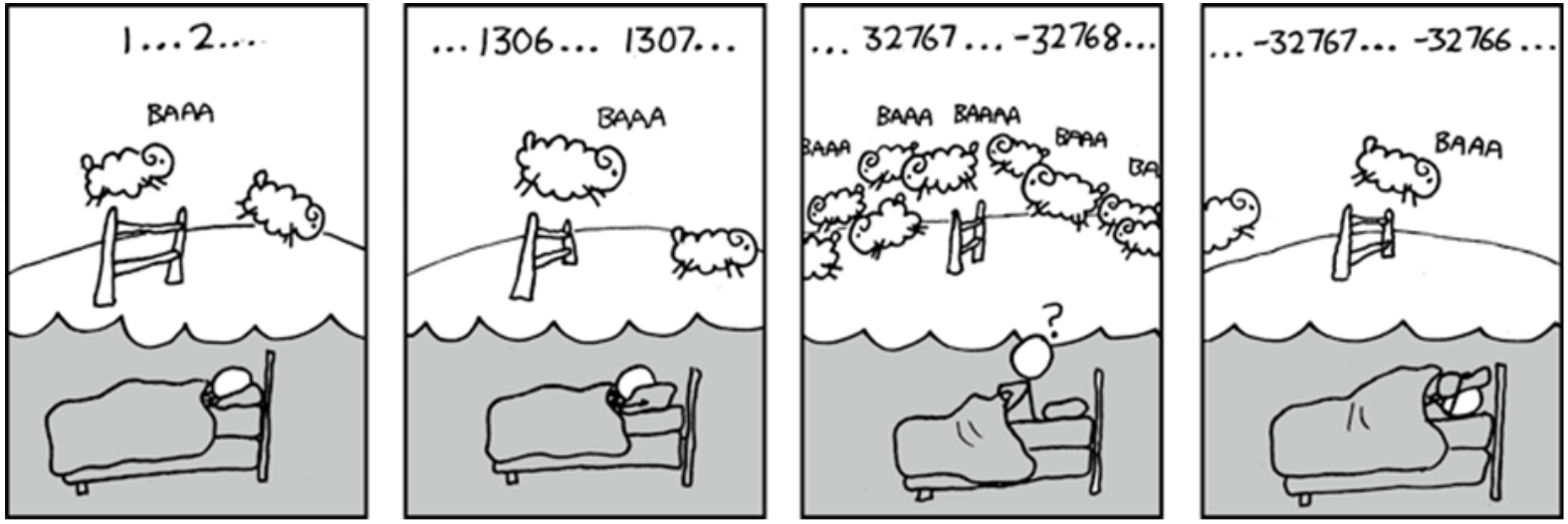
Seguridad en Operaciones con Enteros

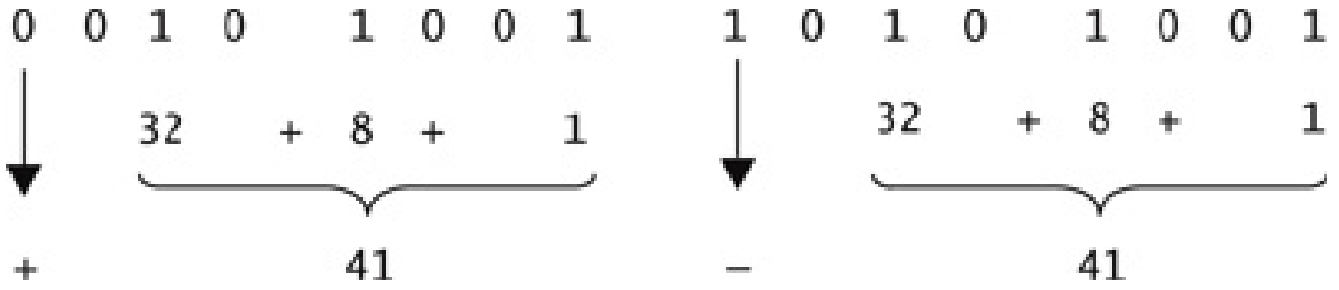
AP 2024



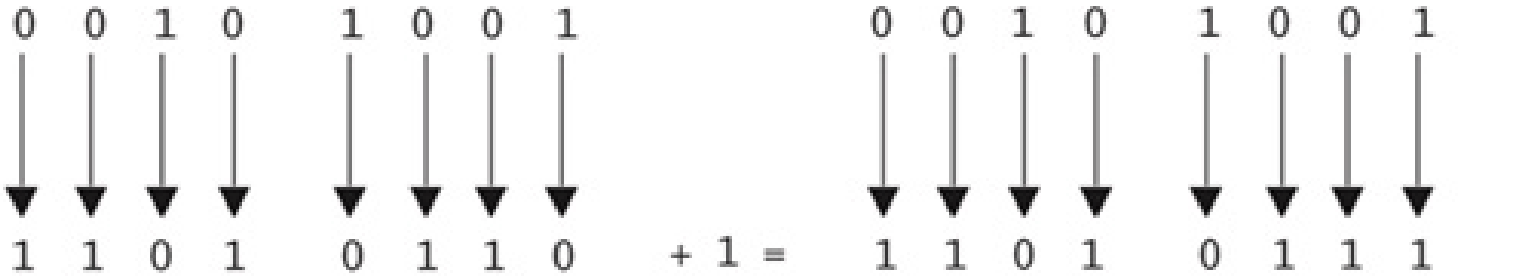
Source: Can’t Sleep, https://imgs.xkcd.com/comics/cant_sleep.png

Representación de Enteros

- Representación de +41 y -41



(a) Módulo y signo



(b) One's complement representation (c) Two's complement representation

https://en.wikipedia.org/wiki/Signed_number_representations
http://ocw.uc3m.es/ingenieria-informatica/programacion-en-fortran/material-de-clase-1/tema1-3.pdf

Complemento a Dos

¿Qué rango se puede representar con 8 y 16 bits?

