UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

Unidad 1. Lógica de programación

Tema 1. Introducción a la programación

UNIVERSIDAD DE LOS ANDES

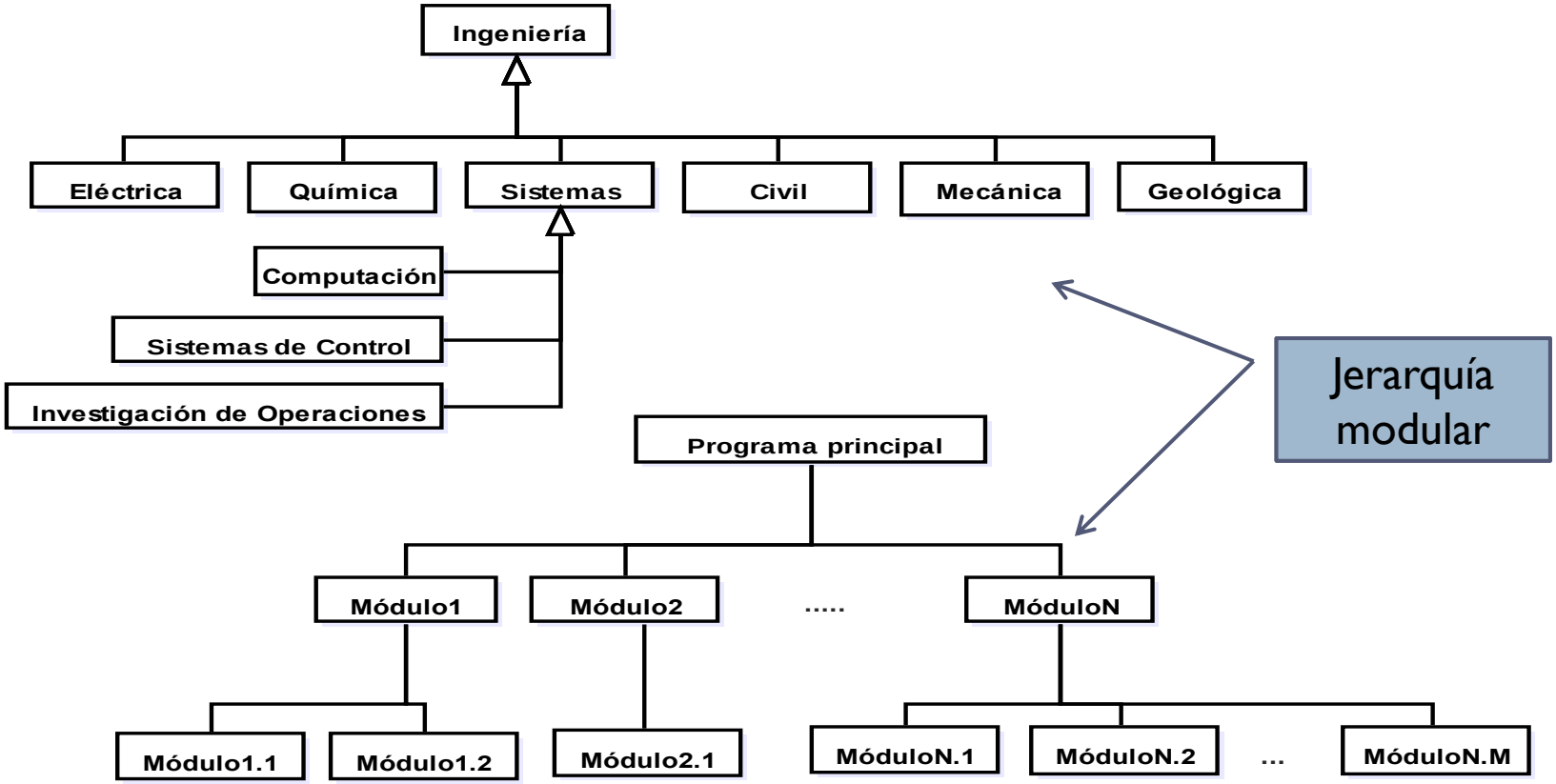
Introducción a la programación

- Contenido
  - Programación modular
  - Características generales de un lenguaje de programación de alto nivel
  - Elementos sintácticos
  - Estructura de un programa
  - Método de desarrollo de programas
  - Un programa muy simple
- Objetivo
  - Lograr una visión general de los lenguajes de programación de alto nivel
- Bibliografía:
  - Deitel y Deitel, cap. 2, sec. 13.1-13.2
  - Joyanes, cap. 3

UNIVERSIDAD DE LOS ANDES

Programación modular

- ✓ Construir programas en forma modular (dividir un programa grande en un cierto número de piezas o componentes más pequeños)



Programación modular

- Módulo: pieza pequeña diseñada para ejecutar una tarea específica.
- Reutilizar los módulos o piezas
- Comprender los mecanismos utilizados para pasar información entre módulos
- Permiten desarrollar y mantener programas grandes
- Tipos:
  - Funciones
  - Procedimientos

