```
ALGORITMO_DE_PIA_MODULO

ESTRUCTURA DE DATOS:
PAISES: Arreglo[1..250] DE REGISTRO
    NOMBRE_P: CADENA
    POBLACION_P: ENTERO
    AREA_P: ENTERO
    DENSIDAD_P: DECIMAL
    REGION_P: CADENA
    SUBREGION_P: CADENA
    IDIOMAS_P: Arreglo[1..5] DE CADENA
    MONEDAS_P: Arreglo[1..3] DE REGISTRO
        CODIGO_M: CADENA
        NOMBRE_M: CADENA

// MODULO_OBTENER_DATOS_PAISES()
1.  Inicio MODULO_OBTENER_DATOS_PAISES()
2.  Declarar URL_API: CADENA → "https://restcountries.com/v3.1/all "
3.  Declarar DATOS_CRUDOS: Lista de diccionarios (datos sin procesar)
4.  Declarar PAISES: Arreglo de registros (estructura definida arriba)
5.  Intentar:
6.      Enviar solicitud HTTP GET a URL_API con timeout de 10 segundos
7.      Si respuesta HTTP 200 (éxito):
8.          Convertir respuesta a JSON y asignar a DATOS_CRUDOS
9.          Para cada PAIS en DATOS_CRUDOS:
10.             Extraer campos: nombre común, población, área, región,
subregión
11.             Procesar idiomas como lista de valores (Arreglo[1..5] DE
CADENA)
12.             Procesar monedas como registros con código y nombre
(Arreglo[1..3] DE REGISTRO)
13.             Calcular densidad → POBLACION / AREA (si AREA > 0)
14.             Agregar registro a PAISES usando estructura secuencial:
15.                 PAISES[i].NOMBRE_P → nombre común
16.                 PAISES[i].POBLACION_P → población
17.                 PAISES[i].AREA_P → área
18.                 PAISES[i].DENSIDAD_P → densidad calculada
19.                 PAISES[i].REGION_P → región
20.                 PAISES[i].SUBREGION_P → subregión
21.                 PAISES[i].IDIOMAS_P[j] → idioma
22.                 PAISES[i].MONEDAS_P[k].CODIGO_M → código de moneda
23.                 PAISES[i].MONEDAS_P[k].NOMBRE_M → nombre de moneda
24.         Retornar PAISES
25.      Sino:
26.         Mostrar mensaje: "Error: Fallo en la conexión con la API"
27.         Retornar vacío
28.  Excepto:
29.     Mostrar mensaje de error y retornar vacío
Fin MODULO_OBTENER_DATOS_PAISES

// MODULO_ESTRUCTURAR_DATOS_PAISES(DATOS)
1.  Inicio MODULO_ESTRUCTURAR_DATOS_PAISES(DATOS)
2.  Declarar paises_estructurados: Arreglo[1..250] DE REGISTRO
3.  Declarar nombre, region, subregion, idiomas, monedas: CADENA
```

```
4.  Declarar poblacion, area: ENTERO
5.  Declarar densidad: DECIMAL
6.  Para cada pais en DATOS:
7.      Si pais contiene campo "name.common":
8.          nombre → pais["name"]["common"]
9.      Sino:
10.         nombre → "N/A"
11.     poblacion → pais.get("population", 0)
12.     area → pais.get("area", 0)
13.     region → pais.get("region", "N/A")
14.     subregion → pais.get("subregion", "N/A")
15.     Si pais contiene "languages":
16.         idiomas → ", ".join(pais["languages"].values())
17.     Sino:
18.         idiomas → "N/A"
19.     Si pais contiene "currencies":
20.         monedas → ", ".join([f"{code} ({info['name']})" for code,
info in pais["currencies"].items()])
21.     Sino:
22.         monedas → "N/A"
23.     Si area > 0:
24.         densidad → Redondear(poblacion / area, 2)
25.     Sino:
26.         densidad → 0
27.     Agregar registro a paises_estructurados con campos normalizados
28. Retornar paises_estructurados
Fin MODULO_ESTRUCTURAR_DATOS_PAISES

// MODULO_CALCULAR_DENSIDAD(POBLACION, AREA)
1.  Inicio MODULO_CALCULAR_DENSIDAD(POBLACION, AREA)
2.  Si AREA > 0:
3.      Retornar Redondear(POBLACION / AREA, 2)
4.  Sino:
5.      Retornar 0
Fin MODULO_CALCULAR_DENSIDAD

// MODULO_GUARDAR_DATOS_JSON(DATOS, Hacer NOMBRE_ARCHIVO →
"DATOS_PAISES.JSON")
1.  Inicio MODULO_GUARDAR_DATOS_JSON(DATOS, Hacer NOMBRE_ARCHIVO →
"DATOS_PAISES.JSON")
2.  Intentar:
3.      Abrir NOMBRE_ARCHIVO en modo escritura con codificación UTF-8
4.      Guardar DATOS en formato JSON con indentación 4
5.      Cerrar archivo
6.      Mostrar mensaje: "Datos guardados en NOMBRE_ARCHIVO"
7.  Excepto:
8.      Mostrar mensaje de error
Fin MODULO_GUARDAR_DATOS_JSON

