Representación de números enteros

Área de Arquitectura y Tecnología de Computadores – Versión 1.0.866, 19/01/2022

Índice

Objetivos

Esta práctica analizará la forma en la que se representan números naturales y enteros en un lenguaje de alto nivel, en concreto en C++. Además, se realizarán diversas operaciones con datos numéricos para entender las repercusiones que tienen los errores de desbordamiento en un programa.

Conocimientos y materiales necesarios

Para aprovechar adecuadamente esta sesión de prácticas, el alumno necesita:

- Conocer los sistemas de codificación de números naturales y enteros (binario natural, hexadecimal, signo-magnitud y complemento a 2).
- Conocer el lenguaje de programación C++.

Durante la sesión se plantearán una serie de preguntas que puedes responder en el correspondiente cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143913) en el Campus Virtual. Puedes abrir el cuestionario en otra pestaña del navegador pinchando en el enlace mientras mantienes pulsada la tecla `Ctrl`.

1. Codificación de enteros y naturales

Para realizar esta práctica se va a utilizar como herramienta el *Visual Studio*. Para simplificar se proporciona un proyecto mínimo ya creado. Descarga del Campus Virtual el fichero `1-Enteros` y descomprímelo.

- Abre el proyecto haciendo doble clic sobre el fichero de solución (extensión `sln`).
- Una vez que está abierto el proyecto compila y ejecuta el programa pulsando la combinación de teclas `Ctrl` + `F5`.

La salida que proporciona este programa es sorprendente: dice que `a` vale 23 y `b`, -5, pero que `a` es menor que `b`. Para entender por qué ocurre esto, en primer lugar hay que codificar estos números. La codificación depende del tipo con el que estén declarados:

- La variable `a` está declarada como `unsigned int`, lo que significa, en la versión de Visual Studio empleada en prácticas, que es un número natural de 32 bits codificado en binario natural. ¿Cuál deberá ser la codificación de `a`? Indica el resultado en hexadecimal. Responde en el cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143913): pregunta 1.
- La variable `b` está declarada como `int`, lo que significa, en la versión de Visual Studio empleada en prácticas, que es un número entero de 32 bits codificado en complemento a 2. ¿Cuál deberá ser la codificación de `b`? Responde en el cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143913): pregunta 2.

Para analizar la codificación de las variables en memoria es necesario ejecutar el programa en modo depuración y pausar su ejecución. La ejecución de un programa se puede pausar utilizando puntos de interrupción, que permiten interrumpir la ejecución del programa en cualquier línea. Realiza los siguientes pasos:

- Sitúa el cursor en la línea en la que se imprime el mensaje por pantalla. Pulsa `F9` para colocar en esa línea un punto de interrupción. El Visual Studio indica que en esa línea hay un punto de interrupción dibujando un círculo rojo en el margen izquierdo. Pulsando con el botón izquierdo del ratón sobre el círculo rojo se elimina el punto de interrupción, y volviendo a pulsar se vuelve a crear.
- Ejecuta el programa en modo depuración pulsando `F5`. Fíjate que para ejecutar el programa en modo depuración no hay que pulsar la tecla `Control`.
- El programa se detendrá justo en la línea donde se colocó el punto de interrupción. La flecha amarilla en el margen izquierdo indica la próxima sentencia que se va a ejecutar.
- En este momento pasando el ratón por encima del nombre de una variable en el programa se puede obtener su valor.
- Otra forma de inspeccionar el valor de las variables es la ventana denominada «Automático», donde se muestran automáticamente variables utilizadas cerca del código ejecutado. También existe la posibilidad de escoger las variables que queremos ver utilizando la ventana «Inspección». Sigue estos pasos para abrir esta ventana: pulsa en el menú Depurar › Ventanas › Inspección › Inspección 1.
- Escribe en el inspector de variables el nombre de las variables `a` (en una línea) y `b` (en otra línea) para ver el valor que almacenan; podrás ver su valor en decimal. Pulsa con el botón derecho del ratón en cualquier parte de la ventana Inspección 1 y selecciona la opción Presentación hexadecimal. El contenido de las variables debería coincidir con la codificación que habías indicado. Si no es así, repasa la codificación y pregunta a tu profesor si no encuentras el problema.

