CLASES DE DATOS

Ing. Edgar Ruiz Lizama

RESUMEN

El artículo presenta un enfoque moderno acerca del estudio de los tipos de datos soportados por un computador. Para el lenguaje C++, los datos se agrupan en *clases* de datos las cuales se estudian y se prueban con programas ejemplo.

ABSTRACT

The article presents a modern focus about the study of the types of data supported by a computer. For the language C++, the data group in classes of data which are studied and they are proven with programs example

INTRODUCCION

Al igual que el C; C++ posee tipos de datos fundamentales o primitivos y tipos de datos derivados o estructurados. Un enfoque más moderno y adecuado para los tipos de datos en C++ es conceptualizarlos como correspondientes a *clases de datos;* siendo éstas, agrupadas en: *clase escalar, clase estructurada y clase apuntador o puntero*. Ver figuras 1, 2, 3, y 4.

"... C++ posee tipos de datos fundamentales o primitivos y tipos de datos derivados o estructurados."

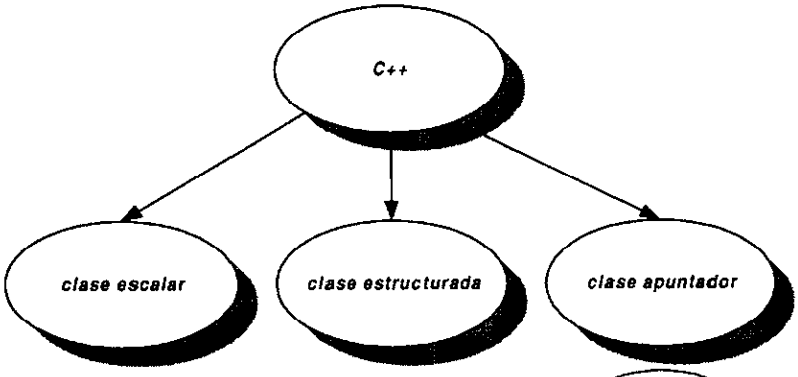

Figura 1: Las clases de datos en C++.

