

Clase #04 de 27

Implementación de Valores & Funciones

Abril 19, Jueves

Agenda para esta clase

- Resoluciones de trabajos #1 y #2
- Intervalo
- Implementación de Valores
- Funciones I: Introducción

Resolución de Trabajos #1 y #2



Intervalo

20 minutos

Implementación de Valores

Representación en Memoria

Representación en Memoria

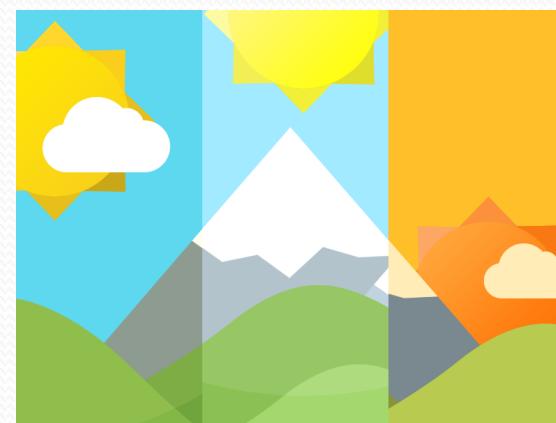
Tipo Matemática	C++	Representación en Memoria Principal
B	bool	1 Byte con un bit (0 ó 1)
S	char	Un byte (8 bits)
N	unsigned	Sigue a int
Z	int	Por lo menos 16 bits.
R	double	Generalmente coincide con double de ISO-IEEE 60559 ó IEEE 754

¿Cómo Representar estos Valores?

- Luces de un semáforo
 - No estados de un semáforo
- Turnos de la factutad
 - Mañana, Tarde, Noche
- Cantidad de bits
- ¿Cuales combinaciones?
- ¿Puede un patrón de bits particular pertenecer a más de un tipo?
- Valores y "Trap values"



0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

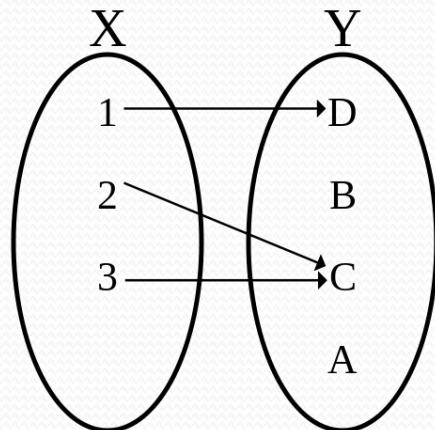




Funciones I

Introducción

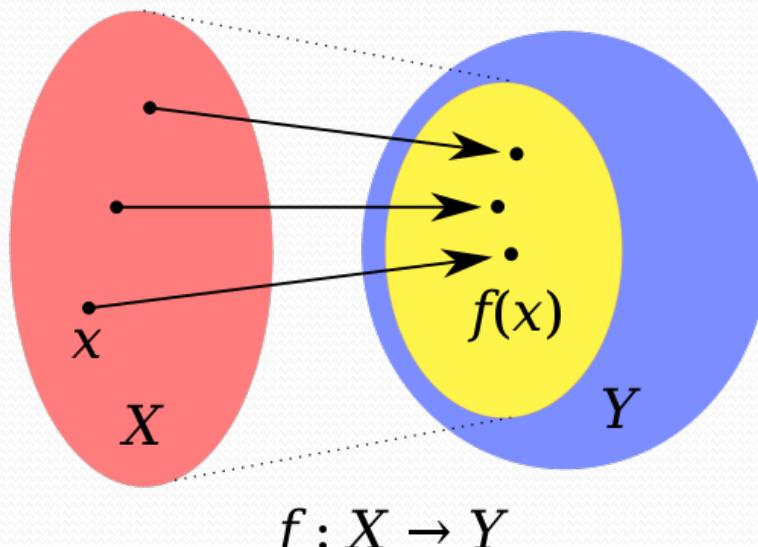
Funciones en Matemática y en Lenguajes de Programación: Ejemplo Introductorio



$$f: \mathbb{R} \rightarrow \mathbb{R} / f(x) = 2x + 1$$

`double f(double);`

```
double f(double x){ return 2*x + 1; }
```



- Conjunto de Salida y Dominio
- Conjunto de Llegada y Codominio

Programa Completo

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

double f(double);

int main(){
    std::cout << f(3);
    assert( 2*3+1 == 7 );
    assert( 7 == f(3) );
}

double f(double x){return 2*x+1;}
```

$$f: \mathbb{R} \rightarrow \mathbb{R}/f(x) = 2x + 1$$

- Assert para pruebas, ya no vamos a usar cout para probar
- Diferencias entre
 - #include <cassert>
 - #include <assert.h>
- Casos de prueba, partición de conjunto de valores
- Terminología
 - Declaración o prototipo de la función
 - Invocación
 - Argumentos
 - Definición o Implementación de la función
 - Parámetros
- Analogía con Matemática

Constante

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

int y(int);

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

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

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

Identidad

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

int Id(int);

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

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

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

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

$$\text{neg}: \mathbb{Z} \rightarrow \mathbb{Z}/\text{neg}(x) = (-1) \cdot x = -x$$

```
int Neg(int);

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

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

Tareas para la próxima clase

1. Trabajo opcional: Probar, Declarar y Definir la función valor absoluto. Restricción: la implementación no debe usar if, ni switch, ni sqrt, ni pow.

Términos de la clase #04

Definir cada término con la bibliografía

- Implementación de Valores
 - Representación en memoria
 - Anchos mínimos
 - ISO-IEEE 60559 ó IEEE 754
 - Combinaciones en función de la cantidad de bits
 - "Trap values"
- Funciones I
 - Definición del concepto matemático
 - Notación matemática
 - Conjunto de Salida y Dominio
 - Conjunto de Llegada y Codominio
 - Imagen
- Prototipo o Declaración de la función
- Definición o Implementación de la función
- Assert
- Casos de pruebas
- Invocación
- Argumentos
- Parámetros



¿Consultas?



Fin de la clase