Parameters and Local Variables for Binary Search

```
int32_t binary_search
    (int32_t array[], int32_t len,
     int32_t value)
{
    int32_t low = 0;
    int32_t high = len - 1;
    int32_t mid;
```

length of array

a sorted array of integers

number to find in array

initialize search bounds [low,high] to [0,len – 1]

Main Iteration: Look Once, Then Adjust Bounds

Keep looking until we have no place to look.

```
    while (high >= low) {
        mid = low + (high - low) / 2;

        // look at one value
        // and adjust bounds
    }
    return -1;
}
```

Look in the middle … but why this way?

Value not found: return -1.

Bugs Can Be Subtle and Hide for Decades

Are these two expressions equivalent?

```
low + (high - low) / 2
(low + high) / 2
```

No …

- but for 20+ years,
- library code for binary search
- used the second expression.

Sums Can Overflow, Producing Negative Array Indices

```
low + (high - low) / 2
(low + high) / 2
```

Consider the following:

- $0 \le$ `low` $< 2^{31}$ and $0 \le$ `high` $< 2^{31}$, and
- `low` $\le$ `high`, so $0 \le$ `(high - low)` $< 2^{31}$ *

What about `low + high`?

Overflow can produce `mid < 0` !

*Technically, one needs to couple this argument with a proof by induction.