El programa ha determinado que `a` es menor que `b` debido a que al realizar la comparación se ha encontrado con que `a` es una variable sin signo y `b` con signo y ha decidido interpretar la codificación de los dos números como variables sin signo. Esto es así porque C hace tipados (*castings*) automáticos y en el caso de una expresión que mezcle números con y sin signo, decide interpretarlos ambos como números sin signo. Para ver qué valor tiene `b` interpretado sin signo, deberás tomar su codificación e interpretarla como binario natural en lugar de complemento a 2. Como es un número muy grande, vamos a utilizar la calculadora de Windows para comprobar su valor:

- Abre la calculadora en Windows.
- En el menú escoge **Ver › Programador**.
- Pon la calculadora en modo hexadecimal pulsando sobre `Hexa` en la parte izquierda.
- Escribe la codificación de `b`.
- Pulsa sobre `Dec` para que pase el número hexadecimal a decimal.

Como verás, es un número muy grande, mucho mayor que 23 (el valor de `a`) y por eso se escribe el mensaje de que `b` es mayor que `a`. En Visual Studio, recompila la aplicación con **Compilar › Recompilar solución** (no vale una compilación normal porque como no se ha cambiado el código, no vuelve a compilar) y muestra la ventana denominada «Lista de errores» (**Ver › Lista de errores**). Podrás ver que, durante la compilación, se generó un aviso («warning») indicando que no coincidían los tipos de las dos variables comparadas. Recuerda que debes considerar los avisos como errores a no ser que sepas qué consecuencias tienen y que en ese caso concreto no son un error.

2. Almacenamiento en memoria

Vamos a ver cómo se almacena la codificación de estas variables en memoria. Para ello realiza los siguientes pasos:

- En primer lugar es necesario determinar la dirección de memoria en la que se almacenan las variables. Ejecuta el programa en modo depuración pulsando F5 y añade al inspector de variables `&a` (en una nueva línea) y `&b` (en otra línea). El operador `&` se utiliza para calcular la dirección a partir de la cual está almacenada una variable. El prefijo `0x` indica que el número que aparece a continuación está representado en hexadecimal. ¿En qué dirección se almacena `a`? ¿Y `b`? Responde en el cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143913): pregunta 3.
- A continuación se examinará el contenido de la memoria en las direcciones indicadas. Para ello pulsa en el menú **Depurar › Ventanas › Memoria › Memoria 1**. Esta acción abrirá el inspector de memoria.
- En el campo dirección del inspector de memoria introduce la dirección de la variable `a`. Los datos se muestran en hexadecimal agrupados por bytes, por tanto los 32 bits que contienen la codificación de la variable `a` son los cuatro primeros grupos (4 bytes son 32 bits). Sin embargo, para poder obtener el valor de la codificación hay que tener en cuenta que los datos en memoria se almacenan en orden inverso siguiendo el criterio *little-endian*. Esto quiere decir que si se quiere almacenar en memoria el dato `12AB34CD`, la secuencia de bytes que se observará será `CD 34 AB 12`. Teniendo esto en cuenta comprueba que la codificación de `a` coincide con lo que habías respondido previamente.
- Siguiendo el mismo procedimiento, comprueba que la codificación de `b` coincide con tu respuesta anterior.
- Para continuar con la ejecución del programa pulsa de nuevo F5. Como no hay más puntos de interrupción, el programa terminará su ejecución.
- En el código C++, cambia el valor que se asigna a la variable `a` por 64 y el valor que se asigna a la variable `b` por -1. Codifica manualmente estos datos y repite el proceso anterior para verificar que coinciden con la codificación que se hace de estas variables en el programa.