UNIVERSIDAD DE LOS ANDES

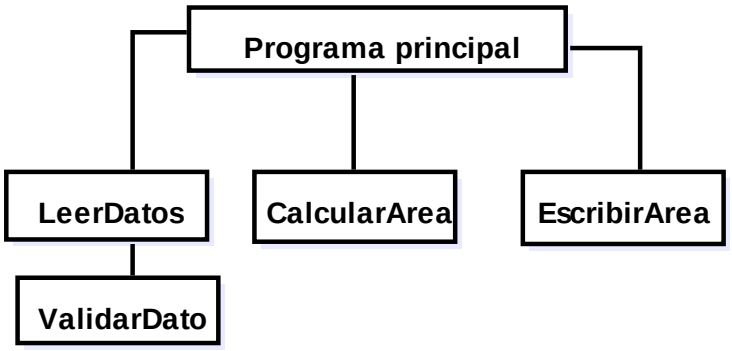
Programación modular

- Diseño descendente (de arriba hacia abajo) o programación top-down o estrategia de diseño llamada divide-y-vencerás
  - Para solucionar problemas complejos y/o grandes, es conveniente dividirlos en problemas más pequeños (subproblemas), los cuales a su vez pueden dividirse en subproblemas más pequeños
  - El proceso de división continua hasta que los subproblemas son tan sencillos que pueden ser resueltos mediante un programa simple llamado módulo
- Los módulos obtenidos en un diseño descendente se codifican en algún lenguaje de programación de alto nivel como: C/C++, Pascal, Fortran, Java, PHP, etc.

Ejemplo de descomposición modular

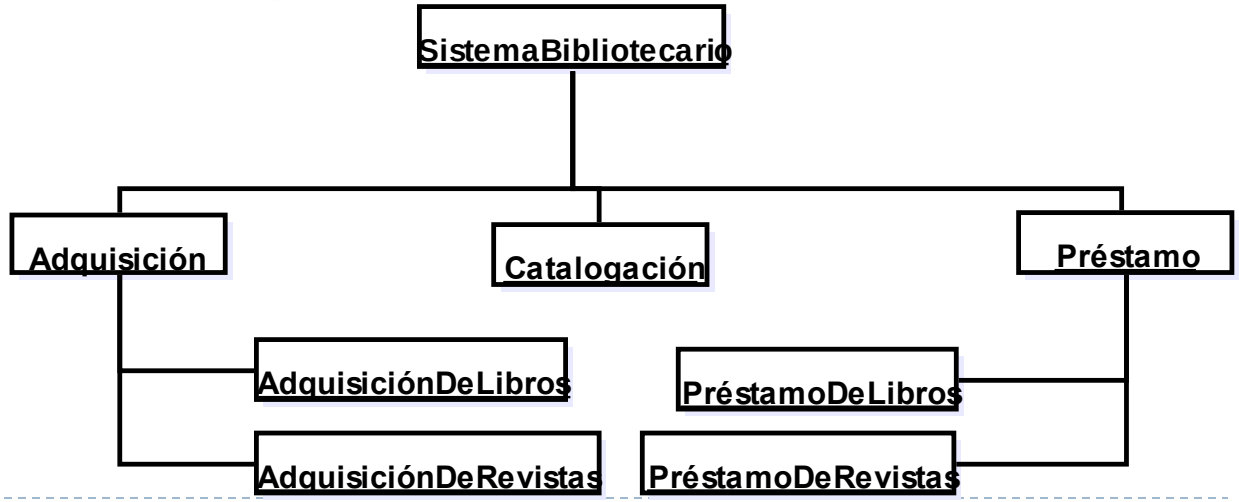
- Calcular el área de un rectángulo

- Subproblemas:
  - Entrada de datos: altura y base
    - Leer datos
    - Validar datos
  - Calcular el área
  - Salida de los resultados
- Módulos:
  - leerDatos(altura, base)
  - validarDato(dato)
  - calcularArea(altura, base, area)
  - escribirArea(area)



Programación modular: ventajas

- Permite repartir el trabajo de desarrollo entre varios programadores de manera simultánea y con un cierto grado de independencia entre ellos
- Facilita la escritura y depuración de un programa, ya que cada módulo representa una parte bien definida del problema
- Permite la modificación de un módulo, sin afectar a los demás
- Permite la localización rápida de errores
- Favorece la comprensión del programa completo



UNIVERSIDAD DE LOS ANDES

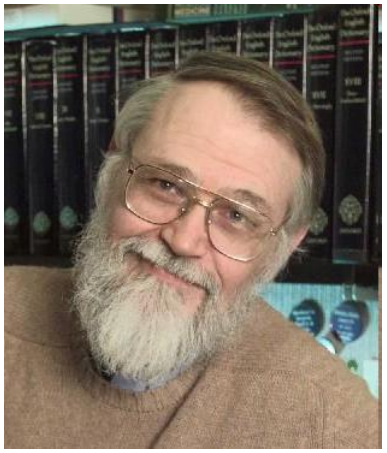
Programación modular: ventajas

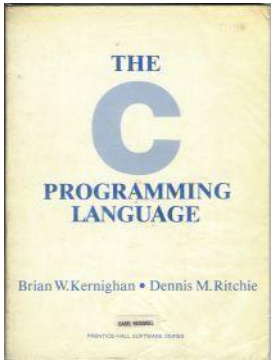
- Permite reutilizar un grupo de sentencias cuando y donde sea necesario
  - Los algoritmos de cada módulo sólo se escriben y codifican una sola vez, aunque se necesiten en distintas ocasiones a lo largo del sistema de software completo e incluso de otros sistemas de software (reutilización) evitando la duplicación innecesaria de código
  - La reutilización de un módulo por otros sistemas de software es un ahorro de tiempo, ya que no es necesario volver a resolver el problema, y si el módulo ha sido previamente probado y verificado también reduce la posibilidad de errores
- Favorece la portabilidad, ya que se pueden escribir programas sin prestar atención a las características de un sistema particular

