Conocimiento de Programación Mínimo Requerido para Construir Programas de Buena Calidad

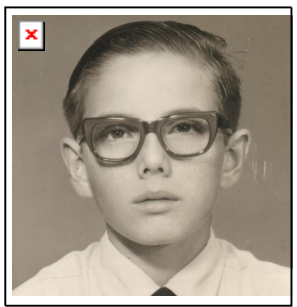
adolfo.dimare@ecci.ucr.ac.cr

Universidad de Costa Rica

Escuela de Ciencias de la Computación e Informática

Agenda

- Especificación
- Documentación Interna
- Espaciado e Indentación
- Datos de Prueba
- ¡No se le Meta Al Rep!
- Ejemplos de Uso
- Descripción Detallada



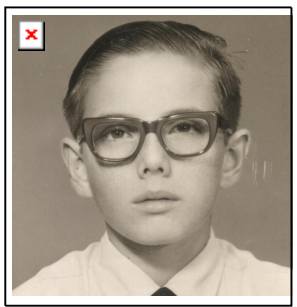
UCR

Conocimiento de Programación Mínimo Requerido para Construir Programas de Buena Calidad

adolfo.dimare@ecci.ucr.ac.cr

7 niveles de entendimiento

1) Se puede recordar 7 cosas
2) Las neuronas tiene una cantidad finita de interconexiones
3) La comprensión se hace exponencialmente más difícil
4) El autismo es una solución
5) Otras solución es limitar a 7 la cantidad de conceptos
6) Ordenamiento ascendente de los conceptos
7) Faltaba uno para llegar a 7



UCR

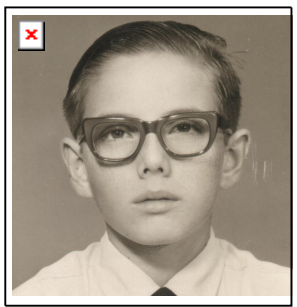
Calidad
Mínimo Conocimiento de Programación

adolfo.dimare@ecci.ucr.ac.cr

#1 Especificación

- Nombre Descriptivo del Módulo
- Funcionalidad
- Quien nos sabe adónde va llega a otro lado
- Frase corta descriptiva
- Nombres adecuados para los parámetros

- Evite el uso de variables globales



UCR

Calcula el Máximo Común Divisor de los números "a" y "b".

- mcd(a,b) >= 1 siempre.
- MCD <==> GCD: Greatest Common Divisor.

Precondición: (b != 0)

Nota:
Se usa el algoritmo de Euclides para hacer el cálculo.

```
{{
//  test::mcd()
    assertTrue( 1 == mcd(1,2) );
    assertTrue( 2*3*5 ==
        mcd( 2*2*2*2 * 3*3 * 5*5, 2*3*5 )
    );
    assertTrue( 30 == mcd( -3600, -30 ) );
}}

template <class INT>
INT mcd(INT a, INT b);
```

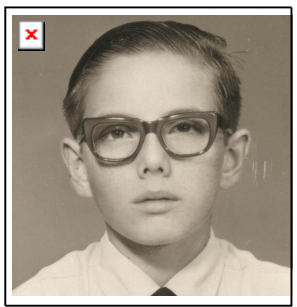
Calidad
Mínimo Conocimiento de Programación

adolfo.dimare@ecci.ucr.ac.cr

#2 Documentación Interna

- Es difícil recordar el algoritmo
- Es difícil recordar aún pára el autor
- Es muy caro reconstruir la lógica del programa
- No es difícil decorar el programa con comentarios cortos y concretos

