UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

Unidad 2. La lógica de programación

Tema 5. Subprogramas

UNIVERSIDAD DE LOS ANDES

Subprogramas

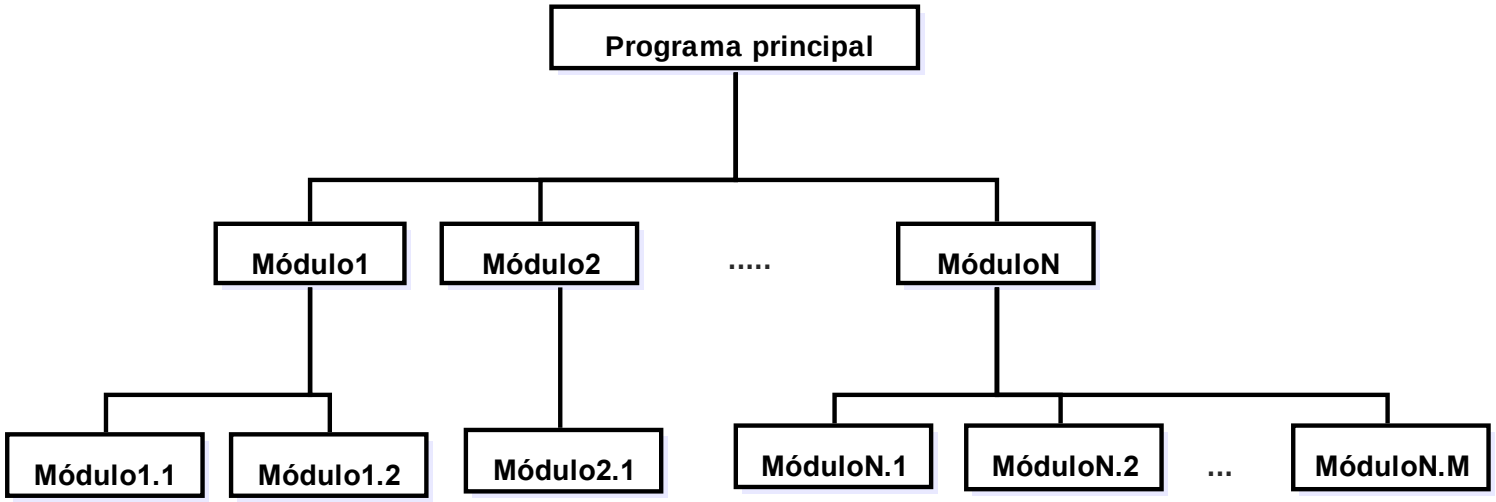
- Contenido:
  - Programación modular
  - Conceptos básicos: módulo, función, parámetros formales y actuales, pre y postcondiciones
  - Tipos:
    - Funciones
      - Funciones de librería: Definición y llamada
    - Procedimientos
  - Representación algorítmica
  - Codificación
- Objetivo:

  Desarrollar habilidades en el uso de subprogramas
- Bibliografía:
  - Deitel y Deitel, cap. 5 y sec. 7.1-7.4.
  - http://www.ing.ula.ve/~ibc/pr1
  - http://www.webdelprofesor.ula.ve/ingenieria/ibc
  - Navas y Besembel, tema VI.
  - Joyanes, sec. 4.2 y 5.1-5.5.

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)
- Reutilizar los módulos o piezas
- Comprender los mecanismos utilizados para pasar información entre módulos

