

Apellido y Nombre: .....

Fecha: ...../...../.....

**CONCEPTOS A TENER EN CUENTA****PROCEDIMIENTOS**

Un procedimiento es un tipo de subprograma o módulo compuesto por un bloque de sentencias, similar a las funciones, pero cuya invocación y retorno de resultados se realiza de forma diferente. Mientras que, para las funciones se obtiene un único resultado y los parámetros sólo representan valores de entrada, para los procedimientos los parámetros pueden ser opcionales y será posible generar tantos resultados como se desee (uno, varios o ninguno). Los resultados obtenidos podrán retornarse a través de la modificación de los parámetros del módulo destinados a tal fin (almacenamiento de resultados).

Respecto a la invocación o llamado a un módulo, las funciones se invocan desde una expresión, en tanto que, para los procedimientos el llamado se realiza como si se ejecutara una instrucción más.



**PROCEDIMIENTO:** refiere a un bloque de instrucciones que puede utilizarse como si de una operación simple se tratase. El bloque puede usar argumentos para ejecutar la tarea para la que fue diseñado.

**ARGUMENTOS DE UN PROCEDIMIENTO:** se constituyen los datos de entrada o salida que utilizará el módulo. Al invocar un procedimiento, sus argumentos deben coincidir en cantidad, orden y tipos con los especificados en la definición del módulo.

**RETORNO AL MAIN DESDE UN PROCEDIMIENTO:** Al finalizar la ejecución de un procedimiento el control del programa se transfiere a la instrucción indicada a continuación de la invocación del módulo.

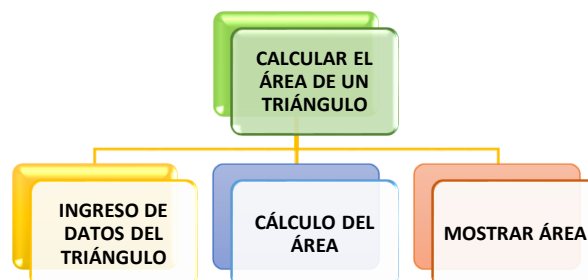
**EJERCICIOS RESUELTOS**

1. Diseñe un programa modular que calcule el área de un triángulo utilizando como datos la base y altura de éste. Codifique el diseño realizado en C/C++.

**Planteo de la Solución**

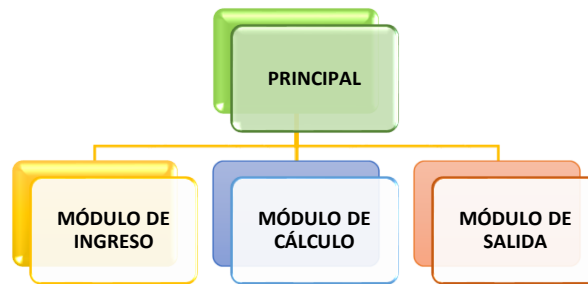
Al diseñar la solución para el problema propuesto deben aplicarse las técnicas de la PE que permitan reducir su complejidad. Para ello, será necesario dividir el problema original en subproblemas sencillos y luego integrar las soluciones de cada uno de ellos para obtener la solución general.

En este ejemplo el problema original es “calcular el área de un triángulo”, problema que puede dividirse en 3 subproblemas sencillos (como se muestra en el siguiente esquema): 1) obtener los datos del triángulo necesarios para el cálculo, 2) Realizar las operaciones asociadas al cálculo del área del triángulo (fórmula) y 3) la presentación del resultado obtenido al usuario.

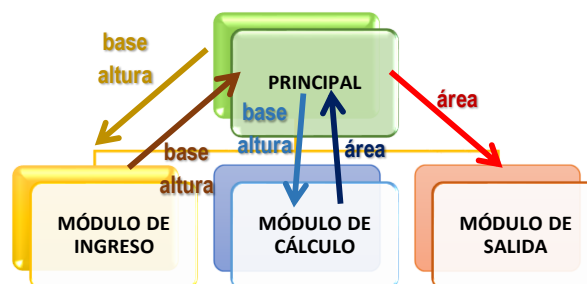


Si fuera necesario cada subproblema puede dividirse nuevamente en problemas más pequeños de modo que se reduzca la complejidad tanto como sea posible. Luego, deben diseñarse pequeños programas especializados en resolver estos subproblemas. Así, los subprogramas o módulos diseñados serán utilizados y controlados por un módulo especial llamado *principal*.

