PROGRAMACIÓN ESTRUCTURADA

06

Trabajo Práctico N° 6

Modularidad: Funciones

Apellido y Nombre: …… *Fecha: ……/……/……*

CONCEPTOS A TENER EN CUENTA

Como ya se ha visto en los trabajos prácticos previos, los problemas sencillos pueden resolverse fácilmente con programas simples de pocas instrucciones. Sin embargo, al tratar con problemas significativamente más complejos será necesario diseñar programas de cientos o miles de instrucciones. Consecuentemente, la detección y eliminación de errores se vuelve una tarea difícil de realizar.

A fin de simplificar la comprobación y mantenimiento de programas, la Programación Modular propone que el conjunto de instrucciones de un programa se divida en unidades independientes (módulos, subprogramas o subrutinas) que realicen tareas específicas. Así, cada una de estas unidades o módulos puede diseñarse, verificarse y mantenerse por separado. Los programas modulares cuentan con un módulo especial, llamado principal, que coordina el trabajo de los restantes subprogramas. Según su comportamiento, los módulos pueden clasificarse en: funciones y procedimientos. Una función, al igual que en matemáticas, comprende una serie de operaciones que deben aplicarse a valores (argumentos) usados por la función para calcular un único resultado simple.

El tema de procedimientos será tratado en el siguiente práctico.

SUBPROGRAMA: *conjunto de sentencias de un programa que realiza determinada tarea y que puede ser ejecutada desde uno o más puntos del programa principal con la simple mención de su nombre y posee todas las características propias de un programa.*

CONTROL DEL PROGRAMA: *Al efectuarse el llamado a un subprograma, el control del programa se transfiere a este módulo independiente el cual después de realizar la tarea encomendada retorna nuevamente el control al programa llamador o principal continuando este la ejecución normal del resto de sus instrucciones.*

PROGRAMA PRINCIPAL (*main*): *modulo que actúa como coordinador el cual controla y relaciona a todos los subprogramas.*

EJERCICIOS RESUELTOS

1. El siguiente programa calcula el cociente entero, mediante restas sucesivas, de 2 números ingresados por el usuario. Modifíquelo para obtener un programa modular, diseñando un módulo especial para el cálculo del cociente.

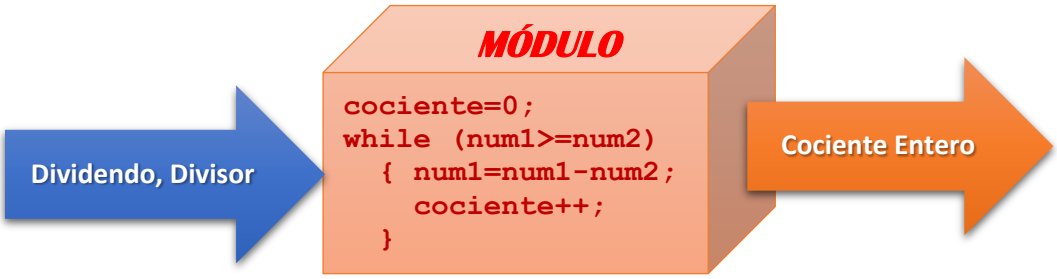
```
#include <iostream>
#include <stdlib.h>
using namespace std;
main()
{ int num1,num2,cociente;
  cout << "Ingrese dividendo: ";
  cin >> num1;
  cout << "Ingrese un divisor: ";
  cin >> num2;
  cociente=0;
  while (num1>=num2)
    { num1=num1-num2;
      cociente++;
    }
  cout << "El cociente es: " << cociente;
  system("pause");
}
```

Instrucciones asociadas a la entrada de datos

Instrucciones asociadas al cálculo del cociente

Instrucciones asociadas a la presentación de resultados

En primer lugar, es necesario identificar las partes del programa asociadas a la entrada, proceso y salida. En este caso, el proceso corresponde al cálculo del cociente mediante restas y será este conjunto de instrucciones el que se utilice para construir el módulo. Para ello, es preciso identificar qué datos serán requeridos por el módulo y cuál será el resultado a generar.



Los valores que deben suministrarse al módulo constituyen los parámetros o argumentos de éste. Mientras que, el resultado o valor de salida determinará el tipo de módulo que se diseñará. Cuando el resultado a generar por un módulo es un valor simple (entero, real, carácter o lógico) se diseñará una función. La función contendrá todas las operaciones de "cálculo" necesarias para obtener, a partir de los parámetros, el resultado final.