UNIVERSIDAD DE LOS ANDES

Lenguaje de programación C

- Creado por:
- Brian Kernighan
- Dennis Ritchie

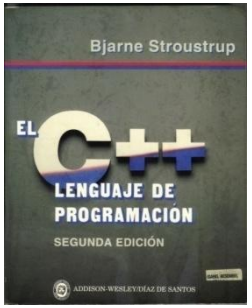




- En 1972, a partir de los lenguajes BCPL (1967) y B (1970)
- Vinculado al sistema operativo UNIX
- Combina características de los lenguajes de alto nivel (sentencias de control y manipulación de datos) y de los lenguajes de bajo nivel (manejo de bits y apuntadores)
- Es independiente del hardware (portable)

Lenguaje de programación C++

- Extensión de C inventada por
- Bjarne Stroustrup



- En 1983 en los laboratorios Bell
- Manejo de programas complejos
- Permite usar la programación estructurada y la orientada por objetos

C con clases

UNIVERSIDAD DE LOS ANDES

Lenguaje de programación C

- Palabras claves o reservadas
  - Aquellas que tienen significado especial para el compilador y que están reservadas para su uso especial en el lenguaje
- Ejemplos:
  - Quick C, Turbo C, Turbo C++, Borland C, Microsoft C, C#

auto	break	case	char	const
goto	if	float	enum	extern
short	signed	sizeof	static	struct
volatile	while	main	scanf	printf

UNIVERSIDAD DE LOS ANDES

Identificadores C/C++

- Identificadores:
  - **Nombres** que permiten indicar, mencionar o hacer referencia a los diferentes objetos manipulados en el programa
  - Deben tener significado, sugiriendo lo que ellos representan
  - **No** pueden ser palabras reservadas
  - Los compiladores de C++ reconocen hasta un máximo de 30 caracteres, aunque el identificador puede tener cualquier longitud
  - Nombres compuestos de una serie de caracteres, dígitos o el subrayado _, donde **el primer caracter no puede ser un dígito**
  - Las mayúsculas y minúsculas son diferentes
  - Ejemplo de identificadores válidos
    - A1, a1, sueldo_base, entero, IVA, sueldoNeto, p, Q, r2, cc456
  - Identificadores inválidos
    - ~~1A, 3p, #sueldoActual, 5_pqr, +nombre, IVA, monto+imp~~

UNIVERSIDAD DE LOS ANDES

Comentarios C/C++

- Permiten que el programador documente su código o programa
- Sirven para facilitar la legibilidad de un programa
- Son ignorados por el compilador
- Tipos
  - Comentarios iniciales: para indicar el objetivo general del programa, identificar el autor, la fecha, etc.
  - Comentarios en cada línea: documenta los pasos cruciales con //
- Ejemplos:

```
/* Esto es un comentario simple. */
// Este también es un comentario simple de una sola línea
/* Esto es un comentario más largo,
    distribuido en varias líneas. El
    texto se suele alinear por la izquierda. */
```

UNIVERSIDAD DE LOS ANDES

Comentarios C/C++

- Características de los comentarios
  - Deben ser coherentes con el programa
  - Deben ser relevantes
  - Deben mantenerse al día
  - Siempre deben enriquecer el programa con conceptos, gráficos, relaciones entre partes, etc.
- Comentarios de varias líneas en C/C++
  - Empiezan con /* y terminan con */
  - Pueden comprender varias líneas
  - Todo lo que esté entre /* y */ es ignorado por el compilador
- Ejemplo:

```
/**********************************************************
* Esto es un comentario de varias                         *
* líneas, encerrado en una caja para                      *
* llamar la atención.                                     *
**********************************************************/
```

UNIVERSIDAD DE LOS ANDES

Delimitadores C/C++

Signo	Significado
; (punto y coma)	Terminación. Debe ir al finalizar cada una de las sentencias o declaraciones para indicar su fin
, (coma)	Separa dos elementos consecutivos de una lista
() (paréntesis)	Enmarca una lista de parámetros, expresiones o condiciones
[] (corchetes)	Enmarcan la dimensión o el subíndice de un arreglo
{} (llaves)	Enmarcan un bloque de sentencias o una lista de valores iniciales
“ “ (comillas)	Enmarcan un comentario de salida o una cadena

UNIVERSIDAD DE LOS ANDES

Programas en C/C++

- Estructura sugerida
  - Comentario con el nombre del programa, autor, fecha de creación y objetivo del mismo
  - Declaraciones de inclusión o importaciones
  - Definiciones de constantes
  - Definiciones de tipos
  - Declaraciones de prototipos
  - Declaraciones de variables globales

```
void main()
{  declaraciones de variables y/o constantes locales
   Conjunto de sentencias
}
```

  - Definiciones de funciones

UNIVERSIDAD DE LOS ANDES

Declaración de inclusión

- Sentencia que le indica al compilador que debe incluir en el código objeto del programa, el contenido del archivo indicado entre los caracteres < >
- Ejemplo:

```
#include <stdio.h>                    // librería de E/S de C
```

- Todo en el lenguaje C/C++ es una función
- La función principal es

```
void main ( )
{
   conjunto de sentencias
}
```

UNIVERSIDAD DE LOS ANDES

Función

- Matemáticamente una función es una operación que toma uno o mas valores llamados argumentos y produce un valor llamado resultado, igual en computación
- Ejemplos:

Función de un solo argumento

$$f(x) = \frac{4}{x^2 + 1}$$

evaluación de la función en 4

f(4)= 4/(16+1) = 4/17 = 0.235

- Función de dos argumentos

$$f(a,b) = \frac{a^2 + b^2}{(a-1)^2 - (b-1)^2}$$

evaluación de la función en a=2 y b=3

f(2,3) = (4 + 9)/(1 – 4) = -13/3 = -4.33

UNIVERSIDAD DE LOS ANDES

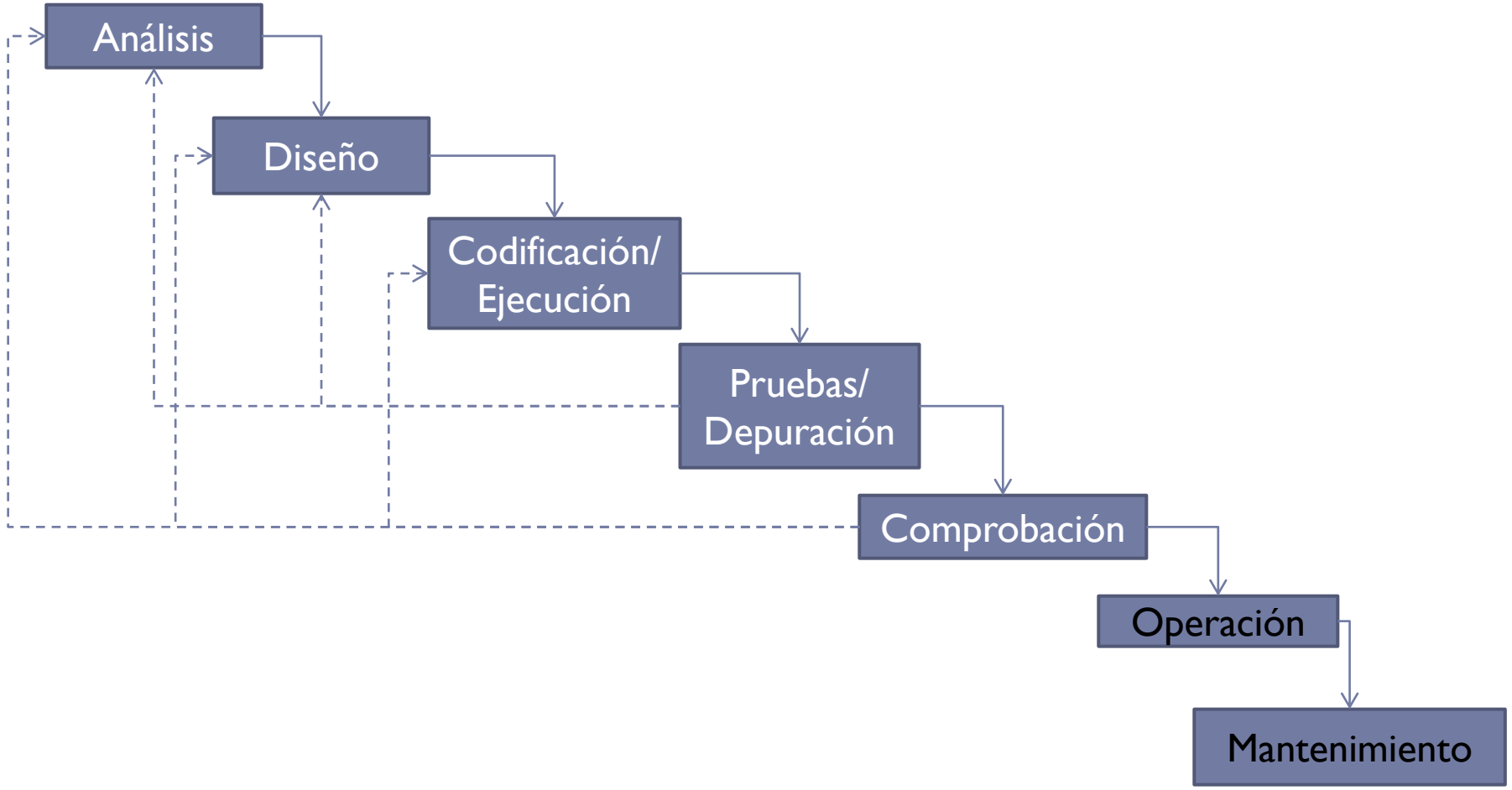
Función en C/C++

- **En C/C++ los módulos** se llaman funciones (unidad básica de los programas) y realizan determinadas tareas bien definidas

- Una función:
  - Tiene un nombre, ejm: f
  - Toma cero o mas valores, denominados argumentos o parámetros de entrada (parámetros formales), ejm: x
  - Según el valor de los parámetros, devuelve un resultado, el cual es obtenido durante su ejecución y devuelto por el ***return***, ejm: resultado

- Una función se define una sola vez pero puede usarse (mediante llamadas) tantas veces como sea necesario

Método de desarrollo de programas



Método de desarrollo de programas

Análisis

- **Análisis del problema** (Análisis E – P - S)
  - Identificar y comprender el problema

> Problema: Calcular el área o superficie de un rectángulo de dimensiones l (largo) y a (ancho) ambas en centímetros

  - Describir los datos de entrada o especificación de entrada (E)
    - Información necesaria para resolver el problema
      - ¿Qué datos se requieren de antemano o entrada?
      - ¿Cuántos datos se introducirán o pedirán?
      - ¿Cuáles datos de entrada son válidos?

