

Programación 1

Tema 6

Enteros



Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza





Índice

- El tipo entero
 - Dominio de valores
 - Representación
 - Operaciones
 - Limitaciones
- Resolución de problemas iterativos con enteros
 - Relativos a cifras
 - Relativos a divisibilidad

Tipos enteros

- Dominio de valores
 - Subconjunto de \mathbb{N} o \mathbb{Z}
 - necesidades de representación interna
- Representación externa en C++
 - `<constante-entera> ::= "0" | ([<signo> (<dígito-no-nulo> {"0" | <dígito-no-nulo>}))`
 - `<signo> := "+" | "-"`
 - `<dígito-no-nulo> ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"`



Tipos enteros

- Representación interna
(en la memoria del computador)
 - Tipos sin signo: en binario
 - Tipos con signo: en binario con complemento a 2

Dominio de valores de tipos enteros en C++

- **short int** $-32768 .. 32767$
- **int** $-2147483648 .. 2147483647$
- **long int** $-2147483648 .. 2147483647$
- **long long int** $-9 \times 10^{18} .. 9 \times 10^{18}$
- **unsigned short int** $0 .. 65535$
- **unsigned int** $0 .. 4294967295$
- **unsigned long int** $0 .. 4294967295$
- **unsigned long long int** $0 .. 18 \times 10^{18}$

Dominio de valores de tipos enteros en C++

- **short** *int* -32768 .. 32767
- **int** -2147483648 .. 2147483647
- **long** *int* -2147483648 .. 2147483647
- **long long** *int* $-9 \times 10^{18} \dots 9 \times 10^{18}$
- **unsigned short** *int* 0 .. 65535
- **unsigned** *int* 0 .. 4294967295
- **unsigned long** *int* 0 .. 4294967295
- **unsigned long long** *int* 0 .. 18×10^{18}



Dominio de valores de los tipos `int` y `unsigned int` en C++

Codificación binaria	Como <code>int</code>	Como <code>unsigned int</code>
00000000000000000000000000000000		0
000000000000000000000000000000001		1
0000000000000000000000000000000010		2
0000000000000000000000000000000011		3
...		...
011111111111111111111111111111110		2 147 483 646
011111111111111111111111111111111		2 147 483 647
100000000000000000000000000000000		2 147 483 648
1000000000000000000000000000000001		2 147 483 649
...		...
1111111111111111111111111111111101		4 294 967 293
1111111111111111111111111111111110		4 294 967 294
1111111111111111111111111111111111		4 294 967 295



Dominio de valores de los tipos `int` y `unsigned int` en C++

Codificación binaria	Como <code>int</code>	Como <code>unsigned int</code>
000000000000000000000000000000000000	0	0
000000000000000000000000000000000001	1	1
000000000000000000000000000000000010	2	2
000000000000000000000000000000000011	3	3
...
01111111111111111111111111111111110	2 147 483 646	2 147 483 646
01111111111111111111111111111111111	2 147 483 647	2 147 483 647
100000000000000000000000000000000000	-2 147 483 648	2 147 483 648
100000000000000000000000000000000001	-2 147 483 647	2 147 483 649
...
111111111111111111111111111111111101	-3	4 294 967 293
111111111111111111111111111111111110	-2	4 294 967 294
111111111111111111111111111111111111	-1	4 294 967 295

Tipos enteros en C++

□ Operadores asociados

■ Aritméticos

- Binarios: +, −, *, /, %
- Unarios: +, −

■ Relacionales

- ==, !=
- <, <=, >, >=

Desbordamiento

```
#include <iostream>
using namespace std;

/*
 * Programa que muestra los efectos de un desbordamiento.
 */
int main() {
    unsigned factorial = 1;           // factorial = 0!
    for (unsigned i = 1; i <= 18; i++) {
        factorial = i * factorial;    // factorial = i!
        cout << i << "! = " << factorial << endl;
    }

    return 1;
}
```



Desbordamiento

1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
11! = 39916800
12! = 479001600
13! = 1932053504
14! = 1278945280
15! = 2004310016
16! = 2004189184
17! = 4006445056
18! = 3396534272



Desbordamiento

1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
11! = 39916800
12! = 479001600
13! = 1932053504
14! = 1278945280
15! = 2004310016
16! = 2004189184
17! = 4006445056
18! = 3396534272

