Ejercicio 8: Porcentajes, IVA e inversiones

1.Escribir un algoritmo que calcula el precio con todos los impuestos incluidos (TII) para un precio sin impuestos y un porcentaje de IVA dado.

Algoritmo 1: ***Precio con IVA***

```
Algoritmo Precio_con_impuestos

Entrada

p: REAL # Precio sin impuestos
i: ENTERO # Porcentanje de IVA a aplicar

Efecto

P: REAL # Precio impuestos incluidos


realización

  Resultado ← p+(p*i)/100
  Resultado = p+(p*i)/100

fin Precio_con_impuestos
```

2. Escribir un algoritmo que calcula el importe de los intereses generados por un capital invertido a un interés dado durante un tiempo dado, expresado en meses.

Algoritmo 2: ***Importe interés generado***

```
Algoritmo importe_interés
Entrada

C: REAL # Capital inicial
i: REAL # Interés anual
t: Entero # Tiempo expresado en meses

Efecto

I: REAL # Importe intereses generados


Precondición
    capital ≥ 0
    interés > 0

Postcondición

   Resultado = C*(i/12)*t


fin importe_ interés
```

Ejercicio 9: Media aritmética ponderada

1.Escribir un algoritmo que calcula la media aritmética de tres números dados.

```
Algoritmo 9: Media aritmética
```

```
Algoritmo media

Entrada

N1,N2, N3: REAL # números dados
K: ENTERO : # cantidad de números dados

Efecto

MA: REAL # Media aritmética


Realización

  Resultado   (N1+N2+N3)/3
  Resultado= (N1+N2+N3)/3

fin media
```

2. La misma pregunta para una media ponderada cuando se dan los números y los coeficientes de ponderación.

```
Algoritmo 2: Cálculo Media ponderada
```

```
Algoritmo media_ponderada

Entrada

n1,n2, n3: REAL # números dados
w1, w1, w2: REAL # coeficientes de ponderación
W: ENTERO : # suma de coeficientes de ponderación

Efecto

MP: REAL # Media ponderada

Precondición
W=1

Postcondición

Realización

  Resultado ← ((n1*w1)+(n2*w2)+(n3*w3))/W
  Resultado= ((n1*w1)+(n2*w2)+(n3*w3))/W

fin media_ponderada
```

Ejercicio 10: Área del triángulo

1.Escribir un algoritmo que calcula el área de un triángulo del que se da la medida de un lado y la de la altura relativa a este lado.

Algoritmo 1: ***Área triángulo***

```
Algoritmo área

Entrada

b: REAL # base
h: REAL # Altura

Efecto

A: REAL # Área


Precondición
    base > 0
    altura > 0


Postcondición
  Resultado= (b*h)/2

fin área
```

2. ¿Se puede utilizar este algoritmo para un triángulo rectángulo si se dan las medidas de sus dos lados perpendiculares?

Sí, ya que cuando nos dan 2 lados perpendiculares tomando como base uno de los dos, el otro, por definición del concepto de altura, será la propia altura de dicho lado. Ese lado cumple las condiciones de altura ya que pasa por el vértice opuesto al otro lado que es el que hemos tomado como base y no solo eso, sino que es perpendicular a la misma, ya que ese dato viene incluido en la hipótesis.

Ejercicio 11: Salario y horas extra

El cálculo de una nómina tiene en cuenta el salario bruto asociado a las horas «normales» que debe hacer el empleado y las horas «extra» trabajadas en el mes. Las horas extra se remuneran según las siguientes normas administrativas:

- Tarifa por hora aumentada en un 125 % para las horas entre la 36.ª y la 43.ª.
- Tarifa por hora aumentada en un 150 % para las horas a partir de la 44.ª.

El aumento se realiza sobre la tarifa por hora normal, calculado a partir del salario mensual bruto para un año de 52 semanas repartidas en 12 meses, sobre la base de 35 horas trabajadas por semana.

Escribir el algoritmo que calcula el importe de las horas extra que hay que pagar, a partir del salario mensual bruto y de la cantidad de horas extra.

Se podrá suponer que el cálculo siempre se usa para una cantidad de horas superior a 8. El problema general supone el estudio previo del capítulo siguiente, que trata de la alternativa.

Encontrará una solución propuesta para este ejercicio en los elementos complementarios de este libro que están disponibles para descargar desde la página Información.

