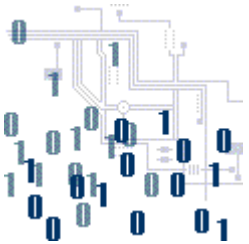
UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA



Unidad 1. Lógica de programación

Tema 2. Tipos básicos de datos y estructuras secuenciales

UNIVERSIDAD DE LOS ANDES

Tipos básicos de datos y estructuras secuenciales

- Contenido
  - Tipos de datos y tipos abstractos de datos (TAD)
  - Variables, memoria y asignación
  - Representación y operaciones:
    - Enteros
    - Reales
    - Apuntador
  - Declaraciones de variables y constantes
    - Locales
    - Globales
  - Estructuras secuenciales
  - Expresiones aritméticas
- Objetivo
  - Desarrollar habilidades en el uso de los tipos básicos de datos con la visión de TAD y de las estructuras secuenciales
- Bibliografía
  - Deitel y Deitel, cap. 2 y 9, sec. 3.11, 3.12, 4.10 y 4.11 y apéndices C y D
  - Navas y Besembel, tema I-III
  - Joyanes, sec. 1.4 - 1.6, 1.9 y 4.4

UNIVERSIDAD DE LOS ANDES

Tipo de dato o dominio

- Dato: expresión general que describe los objetos con lo que opera una computadora
- Conjunto de valores sobre el cual se pueden realizar un conjunto de operaciones
- Valor es similar a elemento perteneciente a un conjunto que representa al tipo de dato.
- El conjunto de valores define el tipo de operaciones que se pueden aplicar sobre sus valores
- **Valor** es equivalente a **elemento**
- Ejemplo:

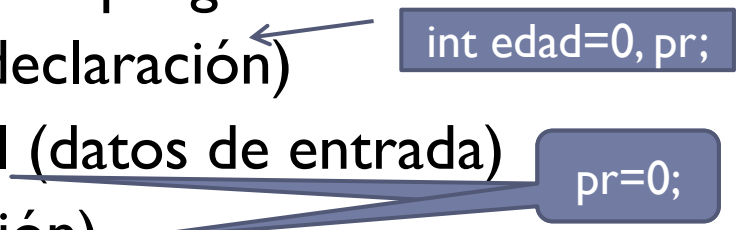
  El valor 2 es un elemento perteneciente al conjunto Enteros

  Enteros = {-∞ …, -3, -2, -1, 0, 1, 2, 3, 4, ... ∞}
- **tipo de dato** es equivalente a **conjunto** o **dominio**

UNIVERSIDAD DE LOS ANDES

Variable

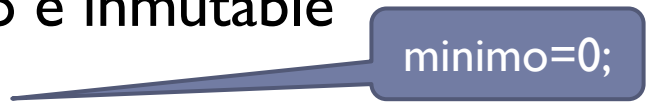
- Elemento de la memoria que sirve para almacenar un valor, referenciado por un **nombre** y perteneciente a un tipo de dato
- Toda variable **debe** estar asociada a un tipo de dato
- A diferencia del valor, la variable tiene propiedades espacio-temporales, es decir, ocupa un espacio determinado de memoria que puede almacenar un valor distinto en cada instante de tiempo
- Las variables contienen el estado de un programa
    1. Se definen, crean e inician (declaración)
    3. Se cargan con un valor inicial (datos de entrada)
    4. Su valor se modifica (asignación)
    5. Llegan a un valor final (resultado de salida)



UNIVERSIDAD DE LOS ANDES

Criterios para variables

- Cuantas menos, mejor
- Cada una con un significado muy claro e inmutable
- No olvidarse de darles un valor inicial



- Controlar (y comprobar) que van tomando valores sensatos: regularmente aplicar predicados que deben satisfacer y notificar si alguna se sale de lo previsto
- Nombre de variable es un identificador que consta de uno o varios caracteres siguiendo las reglas del lenguaje de programación utilizado
- Ejemplos:

  sueldoBase, minimo, maximo, sumaDe1a20, IVA, k1, precio, etc.

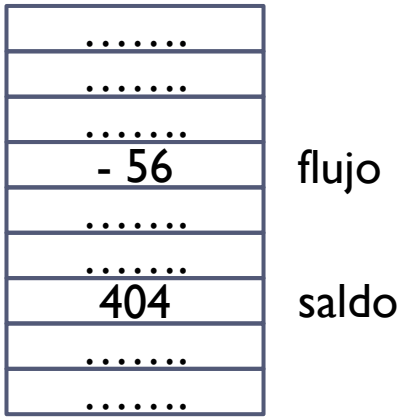
UNIVERSIDAD DE LOS ANDES

Memoria

- Puede ser interpretada como un conjunto de pares (variable, valor) que lleva asociada dos operaciones: Búsqueda (lectura) y Almacenamiento (escritura)
- Memoria = {(Variable$_1$,Valor$_1$), ...,(Variable$_n$,Valor$_n$)}
- Búsqueda (Variable, Memoria) = Valor
- Almacenamiento (Variable, Valor, Memoria)