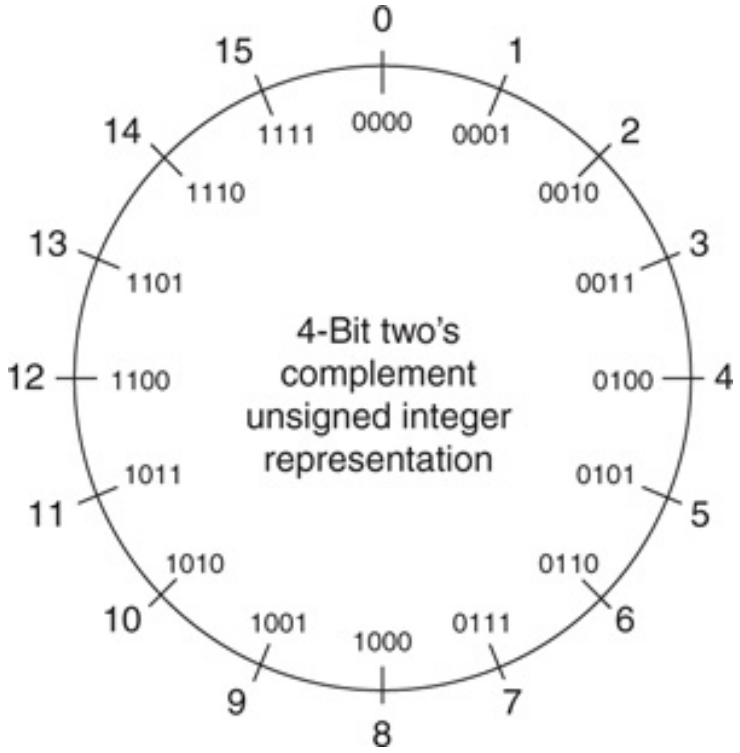
- Desventajas:
  - Rango asimétrico: -2^{n-1}, … , -1, 0, 1, …, $2^{n-1}-1$
- Ventajas:
  - Representación única para el cero
  - **Con números con signo, sumas y restas se pueden hacer con la operación suma y el complemento a Dos**

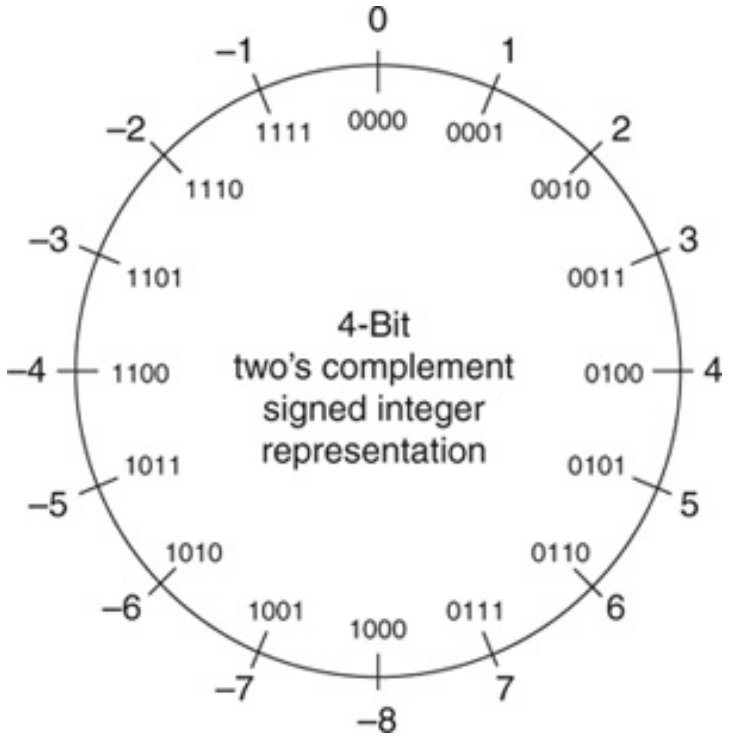
Complemento a Dos

- Si el bit de mayor peso es :
  - 0, el número es positivo.
  - 1, el número es negativo.
- Para cambiar el tamaño del número se repite el bit de mayor peso las veces necesarias
  - +41d = 29h = 0029h = 0000 0029h
  - -41d = D7h = FFD7h = FFFF FFD7h