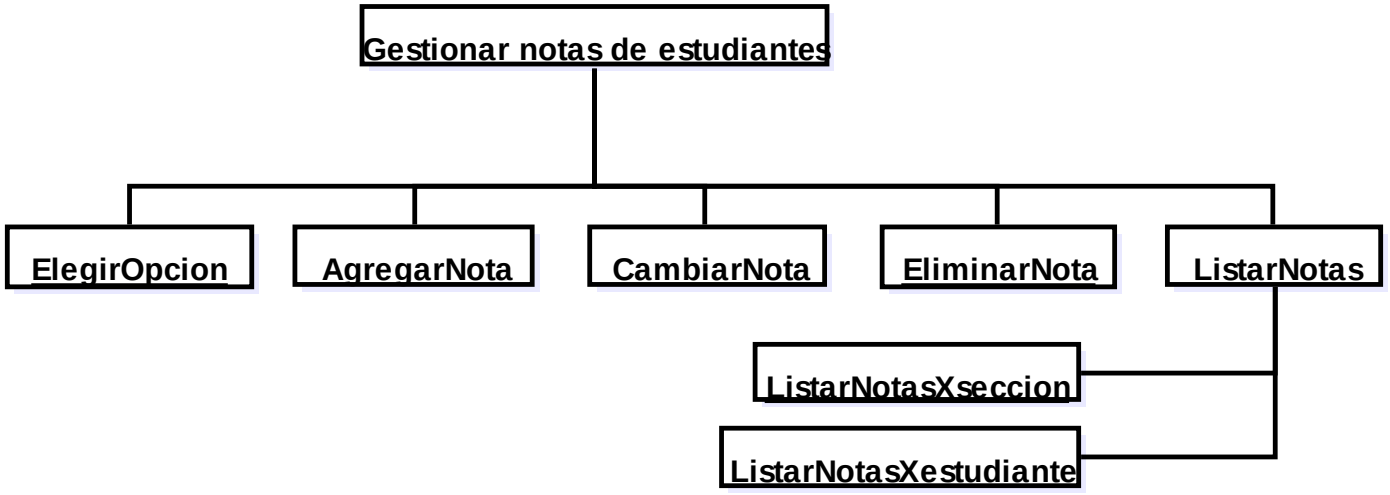


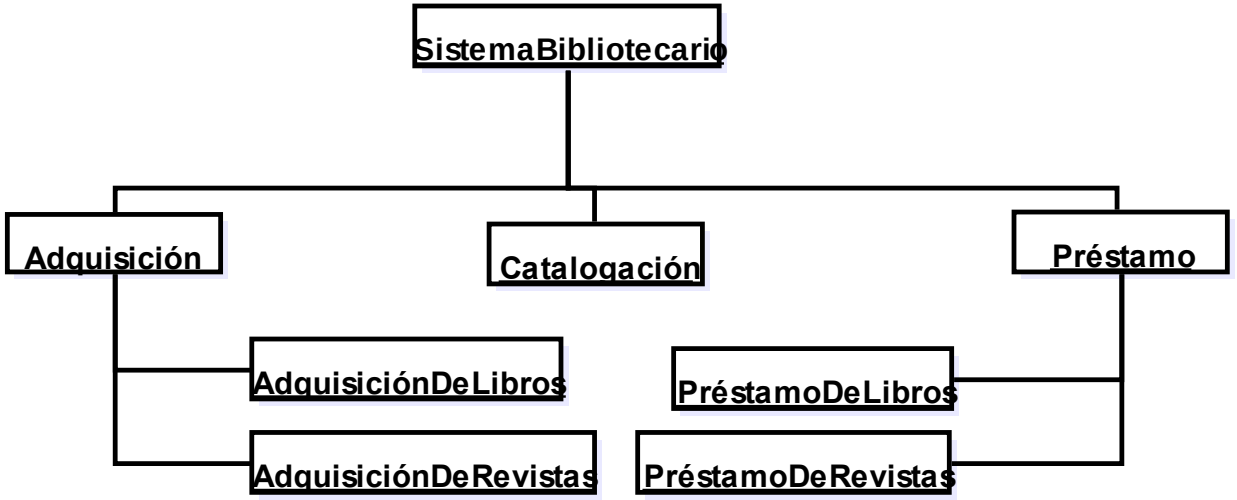
UNIVERSIDAD DE LOS ANDES

Conceptos básicos

- Módulo: pieza pequeña diseñada para ejecutar una tarea específica.
- Permiten desarrollar y mantener programas grandes
- Tipos:
  - Funciones
  - Procedimientos
- Diseño descendente (de arriba hacia abajo) o programación top-down o estrategia de diseño llamada divide-y-vencerás
  - Para solucionar problemas complejos y grandes, es conveniente dividirlo 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.

Ejemplos

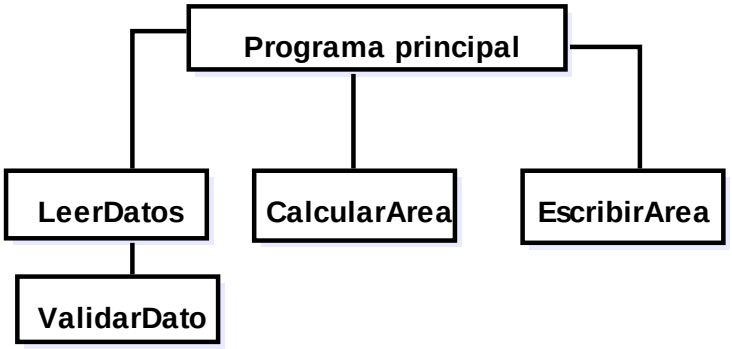

Gestionar notas de estudiantes
ElegirOpcion
AgregarNota
CambiarNota
EliminarNota
ListarNotas
ListarNotasXseccion
ListarNotasXestudiante