Lectura de memoria

Escritura en memoria



Memoria = {…, (flujo, -56), …, (saldo, 404), ….}

UNIVERSIDAD DE LOS ANDES

Asignación

- Expresión sintáctica de la operación de almacenamiento
- Modifica el contenido de una variable, escribe el valor en la dirección de memoria asociada a la variable
- Notación algorítmica

  nombreDeVariable ← valor o resultadoDeExpresión o nombreDeVariable
- Notación en C/C++

  nombreDeVariable = valor | resultadoDeExpresión | nombreDeVariable
- Ejemplo:

  flujo ← -56 — Memoria = {…, (flujo, -56), ….}

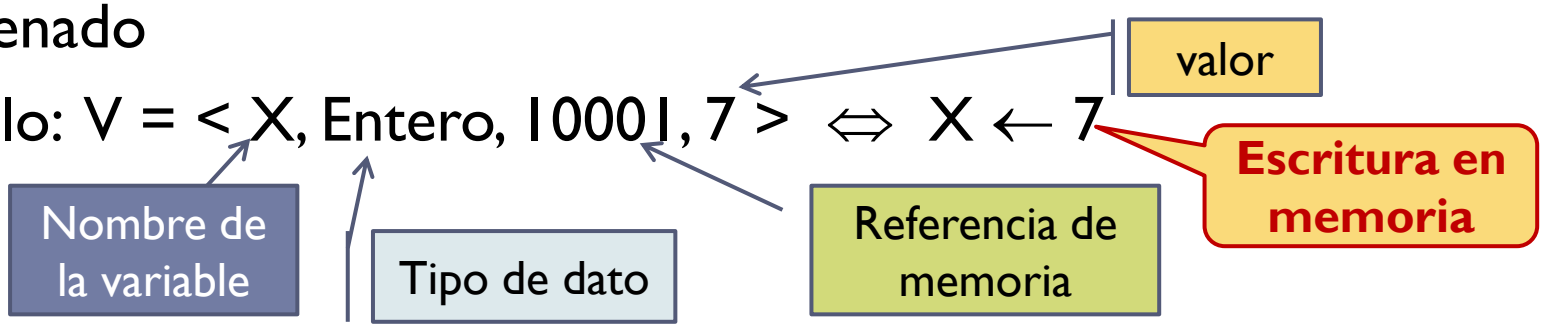
  saldo ← 404 — Memoria = {…, (flujo, -56), …, (saldo, 404), …}

  flujo ← 108 — Memoria = {…, (flujo, 108), …, (saldo, 404), …}
- Cada operación de asignación transforma la memoria de un estado a otro a lo largo del tiempo

UNIVERSIDAD DE LOS ANDES

Asignación

- La asignación X ← 7 equivale a decir que X es una variable a la que se le asigna el valor 7
- La asignación X ← X + 3 evalúa la expresión derecha, en donde la variable con nombre X tiene el valor de 7, sumándole la constante 3 y asignando el resultado (7 + 3) = 10 a la parte izquierda de la expresión, la variable X
- Cuádrupla V = < N, T, R, K >, donde N es el **nombre** de la **variable**, T su **tipo de dato**, R una **referencia de memoria** asignada a la variable para su almacenamiento y K el **valor** almacenado
- Ejemplo: V = < X, Entero, 10001, 7 > ⇔ X ← 7



UNIVERSIDAD DE LOS ANDES

Tipo abstracto de dato (TAD)

- Un tipo de dato T se define "como una clase de valores y una colección de operaciones sobre esos valores. Si las propiedades de esas operaciones son especificadas solamente con axiomas, entonces T es un tipo abstracto de dato o una abstracción de dato" [Gutt-78]
- Una implementación correcta del TAD cumple con todos los axiomas especificados para él
- Ejemplo:

  El TAD Fecha tiene como representación los atributos dia, mes y año, siendo los tres de tipo Entero, donde 1≤ dia ≤ MD, 1 ≤ mes ≤ 12 y 1 ≤ año ≤ MáximoEntero. MD dependerá del valor del mes y de si el año es bisiesto o no

  Operaciones: fijarFecha(), cambiarFecha(), mostrarFecha(), etc.

UNIVERSIDAD DE LOS ANDES

Tipo abstracto de dato (TAD)

- La especificación por axiomas algebraicos para el tipo T se compone de:
  - una **especificación sintáctica** donde se definen los nombres, dominios y rangos de las operaciones sobre T
  - una **especificación semántica** que está compuesta del conjunto de axiomas en forma de ecuaciones, que dicen como opera cada una de las operaciones especificadas sobre las otras.
- La implementación se compone de:
  - una **representación** que especifica como los valores del TAD serán almacenados en la memoria, y
  - los **algoritmos** que especifican como será usada y manipulada la representación, es decir las operaciones del TAD

UNIVERSIDAD DE LOS ANDES

Tipos de datos

- Por el hecho de que distintos valores pertenecientes a diferentes tipos de datos pueden tener la misma representación a nivel de máquina, la especificación del tipo de dato (dominio, rango y operaciones aplicables) permite controlar la interpretación para cada uno
- Ejemplo:

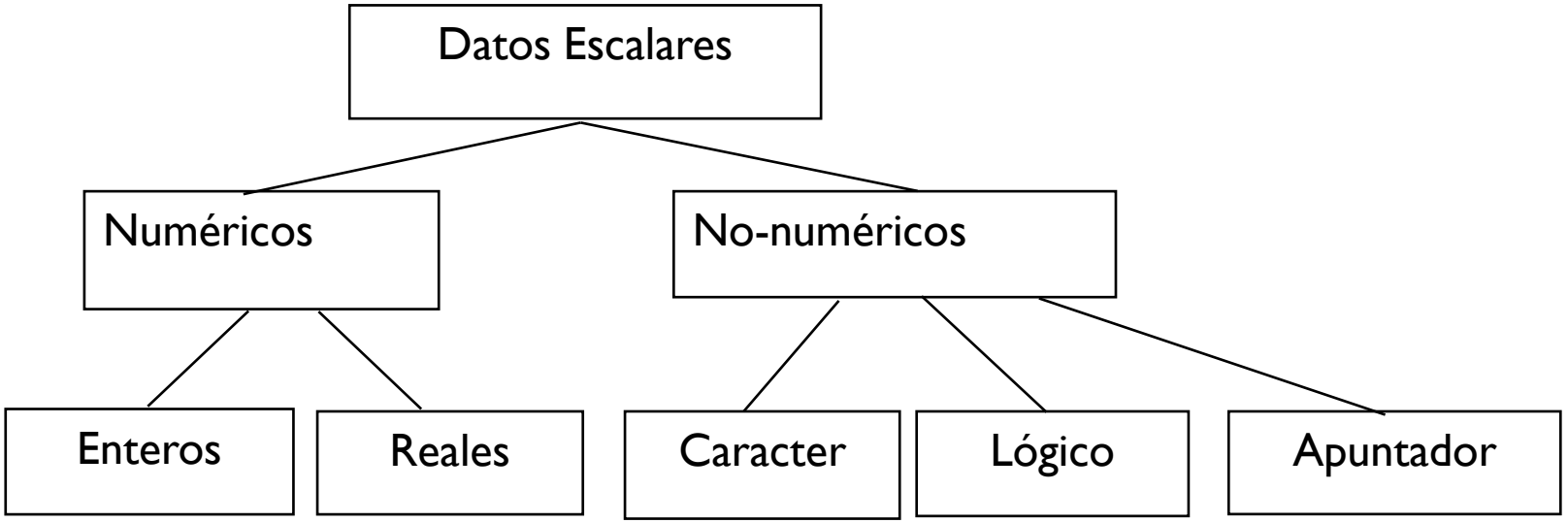
  La secuencia de bits 01000001 (alfabeto binario de longitud 8) puede ser interpretado como:

  1. Caracter 'A' en el tipo de dato Caracter
  2. Entero +65 en el tipo de dato Entero
  3. Real 4.5678 en el tipo de dato Real

UNIVERSIDAD DE LOS ANDES

Tipos de datos

- Puede clasificarse como escalar o estructurado
- Escalar o simple: Aquel cuyo dominio presenta una propiedad de orden (Entero, Real, Caracter, Lógico, Apuntador)
- Estructurado o compuesto: Aquel que se define mediante composición de tipos de datos (vector, cadena de caracteres, matriz, registro)



Tipos de datos escalares numéricos en C/C++

Tipo de dato: Entero C/C++: int, integer, long, etc.

- Subconjunto finito de los números enteros, cuyo rango o tamaño dependerá del lenguaje de programación y de la computadora utilizada
- Dependiendo del número de bits empleado en cada procesador (n), los dominios del tipo de dato Entero varían:

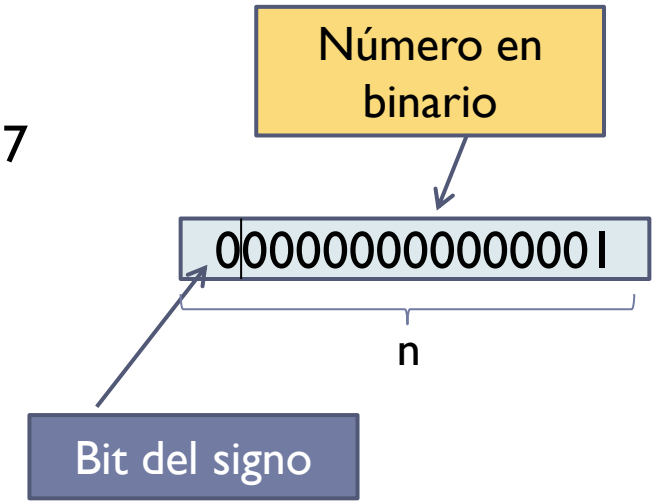
(a) $-2^{(n-1)}, \ldots, -1, 0, 1, \ldots, 2^{(n-1)} - 1$

n=16 -32.768,…,-1,0,-1,…, 32.767

(b) Solo positivos $0, 1, \ldots, 2^{(n-1)}$

n=16 0,1,…, 65.535

n=32 0,1,…, 4.294.967.295



Tipo de dato Entero en C/C++

Tipo	Nro. bits	Rango
int	16	-32768 … +32767
long int	32	-2147483648 … +2147483647
unsigned int	16	0 … +65535
unsigned long int	32	0 … +4294967295

$-2^{31}, \ldots, 2^{31}-1$

$2^{32}-1$

signed int (16), short int (16), unsigned short int (16), signed short int (16), signed long int (32), unsigned long int (32), long long int (64), insigned long long int (64), signed long long int (64)

UNIVERSIDAD DE LOS ANDES

Tipo de dato Entero en C/C++

Operación	Operador	Operador en C/C++	Ejemplo
Suma	+	+	24 + 56
Resta	-	-	5 - 4
Multiplicación	x	*	4 * 5
División	/ o —	/	34 / 6
Resto o módulo		%	68 % 2
Incremento unitario		++	7++
Decremento unitario		--	6--

Tipos de datos escalares numéricos en C/C++

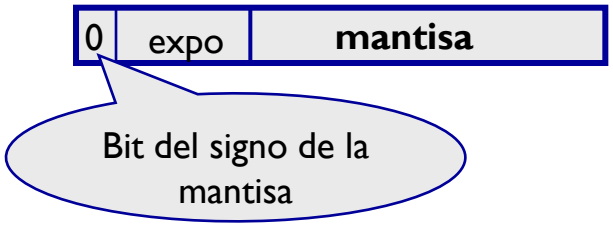
Tipo de dato: Real **C/C++: float, double, long double**

- Subconjunto de los números reales limitado no sólo en el tamaño, sino también en cuanto a la precisión (que depende del procesador)
- Se conocen como números en punto flotante, su representación consta de una mantisa (parte fraccional), de una base y de un exponente (potencia a la que se eleva la base)

Para el número 0.437875×10^3 se tiene:

mantisa = 0.437875
base = 10
exponente = 3

Ejemplos
0.08 3739.41 -3.7452 52.3244 -8.12 3.0



UNIVERSIDAD DE LOS ANDES

Tipo de dato Real en C/C++

Tipo	Nro. bits	Rango	Precisión
float	32	$1.17x10^{-38}$ a $3.4x10^{38}$	6 dígitos decimales
double	64	$2.22x10^{-308}$ a $1.79x10^{308}$	15 dígitos decimales

