

Programación 1

Tema 5

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++
 - $\langle \text{constante-entera} \rangle ::= \text{"0"} \mid ([\langle \text{signo} \rangle] \langle \text{dígito-no-nulo} \rangle \{ \langle \text{dígito} \rangle \})$
 - $\langle \text{signo} \rangle ::= \text{"+"} \mid \text{"-"}$
 - $\langle \text{dígito-no-nulo} \rangle ::= \text{"1"} \mid \text{"2"} \mid \text{"3"} \mid \text{"4"} \mid \text{"5"} \mid \text{"6"} \mid \text{"7"} \mid \text{"8"} \mid \text{"9"}$
 - $\langle \text{dígito} \rangle ::= \text{"0"} \mid \langle \text{dígito-no-nulo} \rangle$

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++ (GNU y MinGW)

- **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++ (GNU y MinGW)

- **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 en C++(GNU y MinGW)

[illegible]



Dominio de valores de los tipos `int` y `unsigned` en C++(GNU y MinGW)

Codificación binaria	Como unsigned	Como int
000000000000000000000000000000000000	0	
000000000000000000000000000000000001	1	
000000000000000000000000000000000010	2	
000000000000000000000000000000000011	3	
...	...	
011111111111111111111111111111111110	2 147 483 646	
011111111111111111111111111111111111	2 147 483 647	
100000000000000000000000000000000000	2 147 483 648	
100000000000000000000000000000000001	2 147 483 649	
...	...	
111111111111111111111111111111111101	4 294 967 293	
111111111111111111111111111111111110	4 294 967 294	
111111111111111111111111111111111111	4 294 967 295	



Dominio de valores de los tipos `int` y `unsigned` en C++(GNU y MinGW)

Codificación binaria	Como unsigned	Como int
000000000000000000000000000000000000	0	0
000000000000000000000000000000000001	1	1
000000000000000000000000000000000010	2	2
000000000000000000000000000000000011	3	3
...
011111111111111111111111111111111110	2 147 483 646	2 147 483 646
011111111111111111111111111111111111	2 147 483 647	2 147 483 647
100000000000000000000000000000000000	2 147 483 648	-2 147 483 648
100000000000000000000000000000000001	2 147 483 649	-2 147 483 647
...
111111111111111111111111111111111101	4 294 967 293	-3
111111111111111111111111111111111110	4 294 967 294	-2
111111111111111111111111111111111111	4 294 967 295	-1

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;
    }
}
```



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;  
}
```



Desbordamiento negativo

-2147483648

Otro desbordamiento negativo

```
/*  
 * Programa que muestra los efectos de un desbordamiento  
 * negativo con datos de tipo unsigned.  
 */  
int main() {  
    // UINT_MAX es mayor entero sin signo representable  
    // como unsigned int. Está definido en <climits>.  
    unsigned i = UINT_MAX;  
    cout << "i = " << i << endl;  
    cout << "¿i == -1??: "  
        << boolalpha << (i == -1) << endl;  
}
```




Otro desbordamiento negativo

```
i = 4294967295  
¿¿i == -1??: true
```

Aritmética de enteros con y sin signo

Cuando el entero con signo es negativo

```
/*  
 * 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;  
}
```



Aritmética de enteros con y sin signo

Cuando el entero con signo es negativo

Suma $a + b$: 4294967291

Producto $a * b$: 4294967272

Aritmética de enteros con y sin signo

Cuando el entero con signo es negativo

```
/*  
 * 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;  
}
```

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

Escriba un número entero: 14063

El número 14063 tiene 5 cifras.

Escriba un número entero: -984

El número -984 tiene 3 cifras.

Escriba un número entero: 0

El número 0 tiene 1 cifras.



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

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

/*
 * Programa que pide un número entero e informa sobre el número de cifras que
 * tiene este.
 */
int main() {
    // Petición del dato
    ...

    // Cálculo del número de cifras
    ...