> Entradas: Dos números reales mayores o iguales a cero para el largo **l** y para el ancho **a**

Método de desarrollo de programas

Análisis

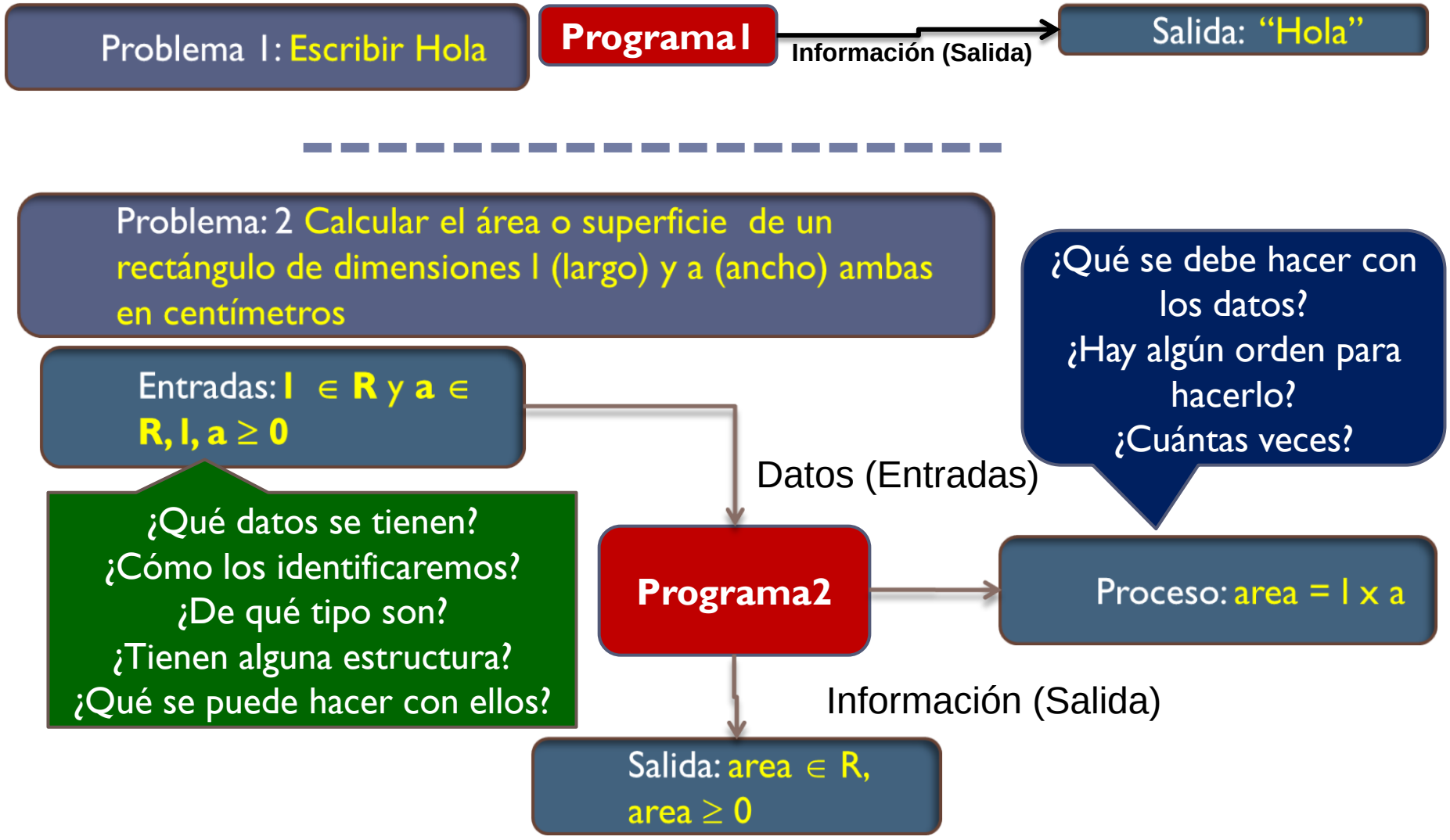
- **Análisis del problema** (Análisis E – P - S)
  - Definición del proceso (P)
    - Operaciones o cálculos necesarios para encontrar la solución del problema
      - ¿Qué tipo de cálculo es necesario? ¿ecuaciones?
      - ¿Cuántas ecuaciones?
      -

Proceso: Calcular el área del rectángulo con la ecuación
area = largo x ancho
area = l x a

  - Especificaciones de salida (S)
    - Resultados finales de los cálculos
      - ¿Cuáles son los datos de salida?
      - ¿Cuántos datos de salida se producirán?
      - ¿Qué precisión tendrán los resultados?
      - ¿Se debe imprimir un encabezado?

Salida: número real mayor o igual a cero que representa el área calculada del rectángulo que está en area