Long double (128) 10 dígitos de precisión

UNIVERSIDAD DE LOS ANDES

Tipo de dato Real en C/C++

Operación	Operador	Operador en C/C++	Ejemplo
Suma	+	+	7.89 + 56.4
Resta	-	-	6.23 – 2.45
Multiplicación	x	*	1.34 * 54.3
División	/ o —	/	34.6 / .96
Incremento unitario		++	7.55++
Decremento unitario		--	634.67--

Incremento o decremento de la parte **entera solamente**

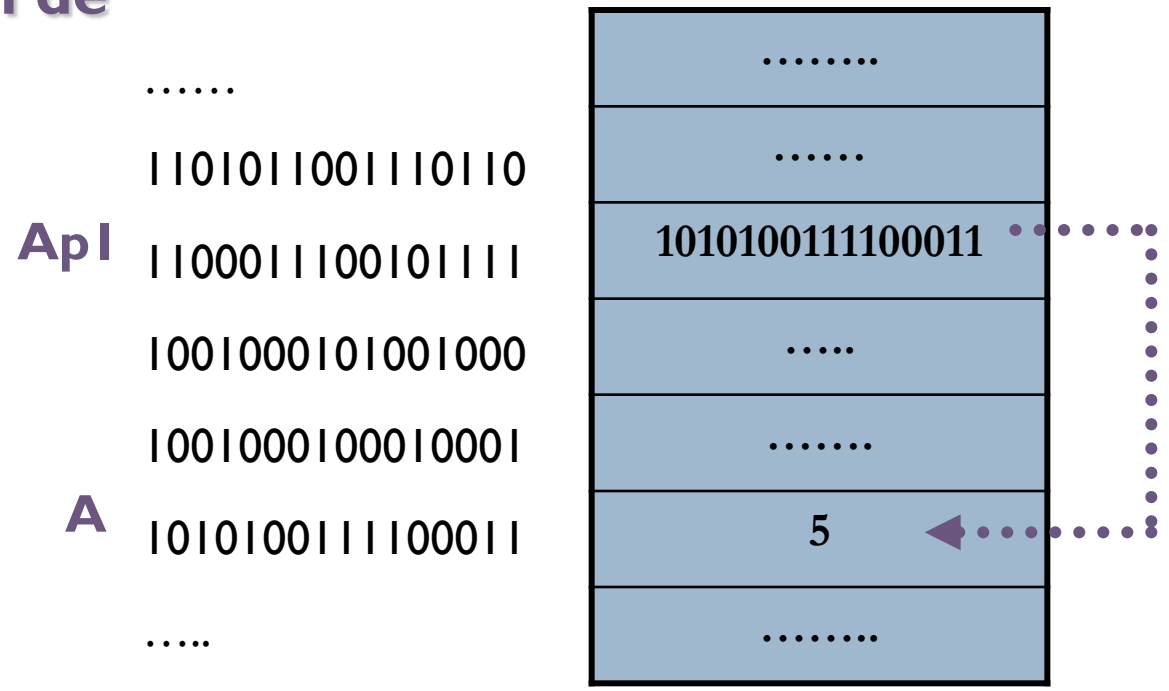
Tipos de datos no escalares en C/C++

Tipo de dato: Apuntador C/C++: prefijo *, &

- Variable especial cuyo contenido es la dirección o localización de memoria de otra variable

&: la dirección de

Ap1 es una variable de tipo apuntador y A una variable de tipo entero



UNIVERSIDAD DE LOS ANDES

Declaración de variables en C/C++

- En C/C++ todas las variables que van a ser usadas en un programa deben ser declaradas antes de ser usadas
- Los objetivos de la declaración de variables son:
  - Asociar un tipo de dato y un identificador (o nombre) a la variable para que el compilador pueda verificar la correctitud de las operaciones en donde interviene la variable
  - Permitir que el compilador sepa cuánto espacio de memoria se necesita para almacenar el valor de la variable, y asignar la dirección de memoria donde este valor se va a almacenar

```
tipoDeDato        listaDeVariablesSeparadasPorComa;
```

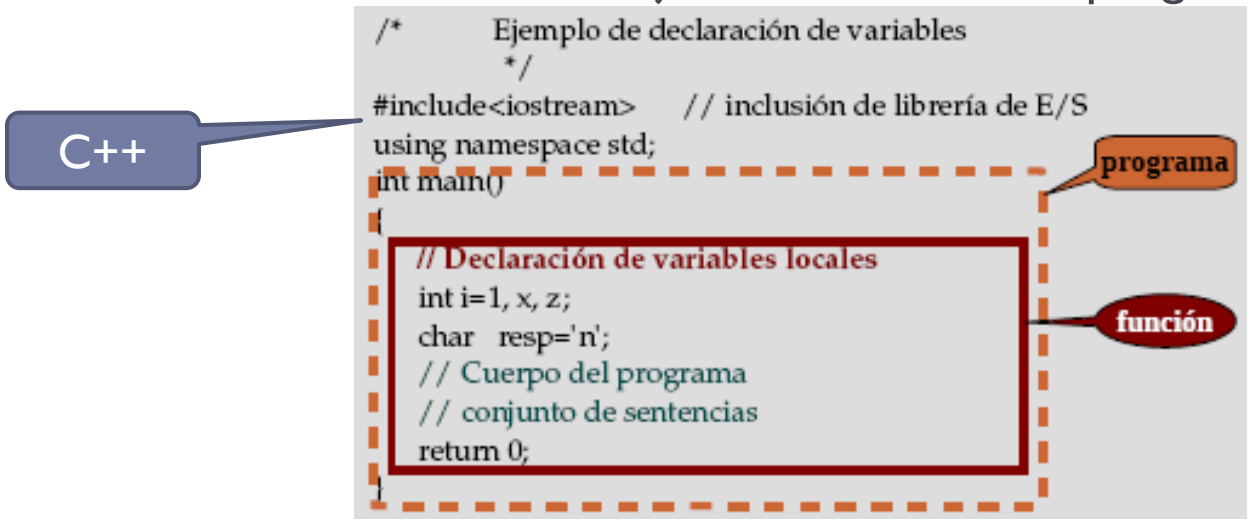
Ejemplos

```
int                 minimo, maximo;
unsigned int        edad=34;   // Iniciación de variable al momento de la declaración
float               pi = 3.14159, sueldoBase, sueldoNeto;
double              peso, temperatura, densidad, precio;
unsigned long int   B = 294967295;
long int            cantidad, distancia;
```

