Clase presencial

Sesion5: Programación Modular

Instructor: David Paúl Porras Córdova
@iscodem

Objetivo General

- Identificar la diferencia entre programación estructurada y programación modular.
- Conocer las ventajas de modularizar las sentencias de código.
- Resolver problemas cotidianos bajo el concepto de módulos.
- Buenas prácticas para la POO.

Contenido de Agenda

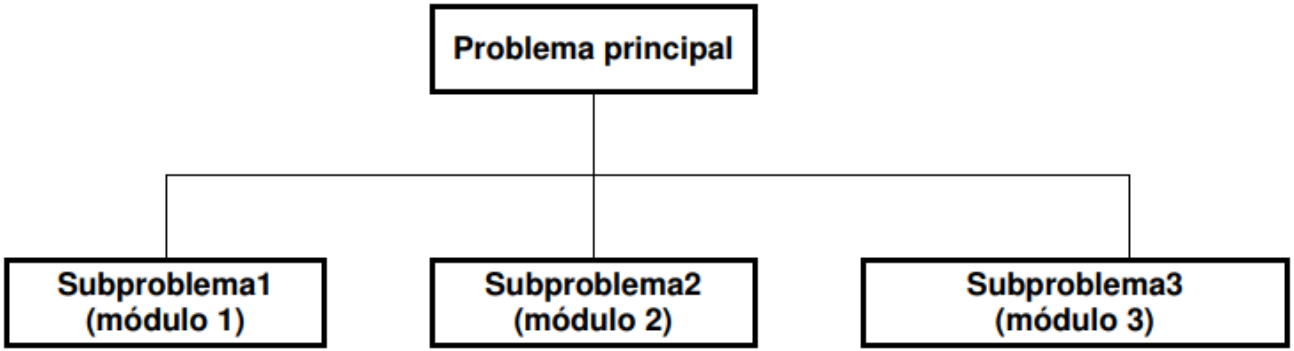
- Programación estructurada
- Programación Módular
  - Funciones
  - Procedimientos
  - Métodos
- Buenas prácticas de programación

1. Programación módular

Uno de los métodos fundamentales para resolver un problema es dividirlo en problemas más pequeños, llamados **subproblemas**. Estos problemas pueden a su vez dividirse repetidamente en problemas más pequeños hasta que los problemas sean de fácil solución. Divide y vencerás … Cada subproblema es deseable que sea independiente de los demás y se denomina **módulo**.

El problema original se resuelve con un programa principal (llamado también driver o main), y los **subproblemas** (módulos) mediante **subprogramas: procedimientos y funciones.**

1. Programación modular



La resolución de un problema comienza con una descomposición modular y luego nuevas descomposiciones de cada módulo en un proceso denominado refinamiento sucesivo.

2. Diseño módular: Los módulos

- Los subproblemas o módulos se diseñan con **subprogramas**, que a su vez se clasifican en **procedimientos** y **funciones**. Los procedimientos y funciones son unidades de programas diseñados para ejecutar un a tarea específica. Por ejemplo los procedimientos predefinidos **LEER** y **ESCRIBIR** están diseñados para realizar operaciones de entrada y salida de datos de un programa.
- El proceso de descomposición de un problema en módulos se denomina **modularización**. Los procedimientos y funciones asisten a la programación modular.

2.1. Funciones y procedimientos

Las funciones, normalmente, devuelven un sólo valor a la unidad de programa (programa que invoca a la función) que las referencia. Los procedimientos pueden devolver cero, uno o varios valores. En el caso de no devolver ningún valor, realiza alguna tarea tal como alguna operación de entrada y/o salida.

A un procedimiento no se le puede asignar valor, y por consiguiente ningún tipo está asociado con el nombre del procedimiento.

Una función se referencia utilizando su nombre en una expresión, mientras que un procedimiento se referencia por su llamada o invocación al mismo.

3. FUNCIONES

Un **subalgoritmo** función es un subalgoritmo que recibiendo o no datos devuelve un único resultado.

Tienen su origen ligado al concepto matemático de función de una o más variables. Ejemplo de este tipo de subalgoritmos son las llamadas funciones internas (sin, cos, abs). Las cuales pueden usarse en expresiones algorítmicas como si se tratara de variables.

Otros ejemplos de funciones matemáticas:

$f(x) = x^2 + 2x - 1$

$g(x, y) = x^3 + 5y$

$h(x, y, z) = 3x + 2y - z$

parámetros formales

3. FUNCIONES

donde **x,y,z**: son los **parámetros formales** o ficticios, es decir permiten expresar la ley o "forma" de la función.

Las funciones pueden tener uno o más parámetros formales (datos) pero siempre devuelven un único resultado.

