C++ bitset and its application

A bitset is an array of bool but each Boolean value is not stored separately instead bitset optimizes the space such that each bool takes 1 bit space only, so **space taken by bitset bs is less than that of bool bs[N] and vector bs(N)**. However, a limitation of bitset is, **N must be known at compile time, i.e., a constant** (this limitation is not there with vector and dynamic array)

As bitset stores the same information in compressed manner the operation on bitset are faster than that of array and vector. We can access each bit of bitset individually with help of array indexing operator [] that is bs[3] shows bit at index 3 of bitset bs just like a simple array. Remember bitset starts its indexing backward that is for 10110, 0 are at 0th and 3rd indices whereas 1 are at 1st 2nd and 4th indices.
We can construct a bitset using integer number as well as binary string via constructors which is shown in below code. The size of bitset is fixed at compile time that is, it can't be changed at runtime.
The **main function defined for bitset class** are operator [], count, size, set, reset and many more they are explained in below code –

```
// C++ program to demonstrate various functionality of bitset
#include <bits/stdc++.h>
using namespace std;

#define M 32

int main()
{
    // default constructor initializes with all bits 0
    bitset<M> bset1;

    // bset2 is initialized with bits of 20
    bitset<M> bset2(20);

    // bset3 is initialized with bits of specified binary string
    bitset<M> bset3(string("1100"));

    // cout prints exact bits representation of bitset
    cout << bset1 << endl; // 00000000000000000000000000000000
    cout << bset2 << endl; // 00000000000000000000000000010100
    cout << bset3 << endl; // 00000000000000000000000000001100
    cout << endl;

    // declaring set8 with capacity of 8 bits

    bitset<8> set8; // 00000000

    // setting first bit (or 6th index)
    set8[1] = 1; // 00000010
    set8[4] = set8[1]; // 00010010
    cout << set8 << endl;

    // count function returns number of set bits in bitset
    int numberof1 = set8.count();

    // size function returns total number of bits in bitset
    // so there difference will give us number of unset(0)
    // bits in bitset
    int numberof0 = set8.size() - numberof1;

    cout << set8 << " has " << numberof1 << " ones and "
         << numberof0 << " zeros\n";

    // test function return 1 if bit is set else returns 0
    cout << "bool representation of " << set8 << " : ";
    for (int i = 0; i < set8.size(); i++)
        cout << set8.test(i) << " ";

    cout << endl;

    // any function returns true, if atleast 1 bit
    // is set
    if (!set8.any())
        cout << "set8 has no bit set.\n";

    if (!bset1.any())
        cout << "bset1 has no bit set.\n";

    // none function returns true, if none of the bit
    // is set
    if (!bset1.none())
        cout << "bset1 has some bit set\n";

    // bset.set() sets all bits
    cout << set8.set() << endl;

    // bset.set(pos, b) makes bset[pos] = b
    cout << set8.set(4, 0) << endl;

    // bset.set(pos) makes bset[pos] = 1  i.e. default
    // is 1
    cout << set8.set(4) << endl;

    // reset function makes all bits 0
    cout << set8.reset(2) << endl;
    cout << set8.reset() << endl;

    // flip function flips all bits i.e.  1 <-> 0
    // and  0 <-> 1
    cout << set8.flip(2) << endl;
    cout << set8.flip() << endl;

    // Converting decimal number to binary by using bitset
    int num = 100;
    cout << "\nDecimal number: " << num
         << "  Binary equivalent: " << bitset<8>(num);

    return 0;
}
```

Run on IDE

Output :

```
00000000000000000000000000000000
00000000000000000000000000010100
00000000000000000000000000001100
```

```
00010010
00010010 has 2 ones and 6 zeros
bool representation of 00010010 : 0 1 0 0 1 0 0 0
bset1 has no bit set.
11111111
11101111
11111111
11111011
00000000
00000100
11111011

Decimal number: 100 Binary equivalent: 01100100
```