Método de desarrollo de programas



UNIVERSIDAD DE LOS ANDES

Método de desarrollo de programas

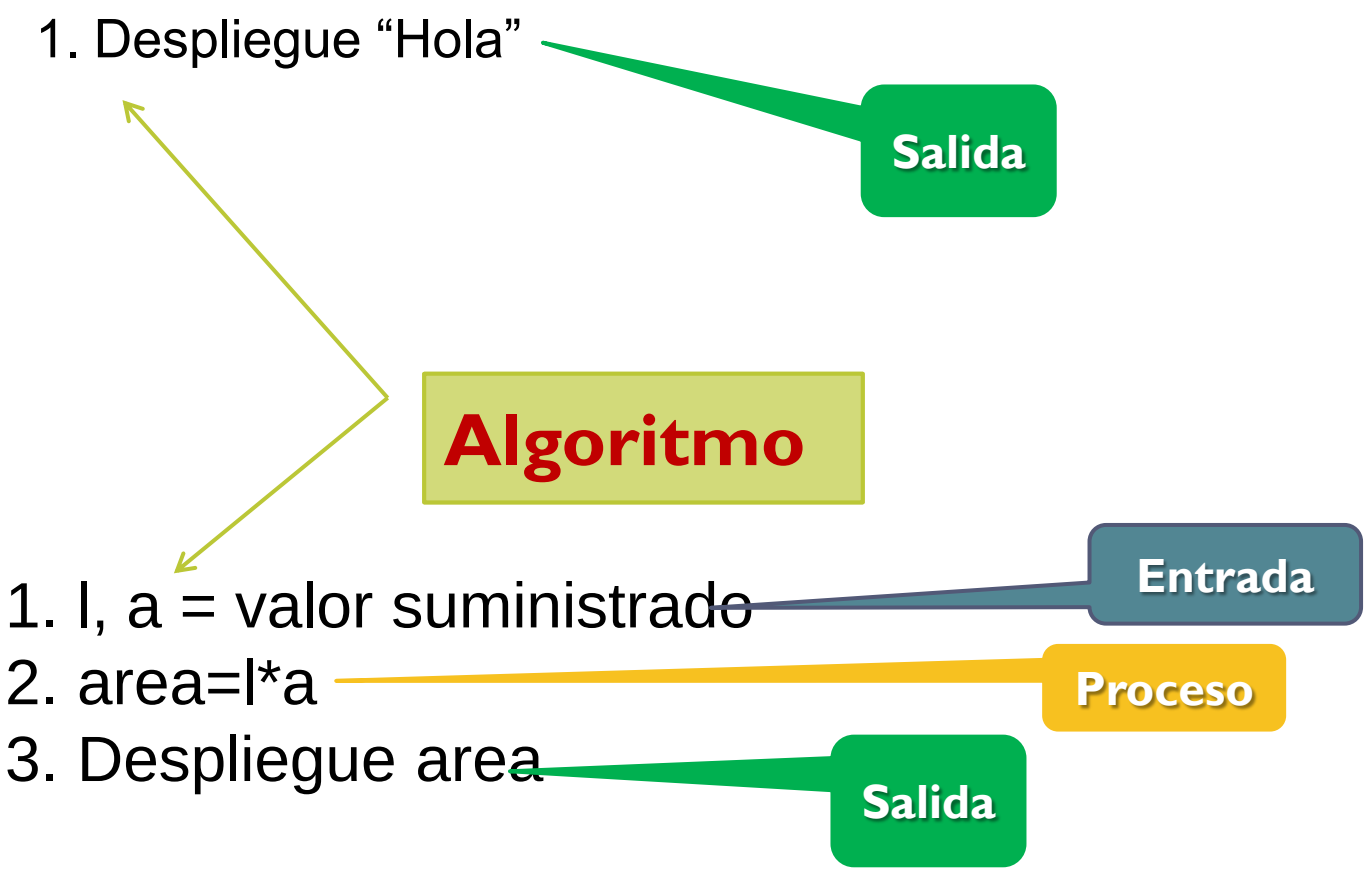
Diseño

- **Diseño de la solución del problema** con un algoritmo
  - Algoritmo: Secuencia finita y ordenada de pasos, sin ambigüedades, que conducen a la solución de un problema dado
    - Un algoritmo debe ser preciso e indicar el orden de realización de cada paso
    - Un algoritmo debe estar definido, si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez
    - Un algoritmo debe ser finito, si se sigue un algoritmo, se debe terminar en algún momento, es decir tiene un número finito de pasos

