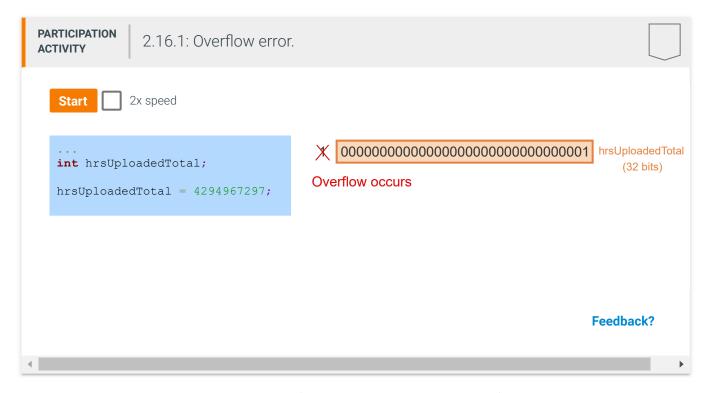
2.16 Integer overflow

An integer variable cannot store a number larger than the maximum supported by the variable's data type. An **overflow** occurs when the value being assigned to a variable is greater than the maximum value the variable can store.



Declaring the variable of type *long long*, (described in another section) which uses at least 64 bits, would solve the above problem. But even that variable could overflow if assigned a large enough value.

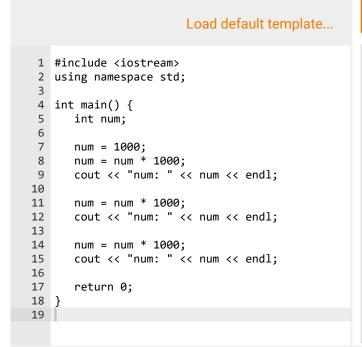
Most compilers detect when a statement assigns to a variable a literal constant so large as to cause overflow. The compiler may not report a syntax error (the syntax is correct), but may output a **compiler warning** message that indicates a potential problem. A GNU compiler outputs the message "warning: overflow in implicit constant conversion", and a Microsoft compiler outputs "warning: '=': truncation of constant value". Generally, good practice is for a programmer to not ignore compiler warnings.

A common source of overflow involves intermediate calculations. Given int variables num1, num2, num3 each with values near 1 billion, (num1 + num2 + num3) / 3 will encounter overflow

in the numerator, which will reach about 3 billion (max int is around 2 billion), even though the final result after dividing by 3 would have been only 1 billion. Dividing earlier can sometimes solve the problem, as in (num1 / 3) + (num2 / 3) + (num3 / 3), but programmers should pay careful attention to possible implicit type conversions.



Run the program and observe the output is as expected. Replicate the multiplicati printing three more times, and observe incorrect output due to overflow. Change r long long, and observe the corrected output.



Run

num: 1000000 num: 1000000000 num: -727379968

Feedback?

PARTICIPATION ACTIVITY

2.16.2: Overflow.



Assume all variables below are declared as int, which uses 32 bits.

1) Overflow can occur at any point in the program, and not only at a variable's initialization.

Correct

Overflow will occur whenever the data type of a variable has insufficient space to store the value assigned.



Yes



O No

- 2) Will x = 1234567890 cause overflow?
 - O Yes
 - No
- 3) Will x = 9999999999 cause overflow?
 - Yes
 - O No
- 4) Will x = 4000000000 cause overflow?
 - Yes
 - O No
- 5) Will these assignments cause overflow?
 - x = 1000;
 - y = 1000;
 - z = x * y;
 - O Yes
 - No
- 6) Will these assignments cause overflow?
 - x = 1000;
 - y = 1000;
 - Z = X * X;
 - z = z * y * y;
 - Yes
 - O No

Correct

The value of about 1 billion assigned to x is within the range of about ± 2 billion, so will not cause overflow.

Correct

This number is greater than about 2 billion, so cannot be assigned to a 32-bit int variable without causing overflow.

Correct

The number is outside the range of a 32-bit int, which is about ±2 billion.

Correct

z is 1,000,000, which is less than 2 billion.

Correct

z is 1,000,000,000,000, which is much larger than 2 billion. Compilers may not detect overflow in calculations and thus not generate a warning; the program simply runs incorrectly.

Feedback?