3. Desbordamiento

El desbordamiento se produce cuando se realiza una operación aritmética y el resultado no es representable para el tipo de formato y número de bits con el que se está trabajando. Realiza los siguientes pasos para ver un ejemplo:

- Modifica el programa anterior para que se lean por teclado dos números enteros (con signo), `a` y `b`.
- Suma `a` y `b`, almacenando el resultado en otra variable entera (con signo) denominada `c`. Imprime el valor de `c` por pantalla.
- Compila y ejecuta el programa pulsando `Ctrl` + `F5`. Introduce valores para `a` y `b`, y verifica que funciona correctamente.
- Vuelve a ejecutar el programa pero esta vez introduce el valor dos mil millones para `a` (un 2 y nueve 0) y lo mismo para `b`. ¿Cuál es el resultado? Responde en el cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143913): pregunta 4.
- Razona el resultado que se ha producido. Si no tienes clara la causa deberías consultar con tu profesor.

Se podría pensar que una forma de evitar el problema del desbordamiento es utilizar un mayor número de bits. Sin embargo, siempre se podrían encontrar dos números tal que su suma no fuera representable para ese número de bits. Por tanto, **el desbordamiento es un problema que no se puede evitar, solo detectar**.

Para detectar el desbordamiento cuando se ha realizado una suma con números enteros basta con que algunas de las siguientes dos condiciones sea cierta:

1. Que `a` y `b` sean positivos y `c` sea negativo.
2. Que `a` y `b` sean negativos y `c` sea positivo (mayor o igual que cero).

- Modifica el programa anterior para que, en caso de que la suma provoque desbordamiento, no se imprima el resultado, sino un mensaje indicándolo.

El problema del desbordamiento en sí no se puede evitar. Sin embargo, hay ciertas operaciones que, dependiendo de la forma en la que se realizan, pueden o no generar problemas de desbordamiento. El cálculo de la media aritmética es una de ellas.

- Añade al programa anterior una nueva variable entera, `d`.
- Asigna a `d` la media entre `a` y `b` e imprime por pantalla el valor de `d`.
- Ejecuta el programa e introduce nuevamente tanto para `a` como para `b` el valor dos mil millones (un 2 y nueve 0).
- Si has calculado la media como (a+b)/2 el resultado te habrá salido incorrecto. El problema en este caso no es que el resultado de la operación genere desbordamiento, ya que la media de dos números iguales es ese mismo número y, por tanto, representable. El problema en este caso es que se produce desbordamiento en una operación intermedia, en concreto en la suma que se realiza antes de la división entre dos.
- Una forma alternativa de calcular la media sería mediante la operación a/2+b/2. Matemáticamente es equivalente a la forma de calcular la media anterior, pero en este caso se evita el problema del desbordamiento en las operaciones intermedias. Modifica el cálculo de la media con esta expresión.
- Ejecuta de nuevo el programa con los mismos datos y observa los resultados.

Utilizando los conocimientos adquiridos responde a la siguiente pregunta: ¿qué valor se imprimirá por pantalla al ejecutar el siguiente fragmento de código? Responde en el cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143913): pregunta 5.

CPP

```cpp
const int UN_MILLON = 1000000;
int contador = 0;
for (int i = 0; i < 3000*UN_MILLON; i++)
  contador++;
cout << "contador " << contador << endl;
```

- Añade el fragmento de código al programa y verifica que tu respuesta fue correcta.

4. Ejercicios adicionales

✓ Crea un programa que realice la suma acumulada de todos los números entre 1 y un millón que son múltiplos de 3. Para acumular la suma, utiliza variables de los siguientes tipos: `int` (entero de 32 bits), `short` (entero de 16 bits) y `long long` (entero de 64 bits). Compara y analiza los resultados.

✓ Crea un programa que lea por pantalla dos números enteros e imprima su diferencia. En caso de que dicha operación aritmética produzca un desbordamiento imprime un mensaje de error indicándolo.