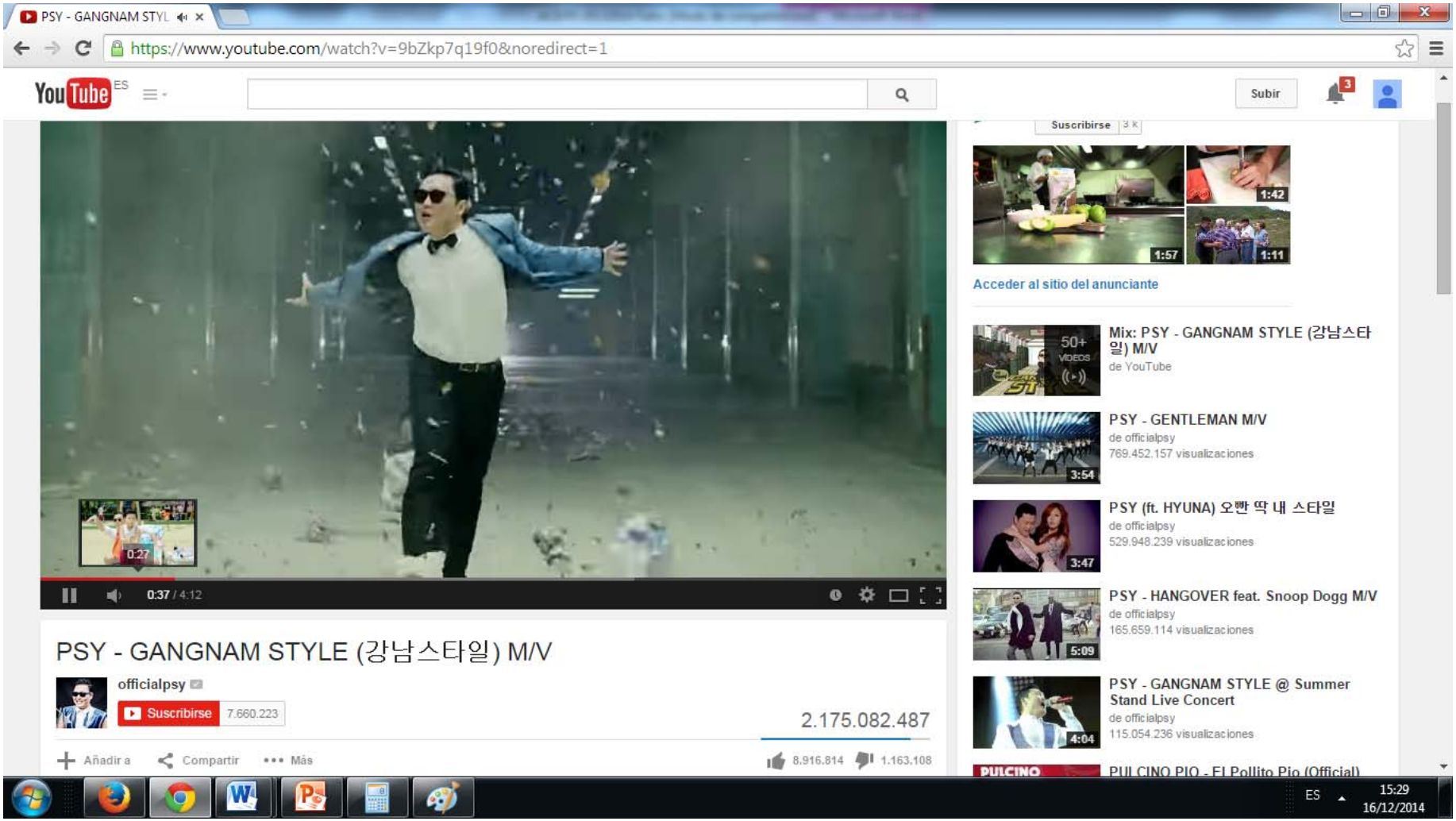
Representación de Enteros

- Enteros de 4 bits





Gangnam Style Overflow



http://elpais.com/elpais/2014/12/04/estilo/1417712408_039258.html

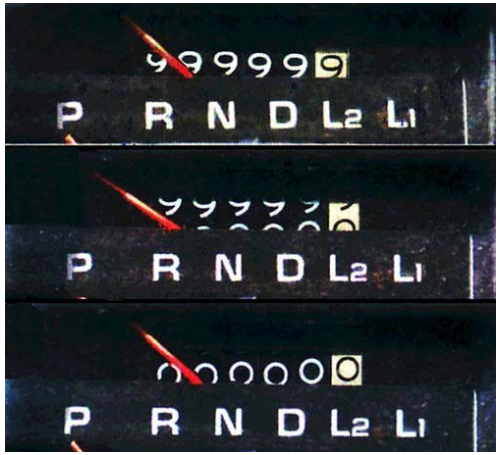
Gangnam Style Overflow

http://www.elmundo.es/enredados/2014/12/03/547efa4ee2704ed6458b4572.html

Gangnam Style Overflow

- int32: -2.147.483.648, …, 2.147.483.647
- https://arstechnica.com/business/2014/12/gangnam-style-overflows-int_max-forces-youtube-to-go-64-bit/