    // Escritura del resultado
    ...
}
```

Problema:

Número de cifras

```
int main() {  
    // Petición del dato  
    cout << "Escriba un número entero: ";  
    int numero;  
    cin >> numero;  
  
    ...  
}
```

Problema:

Número de cifras

```
int main() {  
    ...  
    // Cálculo del número de cifras  
    unsigned cuenta = 1;  
    int n = numero / 10;  
  
    while (n != 0) {  
        cuenta++;  
        n = n / 10;  
    }  
    ...  
}
```

Problema:

Número de cifras

```
int main() {  
    ...  
    // Escritura del resultado  
    cout << "El número " << numero  
         << " tiene " << cuenta  
         << " cifras." << endl;  
}
```



Problema:

Suma de las cifras

Escriba un número entero: 14063

Las cifras de 14063 suman 14.

Escriba un número entero: -984

Las cifras de -984 suman 21.

Escriba un número entero: 0

Las cifras de 0 suman 0.



Problema:

Suma de las cifras

n				
14063				
14063				
1406				
140				
14				
1				
0				



Problema:

Suma de las cifras

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



Problema:

Suma de las cifras

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



Problema:

Suma de las 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:

Suma de las 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

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

/*
 * Programa que pide un número entero e informa sobre la suma de las cifras
 * que lo componen.
 */
int main() {
    // Petición del dato
    ...

    // Cálculo de la suma de las cifras
    ...

    // Escritura del resultado
    ...
}
```

Problema:

Suma de las cifras

```
int main() {  
    // Petición del dato  
    cout << "Escriba un número entero: ";  
    int numero;  
    cin >> numero;  
  
    ...  
}
```

Problema:

Suma de las cifras

```
int main() {  
    ...  
    // Cálculo de la suma de las cifras  
    int n = numero;  
    if (n < 0) {  
        n = -n;  
    }  
  
    unsigned suma = 0;  
    while (n != 0) {  
        suma = suma + n % 10;  
        n = n / 10;  
    }  
    ...  
}
```

Código completo en <https://github.com/prog1-eina/tema-05-enteros/blob/master/3-suma-cifras.cpp>

Problema:

Suma de las cifras

```
int main() {  
    ...  
  
    // Escritura del resultado  
    cout << "Las cifras de "  
        << numero << " suman "  
        << suma << "." << endl;  
}
```

Problema:

Números primos

Escriba un número natural: 104683

El número 104683 es primo.

Escriba un número natural: 47019

El número 47019 no es primo.

Escriba un número natural: 1

El número 1 no es primo.

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

¿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

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

/*
 * Programa que pide un número natural y escribe en la pantalla
 * si es primo o no.
 */
int main() {
    // Petición del dato
    ...

    // Cálculo de la primalidad del número y escritura del
    // resultado
    ...
}
```


Problema:

Números primos

```
int main() {  
    // Petición del dato  
    cout << "Escriba un número natural: ";  
    unsigned n;  
    cin >> n;  
  
    ...  
}
```

Problema:

Números primos

```
int main() {  
    ...  
    // Cálculo de la primalidad del número y escritura del  
    // resultado  
    if (n == 2) {  
        // «n» es igual a 2, luego es primo.  
        cout << "El número " << n << " es primo." << endl;  
    } else if (n < 2 || n % 2 == 0) {  
        // «n» es menor que 2 o par mayor que 2.  
        cout << "El número " << n << " no es primo." << endl;  
    } else {  
        // Se buscan posibles divisores impares a partir del 3:  
        ...  
    }  
}
```

Problema:

Números primos

```
int main() {  
    ...  
    // Se buscan posibles divisores impares a partir del 3:  
  
    // «divisor» es el siguiente impar candidato a dividir «n».  
    unsigned divisor = 3;           // Primer divisor impar a probar  
  
    // «encontrado» indica si se ha encontrado ya un divisor  
    bool encontrado = false;  
  
    while (!encontrado && divisor * divisor <= n) {  
        encontrado = n % divisor == 0;  
        divisor = divisor + 2;  
    }  
    ...  
}
```

Problema:

Números primos

```
int main() {  
    ...  
    // Discriminación del resultado de la  
    // búsqueda de un divisor  
    if (encontrado) {  
        cout << "El número " << n  
            << " no es primo." << endl;  
    } else {  
        cout << "El número " << n  
            << " es primo." << endl;  
    }  
}
```

¿Cómo se puede estudiar este tema?

- Repasando estas transparencias
- Trabajando con el código de estas transparencias
 - <https://github.com/prog1-eina/tema-05-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