A continuación se presenta el código de un programa modular que cuenta con un módulo tipo función para el cálculo del cociente:

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

using namespace std;

int division(int n1,int n2);

main()
{
  int num1,num2,resultado;
  cout << "Ingrese divisor: ";
  cin >> num1;
  cout << "Ingrese dividendo: ";
  cin >> num2;
  resultado=division(num1,num2);
  cout << "Resultado: " << resultado << endl;
  system("pause");
}

int division(int n1,int n2)
{
  int cociente=0;
  while(n1 >= n2)
   { n1=n1-n2;
     cociente++;
   }
  return cociente;
}
```

Prototipo de la función
Argumentos de la función
Tipo de dato de la función
Módulo Principal
Llamada o Invocación
Parámetros Actuales
Definición de la función
Parámetros Formales
Valor de retorno

Como puede observarse al escribir un programa modular se añaden algunos elementos a la estructura básica de programa:

- Prototipo del módulo (función o procedimiento): define qué módulos que serán utilizados en el programa especificando su tipo, nombre y parámetros.
- Llamada o invocación del módulo (función o procedimiento): indica qué módulo será utilizado en ese momento y cuáles serán los valores con los que operará.
- Definición del módulo (función o procedimiento): indica qué datos serán utilizados por el módulo (parámetros formales), el conjunto de operaciones a ejecutar, y el/los resultados a generar. En el caso de las funciones se incluye la instrucción *return* para devolver el resultado del cálculo.

También puede observarse que la estructura del programa principal (*main*) se simplifica al reducir a una sola línea el cálculo del cociente. La "complejidad" de este cálculo queda confinada al módulo que implementa la operación.

EJERCICIOS A RESOLVER

1. El siguiente programa utiliza el módulo COMPARAR para verificar si 2 caracteres introducidos por el usuario son iguales, el primero menor que el segundo o el primero mayor que el segundo. Desafortunadamente parte del código se dañó, ¿serías capaz de restaurarlo?

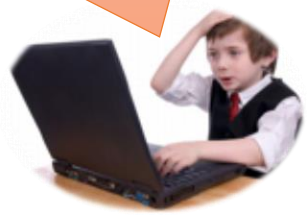
```
#include <iostream>
# include <stdlib.h>

using namespace std;

???? comparar(????,????)

main()
{ char primero, segundo;
  ???? c;
  cout << "Ingrese primer carácter: ";
  cin >> primero;
  cout << "Ingrese segundo carácter: ";
  cin >> segundo;
  ????comparar(primero,segundo)
  if (c==0)
    cout << "IGUALES" << endl;
  else
    if (c==1)
      cout << primero << " es menor que " << segundo << endl;
    else
      cout << segundo << " es menor que " << primero << endl;
}

???? comparar(????, ????)
{int x;
  if (a==b)
     x=0;
  else
    if (a<b)
       x=1;
    else
       x=2;
  return ????;
}
```

2. Codifique un módulo, llamado MAYUSCULA, que determine si un carácter ingresado por el usuario es mayúscula o no. Suponga que el carácter de entrada es ingresado en el programa principal. Indique el pasaje de parámetros utilizado.
3. Codifique un módulo, llamado MINUSCULA, que convierta una letra mayúscula introducida por el usuario a su equivalente minúscula. Suponga que el carácter de entrada es ingresado en el programa principal y que aquellos caracteres que no sean mayúsculas no serán modificados por el módulo. Indique el pasaje de parámetros utilizado.
4. Codifique un módulo, llamado FACTORIAL, que implemente el cálculo utilizando estructuras REPETIR y el criterio de finalización por BANDERA. Suponga que el valor a utilizar en el cálculo es ingresado en el programa principal. Indique el pasaje de parámetros utilizado.
5. Codifique un módulo, llamado PRODUCTO, que calcule el producto de 2 números enteros mediante sumas sucesivas. Considere que el cálculo se realizará utilizando estructuras MIENTRAS y el criterio de finalización por VALOR CENTINELA. Además suponga que los valores a multiplicar son ingresados en el programa principal. Indique el pasaje de parámetros utilizado.
6. Codifique la función POTENCIA mediante productos sucesivos, implementada con estructuras MIENTRAS y finalización por VALOR CENTINELA. Suponga que la base y el exponente son ingresados en el programa principal. Indique el pasaje de parámetros utilizado.

7. Codifique un módulo, llamado CUADRADO, que calcule el cuadrado de un número entero *N* mediante la suma de los *N* primeros impares. Suponga que el valor para el cálculo es ingresado en el programa principal. Indique el pasaje de parámetros utilizado.
8. Codifique un módulo, llamado CUBO, que calcule el cubo de un número entero *N* aplicando el método descripto en el ejercicio 11 del TP5. Suponga que el valor para el cálculo es ingresado en el programa principal. Indique el pasaje de parámetros utilizado.
9. Codifique un módulo, llamado COCIENTE, que calcule el cociente entero (mediante restas sucesivas) entre 2 valores enteros. Considere que el cálculo se implementa aplicando estructuras MIENTRAS y el criterio de finalización por BANDERA. Suponga que los valores para el cálculo son ingresados en el programa principal. Indique el pasaje de parámetros utilizado.
10. Modifique el ejercicio anterior para obtener el resto de la división entera.
11. Codifique un módulo, llamado PRIMO, que determine si un valor ingresado por el usuario es primo o no. Suponga que el valor de entrada es ingresado en el programa principal. Indique el pasaje de parámetros utilizado.
12. Modifique el módulo FACTORIAL, del ejercicio 4, de modo que el cálculo de factorial se realice utilizando el módulo PRODUCTO del ejercicio 5. Suponga que el valor para el cálculo es ingresado en el programa principal. Indique el pasaje de parámetros utilizado.
13. Un grupo de WhatsApp conformado por programadores se utiliza como espacio para compartir códigos e ideas entre desarrolladores. Justamente, los últimos 3 mensajes proponen diferentes versiones de un algoritmo. ¿Serías capaz de identificar el objetivo de éstos algoritmos? ¿Puedes proponer una alternativa?

