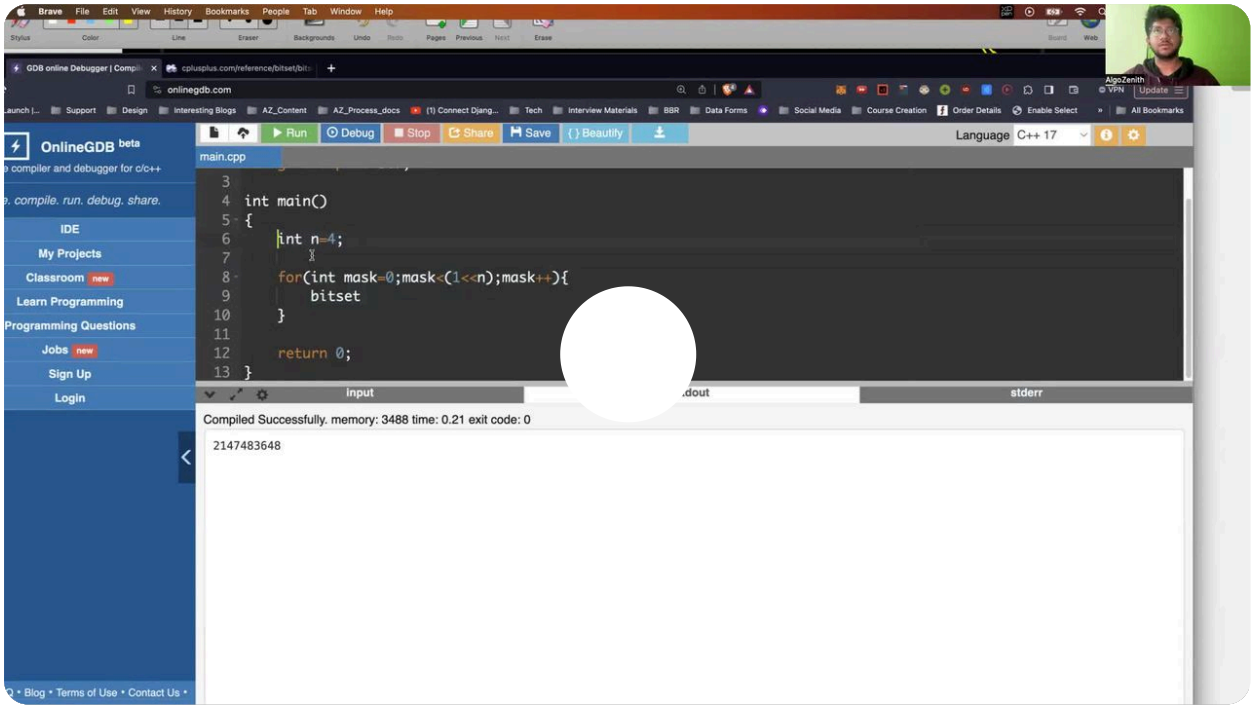
# Bitset in STL




### What is a Bitset?

A `std::bitset` is a container class defined in the C++ Standard Library that represents a fixed-size sequence of bits. Each bit in the sequence can have a value of either 0 or 1. It provides a convenient way to manipulate and perform operations on individual bits or groups of bits efficiently.

### Why is it Used?

- **Memory Efficiency**: Bitset offers memory-efficient storage for boolean flags or switches, where each bit represents the state of a particular condition or option.
- **Bit-level Manipulation**: It allows easy manipulation of individual bits or groups of bits using bitwise operations, enabling efficient implementations of certain algorithms.
- **Compactness**: Especially useful in cases where memory is a concern, like when dealing with large sets of binary flags or representing binary data structures.

### Operations on Bitset:

1. **Construction**:

- Creating a bitset object specifying its size.

```
std::bitset<8> bits; // Creates a bitset with 8 bits, all initialized
```

2. **Setting and Resetting Bits**:
   - Setting a specific bit to 1.
   - Resetting a specific bit to 0.

```
bits.set(2);    // Sets the bit at position 2 to 1.
bits.reset(5);  // Resets the bit at position 5 to 0.
```

3. **Accessing Bits**:
   - Accessing the value of a specific bit.

```
bool bitValue = bits[3]; // Retrieves the value of the bit at position
```

4. **Flipping Bits**:
   - Toggling the value of a specific bit.

```
bits.flip(1); // Toggles the value of the bit at position
```

5. **Counting Set Bits**:
   - Counting the number of set bits in the bitset.

```
size_t count = bits.count(); // Counts the number of set bits in the b
```

6. **Bitwise Operations**:
   - Performing bitwise AND, OR, XOR operations with other bitsets.

```
std::bitset<8> other(0b11001010);
bits |= other;  // Bitwise OR operation with another bitset.
```

```
bits &= other;  // Bitwise AND operation with another bitset.
bits ^= other;  // Bitwise XOR operation with another bitset.
```

7. **Conversion to Integer**:
   - Converting the bitset to an integer value.

```
unsigned long intValue = bits.to_ulong(); // Converts bitset to an
```

8. **String Representation**:
   - Converting the bitset to a string.

```
std::string bitString = bits.to_string(); // Converts bitset to string
```

**Example:**

```
#include <iostream>
#include <bitset>

int main() {
    std::bitset<8> bits(0b10101010); // Creates a bitset with an initial value
    bits.set(2);                     // Sets the bit at position 2 to 1.
    bits.flip(5);                    // Toggles the value of the bit at positi

    std::cout << "Bitset: " << bits << std::endl; // Output: Bitset: 10101110

    bool bitValue = bits[3]; // Retrieves the value of the bit at position 3.
    std::cout << "Bit at position 3: " << bitValue << std::endl; // Output: Bi

    size_t count = bits.count(); // Counts the number of set bits in the bitse
    std::cout << "Number of set bits: " << count << std::endl; // Output: Numb

    return 0;
}
```

This example demonstrates basic operations on a `std::bitset`, including setting, flipping, accessing, and counting bits.