El siguiente esquema (diseño TOP-DOWN) muestra la estructura modular del programa.



El *módulo de ingreso* se ocupa de solicitar al usuario los valores necesarios para resolver el cálculo de área. El *módulo de cálculo* utiliza los datos capturados por el *módulo de ingreso* para obtener el área del triángulo. Finalmente, el *módulo de salida* presenta al usuario el resultado obtenido. El *módulo principal* se ocupa de invocar a los otros módulos en el orden apropiado para resolver el problema, comunicándose con ellos a través de parámetros. En el siguiente esquema se muestran los módulos que conformarán el programa y los datos que se comunican entre ellos.



En el caso del *módulo ingreso*, éste recibe los datos *base* y *altura* del *programa principal* y será el encargado de guardar en ellos la entrada del usuario (los datos entran y salen del módulo). Cuando un módulo tiene permitido modificar los valores que recibe se dice que la comunicación de datos (*pasaje de parámetros*) se realiza *por referencia*.

En el caso del *módulo cálculo* se utilizan los datos de *base* y *altura* (previamente cargados en el módulo ingreso) para generar un único valor de resultado (*área*). El módulo no modificará los datos que recibe sino que los utilizará para calcular un nuevo dato. Este tipo de comunicación de datos se denomina *por valor* (los datos recibidos no se modifican).

Por último, el *módulo salida* recibe el valor de *área* calculado pero no generará valores de retorno para el *programa principal* ya que sólo presentará un mensaje en pantalla. Este módulo también utiliza el *pasaje de parámetros por valor*.

El tipo de cada módulo se elige de acuerdo a los resultados que éste deba generar: el *módulo ingreso* será un procedimiento (genera 2 resultados), el *módulo cálculo* será una función (genera 1 resultado simple) y el *módulo salida* será un procedimiento (no genera valores de retorno).

A continuación se presenta, en pseudocódigo, el programa modular correspondiente.

```

PROGRAMA calculo_triángulo
VARIABLES
    base, altura, area: real
// Declaración de las variables base, altura y area utilizadas en el programa

// Procedimiento ingreso: carga los valores de base y altura de un triángulo
PROCEDIMIENTO ingreso(E/S b: real, E/S h: real)
INICIO
    ESCRIBIR "Ingrese la base del triángulo:"
    LEER b
    ESCRIBIR "Ingrese la altura del triángulo:"
    LEER h
FIN
  
```

```
// Función Calculo: calcula el área de un triángulo usando los valores de base y altura
FUNCIÓN calculo(E b:real, E h:real): real
INICIO
    calculo ← b * h / 2
FIN

// Procedimiento Salida: presenta en pantalla el valor del área calculada
PROCEDIMIENTO salida (E a: real)
INICIO
    ESCRIBIR "El área calculada es:", a
FIN

// Programa Principal: llama a los procedimientos ingreso y salida, y a la función calculo
INICIO
    ingreso(base, altura)
    area ← calculo(base, altura)
    salida(area)
FIN
```

La codificación en lenguaje C/C++ del programa anterior se presenta a continuación:

```
// PROGRAMA calculo_triangulo
#include <iostream>
#include <stdlib.h>

void ingreso(float &b, float &h);
float calculo(float b, float h);
void salida(float a);

// Principal: llama a los módulos ingreso, calculo y salida
main()
{ // Declaración de las variables base, altura y area utilizadas en el programa
    float base, altura, area;
    ingreso(base, altura);
    area = calculo(base, altura);
    salida(area);
    system("pause");
}

// Procedimiento Ingreso: carga los valores de base y altura de un triángulo
void ingreso(float &b, float &h)
{
    cout << "Ingrese la base:";
    cin >> b;
    cout << "Ingrese la altura:";
    cin >> h;
}

// Función Cálculo: calcula el área de un triángulo con los datos de base y altura
float calculo(float b, float h)
{
    return(b * h / 2);
}

// Procedimiento Salida: presenta en pantalla el valor del área calculada
void salida(float a)
{
    cout << "El area calculada es: " << a << endl;
}
```