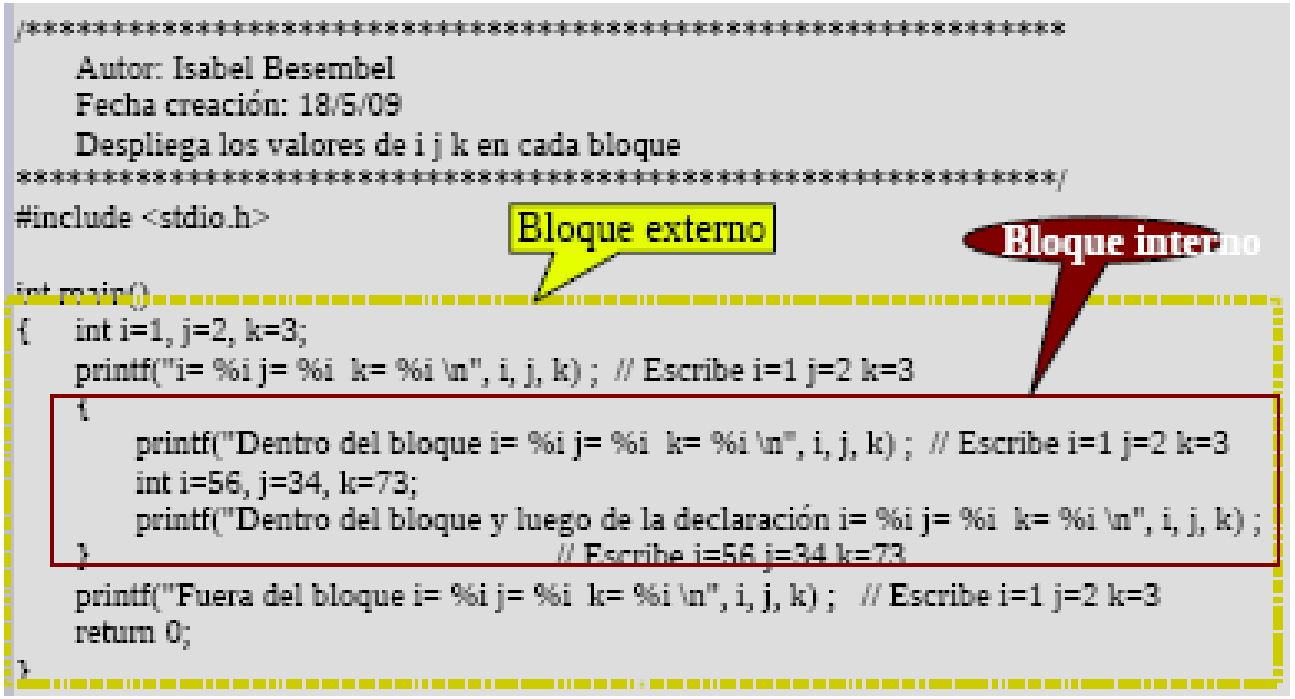
UNIVERSIDAD DE LOS ANDES

Tipos de variables

- Locales
  - Aquella que está declarada dentro de un bloque delimitado por { }
  - Sólo se puede usar dentro del bloque en el que ha sido declarada
- Globales
  - Aquella que está declarada para todo el programa, es decir, fuera de cualquier bloque o función
  - Retiene su valor durante la ejecución de todo el programa



Variables locales




```
/*****************************************************************
      Autor: Isabel Besembel
      Fecha creación: 18/5/09
      Despliega los valores de i j k en cada bloque
*****************************************************************/
#include <stdio.h>

int main()
{     int i=1, j=2, k=3;
      printf("i= %i j= %i  k= %i \n", i, j, k) ;  // Escribe i=1 j=2 k=3
      {
            printf("Dentro del bloque i= %i j= %i  k= %i \n", i, j, k) ;  // Escribe i=1 j=2 k=3
            int i=56, j=34, k=73;
            printf("Dentro del bloque y luego de la declaración i= %i j= %i  k= %i \n", i, j, k) ;
      }                                    // Escribe i=56 j=34 k=73
      printf("Fuera del bloque i= %i j= %i  k= %i \n", i, j, k) ;   // Escribe i=1 j=2 k=3
      return 0;
}
```

UNIVERSIDAD DE LOS ANDES

Constantes

- Valor que no cambia durante la ejecución de un programa
- Puede ser numérica, lógica, caracter, etc.
- **Constantes globales**

  #define IDENTIFICADOR valor

- Asigna el valor al identificador
- Un proceso previo a la compilación sustituirá el identificador por el valor en cualquier parte del programa donde aparezca el identificador
- Ejemplos

```
#define     PI      3.14159
#define     MAXIMO      256
```

UNIVERSIDAD DE LOS ANDES

Constantes globales

```
/*****************************************************************
    Autor: Isabel Besembel
    Fecha creación: 18/5/09
    Uso de constantes globales
*****************************************************************/
#include <stdio.h>
#define LONGITUD 32

int resul;        //  Declaración de variable global
int main()
{   int x=LONGITUD;        //  Declaración de variables locales
    resul=x+2;
    printf("x= %i resul= %i \n", x, resul) ;
    return 0;
}
```