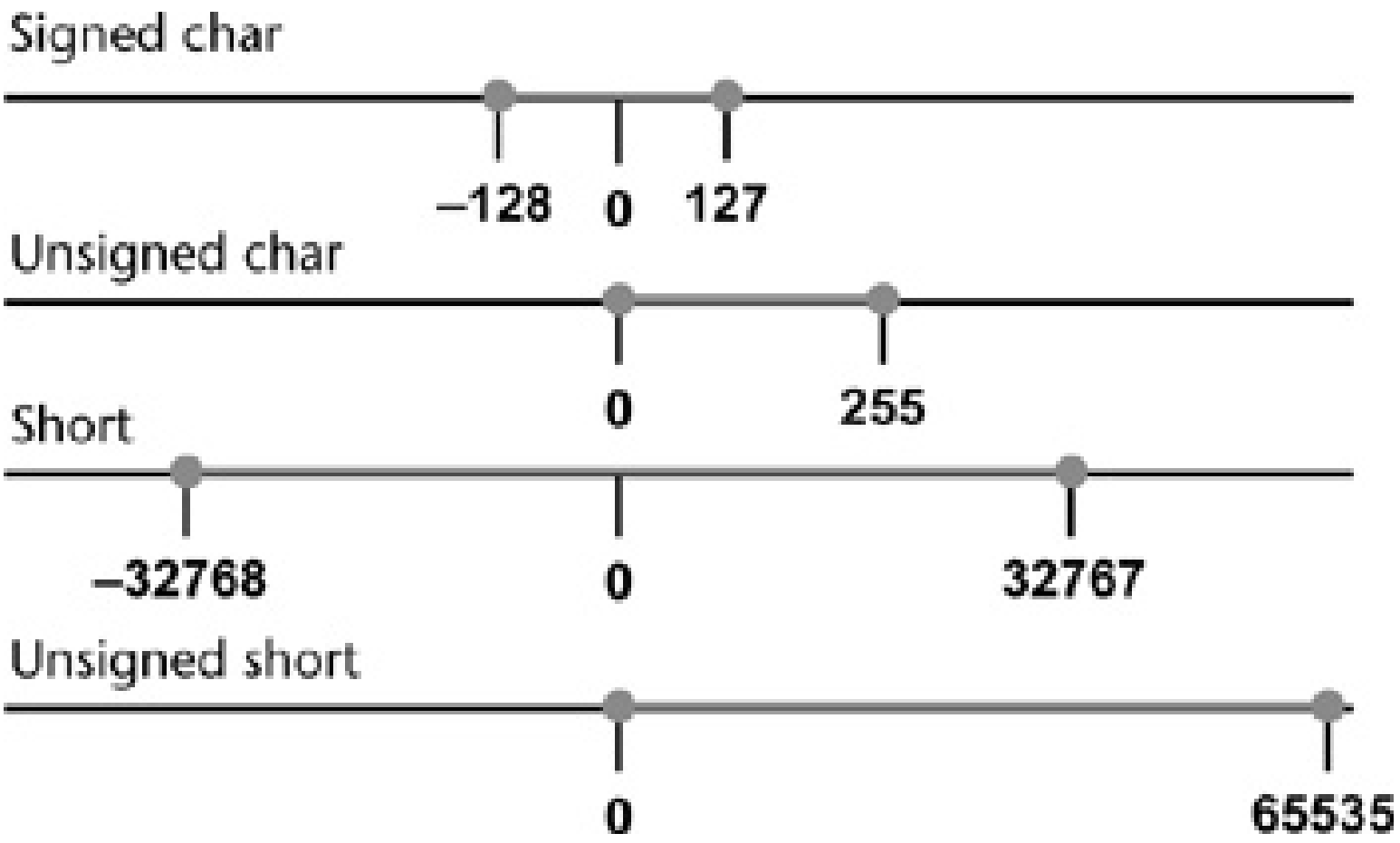
Wrap,
wrap around,
overflow,
darse la vuelta,
desbordar …



Representación de Enteros

- Standard Integer Types, <stdint.h>:
  - (signed) char, int8_t
  - unsigned char, uint8_t
  - (signed) short, int16_t
  - unsigned short, uint16_t
  - (signed) int, int32_t
  - unsigned int, uint32_t
  - (signed) long, int64_t
  - unsigned long, uint64_t

Representación de Enteros



Representación de Enteros
limits.h

SCHAR_MIN	$-2^7 - 1$	-128	Minimum value for an object of type signed char
SCHAR_MAX	$2^7 - 1$	+127	Maximum value for an object of type signed char
UCHAR_MAX	$2^8 - 1$	255	Maximum value for an object of type unsigned char
SHRT_MIN	$-2^{15} - 1$	-32,768	Minimum value for an object of type short int
SHRT_MAX	$2^{15} - 1$	+32,767	Maximum value for an object of type short int
USHRT_MAX	$2^{16} - 1$	65,535	Maximum value for an object of type unsigned short int
INT_MIN	$-2^{31} - 1$	-2,147,483,648	Minimum value for an object of type int
INT_MAX	$2^{31} - 1$	+2,147,483,647	Maximum value for an object of type int
UINT_MAX	$2^{32} - 1$	4,294,967,295	Maximum value for an object of type unsigned int
LONG_MIN	$-2^{63} - 1$		Minimum value for an object of type long int
LONG_MAX	$2^{63} - 1$		Maximum value for an object of type long int
ULONG_MAX	$2^{64} - 1$		Maximum value for an object of type unsigned long int

Errores en Operaciones Enteras

Errores en Operaciones Enteras

```
int i;
i = INT_MAX;                // i = 2,147,483,647
i++;
printf("i = %d\n", i);      // i = -2,147,483,648: VALOR NO ESPERADO
i = INT_MIN;                // i = -2,147,483,648
i--;
printf("i = %d\n", i);      // i = 2,147,483,647

unsigned int j;
j = UINT_MAX;               // 4,294,967,295;
j++;
printf("j = %u\n", j);      // j = 0: VALOR NO ESPERADO
 j = 0;
 j--;
printf("j = %u\n", j);      // j = 4,294,967,295: VALOR NO ESPERADO
```

Errores en Operaciones Enteras

- Operaciones SIN SIGNO:
- SUM= LHS + RHS (sumando izquierdo y derecho);
  - En la suma de número enteros INT sin signo, hay desbordamiento si LHS+RHS>UINTMAX (no evaluable) o RHS>UINTMAX-LHS (sí evaluable)
- DIFF= LHS - RHS;
  - Hay desbordamiento en la resta de enteros sin signo, si LHS < RHS.

