Clase #06 de 27
Funciones Escalares & Comparación de Tipos Flotante

May 14, Miércoles

Agenda para esta clase

- Ejemplo: Aplicación del Proceso de Desarrollo para Funciones Escalares Simples
- Ejercicio: Celsius(fahr)
- Comparación de Puntos Flotante

Ejemplo: Aplicación del Proceso de Desarrollo para Funciones Escalares Simples

Constante

$$y : \mathbb{Z} \rightarrow \mathbb{Z} / y(x) = 42$$

```
#include <cassert>
#include <iostream>

int y(int);

int main(){
     std::cout << y(-9);
     assert( 42 == y(1) );
}

int y(int x){return 42;}
```

Identidad

$$id : \mathbb{Z} \rightarrow \mathbb{Z} \,/\, id(x) = x$$

```
#include <cassert>
#include <iostream>

int Id(int);

int main(){
     std::cout << Id(42);
     assert( 7 == Id(7) );
}

int Id(int x){return x;}
```

Sucesor

$$\mathbf{suc} : \mathbb{Z} \rightarrow \mathbb{Z} / \mathbf{suc}(x) = x + 1$$

```
int Suc(int);

assert( -41 == Suc(-42) );
assert(   0 == Suc( -1) );
assert(   1 == Suc(  0) );
assert(  42 == Suc( 41) );

int Suc(int x){return x+1;}
```

Negación (ó Inverso u Opuesto)

$$\mathrm{neg}:\mathbb{Z}\to\mathbb{Z}\,/\,\mathrm{neg}(x)=\ -x=(-1)\cdot x=0-x$$

```
int Neg(int);

assert( -7 == Neg(  7) );
assert(  0 == Neg(  0) );
assert( 42 == Neg(-42) );

int Neg(int x){return -x;} // return -1*x;
                           // return 0-x;
```

Ejercicio: Desarrollar Función *Celsius(fahr)*

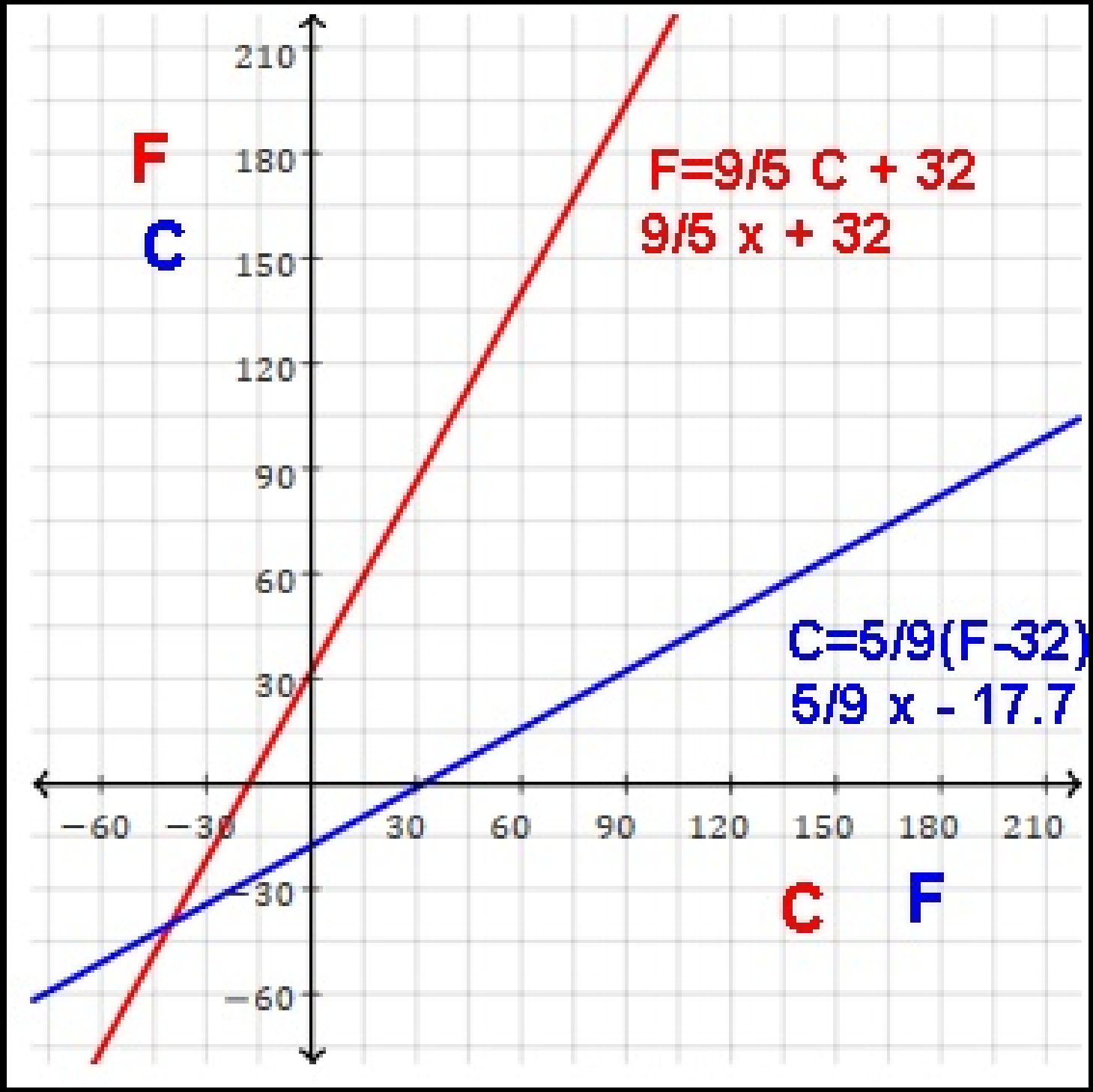
Operaciones Cerradas y Promoción de Tipo