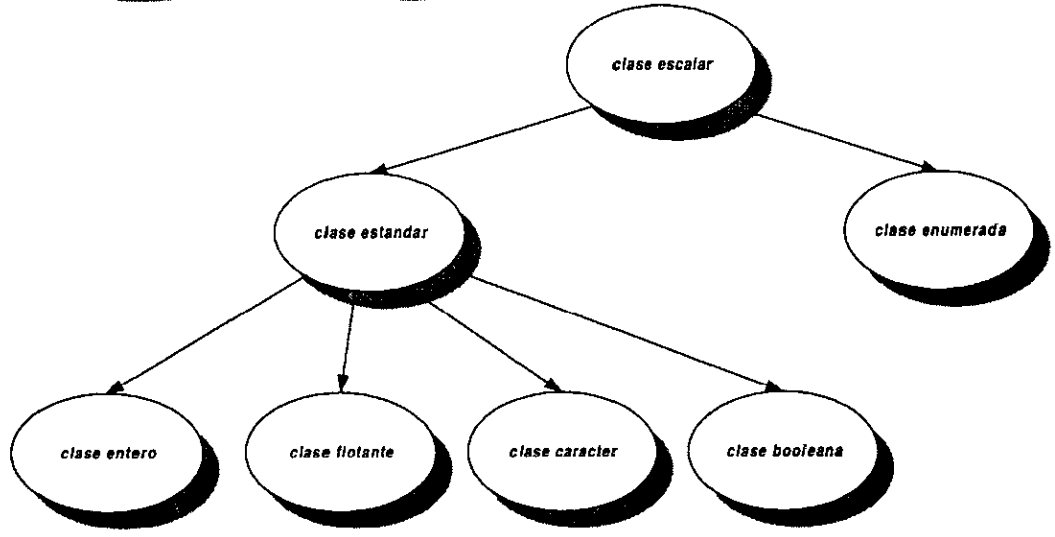

Figura 2: La clase escalar

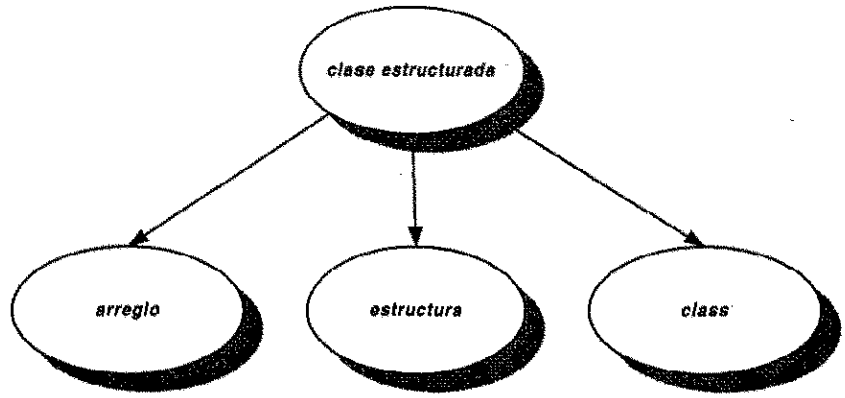

Figura 3:
la clase
estructurada

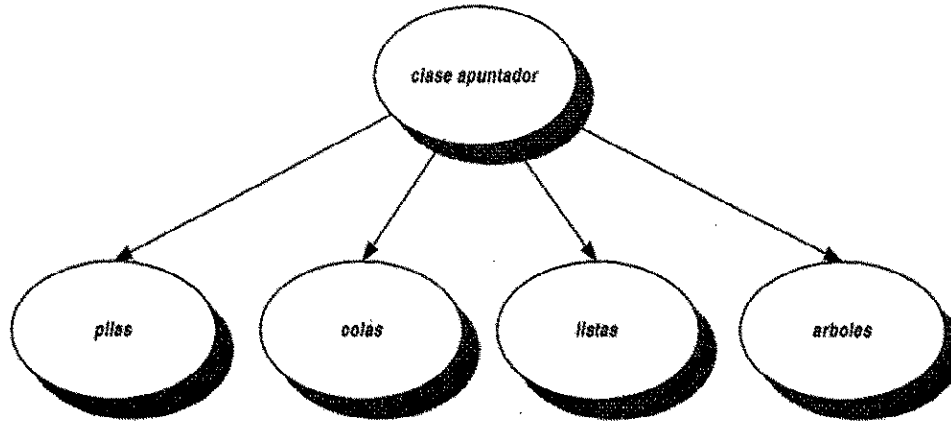

Figura 4:
La clase
apuntador

"... en el mundo real de las computadoras ..., los enteros negativos y positivos tienen un valor definido e incluyen también el cero..."

LA CLASE ESCALAR

La clase escalar como acabamos de ver se divide en clase estándar y clase enumerada. La clase estándar a su vez se subdivide en cuatro clases: *entero, flotante, caracter y booleana.*

CLASE ENTERO

En matemáticas los enteros son positivos y negativos cuya extensión es inconmensurable; es decir, desde -∞ a +∞.

Sin embargo, en el mundo real de las computadoras y para fines prácticos de implementación, los enteros negativos y positivos tienen un valor definido e incluyen también el cero. Para la mayoría de los compiladores C++; el rango de los enteros estándar va desde -32768 a 32767.

Debido a que cada valor que se representa en la computadora ocupa una determinada cantidad en bytes (1 byte = 8 bit) y por cuestiones de eficiencia en el procesamiento; C++ posee clases de enteros; tales como: short int, int, unsigned int, long int y unsigned long; tal como se muestra en la tabla 1.