INT30-C. Ensure that unsigned integer operations do not wrap

SUMA

Noncompliant Code Example

```
void func(unsigned int ui_a, unsigned int
ui_b) {
  unsigned int usum = ui_a + ui_b;
  /* ... */
}
```

Compliant Solution (Precondition Test)

```
#include <limits.h>

void func(unsigned int ui_a,
unsigned int ui_b) {
  unsigned int usum;
  if (UINT_MAX - ui_a < ui_b) {
    /* Handle error */
  } else {
    usum = ui_a + ui_b;
  }
  /* ... */
}
```

INT30-C. Ensure that unsigned integer operations do not wrap

SUMA

Noncompliant Code Example

```
void func(unsigned int ui_a,
unsigned int ui_b) {
  unsigned int usum = ui_a + ui_b;
  /* ... */
}
```

Compliant Solution (Postcondition Test)

```
void func(unsigned int ui_a,
unsigned int ui_b) {
  unsigned int usum = ui_a + ui_b;
  if (usum < ui_a) {
    /* Handle error */
  }
  /* ... */
}
```

INT30-C. Ensure that unsigned integer operations do not wrap

RESTA

Noncompliant Code Example

```
void func(unsigned int ui_a,
unsigned int ui_b) {
  unsigned int udiff = ui_a - ui_b;
  /* ... */
}
```

Compliant Solution (Precondition Test)

```
void func(unsigned int ui_a,
unsigned int ui_b) {
  unsigned int udiff;
  if (ui_a < ui_b){
    /* Handle error */
  } else {
    udiff = ui_a - ui_b;
  }
  /* ... */
}
```

INT30-C. Ensure that unsigned integer operations do not wrap

RESTA

Noncompliant Code Example

```
void func(unsigned int ui_a,
unsigned int ui_b) {
  unsigned int udiff = ui_a -
ui_b;
  /* ... */
}
```

Compliant Solution (Postcondition Test)

```
void func(unsigned int ui_a,
unsigned int ui_b) {
  unsigned int udiff = ui_a -
ui_b;
  if (udiff > ui_a) {
    /* Handle error */
  }
  /* ... */
}
```

Errores en Operaciones Enteras

- Operaciones CON SIGNO
- Hay desbordamiento en la suma de enteros:
  - En el caso de LHS positivo y RHS positivo, si INT_MAX - LHS < RHS.
  - En el caso de LHS negativo y RHS negativo, si LHS < INT_MIN – RHS.

Addition of Signed Integers of Type int

LHS	RHS	Exceptional condition
Positive	Positive	Overflow if INT_MAX - LHS <= RHS
Positive	Negative	None possible
Negative	Positive	None possible
Negative	Negative	Overflow if LHS <= INT_MIN – RHS

INT32-C. Ensure that operations on signed integers do not result in overflow

SUMA

Noncompliant Code Example

```
void func(unsigned int ui_a, unsigned int ui_b) {
  unsigned int udiff = ui_a - ui_b;
  /* ... */
}
```

Compliant Solution

```
#include <limits.h>

void f(signed int si_a, signed int si_b)
{
  signed int sum;
  if (((si_b > 0) && (si_a > (INT_MAX - si_b))) ||
      ((si_b < 0) && (si_a < (INT_MIN - si_b)))) {
    /* Handle error */
  } else {
    sum = si_a + si_b;
  }
  /* ... */
}
```

INT32-C. Ensure that operations on signed integers do not result in overflow

RESTA

Noncompliant Code Example

```
void func(signed int si_a, signed int si_b)
{
  signed int diff = si_a - si_b;
  /* ... */
}
```

Compliant Solution

```
#include <limits.h>

void func(signed int si_a, signed int si_b) {
  signed int diff;
  if ((si_b > 0 && si_a < INT_MIN + si_b) ||
      (si_b < 0 && si_a > INT_MAX + si_b)) {
    /* Handle error */
  } else {
    diff = si_a - si_b;
  }

  /* ... */
}
```

2038

- Many Unix operating systems store time values in 32-bit signed (positive or negative) integers, counting the number of seconds since midnight on January 1, 1970. On Tuesday, January 19, 2038, this value will overflow, becoming a negative number. Although the impact of this problem in 2038 is not yet known, there are concerns that software that projects out to future dates – including tools for mortgage payment and retirement fund distribution – might face problems long before then.

2038

- https://www.infobae.com/america/tecno/2020/01/23/que-es-el-efecto-del-ano-2038-y-a-que-dispositivos-afectaria/
- https://histinf.blogs.upv.es/2012/12/18/el-efecto-2000/

Facebook group

- There is a Facebook group called 'If this group reaches 4,294,967,296 it might cause an integer overflow.' This value is the largest number that can fit in a 32 bit unsigned integer. If the number of members of the group exceeded this number, it might cause an overflow. Whether it will cause an overflow or not depends upon how Facebook is implemented and which language is used – they might use data types that can hold larger numbers. In any case, the chances of an overflow seem remote, as roughly 2/3 of the people on earth would be required to reach the goal of more than 4 billion members.

Boeing 787

- Para mantenimiento existe una variable int32 para contar los centisegundos que el generador de electricidad lleva encendido. Cuando el valor es negativo el generador se apaga (en vuelo también).
- … "after 248 days of continuous power, all four GCUs (generator control units) will go into failsafe mode at the same time, resulting in a loss of all AC electrical power regardless of flight phase."

Resumen

Declaring a variable as type integer:

- ○ Allocates an infinite amount of storage
- ○ Allocates a fixed amount of storage

