Bitset

- This topic will teach how to use Bitsets
  - Declaration
  - set
  - reset
  - flip
  - at
  - test
  - size
  - count
  - any
  - none
  - operators
  - `to_string`
  - `to_ulong`

- Class `bitset` makes it easy to create and manipulate bit sets
  - Useful for representing a set of bit flags
- Fixed in size at compile time
- An alternate tool for bit manipulation

```
bitset< size > b;
```

- creates `bitset` `b`, in which every one of the `size` bits is initially `0` (“off”)

```
b.set( bitNumber );
```

- Sets bit `bitNumber` of `bitset` `b` “on”

```
b.set()
```

Sets all bits in `b` “on”

©1992–2014 by Pearson Education, Inc.
All Rights Reserved.

```
b.reset( bitNumber );
```

- sets bit `bitNumber` of `bitset` `b` “off”

```
b.reset()
```

sets all bits in `b` “off”

```
b.flip( bitNumber );
```

- “flips” bit `bitNumber` of `bitset` `b`
  - If the bit is on, `flip` sets it off

```
b.flip()
```

flips all bits in `b`

©1992–2014 by Pearson Education, Inc.
All Rights Reserved.

`b[ bitNumber ];`

Returns a reference to the bit `bitNumber` of `b`

`b.at( bitNumber );`

Performs *range checking* on `bitNumber` first

- If `bitNumber` is in range, `at` returns a reference to the bit
- Otherwise, `at` throws an `out_of_range` exception

©1992–2014 by Pearson Education, Inc.
All Rights Reserved.

`b.test( bitNumber );`

performs range checking on `bitNumber` first

- If `bitNumber` is in range
  - `test` returns `true` if the bit is on, `false` it's off
- Otherwise, `test` throws an `out_of_range` exception

`b.size()`

returns the number of bits in `bitset b`.

`b.count()`

returns the number of bits that are set in `bitset b`

©1992–2014 by Pearson Education, Inc.
All Rights Reserved.

`b.any()`

returns `true` if any bit is set in `bitset b`

`b.none()`

returns `true` if none of the bits is set in `bitset b`

`b == b1`

`b != b1`

compare the two `bitset`s for equality and inequality, respectively.

©1992–2014 by Pearson Education, Inc.
All Rights Reserved.

- Each of the bitwise assignment operators &=, |= and ^= can be used to combine `bitset`s

```
b &= b1;
```

- performs a bit-by-bit logical AND between bitsets b and b1
  - The result is stored in `b`

```
b |= b1;
b ^= b2;
```

- Performs bitwise logical OR and bitwise logical XOR

```
b >>= n;
```

shifts the bits in `bitset b` right by `n` positions

©1992–2014 by Pearson Education, Inc.
All Rights Reserved.

```
b <<= n;
```

shifts the bits in `bitset` `b` left by `n` positions

```
b.to_string()
b.to_ulong()
```

convert `bitset` `b` to a `string` and an `unsigned long`, respectively

©1992–2014 by Pearson Education, Inc.
All Rights Reserved.

- This taught and showed examples of how to use Bitsets
  - Declaration
  - set
  - reset
  - flip
  - at
  - test
  - size
  - count
  - any
  - none
  - operators
  - `to_string`
  - `to_ulong`