

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 esta 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

- 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

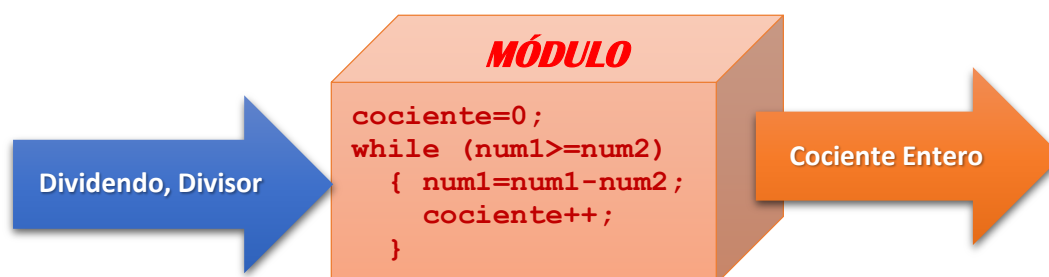
Instrucciones
asociadas a la
presentación
de resultados

¿Qué tipo de
módulo usaré? ¿qué
datos necesitará
este módulo para
hacer el cálculo?

¿Cuáles serán las
operaciones que
realizará el módulo?



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

Ya entendí, sólo debo definir el módulo como un pequeño programa e indicar qué datos necesitará para trabajar



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 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, ya que las funciones sólo pueden obtener un ÚNICO valor como resultado.

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. Dado el siguiente programa, determina su propósito y modifícalo para realizar el cálculo mediante una función.

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

using namespace std;

main()
{ int a,m=0,i=1;
  bool b;
  cout << "Ingrese dato: ";
  cin >> a;
  i=a*(a-1)+1;
  while (a>0)
  {
    m=m+i;
    i=i+2;
    a=a-1;
  }
  cout << "Salida: " << m << endl;
  system("pause");
}
```

2. El siguiente programa utiliza los módulos *VALIDAR* y *AREAC* para verificar que los datos ingresados sean positivos y calcular la superficie de un círculo, respectivamente. Tomando este programa como referencia, codifica uno que calcule el área y volumen de un cilindro circular recto.

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

using namespace std;

const float PI=3.1415926;

bool validar(float num);
float areac (float num);

main()
{ float radio,sup;
  do {
    cout << "Ingrese radio: ";
    cin >> radio;
  } while (validar(radio)==false);
  sup=areac(radio);
  cout << "Area: " << sup << endl;
  system("pause");
}

bool validar(float num)
{
  if (num>0)
    return true;
  else
    return false;
}

float areac(float num)
{
  return PI*pow(num,2.0);
}
```

3. Codifica un programa que permita calcular el área de un triángulo conociendo el valor de sus lados. Para ello, diseñe los módulos *EXISTE* y *HERON*: el módulo *EXISTE* determina si el valor de los lados permite la construcción del triángulo, mientras que el módulo *HERON* aplica la fórmula de Herón para el cálculo del área.
4. Codifica un módulo, llamado *PRODUCTO*, que calcule el producto de 2 números enteros positivos mediante sumas sucesivas. Considera que el cálculo se realizará usando estructuras *MIENTRAS* y criterio de finalización por *CENTINELA*. Además, asume que los valores a multiplicar son ingresados y **validados** en el programa principal. Indica el pasaje de parámetros utilizado.
5. Codifica un módulo, llamado *COCIENTE*, que calcule el cociente de la división entera (mediante restas sucesivas) de 2 valores enteros. Considera que el cálculo se realizará usando estructuras *REPETIR* y criterio de finalización por *BANDERA*. Además, asume que los valores a operar son ingresados y **validados** en el programa principal. Indica el pasaje de parámetros usado.
6. Codifica un módulo, llamado *RESTO*, que calcule el módulo o resto de la división entera (mediante restas sucesivas) de 2 valores enteros. Considera que el cálculo se implementa con estructuras *MIENTRAS* y criterio de finalización a elección. Además, asume que los valores a calcular son ingresados y **validados** en el programa principal. Indica el pasaje de parámetros usado.
7. Codifica un módulo, llamado *CUARTA*, que calcule la potencia cuarta de un número entero positivo *N* aplicando el método descrito en el TP5 (ejercicio 4). Considera que el valor para el cálculo es ingresado y **validado** en el programa principal. Indica el pasaje de parámetros utilizado.
8. Codifica la función *POTENCIA* mediante productos sucesivos, implementada con estructuras *MIENTRAS* y finalización por *BANDERA*. Luego, codifica otra versión que utilice el módulo *PRODUCTO* (ejercicio 4) para resolver el producto acumulado. Considera que la base y el exponente son ingresados y **validados** en el programa principal. Indica el pasaje de parámetros utilizado.
9. Codifica un módulo, llamado *FACTORIAL*, que implemente el cálculo utilizando estructuras repetitivas y criterio de finalización a elección. Luego, codifica otra versión que utilice el módulo *PRODUCTO* (ejercicio 4) para resolver el producto acumulado correspondiente. Asume que el valor a calcular es ingresado y **validado** en el programa principal. Indica el pasaje de parámetros utilizado.
10. Codifica un módulo, llamado *CONTAR_DIGITOS*, que determine cuántas cifras componen un número entero ingresado por el usuario. Considera que el cálculo se implementa con estructuras *REPETIR* y criterio de finalización por *CENTINELA*. Asume que el valor a analizar es ingresado en el programa principal. Indica el pasaje de parámetros utilizado.