- Evite la programación pedante



UCR

```
template <class INT>
INT mcd(INT a, INT b) {
INT &cangrejo  = b, &culantro  = a;
INT chuleta24=((culantro<0)?-culantro
:culantro);INT ron=((cangrejo<0)?-cangrejo:cangrejo);
INT impuesto;do {impuesto=ron;ron=chuleta24%ron;
chuleta24=impuesto;}while(0!=ron);return chuleta24;
}
```

```
template <class INT>
INT mcd(INT a, INT b) {
    INT g = ( (a<INT(0)) ? -a : a ); // solo positivos
    INT r = ( (b<INT(0)) ? -b : b ); // "r" es el resto
    INT tmp;

    do {
        tmp = r;
        r   = g%r; // módulo en común
        g   = tmp;
    } while (INT(0) != r);

    return g;
}
```

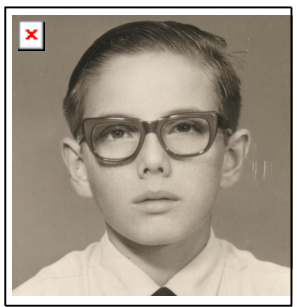
Calidad
Mínimo Conocimiento de Programación

adolfo.dimare@ecci.ucr.ac.cr

#3 Espaciado e Indentación

- El código legible siempre está ordenado
- Muestre la estructura del algoritmo
- Es relativamente fácil usar nombres adecuados

- Use idendificadores “significativos”
- Use embellecedores de código



UCR

```
[CompilerGenerated]
get{// This item is obfuscated and can not be translated.
}
[CompilerGenerated]
set{// This item is obfuscated and can not be translated.
switch ((1 == 1)){case true:goto Label_0020;}
goto Label_0018;if (((0 == 0) ? 0 : 1) != 0)
{}this.eval_t = value;}
```

```
#ifndef Acumulador_h
#define Acumulador_h ///< Evita la inclusión múltiple
class Acumulador{private:long m_total;long m_cantidad;long m_menor;long
m_mayor;public:Acumulador():m_total(0),m_cantidad(0),m_menor(0)
,m_mayor(0){/*borre();*/}~Acumulador(){}Acumulador(const
Acumulador&o){*this=o;}const Acumulador& operator=(const Acumulador&o){
m_total=o.m_total;m_cantidad=o.m_cantidad;m_menor
=o.m_menor;m_mayor=o.m_mayor;return*this;}void borre()
{m_total = m_cantidad = m_menor = m_mayor = 0;}long total() const{
return((m_cantidad <= 2) ? 0 : m_total-m_menor-m_mayor);}long
cantidad() const{return m_cantidad;}void acumule(long
val){m_total += val;m_cantidad++;if(m_cantidad > 1){
m_menor =( val < m_menor ? val : m_menor);m_mayor =( val > m_mayor
? val : m_mayor);}else{m_menor = m_mayor = val;}}void acumule( unsigned
n, const long * val){for(unsigned i=0; i<n; ++i){acumule(val[i]);
}}friend bool operator==(const Acumulador&A,const Acumulador&B){return(
A.m_total==B.m_total&&A.m_cantidad==B.m_cantidad&&A.m_menor==B.m_menor
&&A.m_mayor==B.m_mayor);}friend bool operator!=(const Acumulador&A,const
Acumulador&B){return!(A==B);}friend bool check_ok( const Acumulador&
A);friend class test_Acumulador;}; /* bool */
#endif
```

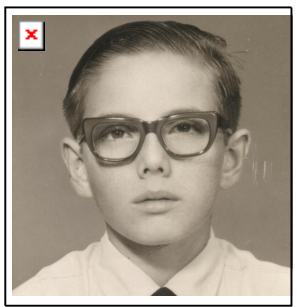
Calidad
Mínimo Conocimiento de Programación

adolfo.dimare@ecci.ucr.ac.cr

#4 Datos de Prueba

- Falibilidad Humana: Inevitable
- Es honeroso reconstruir las pruebas
- Prueba Unitaria complementa la especificación
- Programación ágil / Programación Extrema
- Primero pruebe y luego codifique

- Use **assertTrue(2*3 == mcd(6, 2*2 * 3*3*3*3))**



UCR

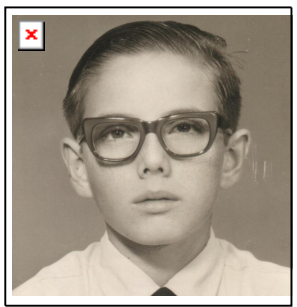
Calidad
Mínimo Conocimiento de Programación

adolfo.dimare@ecci.ucr.ac.cr

#5 ¡No se le Meta Al *Rep*!