- Analice el siguiente programa y determine el valor de las variables *a* y *b* tras la ejecución de los procedimientos *pvalor* y *preferencia*.

El programa *prueba\_parámetros* ilustra el paso de parámetros por Valor y por Referencia.

```

PROGRAMA prueba_parámetros
VARIABLES
  a, b: entero

PROCEDIMIENTO pvalor (E x: entero, E y: entero)
INICIO
  x ← 6
  y ← x*2
FIN

PROCEDIMIENTO preferencia (E/S m: entero, E/S n: entero)
INICIO
  m ← n-1
  n ← m+n
FIN

INICIO
  a ← 7
  b ← 5
  Escribir "Valores Originales", a, b
  ESCRIBIR "Pasando por Valor"
  pvalor(a,b)
  ESCRIBIR "La variable a vale:", a
  ESCRIBIR "La variable b vale:", b
  ESCRIBIR "Pasando por Referencia"
  preferencia(a,b)
  ESCRIBIR "La variable a vale:", a
  ESCRIBIR "La variable b vale:", b
FIN

```

En el programa se utilizan 2 procedimientos que ilustran el paso de parámetros por *VALOR* y por *REFERENCIA*. Observe que los valores iniciales de las variables *a* y *b* se muestran antes de invocar los procedimientos *pvalor* y *preferencia*.

El procedimiento *pvalor* recibe parámetros pasados por *VALOR*, es decir, trabaja con copias de las variables del programa que realiza la invocación. Las modificaciones realizadas sobre las variables *x* y *y* (en el procedimiento) no alteran los valores de las variables *a* y *b* (en el programa principal).

El procedimiento *preferencia* recibe parámetros pasados por *REFERENCIA*, es decir, trabaja directamente con las variables del programa que lleva a cabo la invocación. Los cambios realizados sobre las variables *m* y *n* (en el procedimiento) modifican los valores de las variables *a* y *b* (en el programa principal).

3. Diseñe un programa modular que permite intercambiar el contenido de 2 variables asignadas por el usuario.

```

PROGRAMA intercambio_datos
VARIABLES
  primero, segundo: entero

PROCEDIMIENTO intercambio (E/S num1: entero, E/S num2: entero)
VARIABLES
  auxiliar: entero
INICIO
  auxiliar ← num1
  num1 ← num2
  num2 ← auxiliar
FIN

INICIO
  ESCRIBIR "Ingrese primer número:"
  LEER primero
  ESCRIBIR "Ingrese segundo número:"
  LEER segundo
  intercambio(primero,segundo)
  ESCRIBIR "VALORES CAMBIADOS"
  ESCRIBIR "Primero: ", primero
  ESCRIBIR "Segundo: ", segundo
FIN

```

El programa permite intercambiar el contenido de las variables *primero* y *segundo*, cuyos valores iniciales son asignados por el usuario. El intercambio se realiza mediante un procedimiento que utiliza 2 parámetros (pasados por referencia) que se corresponderán con los valores *primero* y *segundo*.

El módulo *intercambio*, usa una variable *auxiliar* (variable local), para pasar el contenido del primer valor al segundo sin perder ninguno.

**VARIABLES LOCALES:** las variables declaradas dentro de los procedimientos o funciones, son variables locales o privadas de los mismos y solo están disponibles en su propio ámbito.