SistemaBibliotecario
Adquisición
Catalogación
Préstamo
AdquisiciónDeLibros
AdquisiciónDeRevistas
PréstamoDeLibros
PréstamoDeRevistas

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)



UNIVERSIDAD DE LOS ANDES

Ventajas de la programación modular

- 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
- Favorece la portabilidad, ya que se pueden escribir programas sin prestar atención a las características de un sistema particular

UNIVERSIDAD DE LOS ANDES

Ventajas de la programación modular

- 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

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

Ejemplos:

- Función de un solo argumento

  evaluación de la función en 4

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

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

- Función de dos argumentos

  evaluación de la función en a=2 y b=3

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

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

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
  - Toma cero o mas valores, denominados argumentos o parámetros de entrada (parámetros formales)
  - Según el valor de los parámetros, devuelve un resultado, el cual es obtenido durante su ejecución

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

UNIVERSIDAD DE LOS ANDES

Tipos de funciones en C/C++

- Funciones de biblioteca (librerías) : El lenguaje C/C++ tiene sus propias funciones incorporadas que permiten realizar ciertas operaciones o cálculos de uso común
  - Contiene una amplia colección de funciones para llevar a cabo cálculos matemáticos comunes, manipulaciones con cadenas de caracteres, manipulaciones con caracteres, operaciones de entrada/salida y muchas otras operaciones útiles.
  - Esta biblioteca de funciones comunes construida una vez, puede ser re-utilizada por diferentes programas
- Funciones definidas por el programador para realizar determinadas tareas

TDSO **Definición** ⇒ nomFunción(Tipo: parametroFormal,...):tipoResultado
Llamada ⇒ Xj = objeto.operación(listaParmetrosActuales)

C/C++ **Definición** ⇒ tipoResultado nomFuncion(listaParametrosFormales)
Llamada ⇒ nomVar = nomFuncion(listaParametrosActuales)

UNIVERSIDAD DE LOS ANDES

Librerías ANSI C/C++

Librería	Directiva	Descripción
assert	#include <assert.h>	Únicamente define la macro de depuración assert
ctype	#include <ctype.h>	Contiene los prototipos de las funciones y macros de clasificación de caracteres
errno	#include <errno.h>	Define constantes para los códigos de error
float	#include <float.h>	Contiene parámetros de entorno, información sobre limitaciones y rangos para tipos reales

UNIVERSIDAD DE LOS ANDES

Librerías ANSI C/C++

Librería	Directiva	Descripción
limits	#include <limits.h>	Contiene parámetros de entorno, información sobre limitaciones y rangos para tipos enteros
locale	#include <locale.h>	Contiene los prototipos de las funciones, macros, y tipos para manipular y controlar varias opciones pertenecientes a la localidad del sistema
math	#include <math.h>	Contiene los prototipos de las funciones y otras definiciones para el uso y manipulación de funciones matemáticas

UNIVERSIDAD DE LOS ANDES

Librerías ANSI C/C++

Librería	Directiva	Descripción
setjmp	#include <setjmp.h>	Contiene los prototipos para las funciones y un tipo para crear y manipular el entorno al hacer llamadas: registros, pilas, etc..
signal	#include <signal.h>	Contiene las funciones, macros, y tipos para crear y manipular señales del sistema
stdarg	#include <stdarg.h>	Contiene las macros y tipos para crear y manipular argumentos de variables