Problema: Conversión de Fahrenheit a Celsius

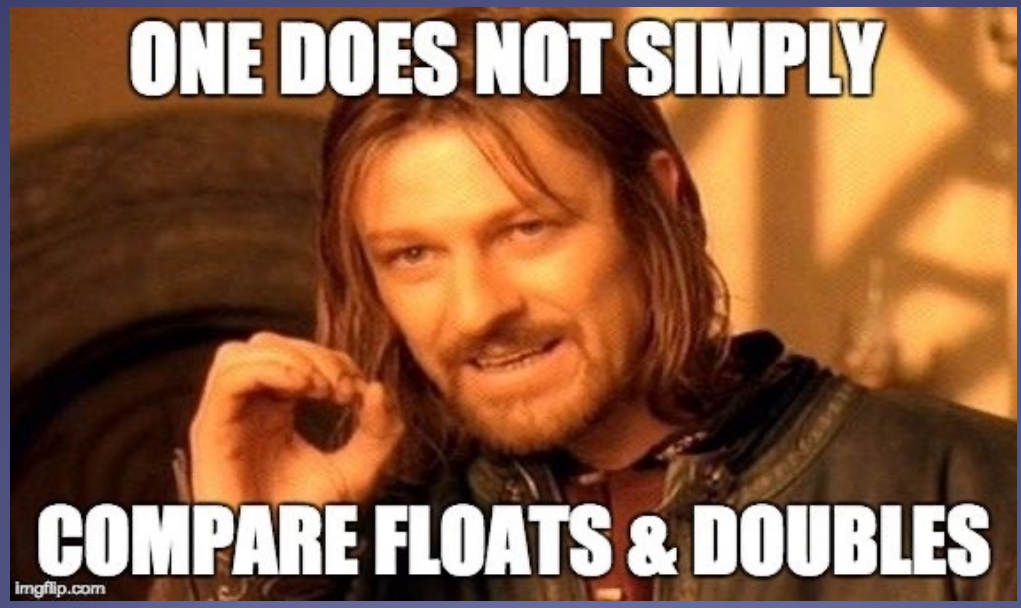
$$celsius : \mathbb{R} \rightarrow \mathbb{R} / celsius(f) = \frac{5}{9}(f - 32)$$

```
double Celsius(double);
```

- Pero hay dos sub-problemas a solucionar antes de poder probar e implementar la función:
  - Valor del racioal cinco novenos versus la división entera en C++ de la expresión 5/9
  - Representación inexacta de los tipos flotantes
- Una solución al primer problema es realizar división entre flotantes.
- Para el segundo problema, debemos incorporar la comparación con tolerancia.