4. Dada la siguiente expresión:

$$\prod_{k=1}^{total} \frac{m^k + n \times k}{k!}$$

Diseñe el **procedimiento/función** que realice el cálculo teniendo en cuenta las siguientes consideraciones:

- Los productos se calculan utilizando la función *producto* (mediante sumas sucesivas) cuyo bucle de cálculo esté controlado por *BANDERA*.
- Las potencias se calculan utilizando la función *potencia* (mediante productos sucesivos) cuyo bucle de cálculo esté controlado por *CONTADOR*.
- Los cálculos de factorial se resuelven utilizando la función *factorial*, implementada con estructuras *MIENTRAS*.
- Diseñe un procedimiento de carga de datos que controle que los valores ingresados sean positivos.

Indique en cada caso el paso de parámetros utilizado.

### Resolución

A fin de resolver el problema planteado es necesario identificar las entradas de datos, el objetivo del programa y las salidas o resultados a producir. Los valores *m*, *n* y *total* corresponden a los datos de entrada que proporcionará el usuario. Con estos datos el programa debe calcular la *productoria* indicada y presentar al usuario el resultado del cálculo (un valor real).

La solución propuesta se plantea de forma modular, lo que permite simplificar la estructura del programa. El ingreso de datos se implementa mediante un procedimiento que trabaja con parámetros por REFERENCIA (se modificarán los datos). Luego, el cálculo de la *productoria*, que utiliza los valores *m*, *n* y *total*, se realiza mediante una función (resultado simple). Esta función, a su vez, se vale de las funciones *producto*, *potencia* y *factorial* para efectuar el cálculo:

- Función Producto: usa 2 parámetros enteros y mediante sumas sucesivas calcula el producto de 2 números enteros.
- Función Potencia: usa 2 parámetros enteros y mediante productos sucesivos calcula la potencia de un número entero elevado a otro.
- Función Factorial: usa 1 parámetro entero y mediante productos acumulados calcula el factorial de 1 número entero.

Para realizar el cálculo, la función *productoria* ejecuta un bucle que acumula los términos generados en cada iteración (repetición). Cada término se obtiene utilizando las funciones *producto*, *potencia* y *factorial* con los valores correspondientes a la iteración actual. El resultado final se muestra en el programa principal.

A continuación se presenta el programa, codificado en C/C++, que resuelve el problema planteado.

```
#include<iostream>
#include<stdlib.h>

using namespace std;

// Declaracion de funciones y procedimientos
void ingreso(int &num1,int &num2, int &num3);
int producto (int nro1, int nro2);
int potencia (int nro1, int nro2);
int factorial (int nro1);
float productoria(int m, int n, int total);

// Programa Principal
main()
{
    int m,n,total; // Variables del programa principal
    ingreso(m, n, total); //Invocación del procedimiento
    cout<<"Resultado:"<<productoria(m,n,total)<<endl; //Invocación de la fcion productoria
    system("pause");
}
```

```
//Definición del procedimiento que permite el ingreso de datos
```

```
void ingreso(int &num1,int &num2, int &num3)
{
    do
    { cout<<" Ingrese primer numero para m:"<<endl;
      cin>> num1;
      cout<<" Ingrese segundo numero para n:"<<endl;
      cin>> num2;
      cout<<" Ingrese tercer numero para total:"<<endl;
      cin>> num3;
    }while((num1<0) || (num2<0) || (num3<0));
}
```

```
//Definición de la función que calcula el producto por sumas sucesivas
```

```
int producto (int nro1, int nro2)
{
    int p=0;
    bool band=true;
    do
    { p=p+nro1;
      nro2--;
      if (nro2==0)
          band=false;
    } while(band==true);
    return p;
}
```

```
//Definición de la función que calcula la potencia por productos sucesivos
```

```
int potencia (int nro1, int nro2)
{
    int pot=1, c=0;
    while(c<nro2)
    {
        pot=pot*nro1;
        c++;
    }
    return pot;
}
```

```
//Definición de la función que calcula el factorial por productos sucesivos
```

```
int factorial (int nro1)
{
    int f=1;
    while (nro1>=1)
    {
        f=f*nro1;
        nro1--;
    }
    return f;
}
```