UNIVERSIDAD DE LOS ANDES

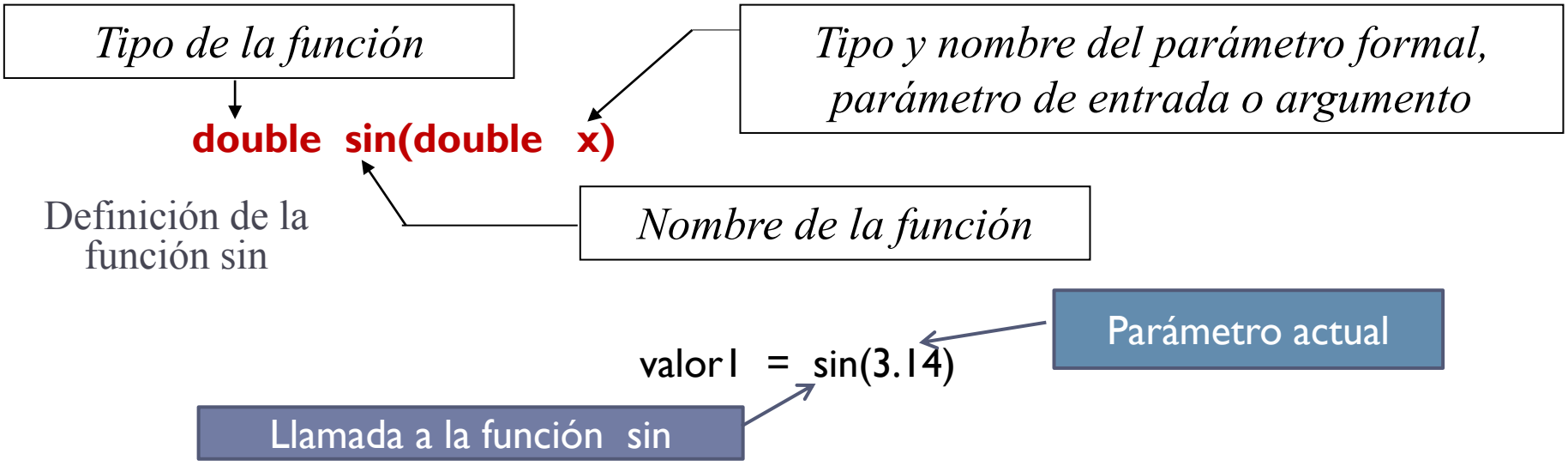
Librerías ANSI C/C++

Librería	Directiva	Descripción
stddef	#include < stddef.h>	Contiene las macros, y tipos comunes
stdio	#include <stdio.h>	Contiene los prototipos de las funciones, macros, y tipos para manipular datos de entrada y salida
stdlib	#include < stdlib.h>	Contiene los prototipos de las funciones, macros,y tipos para utilidades de uso general

UNIVERSIDAD DE LOS ANDES

Librerías ANSI C/C++

Librería	Directiva	Descripción
string	#include < string.h>	Contiene los prototipos de las funciones y macros de clasificación de caracteres
time	#include <time.h>	Contiene los prototipos de las funciones, macros y tipos para manipular la hora y la fecha del sistema



UNIVERSIDAD DE LOS ANDES

Funciones de librería en C/C++

FUNCIÓN	TIPO	PROPÓSITO	ARCHIVO *include*
abs(i)	int	devuelve el valor absoluto de i	stdlib.h
fabs(d)	double	devuelve el valor absoluto de d	stdlib.h
cos(d)	double	devuelve el coseno de d	math.h
cosh(d)	double	devuelve el coseno hiperbólico de d	math.h
exp(d)	double	devuelve el valor e^d	math.h
log(d)	double	devuelve el logaritmo natural de d	math.h
sqrt(d)	double	devuelve la raíz cuadrada de d	math.h
floor(d)	double	devuelve el entero mas grande no mayor que d	math.h
ceil(d)	double	devuelve el entero mas pequeño mayor o igual a d	math.h
pow (d1, d2)	double	devuelve d1 elevado a la potencia d2	math.h
fmod(d1, d2)	double	devuelve el resto de d1/d2 con el mismo signo de d1	math.h
sin(d)	double	devuelve el seno de d	math.h

UNIVERSIDAD DE LOS ANDES

Funciones de librería en C/C++

FUNCIÓN	TIPO	PROPÓSITO	ARCHIVO *include*
acos(d)	double	devuelve el arco coseno de d	math.h
asin(d)	double	devuelve el arco seno de d	math.h
atan(d)	double	devuelve la arco tangente de d	math.h
atan(d1, d2)	double	devuelve la arco tangente de d1/d2	math.h
atof(s)	double	convierte la cadena s a una cantidad en doble precisión	stdlib.h
atoi(s)	int	convierte la cadena s a un entero	stdlib.h
atol(s)	long	convierte la cadena s a un entero largo	stdlib.h
getchar()	int	lee un carácter desde el dispositivo de entrada estándar	stdio.h
putchar(c)	int	escribe un carácter en el dispositivo de salida estándar	stdio.h
tolower(c)	int	convierte una letra a minúscula	ctype.h
toupper(c)	int	convierte una letra a mayúscula	ctype.h

