Trabajo Práctico N° 7

Modularidad: Procedimientos

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++.

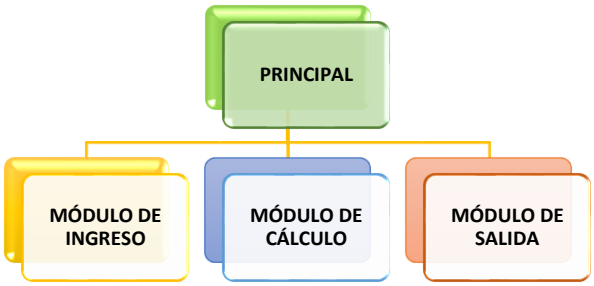
Planteo de la Solución

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.

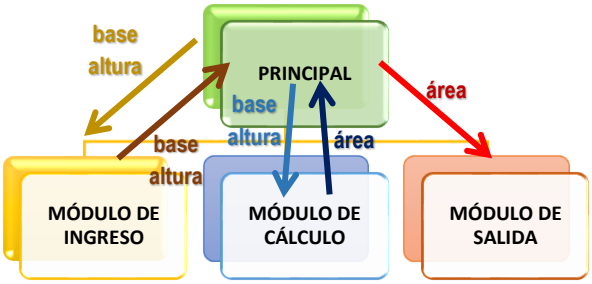
En este ejemplo el problema original es *"calcular el área de un triángulo"*, problema que puede dividirse en 3 subproblemas sencillos (como se muestra en el siguiente esquema): 1) obtener los datos del triángulo necesarios para el cálculo, 2) Realizar las operaciones asociadas al cálculo del área del 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_triangulo
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 al procedimiento Leer_datos y a la función Calculo_area
INICIO
   leer_datos(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 y 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 el pantalla el valor del área calculada
void salida(float a)
{
   cout << "El area calculada es: " << a << endl;
}
```

2. 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* e *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:

a) Los productos se calculan utilizando la función *producto* (mediante sumas sucesivas) cuyo bucle de cálculo esté controlado por *BANDERA*.

b) Las potencias se calculan utilizando la función *potencia* (mediante productos sucesivos) cuyo bucle de cálculo esté controlado por *CONTADOR*.

c) Los cálculos de factorial se resuelven utilizando la función *factorial*, implementada con estructuras *MIENTRAS*.

d) 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

1. Codifique un programa modular que INTERCAMBIE el contenido de dos números ingresados por el usuario, sin utilizar variables auxiliares. Tenga en cuenta que en el TP3 se mostró cómo realizar el intercambio sin variables extra. Indique el pasaje de parámetros utilizado.
2. Codifique un programa modular que ORDENE de forma creciente 3 valores (a, b, c) ingresados por el usuario. Considere que la ordenación se realiza por intercambio de valores entre las variables indicadas. Utilice el módulo de intercambio diseñado en el ítem 1. Indique el pasaje de parámetros utilizado.
3. Codifique un programa modular que, dada una serie de valores ingresada por el usuario, determine el máximo y mínimo valor **primo** de la serie y calcule el resto del cociente entre el máximo y mínimo obtenidos. Tenga en cuenta que el ingreso de datos finalizará a petición del usuario y que el cociente sólo podrá calcularse si se obtuvieron los valores para el dividendo y divisor.
4. Codifique un programa modular que presente un menú con las siguientes opciones: 1) Ingresar 4 valores distintos, 2) Ordenar (de forma decreciente) los 4 valores, y 3) Calcular el factorial del mayor valor, y 4) Salir, que presente el mensaje "Fin de programa". Considere que la opción 2 sólo está disponible si la opción 1 se ejecutó antes y que la opción 3 sólo se habilita si antes se ejecutó la opción 2. Cada opción debe implementarse mediante procedimientos y/o funciones, indicando el pasaje de parámetros utilizado.
5. Codifique un programa modular que presente un menú con las siguientes opciones: 1) Determinar el máximo y mínimo de 10 valores ingresados por el usuario, 2) Calcular el cociente (mediante restas) entre los valores máximo y mínimo, y 3) Salir, que presente el mensaje "Fin de programa". Considere que en la opción 1 se debe controlar que sólo se tengan en cuenta valores positivos. Además, adicione los controles necesarios para que la opción 2 sólo está disponible si la opción 1 se ejecutó antes. Cada opción debe implementarse mediante procedimientos y/o funciones, indicando el pasaje de parámetros utilizado.
6. Codifique un programa modular que calcule las raíces de una ecuación cuadrática. Para ello, considere que el programa presentará un menú con las siguientes opciones: 1) Ingresar los coeficientes de una ecuación cuadrática, 2) Determinar las raíces de la ecuación cuadrática (siempre que sea posible), 3) Indicar si las raíces de la ecuación son iguales, distintas o son imaginarias (no existen en el campo de los reales) y 4) Salir del Programa. Considere que cada opción se implementa mediante módulos, y que debe controlarse que las opciones que dependen de otras no se ejecuten antes de aquellas. Indique cómo se pasan los parámetros.
7. Dada la siguiente expresión:

$$\sum_{i=1}^{N} 2 * i - 1$$

   a) Diseñe (pseudocódigo) el **procedimiento/función** que realice el cálculo teniendo en cuenta las siguientes consideraciones:
      1. Los productos se calculan utilizando la ***función producto*** (mediante sumas sucesivas) cuyo bucle se implemente con estructuras *MIENTRAS*.
      2. Diseñe un ***procedimiento de carga de datos*** que controle que el valor de *N* sea positivo o cero.

      Indique en cada caso el paso de parámetros utilizado.
   b) Realice la **prueba de escritorio** para *N=4,* y para *N=3* determine el valor de la función en cada caso.
   c) Indique el **objetivo de la función**.
8. Dada la siguiente expresión:

$$\sum_{k=1}^{total} \frac{a^k + n \bmod k}{a \times k!}$$

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

a) Los ***productos*** se calculan utilizando la función *producto* (mediante sumas sucesivas) cuyo bucle de cálculo esté controlado por *BANDERA*.

b) Las ***potencias*** se calculan utilizando la función *potencia* (mediante productos sucesivos) cuyo bucle de cálculo esté implementado mediante estructuras *PARA*.

c) Los cálculos de ***factorial*** se resuelven utilizando la función *factorial*, implementada con estructuras *REPETIR*.

d) Los cálculos de ***módulo*** se resuelven utilizando la función *modulo* (mediante restas sucesivas) cuyo bucle de cálculo esté implementado mediante estructuras *MIENTRAS*.

e) Diseñe un procedimiento de carga de datos que controle que los valores ingresados sean positivos, distintos de cero.

Indique en cada caso el paso de parámetros utilizado.

9. Dados los siguiente módulos:

```
PROCEDIMIENTO enigma(E m: ENTERO, E/S n: ENTERO)
VARIABLES
   b: LÓGICO
INICIO
   b<-m<>0
   n<-0
   m<-m*2
   MIENTRAS b=VERDADERO HACER
    b<-FALSO
    SI m<>0 ENTONCES
      SI m mod 2<>0 ENTONCES
         n<-n+m
      FIN_SI
      b<-Verdadero
    FIN_SI
    m<-m-1
   FIN_MIENTRAS
FIN
```

a) Realice la prueba de escritorio para los valores: m=3 y m=4.

b) Determine el objetivo del módulo.

c) Codifique en C/C++ el módulo.

```
FUNCIÓN misterio (E a:entero):??
VARIABLES
  t: ENTERO
  bandera: LÓGICO
INICIO
   bandera<-a>=0
   t<-1
   MIENTRAS bandera=VERDADERO HACER
     SI a=0 O a=1 ENTONCES
        t<-1
        bandera<-FALSO
     SINO
        t<-t*a
        a<-a-1
        SI a < 2 ENTONCES
          bandera<-FALSO
        FIN_SI
     FIN_SI
   FIN_MIENTRAS
FIN
```

a) Realice la prueba de escritorio para los valores: a=4 y a=5

b) Determine el objetivo del módulo (e indique el retorno esperado).

c) Codifique en C/C++ el módulo.

```
void oculta(int a, int b, int &c)
{int i=b;
 bool band;
  c=0;
  band=a-b>=0;
  while (band==true)
   { i=i+b;
     c++;
     if (i>a)
       band=false;
   }
}
```

a) Realice la prueba de escritorio para los valores: a=17 y b=5; a=18 y b=3.

b) Determine el objetivo del módulo.

c) Escriba el pseudocódigo equivalente..