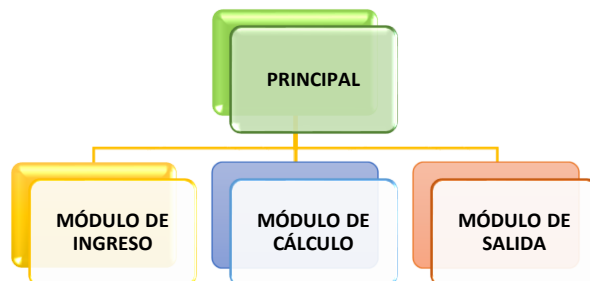
```
//Definición de la función que calcula la productoria invocando las funciones definidas previamente
```

```
float productoria(int m, int n, int total)
{
    int i;
    float prod=1;
    for(i=1;i<=total;i++)
        prod=prod*((potencia(m,i)+producto(n,i))/factorial(i));
    return prod;
}
```

## EJERCICIOS A RESOLVER

Para desarrollar cada programa modular tome como referencia el siguiente esquema (diseño TOP-DOWN):

- entrada de datos: el programa debe incluir un módulo para la captura y validación de la entrada del usuario
- operaciones de transformación o cálculo: el programa puede utilizar uno o más módulos para resolver los cálculos necesarios para obtener el resultado final.
- presentación de resultados: el programa debe contar con un módulo para mostrar al usuario los resultados obtenidos.



1. Sabiendo que el siguiente programa permite calcular el área de un triángulo rectángulo, a partir de los valores de *base* y *altura*, escribe una versión modular equivalente:

```

PROGRAMA ejemplo_trianguulo
VARIABLES
    base, altura:REAL
    area:REAL
INICIO
    REPETIR
        ESCRIBIR "Ingrese base: "
        LEER base
        ESCRIBIR "Ingrese altura: "
        LEER altura
    HASTA QUE (base>0 Y altura>0)
    area←base*altura
    ESCRIBIR "El triángulo de base ",base," y altura ",altura
    ESCRIBIR "Tiene área ",area
FIN
  
```

2. Analiza el siguiente programa e identifica el propósito de cada módulo. ¿Qué ocurre si no se utiliza el pasaje de parámetros por referencia en los módulos *misterio* y *enigma*?

```

#include <iostream>

using namespace std;

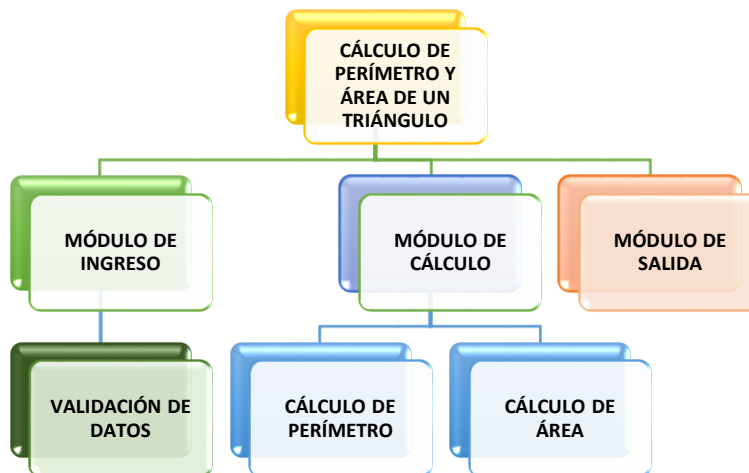
void misterio(int &a, int &b);
void enigma(int &m, int &n);

main()
{ int num1,num2,num3;
  cout << "Ingrese primer valor: ";
  cin >> num1;
  cout << "Ingrese segundo valor: ";
  cin >> num2;
  cout << "Ingrese tercer valor: ";
  cin >> num3;
  enigma(num1,num2);
  enigma(num2,num3);
  enigma(num1,num2);
  cout << "Num1: " << num1 << endl;
  cout << "Num2: " << num2 << endl;
  cout << "Num3: " << num3 << endl;
}

void enigma(int &m, int &n)
{
    if (m-n>0)
        misterio(m,n);
}

void misterio(int &a, int &b)
{
    if (a-b>0)
    {
        a=a-b;
        b=b+a;
        a=b-a;
    }
    else
    if (b-a>0)
    { b=b-a;
      a=a+b;
      b=a-b;
    }
}
  