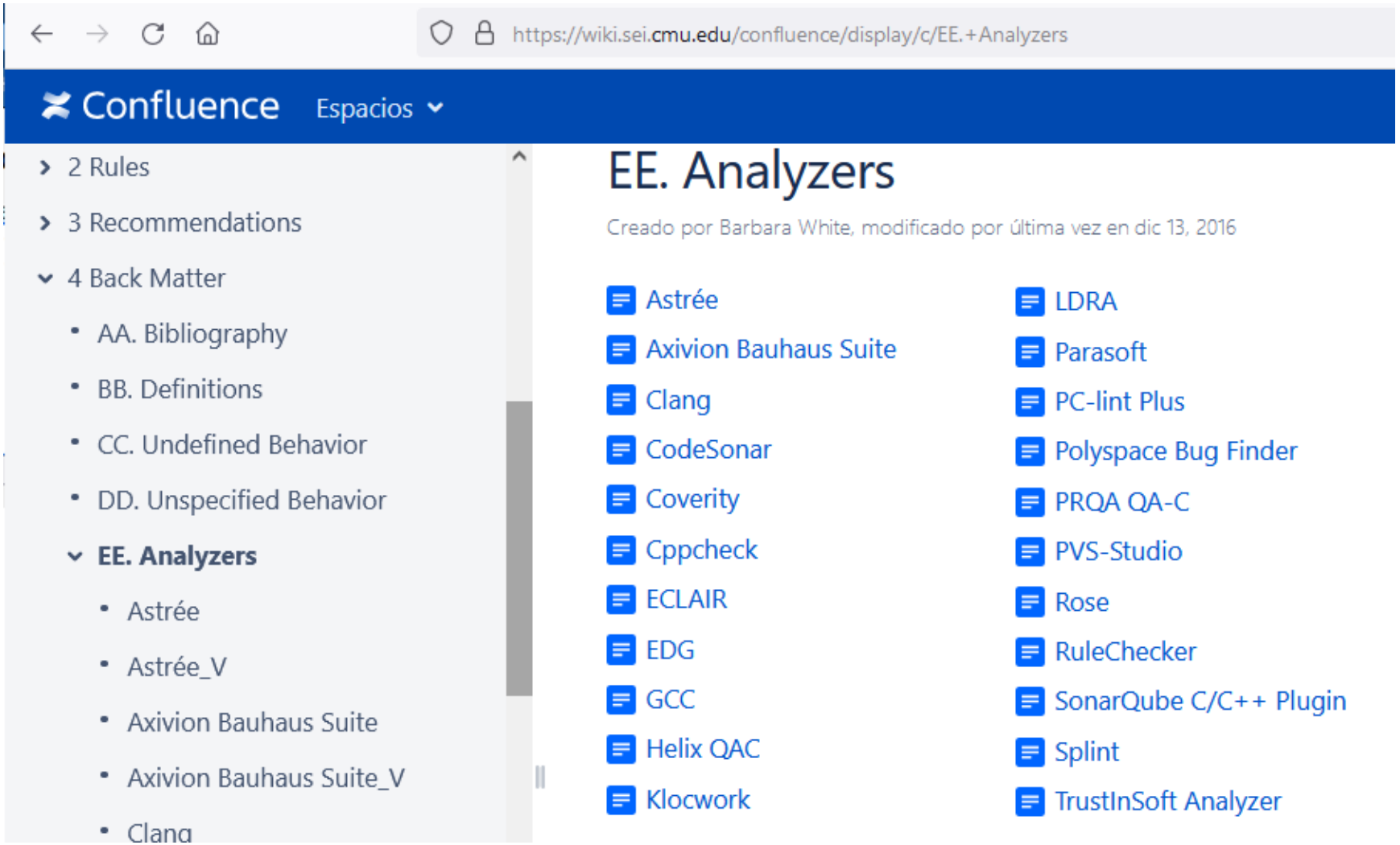
An integer error in C++

- ○ A syntax error
- ○ The program to correct itself
- ○ Unexpected behavior

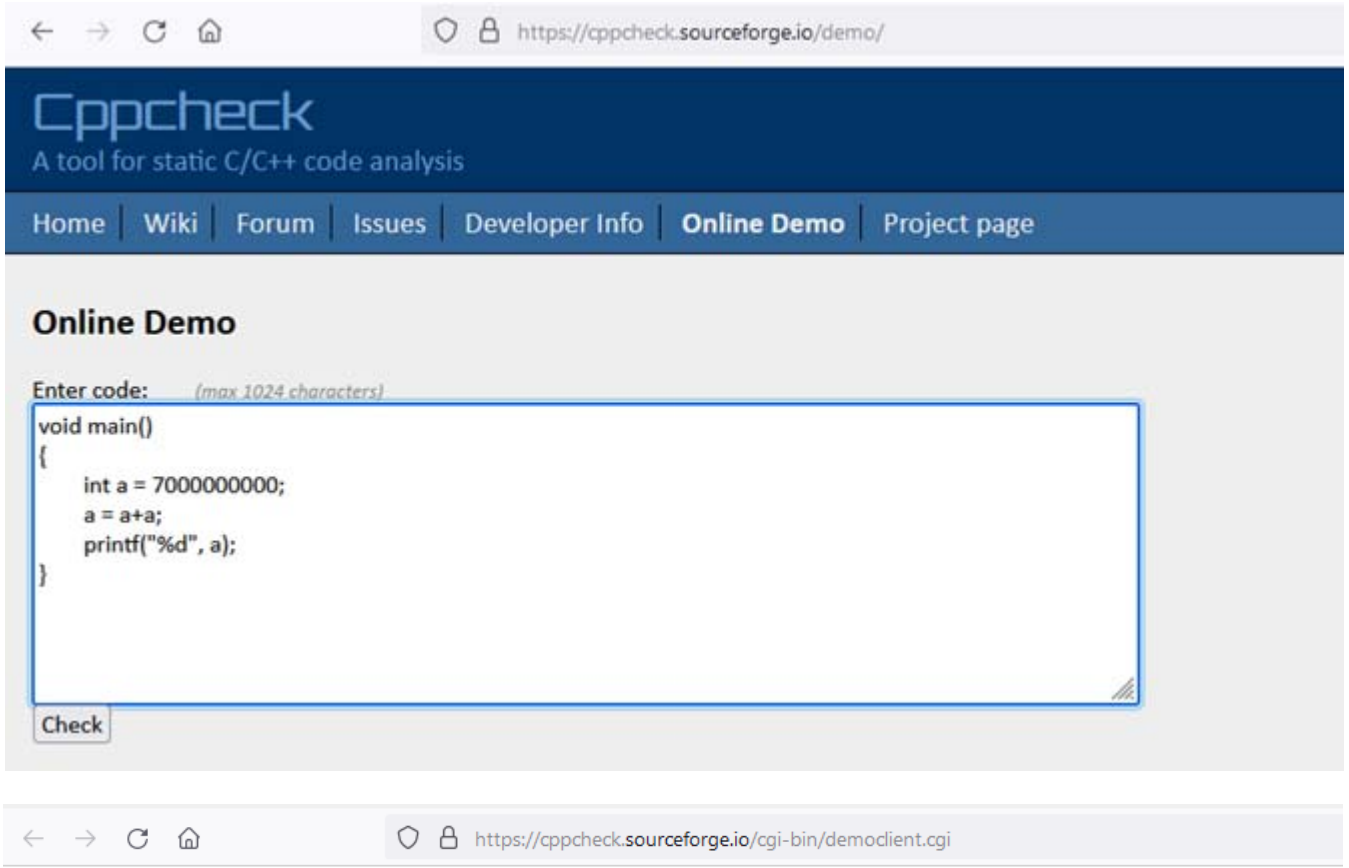
Static Code Analysis

- Static code analysis is a method of debugging by examining source code before a program is run. It's done by analyzing a set of code against a set (or multiple sets) of coding rules.

AI


https://wiki.sei.cmu.edu/confluence/display/c/EE.+Analyzers
Confluence
Espacios
2 Rules
3 Recommendations
4 Back Matter
AA. Bibliography
BB. Definitions
CC. Undefined Behavior
DD. Unspecified Behavior
EE. Analyzers
Astrée
Astrée_V
Axivion Bauhaus Suite
Axivion Bauhaus Suite_V
Clang
EE. Analyzers
Creado por Barbara White, modificado por última vez en dic 13, 2016
Astrée
Axivion Bauhaus Suite
Clang
CodeSonar
Coverity
Cppcheck
ECLAIR
EDG
GCC
Helix QAC
Klocwork
LDRA
Parasoft
PC-lint Plus
Polyspace Bug Finder
PRQA QA-C
PVS-Studio
Rose
RuleChecker
SonarQube C/C++ Plugin
Splint
TrustInSoft Analyzer