Declaración de constante global

UNIVERSIDAD DE LOS ANDES

Constantes locales

- Definición

  const tipoDeDato IDENTIFICADOR = valor;

- Sólo pueden usarse dentro del bloque donde fue declarada

```
int      main()
{  Declaración de constantes locales          // opcional
   Declaración de variables locales           // opcional
   Conjunto de sentencias        // Cuerpo de la función
}
```

- Ejemplo

```
int main()
{  const     int        LONGITUD = 32;
   int       lon = LONGITUD;
   ……                  // Cuerpo de la función
}
```

UNIVERSIDAD DE LOS ANDES

Estructuras secuenciales

- Se ejecutan en secuencia sin posibilidad que la sentencia siguiente a ejecutar pueda ser otra diferente a la que sigue en la secuencia

Tipo de sentencia	Pseudocódigo en inglés	Pseudocódigo en español	Código en C, C++
Comienzo de proceso	begin	inicio	{
Fin de proceso	end	fin	}
Entrada (lectura)	read	leer	scanf, cin
Salida (escritura)	write	escribir	printf, cout
Asignación	A← 5 o A = 5	A← 5 o A = 5	A = 5

UNIVERSIDAD DE LOS ANDES

Variables globales

```
/*****************************************************************
     Autor: Isabel Besembel
     Fecha creación: 18/5/09
     Calcula el producto de tres números enteros
*****************************************************************/
#include <stdio.h>

int resul;          // Declaración de variable global
int main()
{    int x, y, z;   //  Declaración de variables locales
     printf("Introduzca los 3 valores enteros, cada uno menor que 1000 \n") ;
     scanf("%i %i %i",&x, &y, &z);
     resul=x*y*z;
     printf("Resultado de multiplicar %i x %i x %i = %i \n", x, y, z, resul) ;
     return 0;
}
```

Sentencia de entrada (lectura)

- Permite leer valores (datos de entrada) como: 10, -4.6 o "no", y asignarlos a las variables asociadas como: p, r, res
- Los datos de entrada se introducen mediante los dispositivos de entrada (teclados, ratón, etc.)

Notación algorítmica

Leer (listaDeVariablesDeEntrada)
listaDeVariablesDeEntrada = valor suministrado
Leer (p, r, res)
p, r, res = valor suministrado

Notación en C <stdio.h>

scanf ("%TipoDato1 … %tN", &var1, .., &varN);
scanf ("%i %f %d ", &p, &r, &res);

Notación C++ <iostream>

cin >> var1 >> var2 >> … >> varN;
cin >> p >> r >> res;

Sentencia de salida (escritura)

- Permite escribir los valores resultados de un programa, como: 8 ó 6.89, almacenados en las variables asociadas como: p o r
- La salida aparece en algún dispositivo de salida (monitor, impresora, etc.)

Notación algorítmica

Escribir (mensajesY/OlistaDeVariablesDeSalida)
Desplegar mensajesY/OlistaDeVariablesDeSalida
Escribir ("dias=", d, " porcentaje=", p)
Desplegar "dias=", d, "porcentaje=", p

Notación en C
<stdio.h>

printf ("mensaje %TipoDato", var1, .., varN);
printf ("dias = %i porcentaje = %f", d, p);

Notación C++
<iostream>

cout << "mensaje" << var1 << … << varN;
cout << "dias=" << d << " porcentaje=" << p << endl;

UNIVERSIDAD DE LOS ANDES

Asignación en C/C++

- Es un modo de darle valor a una variable

variable = expresión

- La expresión se evalúa y el resultado obtenido se almacena en la variable

variable = constante; var = 3;

variable = variable; var = otraVar;

variable = expresión; var = x * y;

¿Cuál es el valor final de las variables siguientes?

A = 4;

B = 5 * A;

D = (C – A * B)/ 2;

UNIVERSIDAD DE LOS ANDES

Expresiones

- Pueden ser:
  - Una constante
  - Una variable
  - Una combinación de operandos y operadores
- Operandos: constantes, variables u otras expresiones
- Operadores: símbolo que indica al compilador la operación con los operandos
- Tipos:
  - Aritméticas: los operandos que intervienen son numéricos, el resultado es numérico y los operadores son aritméticos (+, -, *, /, etc.)
  - Lógicas: su resultado es Verdadero o Falso
  - Mixtas: mezcla de las anteriores

UNIVERSIDAD DE LOS ANDES

Expresiones aritméticas

- Aritmética entera en C/C++
  - Los operandos son del tipo de dato Entero (int, long, unsigned int, etc.)
  - Los operadores son los del tipo de dato Entero (+, -, *, /, %, --, ++)
    - Operación incremento entero
      - Para n=2, la ejecución de la sentencia

        I = n++; // post-incremento

        Asigna el valor 2 a I y después incrementa el valor de n a 3, los valores finales son:

        I = 2 n = 3
      - Para n=2, la ejecución de la sentencia

        I = ++n; // pre-incremento

        incrementa el valor de n a 4 y después asigna este valor a I, los valores finales son:

        n = 4 I = 4
  - Ejemplos

    X = 5 + Y++; equivalente a X = 5 + Y;
    Y++;

UNIVERSIDAD DE LOS ANDES

Expresiones aritméticas

- Aritmética entera en C/C++
  - Operación decremento entero
    - Para n=5, la ejecución de la sentencia

      I = n--; // post-decremento

      Asigna el valor 5 a I y después decrementa el valor de n a 4, los valores finales son:

      I = 5 n = 4
    - Para n=5, la ejecución de la sentencia

      I = --n; // pre-decremento

      decrementa el valor de n a 4 y después asigna este valor a I, los valores finales son:

      n = 4 I = 4
  - Ejemplos

    X = 8 + Y--; equivalente a X = 8 + Y;
    Y--;

