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 $\mathbb{Z}$
    - necesidades de representación interna
- Representación externa en C++
  - `<constanteEntera> ::= [<signo>] ( “0” | (<dígitoNoNulo> {“0”|<dígitoNoNulo>}))`
  - `<signo> := “+” | “-”`
  - `<dígitoNoNulo> ::= “1”|“2”|“3”|“4”|“5”|“6”|“7”|“8”|“9”`

Tipos enteros

- Representación interna
  (en la memoria del computador)
  - En código binario complemento a 2

Dominio de valores de tipos enteros en C++

- `short` $-32768 .. 32767$
- `int` $-2\times10^9 .. 2\times10^9$
- `long` $-2\times10^9 .. 2\times10^9$
- `long long` $-9\times10^{18} .. 9\times10^{18}$
- `unsigned short` $0 .. 65535$
- `unsigned int` $0 .. 4\times10^9$
- `unsigned long` $0 .. 4\times10^9$
- `unsigned long long` $0 .. 18\times10^{18}$

El tipo `int`

- Operadores asociados
  - Aritméticos
    - Binarios: +, – , *, / , %
    - Unarios: +, –
  - Relacionales
    - ==, !=
    - <, <=, >, >=

Desbordamiento

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

/*
 * Pre:  ---
 * Post: Ha mostrado los efectos de un desbordamiento de datos.
 */
int main() {
   int factorial = 1;                  // factorial = 0!
   for (int i = 1; i <= 18; i++) {
      factorial = i * factorial;       // factorial = i!
      cout << i << "! = " << factorial << endl;
   }

   return 0;
}
```

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! = -288522240
18! = -898433024
```

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! = -288522240
18! = -898433024
```

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:
 */
int 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.
 */
int 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.
 */
int numCifras(int n) {
   int cuenta = 1;       // Lleva la cuenta de las cifras identificadas.
   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.
 */
int sumaCifras(int n) {
  ...
}
```

Problema: Suma de las cifras

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

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

Problema: Números primos

- **Análisis** (distinguiendo pares e impares)
  - $n < 0$ → n no es primo
  - $n = 0$ → 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(int n) {
  ...
}
```

Problema: Números primos

```
bool esPrimo(int 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;
      int divisor = 3;              // Primer divisor impar a probar
      while (!encontrado && divisor * divisor <= n) {
         encontrado = n % divisor == 0;
         divisor = divisor + 2;
      }
      return !encontrado;
   }
}
```