- OOP: Encapsulamiento y Ocultamiento
- Exponga la funcionalidad usando métodos
- Oculte la implementación
- Defina la invariante de cada clase
- Proteja los campos de cada instancia

- Use **get() && set()**



UCR

```
rational r(2*2,3*3);

short m = mcd( r.num, r.den );

if ( m != 2*3 ) goto _LINEAS; // !!??

r.den = 2 * 0; // ???

check_ok( r ); // BOOOM !!!
```

Calidad
Mínimo Conocimiento de Programación

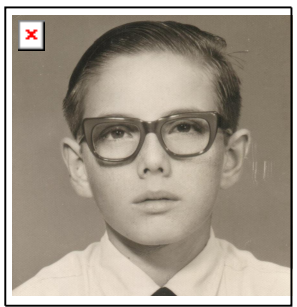
adolfo.dimare@ecci.ucr.ac.cr

#6 Ejemplos de Uso

Una imagen vale por 1,000 palabras

```
assertTrue( 1 == mcd(1,2) );
assertTrue( 2*3*5 == mcd( 2*2*2*2 * 3*3 * 5*5, 2*3*5 ) );
assertTrue( 30 == mcd( -3600,-30 ) );
```

- Use Bunit / JUnit



UCR

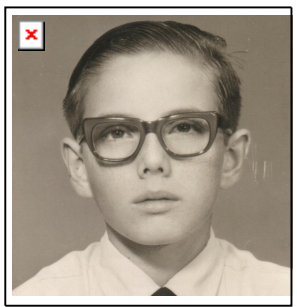
Calidad
Mínimo Conocimiento de Programación

adolfo.dimare@ecci.ucr.ac.cr

#7 Descripción Detallada

- Los Bailarines Bailan
- Los Programadores Programan
- Es difícil redactar especificaciones
- Es necesario redactar especificaciones

- Contrate especialistas para terminar las especificaciones



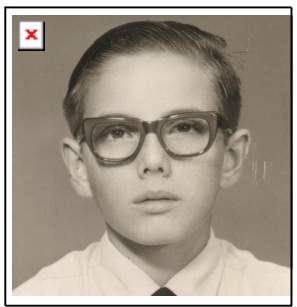
UCR

Calidad
Mínimo Conocimiento de Programación

adolfo.dimare@ecci.ucr.ac.cr

Recomendaciones

- Minimice el uso de variables globales
- Use identificadores significativos
- Incluya la descripción corta en la especificación
- Use documentación interna
- Respete las convenciones de espaciado y de indentación
- Complemente con datos de prueba cada la especificación
- ¡No se le meta al ***Rep***!



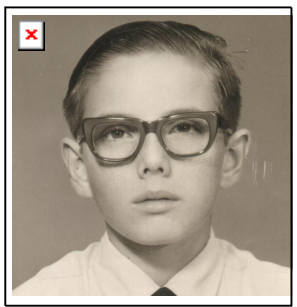
UCR

Conocimiento de Programación Mínimo Requerido para Construir Programas de Buena Calidad

adolfo.dimare@ecci.ucr.ac.cr
Universidad de Costa Rica
Escuela de Ciencias de la Computación e Informática

Análisis Final

- Con muy poco se puede lograr mucho
- Comenzar requiere poco esfuerzo
- Es importante saber qué es lo principal

[http://www.di-mare.com/adolfo/p/qcminspec.htm]

¡¡¡ Muchas gracias !!!