Algoritmo 1: Remuneración de horas extra

```
Algoritmo horas_extra
    # Establece la remuneración de `horas_ext' adicionales para
    # un salario mensual bruto de `salario_mensual_bruto'.

Entrada
    salario_mensual_bruto : REAL
        # Importe del salario mensual bruto
    horas_ext : ENTERO
        # Cantidad de horas extra del mes a pagar

precondición
    salario_mensual_bruto > 0
    horas_ext ≥ 0

constante
    CANTIDAD_SEMANAS : ENTERO ← 52
        # Cantidad de semanas de trabajo
    CANTIDAD_HORAS_SEMANA : ENTERO ← 35
        # Cantidad legal de horas de trabajo semanales
    CANTIDAD_HORAS_MAX_1 : ENTERO ← 8
        # Umbral de cambio de precio de remuneración
    PRECIO_1 : REAL ← 1,25
        # Tarifa de remuneración de CANTIDAD_HORAS_MAX_1 primeras
        # horas extra
    PRECIO_2 : REAL ← 1,50
        # Tarifa de remuneración de las otras horas extra

variable
    horas_ext_1 : ENTERO
        # Cantidad de horas extra con PRECIO_1 %
    horas_ext_2 : ENTERO
        # Cantidad de horas extra con PRECIO_2 %
    precio_hora : REAL
        # Precio hora de la remuneración bruta básica

realización
    calcular el precio_hora de la remuneración bruta básica

    Resultado ← precio_hora x
              (
                inf(horas_ext, CANTIDAD_HORAS_MAX_1) x PRECIO_1
              +
                sup(horas_ext - CANTIDAD_HORAS_MAX_1, 0) x PRECIO_2
              )

postcondición
...
fin horas_extra
```

Ejercicio 12: Cuenta de depósito

Se considera las cuentas de depósitos alojadas en un banco por los clientes. Solo se permite hacer una retirada si el saldo que queda en la cuenta no es negativo.

1.Definir el tipo de datos CUENTA.

*Algoritmo 1: Definición de **abrir** una cuenta*

```
abrir(c : CUENTA ; saldo_inicial : REAL)
    # Inicializar `c' mediante un `saldo_inicial'.

Precondición
    saldo_inicial > 0

realización
    c.descubierto ← 0
    c.saldo ← saldo_inicial

postcondición
    c.descubierto = 0
        # El descubierto no está autorizado
    antiguo(saldo_inicial) = saldo_inicial
    c.saldo = saldo_inicial

fin abrir
```

2. Definir las operaciones aplicables.

*Algoritmo 2: **abonar** una cuenta*

```
abonar(c : CUENTA ; crédito : REAL)
    # Crédito `c' de la suma `crédito'.

Precondición
    c.saldo ≠ NULO
    crédito ≠ NULO

realización
    c.saldo ← c.saldo + crédito

postcondición
    # El descubierto autorizado y el importe del `crédito' no se
    # modifican
    antiguo(c).descubierto = descubierto
    antiguo(c).crédito = crédito

    # El saldo aumenta con el `crédito'
    c.saldo = antiguo(c).saldo + crédito

fin abonar
```

*Algoritmo 3: **cargar** una cuenta*

```
cargar(c : CUENTA ; débito : REAL)
    # Carga `c' con la suma `débito'.

Precondición
    c.saldo ≠ NULO
    débito ≠ NULO
    c.saldo + c.descubierto ≥ débito ≥ 0

realización
    abonar(c, -débito)

postcondición
    # El descubierto autorizado y el importe del `débito' no se
    # modifican
    antiguo(c).descubierto = descubierto
    antiguo(débito) = débito

    # Al saldo se le resta el `débito'
    c.saldo = antiguo(c).saldo - débito

fin cargar
```

En determinadas circunstancias y para determinados clientes, la banca autoriza un descubierto limitado y temporal.

3. Volver a hacer las definiciones previas para permitir estos descubiertos.

*Algoritmo 7: Definición de **abrir** una cuenta con descubierto autorizado durante un tiempo limitado*

```
Algoritmo abrir
    # Inicializar `c' mediante un `saldo_inicial' y un
    # `descubierto_MAX' durante una `duración_max'.

Entrada
    c : CUENTA
    saldo_inicial : REAL
    descubierto_MAX : REAL
    duración_max : FECHA

Precondición
    saldo_inicial > 0
    descubierto_MAX ≥ 0
    duración_max ≥ 0

realización
    c.descubierto ← descubierto_MAX
    c.saldo ← saldo_inicial
    c.fecha_descubierto ← 0
    c.duración_max ← duración_max

postcondición
    c.descubierto = descubierto_MAX
    c.saldo = saldo_inicial
    c.duración_max = duración_max
    c.fecha_descubierto = 0

fin abrir
```