las mayúsculas sigue la misma lógica.
3. La escala de coma de Glasgow se usa en medicina para evaluar el nivel de conciencia de los seres humanos. Se basa en la siguiente escala:



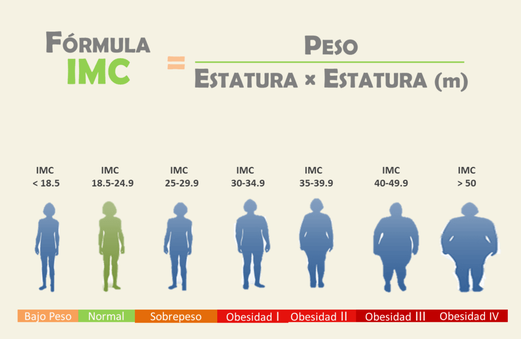
Se anotan las diferentes respuestas de la persona (apertura ocular, respuesta verbal y respuesta motora) y en función de las mismas se obtiene una puntuación. En función de esa puntuación y en el caso de un traumatismo craneoencefálico se puede definir la severidad del cuadro. Se utiliza para aplicar diagnósticos y terapias iniciales.

La tabla para su interpretación es la siguiente (tabla de Gennarelli):

|  |  |
| --- | --- |
| Puntos | Interpretación |
| 13-15 | Traumatismo craneoencefálico leve |
| 9-12 | Traumatismo craneoencefálico moderado |
| <9 | Traumatismo craneoencefálico severo |

Se pide: Crear tres variables de tipo int que guardarán los valores de apertura ocular, respuesta verbal y motora. Inicializarlas con cualquier valor. Dar un mensaje para la interpretación del traumatismo en función de la tabla de Gennarelli y la puntuación obtenida.

1. En función del IMC se puede calcular si el peso de una persona es el correcto o no. La fórmula a aplicar es la siguiente:



Se pide: Inicializar dos variables de tipo float que almacenen el peso y la altura. Calcular el IMC y en función de la tabla superior dar un mensaje en función del IMC obtenido.

Ejemplo: “El paciente tiene está en sobrepeso”

1. El Índice Pronóstico Internacional (IPI) en el linfoma de Hodgkin mide la probabilidad de supervivencia en los próximos 5 años después de haber superado un linfoma de Hodgkin (estar vivo y libre de enfermedad a los cinco años). Se basa en el siguiente cuestionario:
2. Albúmina sérica <4 g/dL (1 punto)
3. Hemoglobina <10.5 g/dL (1 punto)
4. Sexo masculino (1 punto)
5. Enfermedad en estadio IV según la clasificación Ann Arbor (1 punto)
6. Edad >=45 años (1 punto)
7. Recuento leucocitario >=15.000/mm3 (1 punto)
8. Recuento de linfocitos <600/mm3 o <8% de recuento leucocitario (1 punto)

Por cada epígrafe que se cumpla se sumará un punto y en función de los puntos obtenidos se dará un pronóstico de acuerdo a esta tabla:

* 0 puntos: 84 % supervivencia sin progresión y 89 % supervivencia general.
* 1 puntos: 77 % supervivencia sin progresión y 90 % supervivencia general.
* 2 puntos: 67 % supervivencia sin progresión y 81 % supervivencia general.
* 3 puntos: 60 % supervivencia sin progresión y 78 % supervivencia general.
* 4 puntos: 51 % supervivencia sin progresión y 61 % supervivencia general.
* 5 - 7 puntos: 42 % supervivencia sin progresión y 56 % supervivencia general.

Se pide crear las siguientes variables para almacenar los valores del cuestionario:

1. Albúmina sérica (variable de tipo int).
2. Hemoglobina (variable de tipo float).
3. Sexo masculino (variable de tipo boolean).
4. Enfermedad en estadio IV según la clasificación Ann Arbor (variable de tipo int que nos indicará el estadio).
5. Edad: (variable de tipo int).
6. Recuento leucocitario: (variable de tipo int).
7. Recuento de linfocitos: (dos variables de tipo int, una que guarde el recuento de linfocitos en mm3 y otra en porcentaje).

Debemos inicializar las variables con el valor que queramos y en función de esos valores que les hemos dado tenemos que ir calculando los puntos totales. En función de los puntos totales obtenidos debemos de emitir un mensaje indicando la probabilidad de supervivencia. Por ejemplo:

La probabilidad de supervivencia del paciente es de un 67 % supervivencia sin progresión y 81 % supervivencia general.