UNIVERSIDAD DE LOS ANDES

Tipos de parámetros

- **Parámetros formales** (parámetros de entrada o argumentos): Declaraciones de los parámetros en la definición de la función
- **Parámetros actuales**:Valores que toman los parámetros formales y que son proporcionados a la función que es llamada por la función que la llamó
- **Función**: double pow(double x, double z)

Propósito	Ejemplo

Calcula x elevado a la potencia de z
Puede producirse un error de dominio si x es negativo y z no es un valor entero
También se produce un error de dominio si el resultado no se puede representar cuando x es cero y z es menor o igual que cero
Retorna el resultado de x^z
	#include<stdio.h>
	#include<math.h>
	int main()
	{ double x=6.54321, z=0.56789;
	printf("pow(%f, %f) = %f\n", x, z, pow(x,z));
	return 0;
	}

Definición de una función en C/C++

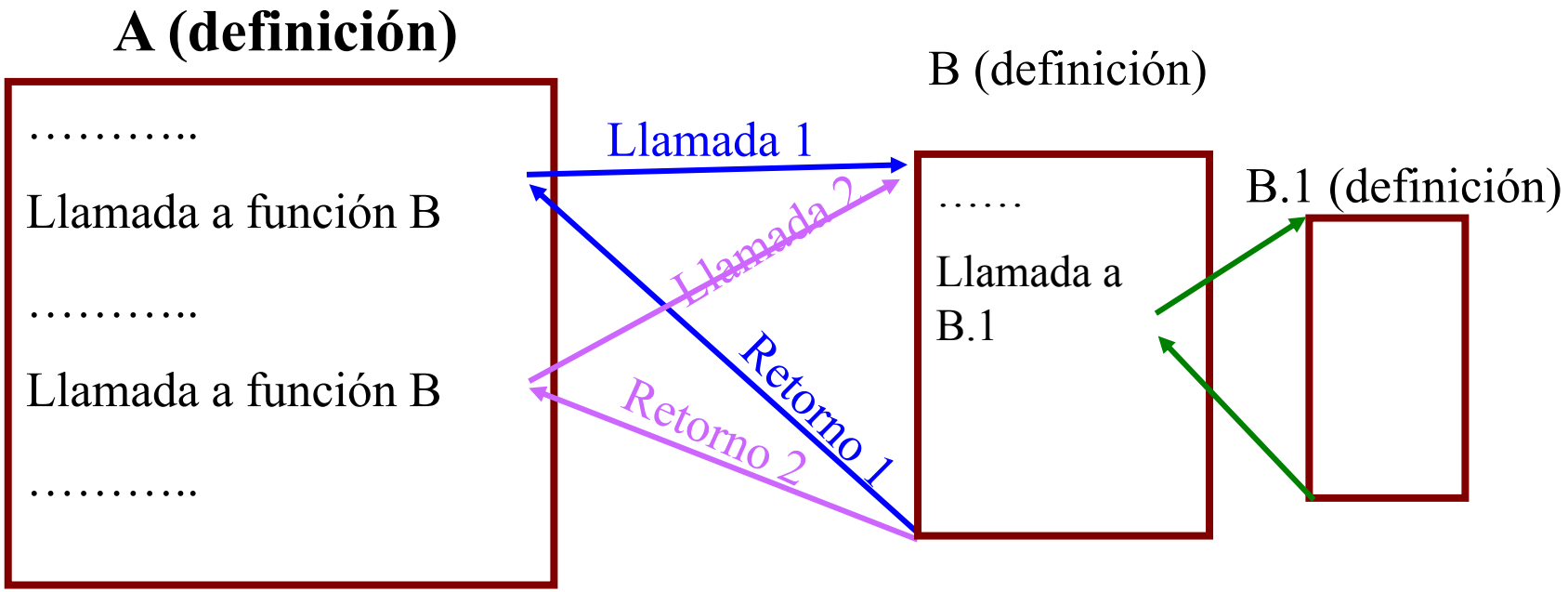
```
tipoResultado  nombreDeLaFunción (listaParámetrosFormales)
{
      DeclaraciónDeVariablesLocales

            Conjunto de sentencias
            return expresiónDelTipoResultado;
}
```