// MODULO_FILTRAR_PAISES_CON_REGEX(DATOS_ESTRUCTURADOS, PATRON_REGEX)
1.  Inicio MODULO_FILTRAR_PAISES_CON_REGEX(DATOS_ESTRUCTURADOS,
PATRON_REGEX)
2.  Declarar regex → Compilar(PATRON_REGEX, re.IGNORECASE)
3.  Declarar RESULTADOS: Arreglo vacío
```

```
4.  Para cada pais en DATOS_ESTRUCTURADOS:
5.      Si regex.search(pais["Nombre"]) no es nulo:
6.          Agregar pais a RESULTADOS
7.  Si RESULTADOS vacío:
8.      Mostrar mensaje: "No se encontraron coincidencias"
9.  Retornar RESULTADOS
Fin MODULO_FILTRAR_PAISES_CON_REGEX

// MODULO_ANALIZAR_ESTADISTICAS(DATOS_ESTRUCTURADOS, Hacer CAMPO →
"POBLACION")
1.  Inicio MODULO_ANALIZAR_ESTADISTICAS(DATOS_ESTRUCTURADOS, Hacer CAMPO
→ "POBLACION")
2.  Declarar VALORES: Lista vacía
3.  Para cada pais en DATOS_ESTRUCTURADOS:
4.      Si pais[CAMPO] > 0:
5.          Agregar pais[CAMPO] a VALORES
6.  Si longitud(VALORES) < 2:
7.      Mostrar mensaje: "Error: Se requieren al menos 2 valores válidos"
8.      Retornar vacío
9.  media → Promedio(VALORES)
10. mediana → Mediana(VALORES)
11. moda → Moda(VALORES)
12. varianza → Varianza(VALORES)
13. desviacion_std → RaízCuadrada(varianza)
14. Retornar diccionario con estadísticas calculadas
Fin MODULO_ANALIZAR_ESTADISTICAS

// MODULO_EXPORTAR_DATOS_EXCEL(DATOS, Hacer NOMBRE_ARCHIVO →
"DATOS_PAISES.xlsx")
1.  Inicio MODULO_EXPORTAR_DATOS_EXCEL(DATOS, Hacer NOMBRE_ARCHIVO →
"DATOS_PAISES.xlsx")
2.  Intentar:
3.      Convertir DATOS a DataFrame de pandas
4.      Exportar DataFrame a NOMBRE_ARCHIVO con engine="openpyxl"
5.      Mostrar mensaje: "Datos exportados exitosamente"
6.  Excepto:
7.      Mostrar mensaje de error
Fin MODULO_EXPORTAR_DATOS_EXCEL

// MODULO_GRAFICAR_DATOS(DATOS, Hacer TIPO_GRAFICO → "BARRAS", Hacer
TITULO → "GRAFICO", Hacer EJE_X → "X", Hacer EJE_Y → "Y", Hacer CAMPO_X →
"NOMBRE", Hacer CAMPO_Y → "POBLACION")
1.  Inicio MODULO_GRAFICAR_DATOS(DATOS, Hacer TIPO_GRAFICO → "BARRAS",
Hacer TITULO → "GRAFICO", Hacer EJE_X → "X", Hacer EJE_Y → "Y", Hacer
CAMPO_X → "NOMBRE", Hacer CAMPO_Y → "POBLACION")
2.  Extraer valores_x → [pais[CAMPO_X] para pais en DATOS]
3.  Extraer valores_y → [pais[CAMPO_Y] para pais en DATOS]
4.  Configurar gráfico:
5.      Tamaño figura → (12, 6)
6.      Título → TITULO
7.      Etiquetas ejes → EJE_X, EJE_Y
8.      Rejilla con estilo punteado
9.  Si TIPO_GRAFICO → "barras":
10.     Dibujar gráfico de barras con valores_x y valores_y
```

```
11. Sino si TIPO_GRAFICO → "lineas":
12.     Dibujar gráfico de líneas con marcadores
13. Sino:
14.     Mostrar mensaje: "Tipo de gráfico no válido"
15. Rotar etiquetas del eje X 45°
16. Guardar gráfico como PNG y mostrar temporalmente
Fin MODULO_GRAFICAR_DATOS

// MODULO_INTERPRETAR_RESULTADOS(ESTADISTICAS, Hacer DATOS_ORIGINALES →
"NONE", Hacer CAMPO → "POBLACION")
1.  Inicio MODULO_INTERPRETAR_RESULTADOS(ESTADISTICAS, Hacer
DATOS_ORIGINALES → "NONE", Hacer CAMPO → "POBLACION")
2.  Si media > mediana + 20%*media:
3.      Mostrar mensaje: "Alta asimetría positiva (valores extremos
altos)"
4.  Sino si mediana > media + 20%*media:
5.      Mostrar mensaje: "Alta asimetría negativa (valores extremos
bajos)"
6.  Sino:
7.      Mostrar mensaje: "Distribución simétrica"
8.  coeficiente_variacion → desviacion_std / media
9.  Si coeficiente_variacion > 1:
10.     Mostrar mensaje: "Alta dispersión entre valores"
11. Sino si coeficiente_variacion > 0.5:
12.     Mostrar mensaje: "Dispersión moderada"
13. Sino:
14.     Mostrar mensaje: "Baja dispersión"
15. Si DATOS_ORIGINALES no es None:
16.     pais_max → Max(DATOS_ORIGINALES, clave=CAMPO)
17.     pais_min → Min(DATOS_ORIGINALES, clave=CAMPO)
18.     Mostrar mensaje: "País con mayor {CAMPO}: {pais_max}"
19.     Mostrar mensaje: "País con menor {CAMPO}: {pais_min}"
Fin MODULO_INTERPRETAR_RESULTADOS

ALGORITMO_DE_PIA_SCRIPT

// DECLARACIÓN DE VARIABLES
DATOS_CRUDOS: LISTA DE DICCIONARIOS
DATOS_ESTRUCTURADOS: LISTA DE DICCIONARIOS
PATRON: CADENA