For bitset set, reset and flip function are defined. Set function sets (1) all bits of bitset if no argument is provided otherwise it sets the bit whose position is given as argument. In same way reset and flip also work if they are called with no argument they perform their operation on whole bitset and if some position is provided as argument then they perform operation at that position only.
For bitset all bitwise operator are overloaded that is they can be applied to bitset directly without any casting or conversion, main overloaded operator are &, |, ==, != and shifting operator <> which makes operation on bitset easy.
Use of above operator is shown in below code.

```cpp
// C++ program to show applicable operator on bitset.
#include <bits/stdc++.h>
using namespace std;

int main()
{
    bitset<4> bset1(9); // bset1 contains 1001
    bitset<4> bset2(3); // bset2 contains 0011

    // comparison operator
    cout << (bset1 == bset2) << endl; // false 0
    cout << (bset1 != bset2) << endl; // true  1

    // bitwise operation and assignment
    cout << (bset1 ^= bset2) << endl; // 1010
    cout << (bset1 &= bset2) << endl; // 0010
    cout << (bset1 |= bset2) << endl; // 0011

    // left and right shifting
    cout << (bset1 <<= 2) << endl; // 1100
    cout << (bset1 >>= 1) << endl; // 0110

    // not operator
    cout << (~bset2) << endl; // 1100

    // bitwise operator
    cout << (bset1 & bset2) << endl; // 0010
    cout << (bset1 | bset2) << endl; // 0111
    cout << (bset1 ^ bset2) << endl; // 0101
}
```

Run on IDE

Output :

```
0
1
```

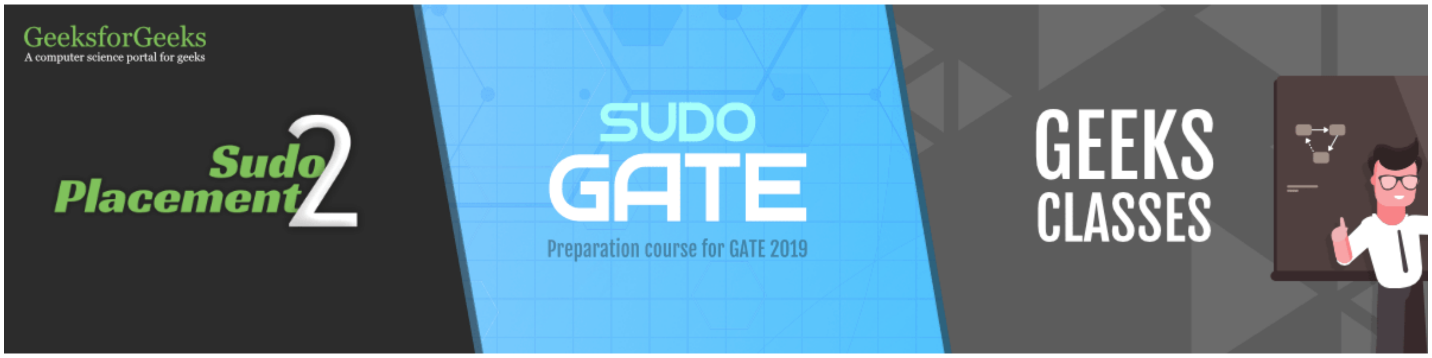
```
1010
0010
0011
1100
0110
1100
0010
0111
0101
```

This article is contributed by Utkarsh Trivedi. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Improved By : gaurav1614

Practice Tags : Bit Magic STL C CPP

Article Tags : Bit Magic C C++ CPP-bitset CPP-Library STL


GeeksforGeeks
A computer science portal for geeks
Sudo Placement 2
SUDO GATE
Preparation course for GATE 2019
GEEKS CLASSES

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content. Login to Improve this Article

Recommended Posts:

C++ bitset interesting facts
Subset sum queries using bitset
std::to_string in C++
unordered_map in C++ STL
Gray to Binary and Binary to Gray conversion
Remove duplicates from a string in O(1) extra space
Calculate the total fine to be collected
Sum of bitwise OR of all subarrays
Value in a given range with maximum XOR
Check if a number can be expressed as 2^x + 2^y

(Login to Rate)

2.5 Average Difficulty : **2.5/5.0**
Based on **15** vote(s)

Basic Easy Medium Hard Expert

Add to TODO List

Mark as DONE

Feedback Add Notes Improve Article

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

Share this post!

A computer science portal for geeks

710-B, Advant Navis Business Park,
Sector-142, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY

About Us
Careers
Privacy Policy
Contact Us

LEARN

Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

PRACTICE

Company-wise
Topic-wise
Contests
Subjective Questions

CONTRIBUTE

Write an Article
Write Interview Experience
Internships
Videos