F
C
210
180
150
120
90
60
30
−30
−60
−60
−30
30
60
90
120
150
180
210
F=9/5 C + 32
9/5 x + 32
C=5/9(F-32)
5/9 x - 17.7
C
F

Comparación de Flotantes



Función Están Cerca (AreNear)

```
bool AreNear(double, double, double);
bool AreNear(double, double, double = 0.001);
bool AreNear(double x, double y, double tolerance = 0.001);

assert(     AreNear( 1.0     , 0.999      ));
assert(     AreNear( 1.0     , 0.9   , .1 ));
assert(     AreNear( 1.0/3.0 , 0.333      ));
assert( not AreNear( 1.0/3.0 , 0.33       ));


bool AreNear(double a, double b, double delta){
     return (a-delta) <= b and b <= (a+delta);
}
```

AreNear y Celsius

```
#include <cassert>
#include <iostream>

double Celsius(double);
bool AreNear(double, double, double = 0.001);

int main(){
        assert( 0  == Celsius(32) );
        std::cout << Celsius(64); // 17.7778
        assert( 17.7778 != Celsius(64) );
        assert( 17.777  <  Celsius(64) );
        assert( 17.778  >  Celsius(64) );
        assert( 17.777  <  Celsius(64) and Celsius(64) < 17.778);
        assert( AreNear( 42,  42, 0));
        // 1 == 10*(1/10)
        assert( not AreNear( 1.0, 0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1, 0));
        assert( AreNear(     1.0, 0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1   ));
        assert( AreNear(1.0, 0.999  ));
        assert( AreNear(1.0, 0.9, .1));
        assert( AreNear(    1.0/3.0, 0.333));
        assert( not AreNear(1.0/3.0, 0.33 ));
        assert( AreNear(-35.5556,  Celsius(-32)));
        assert( AreNear(-17.7778,  Celsius(  0)));
        assert( AreNear( 17.7778,  Celsius( 64)));
        assert( AreNear( 17.77777777778,  Celsius( 64), 0.00000000001 ));
}

double Celsius(double f){return 5.0/9.0*(f-32);}

bool AreNear(double a, double b, double delta){
        return (a-delta) <= b and b <= (a+delta);
}
```

AreNear y Celsius (Zoom)