Algoritmo = Pseudocódigo

X1,…,Xn = valor suministrado
Despliegue X1,…,Xn
X1,…,Xn = E1,…,En

Método de desarrollo de programas



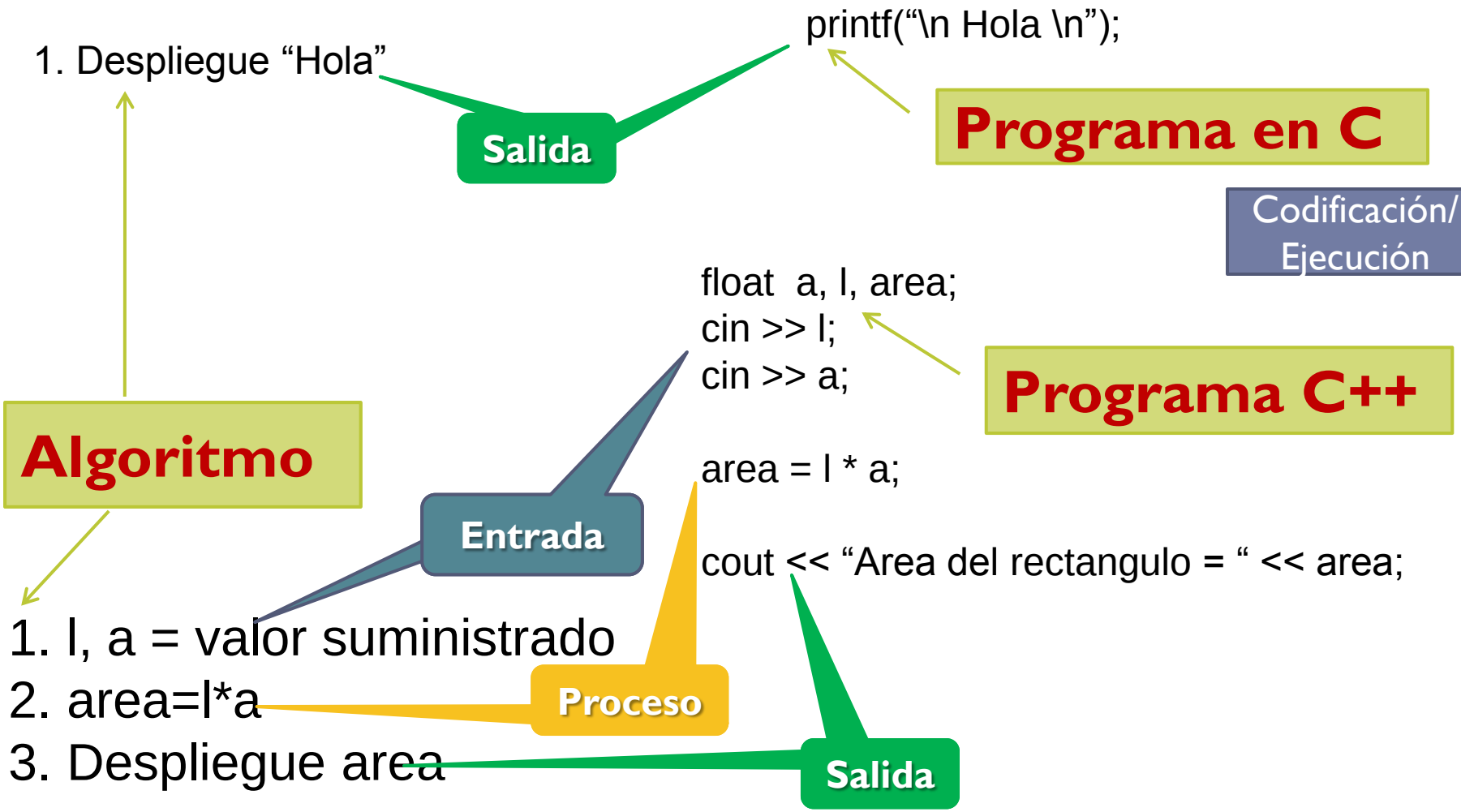
UNIVERSIDAD DE LOS ANDES

Método de desarrollo de programas

- **Codificación**: es la traducción del diseño de la solución del problema (algoritmo) a un programa escrito en un lenguaje de programación
- El programa se escribe en un editor de texto
- **Ejecución o corrida del programa**: invocación del programa objeto obtenido luego de la compilación del programa fuente
- **Programa**: secuencia de sentencias, cada una expresa las operaciones que debe ejecutar la computadora
- Lenguaje de programación: C/C++

Codificación/ Ejecución

Método de desarrollo de programas



Método de desarrollo de programas

- Corrida en frío del programa
  - Prueba manual del programa
    - Selección de un conjunto de datos de entrada
    - Ejecución manual de cada sentencia del programa fuente
    - Verificación de los resultados obtenidos en comparación con los resultados esperados según los datos de entrada
  - El conjunto de datos de entrada debe permitir la prueba de todos los caminos de ejecución posibles dentro del programa
  - Ejemplo:

Pruebas/ Depuración

l	a	area
0	0	0
4.1	5.3	21.73

l	a	area
0	4	0
0	5	0

Método de desarrollo de programas

- Depuración del programa
  - Identificación y eliminación de errores una vez que el programa ha sido compilado
  - Errores de sintaxis
    - Violan las reglas del lenguaje de programación
    - El compilador localizará e identificará la mayoría de estos automáticamente
  - Errores lógicos
    - Equivocaciones que causan que el programa se ejecute de forma inesperada o incorrecta
- Tipos básicos de sentencias: secuenciales (leer, escribir y asignar valores)

Pruebas/ Depuración

UNIVERSIDAD DE LOS ANDES

Método de desarrollo de programas

- Ejemplo de errores de sintaxis

```
cout << "Area del rectangulo = " << area          // falta el ; al final
```

- Una vez editado y corregido el error, el programa fuente debe ser compilado nuevamente, es decir traducido a lenguaje de máquina (código objeto)
- Si el programa fuente tiene errores de sintaxis, no se genera el programa objeto
- Cuando el programa está sintácticamente correcto, el código objeto se encadena con las funciones en las librerías incluidas usando **un encadenador**

UNIVERSIDAD DE LOS ANDES

Método de desarrollo de programas

- El código objeto compilado y encadenado es cargado en la memoria principal para su ejecución por un software denominado **cargador**
- El código objeto compilado y encadenado y cargado (código ejecutable) es **ejecutado** o corrido en el procesador, haciendo uso de los datos de entrada
- **Comprobación del programa**
  - Verificación de los resultados obtenidos después de la ejecución del programa, los cuales deben corresponder con los resultados esperados