Las funciones son evaluadas utilizando **parámetros actuales** o reales, es decir los valores con los que se quiere evaluar la función:

f(3)

g(-1 , 5)

h(2 , 0 , 7)

parámetros actuales

3. FUNCIONES

Una función es un objeto del ambiente, con nombre, tipo y valor único. El tipo se asocia al valor que retorna la función cuando es evaluada para un conjunto dado de valores de sus argumentos

```
Funcion nombre (lista de parámetros formales): Tipo de resultado
    Declaración de variables
Inicio
    Acciones
    Devolver (constante, variable o expresión)
Fin función
```

Lista de parámetros formales: contiene las variables que pasan alguna información necesaria para que la función ejecute el conjunto de acciones.

Tipo de resultado: señala el tipo de dato que devuelve la función.

Declaración de variables: en este lugar se deben declarar los parámetros formales y también aquellas variables que se usarán en la función.

Cuerpo de la función: lo constituye el conjunto de acciones a realizar por la función.

Retornar el resultado: el único resultado que devuelve la función puede ser un valor constante, o una variable o una expresión válida, la cual debe colocarse entre paréntesis al lado de la acción Devolver. Cuando se ejecuta esta acción se devuelve el control del programa al lugar donde se ha llamado a la función.

3. FUNCIONES

- definición de la función $f(x) = x^2 + 2x - 1$

```
funcion f (x: real): real
inicio
    Devolver ( x * x + 2 * x – 1 )
fin funcion
```

- definición de la función $y = x^n$ (n entero)

```
funcion potencia (x: real, n: entero): real
variables
    entero  i
    real  y
inicio
    y ← 1
    repetir para i ← 1 , abs (n)
        y ← y * x
    fin para
    si  n < 0  entonces
            y ← 1 / y
    fin si
    Devolver ( y )
fin funcion
```

Laboratorio: 5. Funciones

- Ejercicio 1:
  - Tiempo: 20 minutos
  - Escribir un algoritmo que utilice funciones para calcular la potencia n de x. Los valores de n y x deberán ser ingresados por teclado, siendo n un entero y x un valor real.

Laboratorio: 5. Funciones

- Ejercicio 2:
  - Tiempo: 20 minutos
  - Realizar un algoritmo que permita evaluar los requisitos de ingreso de una persona a una sala de cine.
    - Poseer DNI vigente
    - Tener mayoría de edad ó estas acompañado de un apoderado.
    - Tener reserva vigente.
  - El programa además deberá calcular el monto total a pagar después de haber comprado entre popcorn, gaseosas ó dulces.

4. PROCEDIMIENTOS o SUBRUTINAS

Un procedimiento o subrutina es un **subalgoritmo** que recibiendo o no datos permite devolver varios resultados, un resultado o ninguno.

Un procedimiento está compuesto por un grupo de sentencias a las que asigna un nombre (identificador o simplemente nombre del procedimiento) y constituye una unidad de programa. La tarea asignada al procedimiento se ejecutará siempre que se encuentre el identificador (nombre del procedimiento) en el conjunto de sentencias que definen el programa.

4. ¿Cómo trabajar con procedimientos?

- Declaración de un procedimiento
- Llamada a un procedimiento .
- Dónde escribir un procedimiento?
- Transferencia de información a/desde procedimientos: parámetros
- Ventajas de utilizar un procedimiento

5. Declaración de un procedimiento

La declaración de un procedimiento no indica a la computadora que ejecute las instrucciones dadas, sino que indica a la computadora cuáles son estas instrucciones y dónde están localizadas cuando sea necesario.

5. Declaración de un procedimiento

Declaración

- **Formato 1**

```
Subrutina nombre()
    Declaración de variables
Inicio
    Acciones
Fin subrutina
```

- **Formato 2**

```
Subrutina nombre (lista de parámetros formales)
    Declaración de variables
Inicio
    Acciones
Fin subrutina
```

Nombre: identificador válido

Lista de parámetros formales: parámetros formales del procedimiento; sirven para pasar información al procedimiento y/o devolver información del procedimiento a la unidad de programa que le invoca.

Están separados por comas, y precedidos por las letras E (entrada), S (Salida) o E/S (Entrada/Salida)

5. Llamada al procedimiento

Los procedimientos se llaman dentro de un programa o de otro procedimiento directamente por su nombre, de acuerdo a los formatos 1 o 2 .

- **Formato 1**

```
nombre
```

- **Formato 2**

```
nombre (lista de parámetros formales)
```

La sentencia nombre indica la ejecución del procedimiento cuyo identificador coincide con nombre. Después que ha terminado la ejecución, se ejecuta la sentencia que sigue a la llamada al procedimiento.

En resumen, un procedimiento, al igual que un programa, consta de tres partes:

- Una cabecera del procedimiento que proporciona el nombre del mismo y, caso de existir, una lista de parámetros formales.
- Una sección de declaración que puede contener constantes variables, etc.
- Una sección ejecutable: cuerpo de acciones del procedimiento.