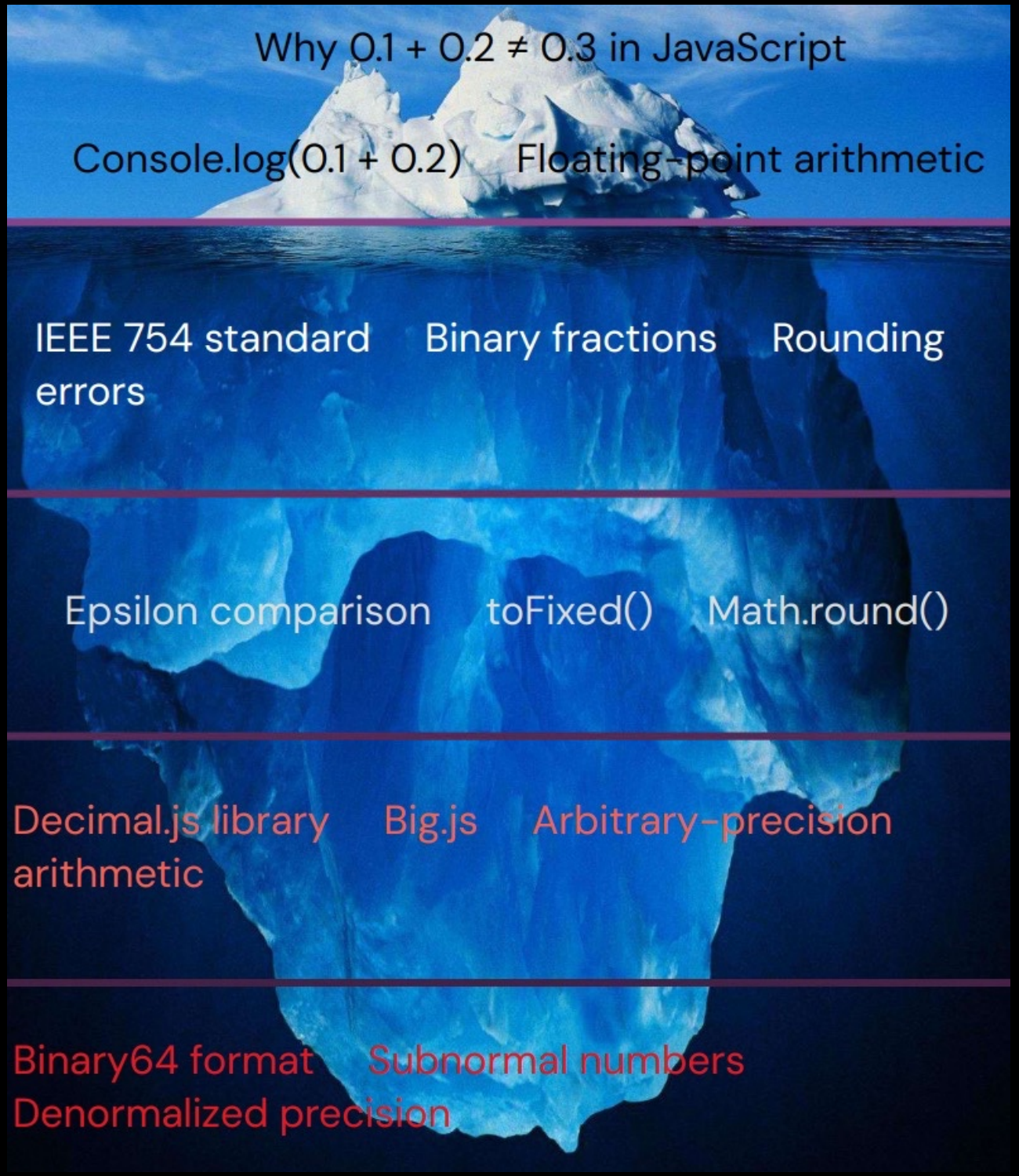
```
#include <cassert>
#include <iostream>

double Celsius(double);
bool AreNear(double, double, double = 0.001);

int main(){
      assert( AreNear( 17.7778,  Celsius( 64)));
      assert( AreNear( 17.77777777778,  Celsius( 64), 0.00000000001 ));
}

double Celsius(double f){return 5.0/9.0*(f-32);}

bool AreNear(double a, double b, double delta){
      return (a-delta) <= b and b <= (a+delta);
}
```


Why 0.1 + 0.2 ≠ 0.3 in JavaScript
Console.log(0.1 + 0.2)
Floating-point arithmetic
IEEE 754 standard errors
Binary fractions
Rounding
Epsilon comparison
toFixed()
Math.round()
Decimal.js library
Big.js
Arbitrary-precision arithmetic
Binary64 format
Subnormal numbers
Denormalized precision

Términos de la clase #06

Definir cada término con la bibliografía

- Función Constante
- Función Identidad
- Función Sucesor
- Función Negación, o Inversa u Opuesto
- Múltiples implementaciones para una misma especificación
- Comparación de Flotantes
  - Truncamiento por división entera
  - Representación no precisa de tipos flotantes
  - Comparación con tolerancia
  - Argumentos por defecto

Tareas para la próxima clase

1. Entregar trabajo “Ejemplos de Valores y Operaciones de Tipos de Datos”
2. Desarrollar la función Celsius; no es entrega formal, todavía.

¿Consultas?

Fin de la clase