```

3. Tomando como referencia el siguiente diagrama *Top-Down*, codifica un programa que calcule el perímetro y área de un triángulo (fórmula de Herón) a partir del valor de sus lados. Para ello, considere lo siguiente:
- Módulo de Ingreso: permite capturar la entrada del usuario (lados del triángulo) garantizando que los valores sean válidos para realizar el cálculo.
  - Módulo de Cálculo: permite calcular el perímetro y el área de un triángulo, cada cálculo se implementa en un módulo separado.
  - Módulo de Salida: permite presentar en pantalla los resultados del cálculo de perímetro y área.



4. Codifica un programa modular que permita aplicar el Teorema de Pitágoras para calcular la hipotenusa de un triángulo rectángulo. Ten en cuenta que el programa debe incluir un módulo de ingreso de datos (con validación de datos), y un módulo de cálculo de hipotenusa. Considera que el cálculo de raíces debe realizarse aplicando el método de aproximación visto en el ejercicio 11 del TP5. Indica el pasaje de parámetros utilizado.
5. Codifica un programa modular que permita calcular el cociente y resto de la división entera entre 2 valores ingresados por el usuario, considerando 2 propuestas para llevar a cabo el cálculo:
- una función que permita calcular, según un parámetro de opción, el cociente o resto de la división (mediante restas).
  - un procedimiento que permita calcular el cociente y resto de la división entera (mediante restas).
- Considera que el programa sólo admitirá valores positivos para realizar el cálculo. Indica el pasaje de parámetros utilizado.
6. Codifica un programa modular que permita calcular las raíces de una ecuación cuadrática, considerando 2 propuestas para llevar a cabo el cálculo:
- una función que permita calcular, según un parámetro de opción, la primera (+) o la segunda (-) raíz de la ecuación.
  - un procedimiento que permita calcular las 2 raíces de la ecuación cuadrática.
- Considera que el programa debe validar los coeficientes de la ecuación de modo que sólo se acepten valores que permitan realizar el cálculo. Indica el pasaje de parámetros utilizado.
7. Una propiedad matemática interesante es que un número entero positivo puede expresarse como el producto de valores primos. A partir del método explicado en <https://www.youtube.com/watch?v=NPaBFe6QBDQ> codifica un módulo, llamado FPRIMO, que muestre los factores primos correspondientes a un valor ingresado por el usuario. Utiliza en la solución propuesta los módulos de cálculo de producto, cociente, resto, potencia, cuadrado, cubo, etc. desarrollados en el TP7.
8. Codifica un programa modular que presente un menú de opciones que permita: 1) Generar un término de la serie de Fibonacci solicitado por el usuario, 2) Determinar si un número ingresado por el usuario pertenece o no a la serie de Fibonacci, 3) Mostrar  $N$  términos de la serie de Fibonacci y 4) Salir del programa. Considera que cada opción debe implementarse mediante módulos.