- **Todas las funciones devuelven UN SOLO VALOR**

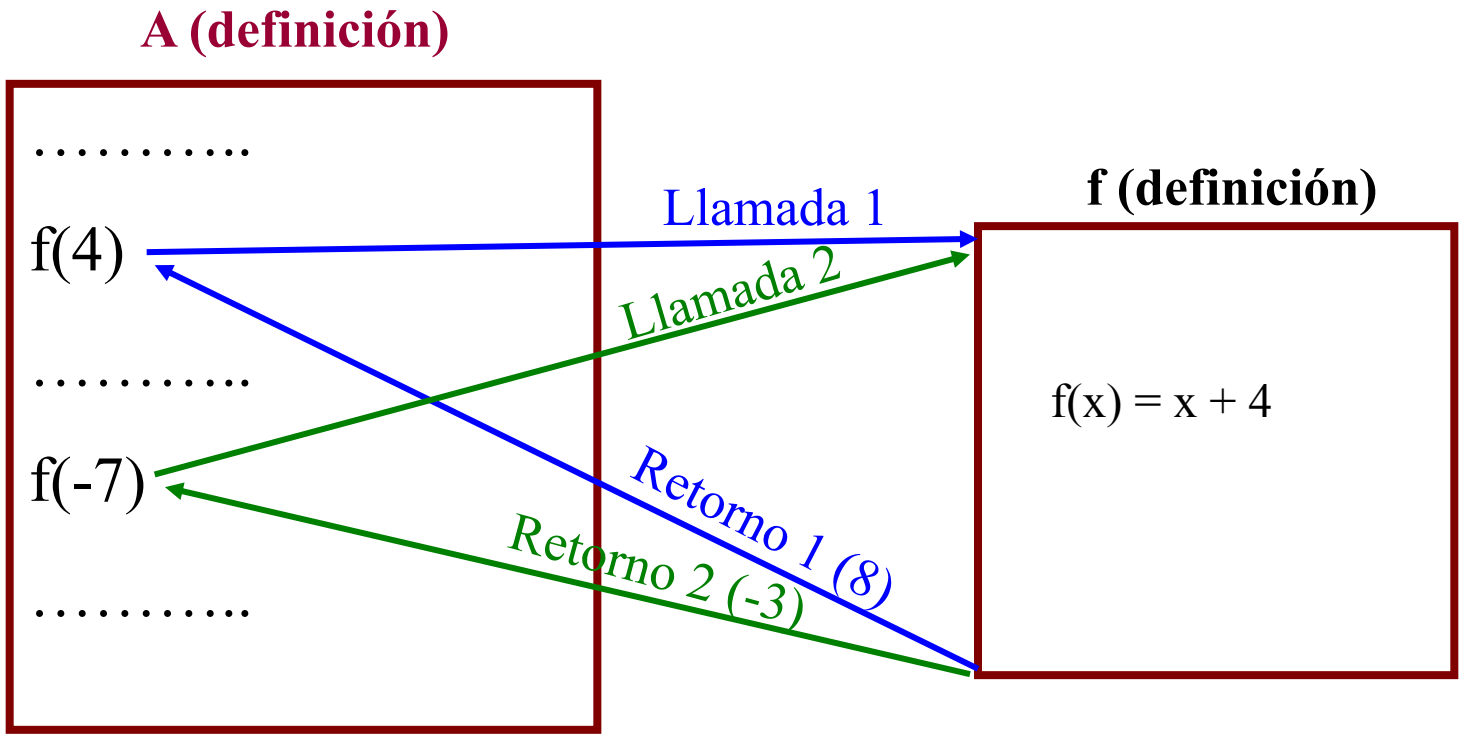
- La lista de parámetros formales y la declaración de variables locales son opcionales