cppcheck

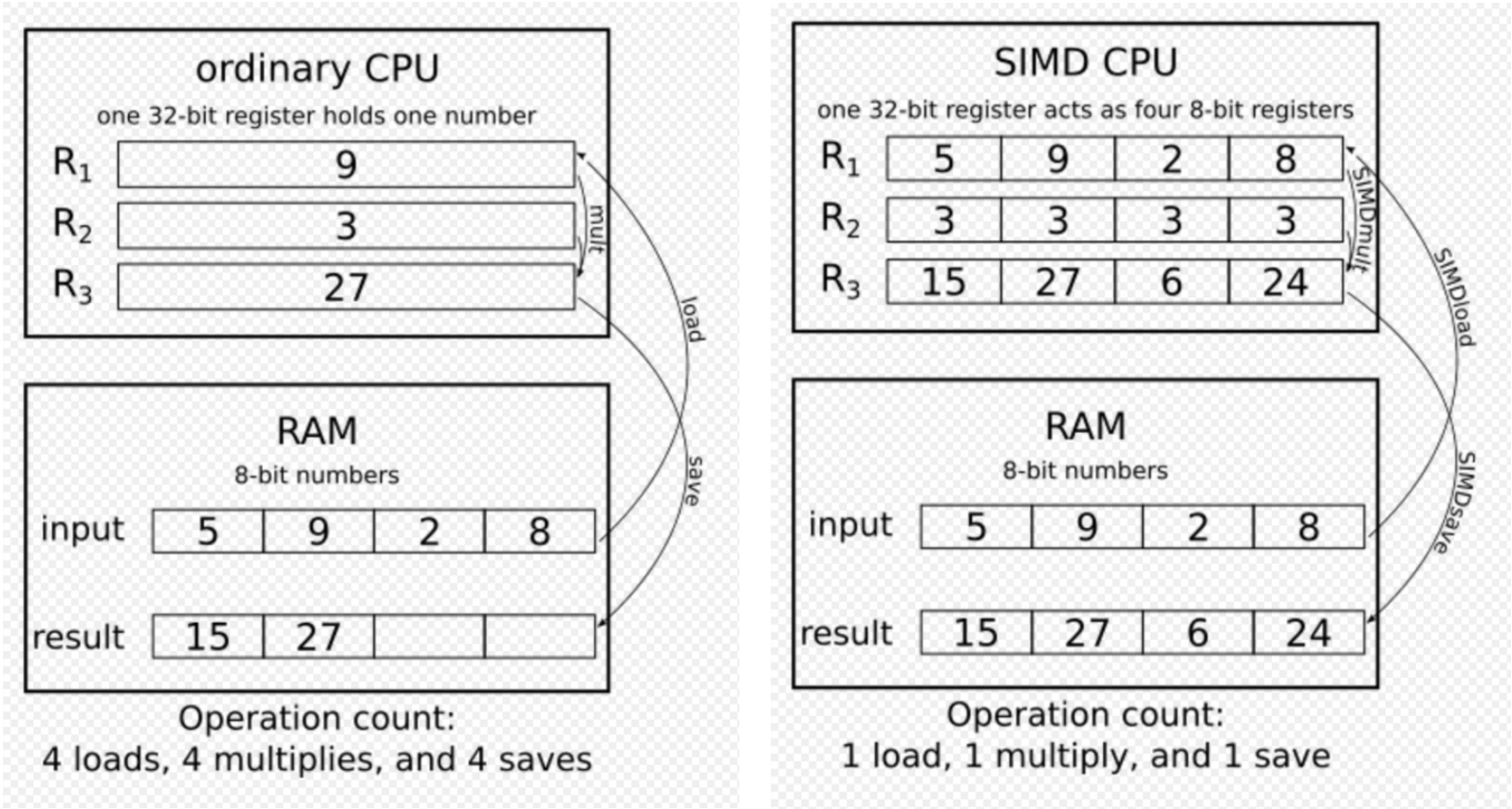


https://cppcheck.sourceforge.io/demo/

java.math
Class BigInteger

- https://docs.oracle.com/javase/7/docs/api/java/math/BigInteger.html
- Immutable arbitrary-precision integers. All operations behave as if BigIntegers were represented in two's-complement notation (like Java's primitive integer types). BigInteger provides analogues to all of Java's primitive integer operators, and all relevant methods from java.lang.Math. Additionally, BigInteger provides operations for modular arithmetic, GCD calculation, primality testing, prime generation, bit manipulation, and a few other miscellaneous operations.

Enteros y SIMD



https://www.cita.utoronto.ca/~merz/intel_c10b/main_cls/mergedProjects/intref_cls/common/intref_sse2_integer_arithmetic.htm

`__mm128i _mm_add_epi8(__m128i a, __m128i b)`

Adds the 16 signed or unsigned 8-bit integers in `a` to the 16 signed or unsigned 8-bit integers in `b`.

R0	R1	...	R15
a0 + b0	a1 + b1;	...	a15 + b15

`__mm128i _mm_add_epi16(__m128i a, __m128i b)`

Adds the 8 signed or unsigned 16-bit integers in `a` to the 8 signed or unsigned 16-bit integers in `b`.