Comprobación

Método de desarrollo de programas

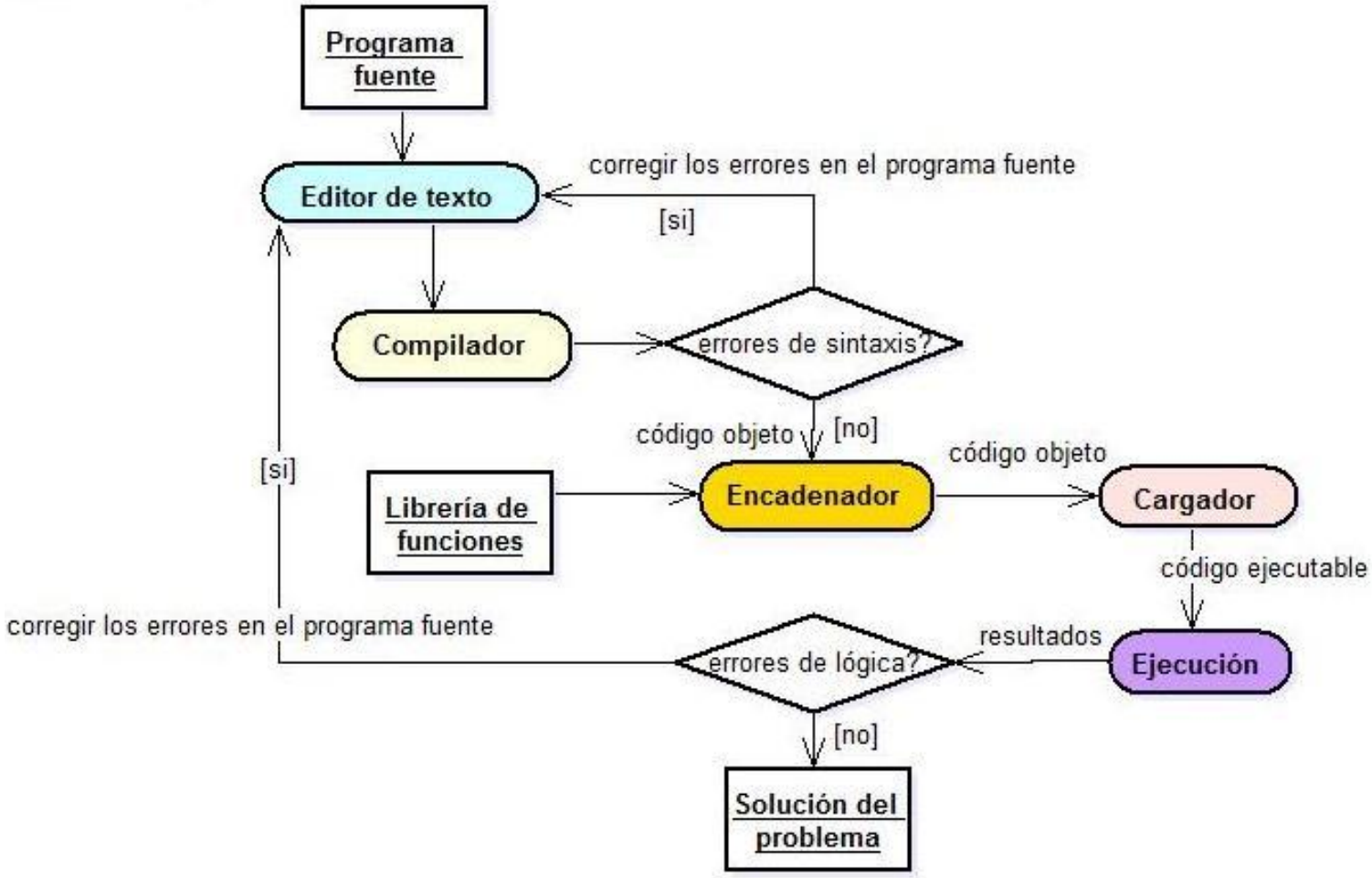
- Comprobar que el programa realiza las tareas para las cuales ha sido diseñado y produce el resultado correcto y esperado
- Si el programa tiene errores de lógica (errores en el método de solución), ellos deben ser corregidos en el programa fuente usando el editor de texto
- Ejemplo:

Comprobación

B = 0

C = 5 / B ⇐ OJO ----- división entre cero!!!!!

Proceso de ejecución de un programa



UNIVERSIDAD DE LOS ANDES

Método de desarrollo de programas

- Puesta en **operación** del programa
  - Instalación del hardware y software
  - Capacitación
- **Mantenimiento** del programa
  - Comienza tan pronto como el producto es puesto en operación
  - Permite corregir defectos menores o
  - Añadir una mayor funcionalidad, bien sea por peticiones del usuario o cliente o por demanda del mercado

Operación

Mantenimiento

Mi primer programa en C

Comentario

```
/* Ejemplo 1
   Mayo, 2009
   Autora: Isabel Besembel
   Despliega el mensaje "Mi primer programa del curso
   PROGRAMACIÓN 1" */
```

Declaración de inclusión o importación de la librería

Inicio del programa Palabras reservadas

```
#include <stdio.h>

int main()
{
    printf("Mi primer programa del curso PROGRAMACIÓN 1");
    return 0;
}
```

Inicio y fin del bloque de sentencias

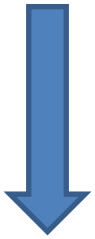
Sentencias

UNIVERSIDAD DE LOS ANDES

Conjunto de sentencias

- El proceso de diseño del algoritmo y posterior codificación del programa, consiste en definir las acciones o sentencias que resolverán el problema
- En un programa, las sentencias por lo general se ejecutan una después de la otra, en el orden en que aparecen escritas
- Programa lineal: las sentencias se ejecutan secuencialmente

Sentencia 1
Sentencia 2
…..
Sentencia n



UNIVERSIDAD DE LOS ANDES

Recordar!!!!

- La programación estructurada utiliza tres estructuras de control
- Todos los programas pueden ser escritos en términos de estas estructuras de control (C. Bohm y G. Jacopini)
- Estructuras de control
  - Secuenciales
  - Selección o de decisión
  - De repetición

- Características del lenguaje de programación C
- ¿Qué son identificadores en C/C++?
- ¿Cómo son los comentarios en C/C++?
- ¿Cuáles son los delimitadores en C/C++?
- ¿Cómo es la estructura de un programa en C/C++?
- ¿Cuál es el método de desarrollo de programas?
- ¿Cómo hacer un programa muy simple?

Resumen

¿Cuáles son los conceptos relevantes de esta clase?

UNIVERSIDAD DE LOS ANDES

Ejercicios

Escriba un programa en C++ que:

1. Dibuje en pantalla un triángulo de hasta 15 asteriscos (*) de base
2. Escriba su nombre encerrado en un recuadro
3. Dibuje un triángulo de dígitos de la manera siguiente

```
1
22
333
4444
```

4. Imprima en pantalla las figuras dada a continuación

```
  *          *
 ***        * *
  *          *
 ***        * *
  *          *
```