Desbordamiento negativo

```
/*  
 * Programa que muestra los efectos de un  
 * desbordamiento negativo.  
 */  
int main() {  
    int i = 2147483647;    //  $2^{31} - 1$   
    i++;  
    cout << i << endl;  
  
    return 1;  
}
```



Desbordamiento negativo

-2147483648

Aritmética de enteros con y sin signo

```
/*  
 * Programa que muestra resultados de una suma y una  
 * multiplicación utilizando enteros con y sin signo.  
 */  
int main() {  
    int a = -8;  
    unsigned b = 3;  
  
    cout << "    Suma a + b: " << a + b << endl;  
    cout << "Producto a * b: " << a * b << endl;  
  
    return 1;  
}
```



Aritmética de enteros con y sin signo

Suma $a + b$: 4294967291

Producto $a * b$: 4294967272

Aritmética de enteros con y sin signo

```
/*  
 * Programa que muestra resultados más lógicos de una suma  
 * y una multiplicación utilizando enteros con y sin  
 * signo.  
 */  
int main() {  
    int a = -8;  
    unsigned b = 3;  
  
    cout << "    Suma a + int(b): " << a + int(b) << endl;  
    cout << "Producto a * int(b): " << a * int(b) << endl;  
  
    return 0;  
}
```

Problemas con enteros

- Tratamiento de cifras
 - Número de cifras
 - Suma de cifras
 - Cálculo de la i -ésima cifra
 - Imagen especular
- Divisibilidad
 - Primalidad
 - Máximo común divisor

Problema:

Número de cifras

```
/*  
 * Pre:  
 * Post:  
 */  
unsigned numCifras(int n) {  
    ...  
}
```

Problema:

Número de cifras

```
/*  
 * Pre: ---  
 * Post: Ha devuelto el número de cifras  
 * de «n» cuando este número se  
 * escribe en base 10.  
 */  
unsigned numCifras(int n) {  
    ...  
}
```



Problema:

Número de cifras

n

14063



Problema:

Número de cifras

n
14063
1406



Problema:

Número de cifras

n
14063
1406
140



Problema:

Número de cifras

n
14063
1406
140
14
1
0



Problema:

Número de cifras

n	cuenta
14063	0
1406	1
140	2
14	3
1	4
0	5

Problema:

Número de cifras

```
/*
 * Pre:  ---
 * Post: Ha devuelto el número de cifras de «n» cuando este número se
 *       escribe en base 10.
 */
unsigned numCifras(int n) {
    unsigned cuenta = 1;           // Lleva la cuenta de las cifras.
    n = n / 10;                    // Elimina la cifra menos significativa de «n».
    // Empezamos la cuenta en 1 y quitamos una cifra antes de entrar al
    // bucle para que numCifras(0) devuelva 1.
    while (n != 0) {
        // El valor de «cuenta» es igual al de cifras identificadas en «n»
        cuenta++;                  // Cuenta la cifra menos significativa de «n»
        n = n / 10;                // y la “elimina”.
    }
    return cuenta;
}
```

Problema:

Suma de las cifras

```
/*  
 * Pre: ---  
 * Post: Ha devuelto La suma de Las  
 * cifras de «n» cuando «n» se  
 * escribe en base 10.  
 */  
unsigned sumaCifras(int n) {  
    ...  
}
```



Problema:

Número de cifras

n				
14063				
14063				
1406				
140				
14				
1				
0				



Problema:

Número de cifras

n	n / 10			
14063				
14063	1406			
1406	140			
140	14			
14	1			
1	0			
0				



Problema:

Número de cifras

n	$n / 10$	$n \% 10$		
14063				
14063	1406	3		
1406	140	6		
140	14	0		
14	1	4		
1	0	1		
0				



Problema:

Número de cifras

n	n / 10	n % 10		
14063				
14063	1406	3	3	
1406	140	6	3+6	
140	14	0	3+6+0	
14	1	4	3+6+0+4	
1	0	1	3+6+0+4+1	
0				



Problema:

Número de cifras

n	n / 10	n % 10	suma	
14063				0
14063	1406	3	3	3
1406	140	6	3+6	9
140	14	0	3+6+0	9
14	1	4	3+6+0+4	13
1	0	1	3+6+0+4+1	14
0				14

Problema:

Suma de las cifras

```
/*
 * Pre:  ---
 * Post: Ha devuelto la suma de las cifras de «n» cuando «n» se escribe
 *       en base 10.
 */
unsigned sumaCifras(int n) {
    if (n < 0) {
        n = -n;    // cambia el signo de «n», si es preciso, para que sea positivo
    }

    unsigned suma = 0;    // valor de la suma de las cifras “eliminadas” de «n»
                          // (inicialmente 0)

    while (n != 0) {
        suma += n % 10;    // suma la cifra menos significativa de «n»
        n = n / 10;        // y la “elimina” de «n»
    }

    return suma;
}
```

Problema:

Números primos

```
/*  
 * Pre: ---  
 * Post: Ha devuelto «true» si y solo si  
 * «n» es un número primo.  
 */  
bool esPrimo(unsigned n) {  
    ...  
}
```

Problema:

Números primos

- **Número primo**
 - Número natural mayor que 1 que tiene únicamente dos divisores distintos: él mismo y el 1
- **Número compuesto**
 - Número natural que tiene algún divisor natural aparte de sí mismo y del 1
- **El número 1**, por convenio, no se considera ni primo ni compuesto.

Problema:

Números primos

□ **Análisis**

- $n = 0$ $\rightarrow n$ no es primo
- $n = 1$ $\rightarrow n$ no es primo
- $n > 1$
 - Hay un número en el intervalo $[2, \sqrt{n}]$ que divide a n $\rightarrow n$ no es primo
 - No hay ningún número en $[2, \sqrt{n}]$ que divide a n $\rightarrow n$ es primo

Problema:

Números primos

- **Análisis** (distinguiendo pares e impares)
 - $n = 0$ → n no es primo
 - $n = 1$ → n no es primo
 - $n = 2$ → n es primo
 - $n > 2$
 - n par → n no es primo
 - n impar y hay otro impar en el intervalo $[3, \sqrt{n}]$ que divide a n → n no es primo
 - n impar y no hay otro impar en el intervalo $[3, \sqrt{n}]$ que divide a n → n es primo

Problema:

Números primos

```
/*  
 * Pre: ---  
 * Post: Ha devuelto «true» si y solo si  
 * «n» es un número primo.  
 */  
bool esPrimo(unsigned n) {  
    ...  
}
```

¿Es 437 primo?

- Mayor que 2 e impar
 - ¿Es divisible por 3? No
 - ¿Es divisible por 5? No
 - ¿Es divisible por 7? No
 - ¿Es divisible por 9? No
 - ¿Es divisible por 11? No
 - ¿Es divisible por 13? No
 - ¿Es divisible por 15? No
 - ¿Es divisible por 17? No
 - ¿Es divisible por 19? Sí → No es primo

¿Es 443 primo?

- Mayor que 2 e impar
 - ¿Es divisible por 3? No
 - ¿Es divisible por 5? No
 - ¿Es divisible por 7? No
 - ¿Es divisible por 9? No
 - ¿Es divisible por 11? No
 - ¿Es divisible por 13? No
 - ¿Es divisible por 15? No
 - ¿Es divisible por 17? No
 - ¿Es divisible por 19? No
 - ¿Es divisible por 21? No
 - $23 > \sqrt{443} \rightarrow$ Es primo

Problema:

Números primos

```
bool esPrimo(unsigned n) {  
    if (n == 2) {  
        return true;           // «n» es igual a 2, luego es primo  
    }  
    else if (n < 2 || n % 2 == 0) {  
        return false;         // «n» es menor que 2 o divisible por 2  
    }  
    else {  
        // Se buscan posibles divisores impares de «n»  
        bool encontrado = false;  
        unsigned divisor = 3;  // 1.ª divisor impar a probar  
        while (!encontrado && divisor * divisor <= n) {  
            encontrado = n % divisor == 0;  
            divisor = divisor + 2;  
        }  
        return !encontrado;  
    }  
}
```

¿Cómo se puede estudiar este tema?

- Repasando estas transparencias
- Trabajando con el código de estas transparencias
 - <https://github.com/prog1-eina/tema-06-enteros>
- Leyendo el material adicional dispuesto en Moodle:
 - Capítulo 6 de los apuntes del profesor Martínez
 - Enlaces al tutorial de Tutorials Point
- Realizando los problemas de los temas 4, 5 y 6
- Realizando las prácticas 2 y 3