¿Te animas a reemplazar las operaciones de cociente y resto por los módulos de los ejercicios 5 y 6?

11. Codifica un módulo, llamado *MAXIMO_DIGITO*, que obtenga el dígito de mayor valor de un número entero ingresado por el usuario. Considera que el cálculo se implementa con estructuras *MIENTRAS* y criterio de finalización por *BANDERA*. Asume que el valor a analizar es ingresado en el programa principal. Indica el pasaje de parámetros utilizado.

¿Te animas a reemplazar las operaciones de cociente y resto por los módulos de los ejercicios 5 y 6?

12. Codifica un módulo, llamado *SUMA_DIGITOS*, que obtenga la suma de los dígitos de un número entero ingresado por el usuario. Considera que, a elección del usuario, el módulo podrá sumar todos los dígitos, sólo los impares o sólo los pares del número; indicándose esta elección mediante un parámetro extra. Asume que el valor a analizar es ingresado en el programa principal. Indica el pasaje de parámetros utilizado.

¿Te animas a reemplazar las operaciones de cociente y resto por los módulos de los ejercicios 5 y 6?

13. Codifica un módulo, llamado *INVERSO*, que calcule el inverso de un número. Para ello, ten en cuenta el método de descomposición de dígitos descrito en el ejercicio 7 del TP4. Considera que el cálculo se implementa con estructuras *MIENTRAS* y criterio de finalización por *BANDERA*, reemplazando las operaciones de división entera y resto por los módulos de los ejercicios 5 y 6. Asume que el valor para el cálculo es ingresado en el programa principal. Indica el pasaje de parámetros utilizado.

14. Codifica un módulo, llamado *CAPICUA*, que determine si un número entero positivo *N* es capicúa o no. Para ello, ten en cuenta que puede reusar el módulo *INVERSO*. Asume que el valor a analizar es ingresado y validado en el programa principal. Indica el pasaje de parámetros utilizado.
15. ¿Sabías que es posible obtener un número capicúa a partir de otro que no lo es? El proceso es bastante sencillo, dado un valor *N* (valor no capicúa) debe sumarse a éste su inverso *M* (dígitos invertidos) y si la suma obtenida es capicúa entonces el proceso finaliza; sino se repite lo anterior hasta obtener el valor deseado. Por ejemplo, dado el valor 37 y su inverso 73 la suma de éstos es 110; un valor no capicúa, esto implica que será necesario calcular nuevamente el inverso (11) y realizar una vez más la suma (121). Siendo 121 un número capicúa, entonces finaliza el proceso.
16. Codifica un módulo, llamado *PRIMO*, que determine si un valor positivo ingresado por el usuario es primo o no. Considera que el valor de entrada es ingresado y **validado** en el programa principal. Indica el pasaje de parámetros utilizado.
¿Te animas a reemplazar la operación resto por el módulo del ejercicios 6?
17. Codifica un módulo, llamado *FIBO*, que calcule un término de la serie de Fibonacci. Considera que el término a calcular es ingresado en el programa principal. Indica el pasaje de parámetros utilizado.
18. Codifica un módulo, llamado *FIBODETECT*, que dado un número entero positivo *N*, determine si se trata o no de un valor perteneciente a la serie de Fibonacci. Considera que el número a analizar es ingresado y validado en el programa principal. Indique el pasaje de parámetros utilizado.
19. En el TP6 (ejercicio 8) codificaste una calculadora que realiza varias operaciones, algunas de ellas aplicando estructuras repetitivas. En este trabajo, y considerando los módulos desarrollados en los ítems anteriores, te proponemos codificar un programa que emule una calculadora cuyas operaciones se implementen mediante funciones.
20. En el TP6 (ejercicio 16) codificaste la versión avanzada del juego “Adivina el número”. Ahora, aplicando funciones, escribe la versión modular del juego.