R0	R1	...	R7
a0 + b0	a1 + b1	...	a7 + b7

`__m128i _mm_adds_epi8(__m128i a, __m128i b)`

Adds the 16 signed 8-bit integers in `a` to the 16 signed 8-bit integers in `b` using saturating arithmetic.

R0	R1	...	R15
SignedSaturate (a0 + b0)	SignedSaturate (a1 + b1)	...	SignedSaturate (a15 + b15)

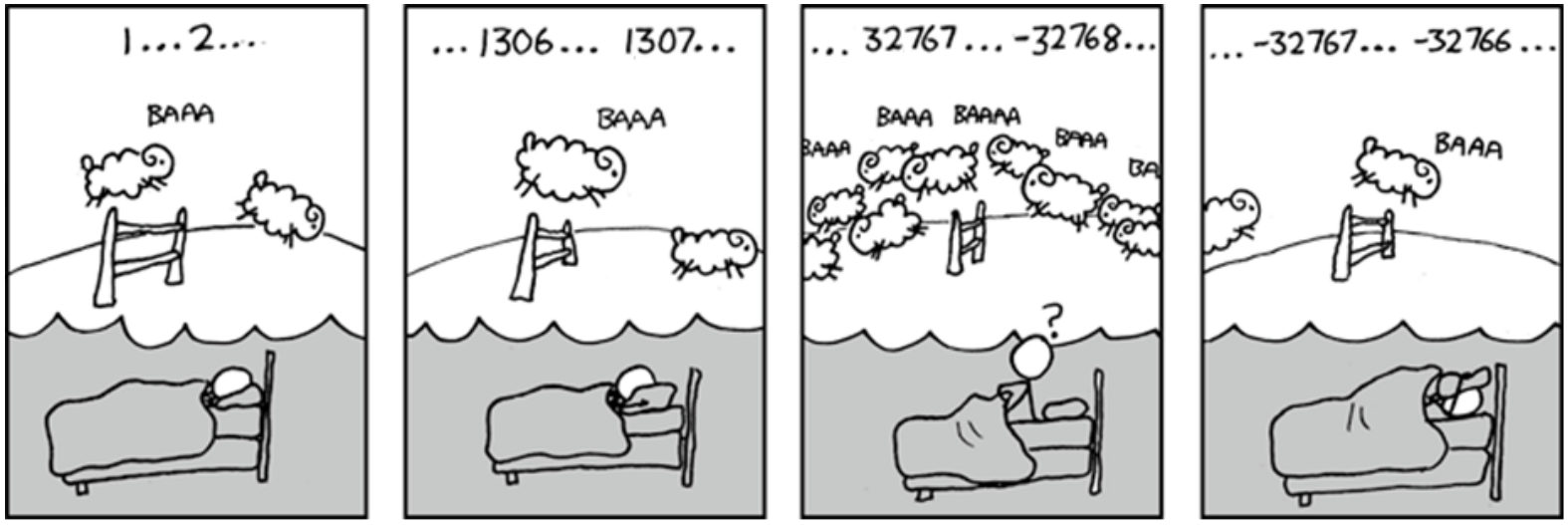
`__m128i _mm_adds_epi16(__m128i a, __m128i b)`

Adds the 8 signed 16-bit integers in `a` to the 8 signed 16-bit integers in `b` using saturating arithmetic.

R0	R1	...	R7
SignedSaturate (a0 + b0)	SignedSaturate (a1 + b1)	...	SignedSaturate (a7 + b7)

Pregunta de Examen

- ¿Qué tipo de datos está usando nuestro amigo programador para contar ovejas?



Bibliografía

- SEI CERT C Coding Standard:
  - Enteros: https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152052
  - Punto Flotante: https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152181
- Secure Coding in C and C++, By Robert C. Seacord, Addison Wesley Professional, September 09, 2005, Print ISBN-13: 978-0-321-33572-2, Web ISBN-13: 978-0-7686-8592-3