Clase entero	Tamaño en bytes	Rango de valores
short int o short	*1*	*-128 a 127*
int	*2*	*32768 a +32767*
unsigned int	*2*	*0 a 6555*
long int o long	*4*	*-2147483648 a +2147483647*
unsigned long	*4*	*0 a 4294967295*

Tabla 1: La clase entero

Ejemplo 1: el programa bytes1.cpp realiza la comprobación del tamaño en bytes para la clase entero. (1)

```
#include <iostream.h>

int main() // bytes1.cpp
{
 cout<<"Un short int tiene "<<sizeof(short int)<<" bytes\n";
 cout<<"Un int tiene "<<sizeof(int)<<" bytes\n";
 cout<<"Un unsigned int tiene "<<sizeof(unsigned int)<<" bytes\n";
 cout<<"Un long int o long tiene "<<sizeof(long int)<<" bytes\n";
 cout<<"Un unsigned long tiene "<<sizeof(unsigned long)<<" bytes\n";

 return 0;
```

La salida para bytes1.cpp es:

```
Un short int tiene 2 bytes
Un int tiene 2 bytes
Un unsigned int tiene 2 bytes
Un long int o long tiene 4 bytes
Un unsigned long tiene 4 bytes
```

Ejemplo 2: El programa rangos1.cpp muestra el rango de valores para la clase entero.

```
#include <iostream.h>
#include <limits.h>

void main() // rangos.cpp
{
 cout<<"Entero sin signo maximo "<<UINT_MAX;
 cout<<"\nEntero largo sin signo maximo "<<LONG_MAX;
 cout<<"\nEntero corto sin signo maximo "<<SHRT_MAX;
 cout<<"\nEntero largo sin signo maximo "<<LONG_MAX;
 cout<<"\nEntero largo con signo minimo "<<LONG_MIN;
 cout<<"\nEntero con signo maximo "<<INT_MAX;
 cout<<"\nEntero con signo minimo "<<INT_MIN;
 cout<<"\nEntero corto con signo minimo "<<SHRT_MIN;
}
```

La salida para rangos1.cpp

```
Entero sin signo maximo 65565
Entero largo sin signo maximo 2147483647
Entero corto sin signo maximo 32767
Entero largo sin signo maximo 2147483647
Entero largo con signo minimo -2147483648
Entero con signo maximo 32767
Entero con signo minimo -32768
Entero corto con signo minimo -32768
```

CLASE FLOTANTE

En términos matemáticos el conjunto de los números reales incluye a todos aquellos valores cuya representación necesita el punto decimal y que van desde $-\infty$ hasta $+\infty$. En notación sintáctica BNF (Backus-Naur Form) un número real se representa como:

<numero-real> ::= (+|-) <entero>. <entero>

El cual se lee como: *un número real es un signo + ó un signo - seguido de un entero, seguido de un punto, seguido de un entero*; donde *entero* puede ser un *dígito decimal* o una *secuencia de dígitos*.

La clase flotante en C++, incluye a todos los enteros y a todo aquel valor que para su representación necesita del punto decimal; tal como por ejemplo los números: 1, 15.24, -4.05, 3.2728.

La tabla 2; muestra los tipos soportados por la clase flotante: float, double y long double.

Clase flotante	Tamaño en bytes	Rango de valores
float	4	$3.4x10^{-38}$ a $3.4x10^{38}$
double	8	$1.7x10^{-308}$ a $1.7x10^{308}$
long double	10	$3.4x10^{-4932}$ a $1.7x10^{4932}$

Tabla 2: La clase de flotante

(1) Si bien es cierto el tipo short y short int tiene un byte, esto se reporta en los compiladores Turbo de C/C++ que corren en DOS, más no así en un compilador Borland C++ 4.5 bajo Windows.

Ejemplo 3: el programa bytes2.cpp comprueba el tamaño de bytes de los miembros de la clase flotante.

```
#include <iostream.h>

int main() // bytes2.cpp
{
  cout<<"Un float tiene "<<sizeof(float)<<" bytes\n";
  cout<<"Un double tiene "<<sizeof(double)<<" bytes\n";
  cout<<"Un long double tiene "<<sizeof(long double)<<" bytes\n";

  return 0;
}
```

La salida para bytes2.cpp es:

```
Un float tiene 4 bytes
Un double tiene 8 bytes
Un long double tiene 10 bytes
```

CLASE CARACTER

Los símbolos representados en una computadora son caracteres. El teclado de su computadora muestra el conjunto de caracteres representables en ella. Las letras, dígitos, signos de puntuación y símbolos especiales son caracteres, este conjunto de caracteres se representan conforme el código ASCII (American Standard Code for Information Interchange).