Laboratorio: 5. Procedimientos

- Ejercicio 2:
  - Tiempo: 20 minutos
  - Se requiere realizar un programa que calcule la tasación de un departamento, considerando lo siguiente
    - Área y número de habitaciones.
    - Ubicación geográfica.
    - Número de cocheras y área de estacionamiento.
    - Servicios básicos instalados (agua, luz, gas).

5. Dónde escribir el procedimiento?

La posición adecuada de dónde escribir el procedimiento depende del lenguaje de codificación elegido. En seudocódigo, será indistinto el orden en que se escriben el algoritmo y los **subalgoritmos**. En este ejemplo, adoptamos arbitrariamente escribirlo luego del programa principal.

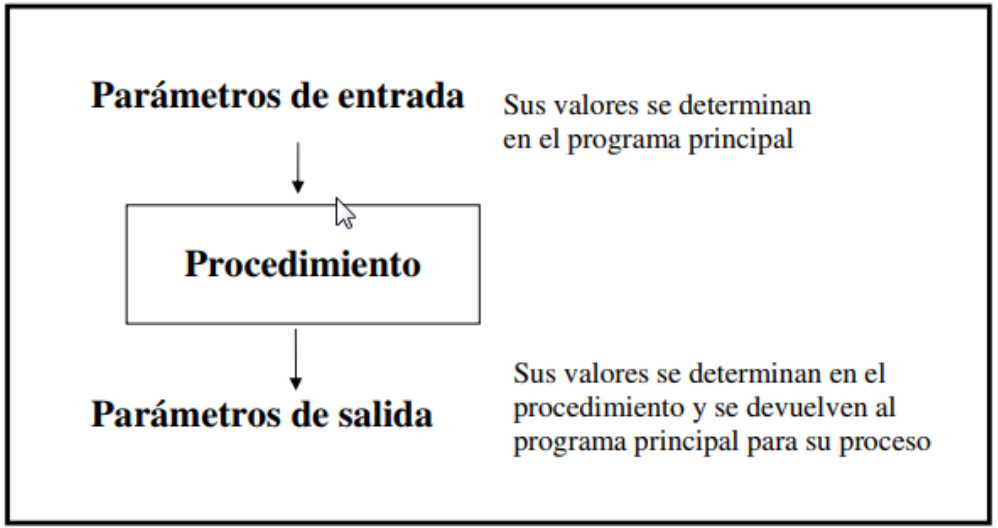
```
Programa Círculo
Variables
    Real : radio, area
Inicio
    Escribir('ingrese radio del círculo')
    Leer(radio)

    Superficie(radio, area)

    Escribir('el área del círculo es',area)
Fin

Subrutina Superficie(E: r: real , S: A: real)
Inicio
    A ← 3.1415192 * r ^ 2
Fin subrutina
```

6. Transferencia de información a/desde procedimientos: parámetros

Un parámetro es un método para pasar información – valores a variables – del programa principal a un procedimiento.

Procedimientos sin parámetros: no existe comunicación entre el programa principal y los procedimientos o entre dos procedimientos.

Procedimientos con parámetros: existe comunicación entre el programa principal y los procedimientos o entre dos procedimientos.



5. Parámetros actuales y parámetros formales

Las acciones que contienen el llamado al procedimiento constan de dos partes: un identificador (nombre del procedimiento) y una lista de **parámetros actuales**

```
Nombre (par1, par2, par3, ….)
```

Los parámetros actuales par1, par2, etc. Contienen los valores que se transferirán al procedimiento.

En la declaración de un procedimiento, cuando se incluyen los parámetros, éstos se denominan **parámetros formales** parf1,parf2, parf3, etc. Ellos sirven para contener los valores de los parámetros actuales cuando se invoca al procedimiento.

```
Procedimiento Nombre (parf1, parf2, parf3, …)
```

6. Parámetros valor y referencia (variable)

Algoritmo EJEMPLO **variables** **entero : A,B,C** **Inicio** A ← 3 B ← 5 C ←17 SUMAR (A, A, A+B, C) **Escribir** ('el valor en C es :', C) **fin**	• **Por valor** **Subrutina** SUMAR **(E:** x, y, z, v : entero**)** **inicio** x ← x +1 v ← y + z **fin subrutina** • **Por referencia** **Subrutina** SUMAR **(E/S:** x, y : entero, **E:** z: entero, **E/S:** v: entero**)** **inicio** x ← x +1 v ← y + z **fin subrutina**

7. Ámbito de variables

Las variables pueden clasificarse según su utilización en:

- **variables locales**: son aquellas que están declaradas dentro del algoritmo o subalgoritmo, y son propias al ámbito de la declaración, en el sentido que cada una es distinta de otra variable declarada con el mismo nombre en cualquier parte del algoritmo principal u otros subalgoritmos.

- **variables globales**: son aquellas que están declaradas en el algoritmo o subalgoritmo y son accesibles para los subalgoritmos que de él dependen.