Mecanismo de llamadas entre funciones



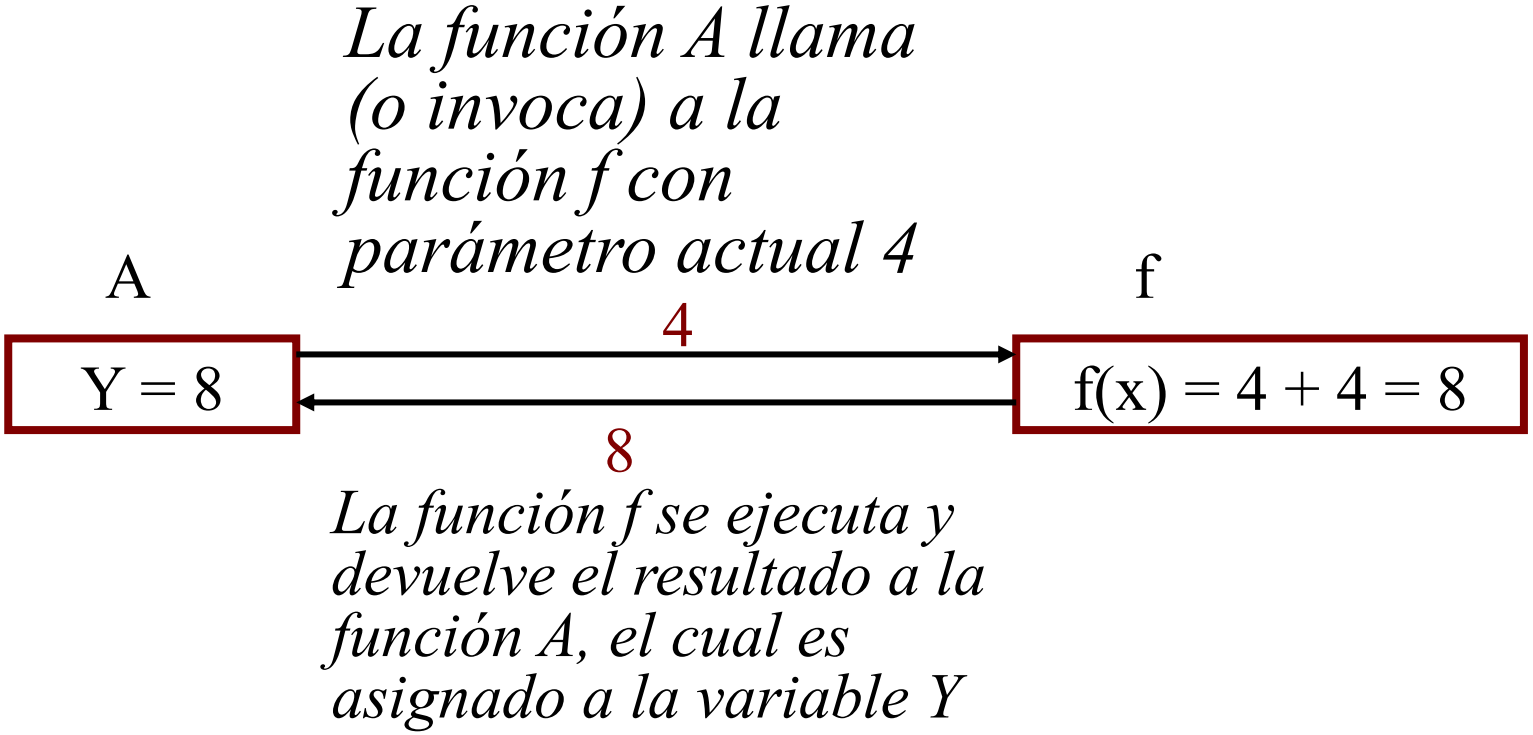
Nota: Cada vez que una función es llamada, se ejecuta y retorna el control al lugar desde donde fue hecha la llamada

Mecanismo de llamadas entre funciones



Mecanismos utilizados para pasar información entre funciones

- Parámetros: proporcionan la forma de comunicar información entre funciones



UNIVERSIDAD DE LOS ANDES

Ejercicio resuelto 1

- Enunciado del problema

  Realizar la función que calcule $f(x) = \frac{4}{x^2 + 1}$

- Análisis E-P-S

Entrada	Variable	Descripción	Tipo de dato	Rango válido
	x	Variable asociada a la función	Real	El disponible en el lenguaje
Proceso	Calcular fx = 4 / (x * x + 1)			
Salida	Mensaje indicando fx	Valor resultante de aplicar la fórmula	Real	El disponible en el lenguaje

Ejercicio resuelto 1. Diseño

Precondiciones: cosas que son ciertas al entrar al subprograma

Postcondiciones: cosas que son ciertas al salir del subprograma

	f(Real x): Real	
	{pre: }	{pos: fx ∈ ℜ}
1 2	fx = 4 / (x * x + 1) Devolver fx	•**x:** Real. Resultado que se obtiene al evaluar la función f(x) en el valor de x enviado por el parámetro formal
1 2 3	x = 4 = fx = 0.235 x = 0 = fx = 4.0 x = -4 = fx = 0.235	Caso exitoso Caso borde Caso exitoso

Nombre de la función

Tipo de dato del resultado

Lista de parámetros formales

UNIVERSIDAD DE LOS ANDES

Ejercicio resuelto 1. Implementación 1

```
#include <stdio.h>
float    fx( float   x)
{  return (x*x+1);
}
int main()
{  float            x;
   printf("Introduzca el valor  x =");
   scanf("%f", &x);
   printf("Calculo de f(%f) = ",  fx(x));
   return 0;
}
```