El ASCII es un conjunto que posee 256 códigos (0 al 255); donde cada código es un byte y a cada uno le corresponde un número, por ello se dice que los caracteres ASCII son ordinales.

La tabla 3, muestra los elementos de la clase caracter en C++: char y unsigned char.

" La clase flotante en C++, incluye a todos los enteros y a todo aquel valor que para su representación necesita del punto decimal"

Clase caracter	Tamaño en bytes	Rango de valores
char	1	-128 a 127
unsigned char	1	0 a 255

Tabla 3: La clase caracter

Ejemplo 4: El programa bytes3.cpp muestra el tamaño en bytes para la clase caracter.

```
#include <iostream.h>

int main() // bytes3.cpp
{
  cout<<"Un char tiene "<<sizeof(char)<<" bytes\n";
  cout<<"Un unsigned char "<<sizeof(unsigned char)<<" bytes\n";

  return 0;
}
```

La salida para bytes3.cpp es:

```
Un char tiene 1 bytes
Un unsigned char tiene 1 bytes
```

Ejemplo 5: el programa carascii.cpp muestra los caracteres alfabéticos tanto en mayúsculas como en minúsculas y sus correspondientes valores ASCII.

```
#include <iostream.h>
#include <ctype.h> // para toascii()
void main()  // carascii.cpp
{
  char c;
  cout<<"\nCaracteres alfabeticos en Mayusculas"<<endl;
  for (c='A'; c<='Z'; c++)
        cout<<c<<" = "<< toascii(c)<<"\t";

  cout<<"\nCaracteres alfabeticos en Minusculas"<<endl;
  for (c='a'; c<='z'; c++)
        cout<<c<<" = "<< toascii(c)<<"\t";
}
```

La salida para carascii.cpp es:

```
Caracteres alfabeticos en Mayusculas
A = 65  B = 66  C = 67  D = 68  E =69
F = 70 G = 71  H = 72  I = 73   J =74
K = 75 L = 76 M = 77  N =78  O =79
P = 80 Q = 81  R = 82  S = 83  T = 84
U = 85 V =86  W = 87  X = 88  Y = 89
Z = 90

Caracteres alfabeticos en Minusculas
a =97 b = 98  c =99 d =100  e =101
f=102  g=103   h=104  i=105  j=106
k=107 l=108 m=109  n=110  o=111
p=112 q=113  r=114 s=115  t=116
u=117 v=118 w=119  x=120  y=121
z=122
```

CLASE BOOLEANA

Todo objeto booleano, posee sólo uno de dos valores; TRUE (verdad) y FALSE (falso). Estos tienen gran importancia en todo programa que utiliza toma de decisiones; pues una decisión implica evaluar una condición y dependiendo del resultado (verdad o falso) se toma una acción o camino a seguir. La clase *bool* de C++, define false y true de tal forma que false siempre es menor que true. En general para todo programa C/C++, FALSE=0 y TRUE=1.

La clase *bool* es soportada por compiladores modernos de C++ de acuerdo con el estándar ANSI C++. Si su compilador C++ no la posee, usted puede implementarla mediante la declaración de objetos enumerados tal como se verá enseguida.