9. Codifica un programa modular que implemente las siguientes operaciones de triángulos: 1) Ingresar lados del triángulo, 2) Calcular área del triángulo (fórmula de Herón, 3) Determinar tipo de triángulo (equilátero, isósceles, escaleno), 4) Identificar triángulo rectángulo y 5) Salir del Programa. Ten en cuenta que el módulo de ingreso debe verificar que el valor de los lados sea válido para construir el triángulo. Cada opción debe implementarse mediante módulos, debiendo controlarse que las opciones sólo se ejecuten si se dispone de los datos necesarios. Indica cómo se pasan los parámetros.
10. Codifica un programa modular que presente un menú de opciones para realizar las operaciones de una calculadora sencilla. Considera que tanto el menú como las operaciones deben implementarse a través de módulos. Incluye las validaciones necesarias para los datos que se utilizarán en cada operación.
- |   |                                 |
|---|---------------------------------|
| a) Producto (mediante sumas)                            | g) Raíz cuadrada                |
| b) Cociente (mediante restas)                           | h) Factorial                    |
| c) Resto (mediante restas)                              | i) Inverso (dígitos invertidos) |
| d) Cuadrado (suma de impares)                           | j) Generador de capicúa         |
| e) Cubo (suma de impares)                               | k) Término Fibonacci            |
| f) Cuarta (potencia cuarta mediante de suma de impares) | l) Factores primos              |



11. Dados los siguiente módulos:

**A) PROCEDIMIENTO** omega (E n:ENTERO, E/S p:ENTERO, E/S q:ENTERO)

```
VARIABLES
  x:ENTERO
INICIO
  p←n MOD 10
  q←n MOD 10
  MIENTRAS (n>0) HACER
    x←n MOD 10
    n←n DIV 10
    SI (p>x) ENTONCES
      p←x
    FIN_SI
    SI (q<x) ENTONCES
      q←x
    FIN_SI
  FIN_MIENTRAS
FIN
```

**B) int alfa(int x, int y)**  
 { int z=0;  
 do  
 { if (y>0)  
 { z=z+x;  
 y--;  
 }  
 } while(y!=0);  
 return z;  
 }

```
void beta(int a, int &b)
{ b=1;
  while (a>0)
  { b=alfa(b,a);
    a--;
  }
}
```

- a) Realice la prueba de escritorio para los valores: n=9847 y n=5728
- b) Determine el objetivo del módulo (e indique el retorno esperado).
- c) Codifique en C/C++ el módulo.

- a) Realice la prueba de escritorio para los valores: a=5 y a=7.
- b) Determine el objetivo del módulo.
- c) Escriba el pseudocódigo equivalente.

C) PROCEDIMIENTO secreto(E n:ENTERO)

VARIABLES

a,b,c,i:ENTERO

INICIO

b←1

c←1

i←1

MIENTRAS (i≤n) HACER

SI i≤2 ENTONCES

ESCRIBIR c

SINO

a←b

b←c

c←a+b

ESCRIBIR c

FIN\_SI

i←i+1

FIN\_MIENTRAS

FIN

D) PROCEDIMIENTO omega(E/S n:ENTERO, E m:LÓGICO)

VARIABLES

x,p:ENTERO

INICIO

SI m=FALSO ENTONCES

x←n\*(n-1)+1

SINO

x←1

FIN\_SI

p←0

MIENTRAS (n>0) HACER

p←p+x

x←x+2

n←n-1

FIN\_MIENTRAS

n←x

FIN

E) PROCEDIMIENTO misterio(E/S a:ENTERO, E/S b:ENTERO,  
E/S c:ENTERO)

VARIABLES

b1,b2:LÓGICO

INICIO

REPETIR

ESCRIBIR "Ingrese valor: "

LEER a

ESCRIBIR "Ingrese valor: "

LEER b

ESCRIBIR "Ingrese valor: "

LEER c

b1←a>0 Y b>0 Y c>0

b2←(a+b>c) Y (a+c>b) Y (b+c>a)

HASTA QUE (b1=Verdadero) Y (b2=Verdadero)

FIN

a) Realice la prueba de escritorio para los valores:

n=4 y n=6

b) Determine el objetivo del módulo.

c) Codifique en C/C++ el módulo.

a) Realice la prueba de escritorio para los valores:

n=3, m=VERDADERO y n=5, m=FALSO.

b) Determine el objetivo del módulo.

c) Codifique en C/C++ el módulo.

a) Realice la prueba de escritorio para los valores:

a=3, b=4 y c=5; a=3, b=9 y c=2.

b) Determine el objetivo del módulo.

c) Codifique en C/C++ el módulo.