1. El Índice de Pronóstico Internacional del Linfoma folicular (FLIPI) mide la probabilidad de supervivencia en los próximos 5 años después de haber superado un linfoma folicular. Se basa en el siguiente cuestionario que asigna un punto por cada uno de los siguientes criterios:
2. Edad más de 60 años.
3. Estado Ann-Arbor III o IV.
4. Hemoglobina < 12 g/dl.
5. LDH > valor límite máximo de la normalidad. La normalidad depende del laboratorio, pero vamos a considerar un valor de 250.
6. Número de afectaciones ganglionares mayor o igual a 5.

Se pide crear seis variables para almacenar cada uno de los puntos del cuestionario que inicializaremos con un valor cualquiera:

1. Edad: (Variable int que guarde la edad del paciente)
2. Estado Ann-Arbor III o IV. (Variable booleana que nos indique si es el estado Ann-Arbor III o IV)
3. Hemoglobina: (Variable int que guarde la hemoglobina del paciente)
4. LDH: (variable de tipo int que guarde el LDH).
5. Número de afectaciones ganglionares: (Variable de tipo int que guarde el número de afectaciones ganglionares).
6. Una variable contador que vaya acumulando los puntos.

En función del número obtenido tendremos que hacer un diagnóstico de acuerdo a esta tabla:

|  |  |
| --- | --- |
| Grupo de riesgo | Puntuación |
| Bajo | 0-1 |
| Intermedio | 2 |
| Alto | 3, 4 o 5 |

El programa tendrá que emitir un mensaje de este tipo:

El paciente tiene riesgo bajo de muerte o recaída en linfoma folicular.

1. Linfoma Burkitt. Al igual que los anteriores para medir la probabilidad de supervivencia en los próximos 5 años después de haber superado un linfoma Burkitt se utiliza los siguientes criterios:
2. Edad 40 a 59 años: 1 punto.
3. Raza negra: 1 punto.
4. Edad 60 a 79 años: 2 puntos.
5. estadio III/IV: 2 puntos.
6. Edad igual o mayor a 80 años: 4 puntos.

Se pide: crear tres variables para almacenar cada uno de los puntos del cuestionario e inicializarlas con un valor cualquiera:

1. Edad: variable de tipo int.
2. Raza negra: variable de tipo boolean (será true si es raza negra).
3. estadio III/IV: (variable de tipo int que guarde el número del estadio).

En función de la siguiente tabla y la puntuación obtenida dar un diagnóstico:

|  |  |
| --- | --- |
| **Grupo de riesgo** | **Puntos** |
| Bajo (supervivencia a 5 años: 71%) | 0-1 |
| Intermedio-bajo (supervivencia a 5 años: 55%) | 2 puntos |
| Intermedio-alto (supervivencia a 5 años: 41%) | 3 puntos |
| Alto (supervivencia a 5 años: 29%) | 4 o más puntos |

1. Linfoma T. Sigue los siguientes criterios (suma un punto por cada uno):
2. Edad > 60 años.
3. ECOG >= 2.
4. LDH > valor límite máximo de la normalidad. La normalidad depende del laboratorio, pero vamos a considerar un valor de 250.
5. Infiltración de médula ósea.

Para ello usar las siguientes variables:

1. Edad (int).
2. ECOG (int).
3. LDH (int).
4. Infiltración de médula ósea. (boolean).

|  |  |
| --- | --- |
| **Grupo de riesgo** | **Puntos** |
| Bajo | 0 |
| Intermedio-bajo | 1 puntos |
| Intermedio-alto | 2 puntos |
| Alto | 3-4 puntos |

Notas:

* Se deben respetar las convenciones de Java en cuanto a nombres: Clases en PascalCase, métodos en camelCase, nombres de variables en camelCase, nombres de constantes en mayúsculas, etc. Por cada norma que no se cumpla restará 0,5 puntos.
* Las salidas de los programas deben respetar el formato que se muestra en los pantallazos (comas, puntos, retornos de línea, etc.)
* Se trata de trabajar las estructuras de control, por tanto, se debe trabajar única y exclusivamente con estas (bucles, ifs, switch, etc.) Cualquier otro método usado para resolver algún problema no contará como bueno.