ST0244-032
Clase 15

J.F. Cardona

Universidad EAFIT

9 de septiembre de 2015

Agenda

1. Capítulo 3. Programación orientada a objeto con C++
   - Tipos de datos en C++
     - Operaciones de los tipos básicos
   - Declaración
   - Apuntadores y referencias

Programación orientada a objetos con C++

Tipos de datos en C++ - Tipos fundamentales

Void

- Es un tipo básico que no tiene valores.
- Es decir que no tiene literales.
- Ejemplos:

```
1 void a;  // No hay objetos tipo void
  void& ref;
3 void funcion(); // Un procedimiento
  void* pv;       // Puntero a un tipo void
```

Programación orientada a objetos con C++

Tipos de datos en C++ - Tipos fundamentales - Operaciones

- Los operadores aritméticos puede ser utilizados con los tipos de datos fundamentales númericos (para enteros).

```
a + b // suma
+a    // operador unario mas
a - b // menos
-a    // operador unario menos
a * b // multiplicacion
a / b // division
a %b // modulo
```

- También se puede utilizar operadores de comparación.

```
a == b  // igual
a != b  // no igual
a <  b  // menos que
a >  b  // mayor que
a <= b  // menor o igual que
a >= b  // mayor o igual que
```

¡

Programación orientada a objetos con C++

Tipos de datos en C++ - Tipos fundamentales

- También existen operadores lógicos:

```
a & b    // operador y a nivel de bits
a | b    // operador o a nivel de bits
a ^ b    // operador o-exclusivo a nivel de bits
~a       // operador de complemento a bits
a && b   // operador y logico
a and b  // operador y logico
a || b   // operador o logico
a or b   // operador o logico
! a      // negacion logico
```

Programación orientada a objetos con C++

Declaración

- Antes que un nombre pueda ser utilizado en C++, este debe ser declarado.
- Esto, especificar su tipo para informa al compilador que clase de entidad el nombre se refiere.

```
1 char c;
  string cadena;
3 int contador = 1;
  const double pi = 3.141592656;
5 extern int error_number;

7 char nombre[] = "Juan";
  const char *temporada[] = { "vacaciones", "estudio" };
9
  struct Fecha { int d, m, a; };
```

- Alguna de las anteriores declaraciones son una definición, esto es que ellas también definen una entidad para el nombre las cuales

Declaración - Estructura de una declaración

- Una declaración consiste de cuatro partes:
  - Un especificador opcional,
  - Un tipo base,
  - Un declarador,
  - Inicializador opcional.
- Por ejemplo

```
const char *jugadores[] = { "James", "Falcao", "Jackson" };
register int value { 100.0 }; // C++11
auto det = 123.34;
```

Programación orientada a objetos en C++

Declaración - Estructura declaración

- Un especificador es una palabra reservada como `virtual` `extern`, que especifica un atributo no relacionado con el tipo que esta siendo declarado.
- Un declarador está compuesto de un nombre y opcionalmente algunos operadores de declaración:

*	apuntador	prefijo
*const	apuntador constante	prefijo
&	referencia	prefijo
[]	arreglo	postijo
()	función	postifjo

- Estos se pueden combinar varias veces.

```
1 int *a; // apuntador a entero
  int **b; // apuntador a apuntador de entero
3 double *arreglo[10]; // arreglo de enteros
  void (*f)(void); // un apuntador a procedimiento
5 int *(*f)(int *); // un apuntador a funcion
```

Inicialización

- Son varias las formas de inicialización:

```
1 double vald1 = 4.2; // Inicializa vald1 a 4.2
  double vald2 {4.2}; // Inicializa vald2 a 4.2
```

- Los apuntadores permiten obtener la dirección de un valor de memoria.
- Su manejo se hace a través de varias primero a través de tipos de datos apuntadores.
- Operador para obtener la dirección.
- Operador para obtener el valor de dirección.

- Los apuntadores son difíciles de manejar, por ello se han establecidos las referencias como un mecanismo para establecer una alias a una dirección.
- Una referencia se establece a través de una declaración y esta debe ser inicializada.

```
int a = 10;
int &ra = a;
```

- Una referencia siempre debe ser inicializada.