UNIVERSIDAD DE LOS ANDES

Ejercicio resuelto 1. Implementación 2

```
#include <stdio.h>
float    fx( float   x)
{  return (pow(x, 2.0)+1);
}
int main()
{  float            x;
   printf("Introduzca el valor  x =");
   scanf("%f", &x);
   printf("Calculo de f(%f) = ",  fx(x));
   return 0;
}
```

UNIVERSIDAD DE LOS ANDES

Resumen

tipoResultado nombreDeLaFunción (*listaParámetrosFormales*)

{

DeclaraciónDeVariablesLocales

Conjunto de sentencias

return *expresiónDelTipoResultado;*

}

nombreDeVariable = nombreDeLaFunción(listaParámetrosActuales)

TDSO	**Definición** ⇒ **nomFunción**(Tipo: parámetroFormal,...):*tipoResultado* **Llamada** ⇒ Xj = **nomFunción**(listaParámetrosActuales)
C/C++	**Definición** ⇒ *tipoResultado* **nomFuncion**(listaParametrosFormales) **Llamada** ⇒ nomVar = **nomFuncion**(listaParametrosActuales)

UNIVERSIDAD DE LOS ANDES

Ejercicio resuelto 2

- Enunciado del problema

  Leer una letra en minúsculas y convertirla en mayúsculas

- Análisis E-P-S

Entrada	Variable	Descripción	Tipo de dato	Rango válido
	letra	Letra leída desde el teclado	Caracter	$\in$[a, z]o[A, Z]
Proceso	Convertir la letra en su correspondiente letra mayúscula			
Salida	Mensaje indicando la letra original y su correspondiente mayúscula		Caracter	[A, Z]

Ejercicio resuelto 2. Diseño y codificación en C++

converstidorAmayuscula {pre: letra ∈ {Caracteres}}		{pos: letraM ∈ ['A', 'Z'] }
1 2	[Desplegar "Introduzca una letra " letra = valor suministrado] (letra ∉ {letras}) Desplegar "La mayuscula de ", letra, "es ", mayuscula(letra)	●**letra:** Caracter. Letra leida del teclado y validada para que este entre la 'a' y la 'z'.
1 2	letra = 'z' => letraM = 'Z' letra = 'a' => letraM = 'A'	Caso límite Caso exitoso

```
#include <iostream>
#include <stdio.h>
#include <ctype.h>
using spacename std;
void main()
{   char letra;
    do
    {    cout << "\nIntroduzca una letra ";
         letra= getchar();
    }while(letra<'a'||letra>'z');
    cout<<"La mayuscula de "<<letra<<" es "<<toupper(minuscula);
}
```

- ¿Qué es la programación modular?
- ¿Cuáles son las ventajas de esta programación?
- ¿Qué es un módulo?
- ¿Qué es una función?
- ¿Qué son parámetros formales y actuales?
- ¿Qué son pre y post condiciones?
- ¿Cómo se define y se invoca o llama a un subprograma?
- ¿Qué son funciones de librería?

Resumen

¿Cuáles son los conceptos relevantes de esta clase?

UNIVERSIDAD DE LOS ANDES

Ejercicios

- **Para cada uno de los enunciados dados a continuación realizar: Análisis en E-P-S, diseño en pseudocódigo y codificación en C o C++ utilizando funciones de librería**

1. Ordene descendentemente un conjunto de valores dados que pertenecen al intervalo [2, 120]
2. Encuentre el valor del polinomio $T(x,y)=x^2+y^4+xy^3-30$, para valores de x y dados. Valide x y para que el resultado de T(x,y) sea correcto.
3. Ordene de menor a mayor un conjunto de letras dadas
4. Si se tiene que:

```
int i = 8, j = 5;
double x = 0.004, y = -0.01;
char c = 'c', d = 'd';
```

Determinar el valor de cada una de las siguientes expresiones, que hacen uso de funciones de biblioteca.

a) abs(i - 2 * j)	b) fabs(x + y)	c) ceil (x)
d) ceil (x + y)	e) floor (x)	f) floor(x + y)
g) exp(x)	h) log(x)	i) log(exp(x))
j) sqrt(pow(x, 2) + y*y)	k) sqrt(sin(x) + cos(y))	l) pow(x - y, 3.0)