UNIVERSIDAD DE LOS ANDES

Expresiones aritméticas

- Aritmética entera en C/C++
  - División entera arroja un resultado entero

    17 / 5 = 3
  - Operador módulo (%) o cálculo del resto de la división entera
    - Sólo puede utilizarse con operandos enteros

    17 % 5 = 2
- Los operandos de una expresión aritmética en C/C++ satisfacen lo siguiente:

1. Si ambos operandos son de tipo Entero entonces el resultado es Entero
2. Si uno o ambos operandos son de tipo Real entonces el resultado es de tipo Real

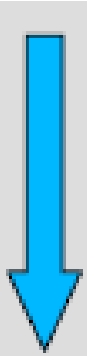
UNIVERSIDAD DE LOS ANDES

Expresiones aritméticas

- Los operandos de una expresión aritmética en C/C++ satisfacen lo siguiente:

3. Si la expresión evaluada da como resultado un **valor de tipo Entero** y la **variable** a la izquierda de la asignación es de tipo **Real**, entonces el resultado será convertido automáticamente al tipo **Real** antes de realizar la asignación
4. Si la expresión evaluada da como resultado un **valor de tipo Real** y la **variable** a la izquierda de la asignación es de tipo **Entero**, entonces el resultado será convertido automáticamente al tipo **Entero** antes de realizar la asignación

Precedencia de los operadores en C/C++



( ) Subexpresiones encerradas entre paréntesis se evalúan primero (mayor nivel de precedencia)

++ --

* / %

+ - (menor nivel de precedencia)

Regla 1: La evaluación de las operadores con la misma prioridad se realiza de izquierda a derecha

Evaluar la siguiente expresión aritmética:

(a + b + c + d + e) % 5

R1 = a + b

R2 = R1 + c

R3 = R2 + d

R4 = R3 + e

R5 = R4%5

Regla 2: Se toman los dos primeros operadores, si son de igual jerarquía, se realiza el más a la izquierda

Evaluar la expresión aritmética

(a + b + c + d + e)%5

para los valores:

a = 3, b = 2, c = 1, d = 8, e = 4

R1 = a + b = 3 + 2 = 5

R2 = R1 + c = 5 + 1 = 6

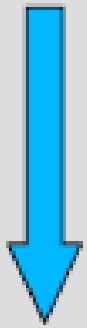
R3 = R2 + d = 6 + 8 = 14

R4 = R3 + e = 14 + 4 = 18

R5 = R4 % 5 = 18%5 = 3

Precedencia de los operadores en C/C++

UNIVERSIDAD DE LOS ANDES



( ) Subexpresiones encerradas entre paréntesis se evalúan primero (mayor nivel de precedencia)

++ --

* / %

+ - (menor nivel de precedencia)

Regla 1: La evaluación de las operadores con la misma prioridad se realiza de izquierda a derecha

Regla 2: Se toman los dos primeros operadores, si son de igual jerarquía, se realiza el más a la izquierda

Evaluar la siguiente expresión aritmética:

a % 2 / b – c * a * y + 18 / c – 3 * h

R1 = a % 2
R2 = R1 / b
R3 = c * a
R4 = R3 * y
R5 = R2 – R4
R6 = 18 / c
R7 = R5 + R6
R8 = 3 * h
R9 = R7 – R8

Regla 3: Se toman los dos primeros operadores, si son de diferente jerarquía, se mira el tercer operador y se realiza el de mayor jerarquía que esté más a la izquierda

Evaluar la expresión aritmética

a % 2 / b – c * a * y + 18 / c – 3 * h

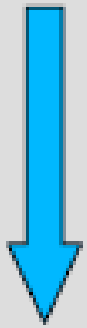
para los valores:

a = 3, b = 2, c = 1, h = 8, y = 4

R1 = a % 2 = 3 % 2 = 1
R2 = R1 / b = 1 / 2 = 0
R3 = c * a = 1 * 3 = 3
R4 = R3 * y = 3 * 4 = 12
R5 = R2 – R4 = 0 – 12 = -12
R6 = 18 / c = 18 / 1 = 18
R7 = R5 + R6 = -12 + 18 = 6
R8 = 3 * h = 3 * 8 = 24
R9 = R7 – R8 = 6 – 24 = -18

Precedencia de los operadores en C/C++

UNIVERSIDAD DE LOS ANDES



( ) Subexpresiones encerradas entre paréntesis se evalúan primero (mayor nivel de precedencia)

++ --

* / %

+ - (menor nivel de precedencia)

Regla 1: La evaluación de las operadores con la misma prioridad se realiza de izquierda a derecha

Regla 2: Se toman los dos primeros operadores, si son de igual jerarquía, se realiza el más a la izquierda

Regla 3: Se toman los dos primeros operadores, si son de diferente jerarquía, se mira el tercer operador y se realiza el de mayor jerarquía que esté más a la izquierda

Evaluar la siguiente expresión aritmética:

a * (b + c) + c * (d + e)

R1 = b + c
R2 = a * R1
R3 = d + e
R4 = c * R3
R5 = R2 + R4

Regla 4: Los paréntesis se resuelven completamente antes de realizar cualquier operación fuera de ellos, siguiendo las reglas dadas

Evaluar la siguiente expresión aritmética

a * (b + c) + c * (d + e)

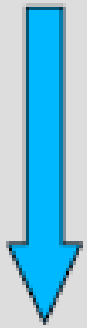
para los valores:

a = 3, b = 2, c = 1, d = 8, e = 4

R1 = b + c = 2 + 1 = 3
R2 = a * R1 = 3 * 3 = 9
R3 = d + e = 8 + 4 = 12
R4 = c * R3 = 1 * 12 = 12
R5 = R2 + R4 = 9 + 12 = 21

Precedencia de los operadores en C/C++

UNIVERSIDAD DE LOS ANDES



( ) Subexpresiones encerradas entre paréntesis se evalúan primero (mayor nivel de precedencia)