```
FUNCIÓN idea1 (E n: ENTERO)
VARIABLES
   x,s,i,p: ENTERO
INICIO
   x<-1
   MIENTRAS x<4 HACER
      s<-1
      PARA i DESDE 1 HASTA x CON PASO 1 HACER
          s<-s*n
      FIN_PARA
      p<-s
      x<-x+1
   FIN_MIENTRAS
   idea1<-p
FIN
```

```
FUNCIÓN idea3 (E n: ENTERO)
VARIABLES
   s,i: ENTERO
INICIO
   i<-0
   s<-0
   MIENTRAS i<n HACER
     s<-s+n^2-n+1+2*i
     i<-i+1
   FIN_MIENTRAS
   idea3<-s
FIN
```



```
FUNCIÓN idea2 (E n: ENTERO)
VARIABLES
   x,s: ENTERO
   b: LÓGICO
INICIO
   x<-n*(n-1)+1
   s<-0
   b<-n>0
   MIENTRAS b=VERDADERO HACER
      s<-s+x
      x<-x+2
      n<-n-1
      SI n=0 ENTONCES
         b<-FALSO
      FIN_SI
   FIN_MIENTRAS
   idea2<-s
FIN
```

14. Diseñe programas modulares que resuelvan la siguientes sumatorias/productorias (reutilice los módulos diseñados en los ejercicios anteriores):

a) $\sum_{k=0}^{n} 2 \times i + 1$ b) $\sum_{i=1}^{total} i!\ mod\ 2 \times i$ c) $\prod_{h=1}^{final} final\ div\ h + h!$

15. Dados los siguientes módulos

a) Realice pruebas de escritorio, teniendo en cuenta el tipo de los argumentos de cada uno.
b) Determine el objetivo de cada módulo.
c) Codifique en C++, añadiendo el programa principal y las invocaciones necesarias para la ejecución.

```
FUNCIÓN enigma (E m: ENTERO):ENTERO
VARIABLES
   b:LÓGICO
   s:ENTERO
INICIO
   s←0
   b←m > 0
   MIENTRAS (b=VERDADERO) HACER
      s←s+m%10;
      m←m/10;
      SI (m=0) ENTONCES
        b←FALSO
      FIN_SI
   FIN_MIENTRAS
   enigma←s
FIN
```

```
FUNCIÓN misterio (E x: ENTERO): ENTERO
VARIABLES
   band: LOGICO
   z: ENTERO
INICIO
   z←1
   band←x >= 0
   MIENTRAS band HACER
     SI (x=0) ENTONCES
       band←FALSO
     SINO
       z←z*x
       x←x-1
     FIN_SI
   FIN_MIENTRAS
   misterio←z
FIN
```

```
FUNCIÓN oculta (E a: ENTERO, E
b:ENTERO):ENTERO
VARIABLES
   band: LOGICO
   c: ENTERO
INICIO
   c←0
   band←VERDADERO
   MIENTRAS (band <> FALSO) HACER
      c←c+b
      a←a-1
      SI (a=0) ENTONCES
        band←FALSO
      FIN_SI
   FIN_MIENTRAS
   oculta←c
FIN
```

```
FUNCIÓN desconocida (E d:CARACTER): LÓGICO
VARIABLES
   m: LÓGICO
INICIO
   m←FALSO
   SI (d>='a' && d<='z') ENTONCES
      m←VERDADERO
   SINO
     SI (d>='A' && d<='Z') ENTONCES
        m←VERDADERO
     FIN_SI
   FIN_SI
   desconocida←m
FIN
```