CLASE ENUMERADA

La clase *enum* viene ya desde C; la cual permite al usuario declarar y usar tipos enumerados en sus programas. La declaración de datos enumerados se realiza con la palabra reservada enum, siendo su formato el siguiente,

```
enum <identificador clase>{ valor1, valor2, ... , valorN);
<identificador clase> <identificador objeto>;
```

Los siguientes son ejemplos de declaración válidos de objetos de la clase *enum*.

```
enum Semana { Dom, Lun, Mar, Mie, Jue, Vie, Sab};
Semana Dia;

enum Paradigma { Estructurado, OO, Funcional, Logico};
Paradigma Lenguaje;
```

Ejemplo 6: El programa boolea1.cpp muestra los valores de los objetos enumerados FALSE y TRUE.

```
#include <iostream.h>

enum boolea {FALSE, TRUE};

main()   // boolea1.cpp
{
 cout<<"\nValor de FALSE "<<FALSE;
 cout<<"\nValor de TRUE  "<<TRUE;

 return 0;
}
```

```
La salida para boolea1.cpp es
Valor de FALSE  0
Valor de TRUE   1
```

Los objetos de la clase *enum* son ordinales y el compilador C/C++, asigna por defecto el valor 0 al primer objeto, 1 al segundo objeto y así sucesivamente. Por ello es que a los objetos de esta clase se les puede aplicar operaciones relacionales tal como se ve en los ejemplos 7 y 8.

Ejemplo 7: el programa boolea2.cpp es una variante de boolea1.cpp

```
#include <iostream.h>

enum boolea {FALSE, TRUE};

main()   // boolea2.cpp
{
        int a=10, b=5;
        if (a>b)
                cout<<"\nEs Verdad "<<TRUE;
        else
                cout<<"\nEs Falso "<<FALSE;
        return 0;
}
```

```
La salida para boolea2.cpp es
Es Verdad 1
```

Ejemplo 8: El programa semana1.cpp muestra cómo los objetos de la clase enumerada Semana son ordinales y para probarlo utiliza el objeto Dia.

```
#include <iostream.h>

enum Semana {Dom=1, Lun, Mar, Mie, Juev, Vier, Sab};
Semana Dia;

int main() // semana1.cpp
{
 for (Dia=Dom; Dia<=Sab; Dia++)
 { switch(Dia) {
   case 1 : cout<<"\nEl Dia "<<Dia<<" es Domingo";
            break;
   case 2 : cout<<"\nEl Dia "<<Dia<<" es Lunes";
            break;
   case 3 : cout<<"\nEl Dia "<<Dia<<" es Martes";
            break;
   case 4 : cout<<"\nEl Dia "<<Dia<<" es Miercoles";
            break;
    case 5 : cout<<"\nEl Dia "<<Dia<<" es Jueves";
            break;
   case 6 : cout<<"\nEl Dia "<<Dia<<" es Viernes";
            break;
   case 7 : cout<<"\nEl Dia "<<Dia<<" es Sabado";
            break;
            }
 }
 return 0;
}
```

```
La salida para semana1.cpp es
El Dia 1 es Domingo
El Dia 2 es Lunes
El Dia 3 es Martes
El Dia 4 es Miercoles
El Dia 5 es Jueves
El Dia 6 es Viernes
El Dia 7 es Sabado
```

Las clases estructurada y la clase puntero se presentarán en un próximo artículo.

" La clase *enum* ... permite al usuario declarar y usar tipos enumerados en sus programas"

REFERENCIAS

1. Deitel H.M. y Deitel P.J. *C++ How to program*. E.U. A. Prentice-Hall Inc., 1130 p. (1998).
2. Ruiz Lizama, Edgar. *Curso de Lenguaje C* Lima, UNMSM Facultad de Ingeniería Industrial 150 p. (1999).
3. Staguaard, Andrew. *Técnicas estructuradas y orientadas a objetos: una introducción utilizando C++*. México. Prentice-Hall Hispanoamericana, 770 p. (1998).
4. Stroustrup, Bjarne. *El Lenguaje de Programación C++*. U.S.A. Addison Wesley Iberoamericana. 710 p. (1993).

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

FACULTAD DE INGENIERIA INDUSTRIAL

ALTA CALIDAD AL ALCANCE DE TODOS

AV. VENEZUELA S/N CDRA 34. LIMA

Teléfono:511-451-5135 ó 4644053
Fax: 511-452-0349
Correo: decanoii@unmsm.edu.pe

URL: http://www.unmsm.edu.pe/industrial