++ --

* / %

+ - (menor nivel de precedencia)

Regla 1: La evaluación de las operadores con la misma prioridad se realiza de izquierda a derecha

Regla 2: Se toman los dos primeros operadores, si son de igual jerarquía, se realiza el más a la izquierda

Regla 3: Se toman los dos primeros operadores, si son de diferente jerarquía, se mira el tercer operador y se realiza el de mayor jerarquía que esté más a la izquierda

Regla 4: Los paréntesis se resuelven completamente antes de realizar cualquier operación fuera de ellos, siguiendo las reglas dadas

Evaluar la siguiente expresión aritmética:

$(a * (b + c)) - 2 * a + (4 * d - f)$

R1 = b + c
R2 = a * R1
R3 = 2 * a
R4 = R2 – R3
R5 = 4 * d
R6 = R5 - f
R7 = R4 + R6

Regla 5: El paréntesis más interno se resuelve primero, siguiendo las reglas dadas

Evaluar la expresión aritmética

$(a * (b + c)) - 2 * a + (4 * d - f)$

para los valores:

$a = 3, b = 2, c = 1, d = 8, f = 4$

$R1 = b + c = 2 + 1 = 3$
$R2 = a * R1 = 3 * 3 = 9$
$R3 = 2 * a = 2 * 3 = 6$
$R4 = R2 - R3 = 9 - 6 = 3$
$R5 = 4 * d = 4 * 8 = 32$
$R6 = R5 - f = 32 - 4 = 28$
$R7 = R4 + R6 = 3 + 28 = 31$

- ¿Qué son tipos de datos y tipos abstractos de datos?
- ¿Qué son variables y constantes en C/C++?
- ¿Cuáles son los tipos de variables y constantes en C/C++?
- ¿Cuáles son los operadores de tipo Entero y de tipo Real en C/C++?
- ¿Cómo es la declaración de variables y constantes en C/C++?
- ¿Cómo son las sentencias de entrada y salida en C/C++?
- ¿Cómo son las expresiones aritméticas y cómo se evalúan en C/C++?
- ¿Cómo hacer un programa de cálculo sencillo?

Resumen

¿Cuáles son los conceptos relevantes de esta clase?

UNIVERSIDAD DE LOS ANDES

Ejercicios

1. Evaluar cada una de las siguientes expresiones aritméticas:

a) A % B + C / D – 6

b) A + 2 * (3 + B)

c) 3 * (A % (B / C)) + 5

d) 6 * 5 / 10 * 2 + 10

e) (6 * 5)/ (10 * 2) + 10

f) (6 * 5) /(10 * 2 + 10)

g) (6 * 5) / (10 * (2 + 10))

h) A * B / C * D

UNIVERSIDAD DE LOS ANDES

Ejercicios

2. Evaluar la expresión

4 / 2 * 3 / 6 + 6 / 2 / 1 / 5 % 2 / 4 * 2

3. Escribir las siguientes expresiones algebraicas como expresiones en C++

a) $4x - 2y + 7$

b) $\frac{a + b}{c - d}$

c) $\frac{3x + 2y}{2z}$

d) $\frac{x + y}{x} - \frac{3x}{5}$

e) $\frac{y2 - y1}{x2 - x1}$

UNIVERSIDAD DE LOS ANDES

Ejercicios

4. Dadas las declaraciones:

```
float w;
int i, j = 2, k = 4;
```

Determinar el valor de las variables i, w y k después de la ejecución de las siguientes sentencias de asignación:

```
i = j / k;
w = i / j;
k = i % j;
w = 8.0 / j;
```

5. La fuerza de atracción entre dos cuerpos es igual al producto de una constante k por el cociente que resulta de dividir el producto de las masas de los cuerpos por el cuadrado de su distancia. Realice el análisis E-P-S, diseño en TDSO y codificación en C++ para resolver este problema y realice la corrida en frío para los valores de k = 0,1, masa del cuerpo 1 = 3 kg., masa del cuerpo 2 = 5 kg., y la distancia entre los dos cuerpos = 2,5 m

UNIVERSIDAD DE LOS ANDES

Ejercicios

6. Realice el análisis E-P-S, diseño en TDSO y codificación en C/C++ para cada uno de los enunciados dados a continuación:
   1. Calcular el número total de hojas que tiene un árbol, si por ejemplo, tiene doscientos ochenta y cuatro ramas y sabiendo que cada rama tiene como promedio trescientas cuarenta y siete hojas
   2. Un joyero vino de Siria para vender joyas en Bagdad. Prometió que pagaría por el hospedaje 20 dinares si el dueño de la hostería vendía todas las joyas por 100 dinares; y 35 dinares si las vendía por 200 dinares. Al cabo de varios días, tras andar de aquí para allá, acabó vendiéndolas todas por 140 dinares. ¿Cuánto debe pagar el joyero de acuerdo con el trato de hospedaje?
   3. Calcular el promedio de cinco notas
   4. Multiplicar tres números enteros x, y, z

UNIVERSIDAD DE LOS ANDES

Ejercicios

5. Dado el peso de una persona en libras, calcular su peso en kilogramos y gramos
6. Leer cuatro números reales. Calcular y escribir su producto, suma y su media aritmética
7. Leer el radio de un círculo y calcular e imprimir su superficie y la longitud de la circunferencia
8. Si un cuerpo pesa a kp (kilopondios) en un lugar en el que la gravedad es g = 9.8 m/s2, calcular su masa
9. Una fuerza actúa sobre un cuerpo de n kg de masa, pasando la velocidad de éste de v0 a vf m/s en t segundos. Calcular la fuerza.
10. Calcular el perímetro de un cuadrado de lado igual a 5.
11. Calcular el perímetro de un triángulo equilatero de lado igual a 8
12. Calcular la circunferencia de un círculo de radio dado
13. Calcular el área de un rectángulo de ancho y largo dados