PAISES_FILTRADOS: LISTA DE DICCIONARIOS
ESTADISTICAS_POBLACION: DICCIONARIO
INTERPRETACION: CADENA

1. Inicio ALGORITMO_DE_PIA_SCRIPT()
2. Si nombre del programa es "__main__":
3.     // 1. Descargar datos desde la API REST Countries
4.     datos_crudos → obtener_datos_paises()
5.     Si datos_crudos no es vacío:
6.         // 2. Transformar datos crudos en estructura tabular
7.         datos_estructurados → estructurar_datos_paises(datos_crudos)
8.         // 3. Persistir datos en archivo JSON
9.         guardar_datos_json(datos_estructurados, "datos_paises.json")
10.        // 4. Filtrar países usando expresiones regulares
```

```
        Mostrar mensaje: "Ingrese un patrón de búsqueda (ej.: '^A' o
'land$'):"
        Leer entrada: patron
        paises_filtrados →
filtrar_paises_con_regex(datos_estructurados, patron)
        // 5. Mostrar resultados filtrados
        Si longitud(paises_filtrados) > 0:
            Mostrar mensaje: "Países que coinciden con el patrón
'{patron}':"
            Repetir para cada pais en paises_filtrados:
                Mostrar mensaje: "- {pais['Nombre']} (Región:
{pais['Región']}, Idiomas: {pais['Idiomas']})"
        Sino:
            Mostrar mensaje: "No se encontraron países que coincidan
con el patrón '{patron}'."
        FinSi
        // 6. Calcular estadísticas básicas sobre población
        Mostrar mensaje: "Análisis estadístico de población:"
        estadisticas_poblacion →
analizar_estadisticas(datos_estructurados, campo="Población")
        Si estadisticas_poblacion no es vacío:
            Repetir para cada clave, valor en
estadisticas_poblacion.items():
                Mostrar mensaje: "{clave}: {valor}"
        FinSi
        // 7. Exportar datos a Excel
        exportar_datos_excel(datos_estructurados, "datos_paises.xlsx")
        Si paises_filtrados no es vacío:
            exportar_datos_excel(paises_filtrados,
"paises_filtrados.xlsx")
        FinSi
        // 8. Visualizar datos con gráficos
        Mostrar mensaje: "Visualizando datos..."
        // Ejemplo 1: Top 10 países más poblados
        top_poblacion → ordenar(datos_estructurados, clave=lambda x:
x["Población"], descendente=True)[:10]
        graficar_datos(
            top_poblacion,
            tipo_grafico="barras",
            titulo="Top 10 Países por Población",
            eje_x="País",
            eje_y="Población",
            campo_x="Nombre",
            campo_y="Población"
        )
        // Ejemplo 2: Densidad poblacional de países filtrados
        Si paises_filtrados no es vacío:
            graficar_datos(
                paises_filtrados,
                tipo_grafico="lineas",
                titulo="Densidad Poblacional de Países Filtrados",
                eje_x="País",
                eje_y="Densidad (hab/km²)",
                campo_x="Nombre",
```

```
56.                 campo_y="Densidad (hab/km²)"
57.             )
58.         FinSi
59.         // 9. Interpretar resultados estadísticos
60.         Mostrar mensaje: "Interpretación del análisis de población:"
61.         interpretacion →
interpretar_resultados(estadisticas_poblacion, datos_estructurados,
campo="Población")
62.         Mostrar mensaje: interpretacion
63.         // 10. Interpretar estadísticas de área en países filtrados
64.         Si paises_filtrados no es vacío:
65.             Mostrar mensaje: "Interpretación del análisis de área en
países filtrados:"
66.             estadisticas_area →
analizar_estadisticas(paises_filtrados, campo="Área (km²)")
67.             Si estadisticas_area no es vacío:
68.                 interpretacion_area →
interpretar_resultados(estadisticas_area, paises_filtrados, campo="Área
(km²)")
69.                 Mostrar mensaje: interpretacion_area
70.             FinSi
71.         FinSi
72.     Sino:
73.         Mostrar mensaje: "No se pudieron obtener datos de la API."
74.     FinSi
75. Fin ALGORITMO_DE_PIA_SCRIPT
```