Bit Shifting Technique, Shift Left (<<) and Shift Right (>>)

Multiplication by 2^n / Shift Left (<<)

If you shift 1 bit to left equals to multiply by 2.
If you shift 2 bit to left equals to multiply by 4.
If you shift n bit to left equals to multiply by 2^n.

Division by 2^n / Shift Right (>>)

If you shift 1 bit to right equals to divide by 2.
If you shift 2 bit ro right equals to divide by 2.
If you shift n bit to right equals to divide by 2^n.

Bit Shifting + Addition / Subtraction

We can only handle multiplication by 2^n using bit shifting.
To handle multiplication we can use bit shifting and addition / subtraction.
If you want to multiply by n denote n as addition / subtraction of powers of 2.

```
a = a * 38
  = a * (2^1 + 2^2 + 2^5)
  = (a * 2^1) + (a * 2^2) + (a * 2^5)
  = (a << 1) + (a << 2) + (a << 5)
```

Type Casting Technique

```
cout << (char) 65 << endl;
cout << (int) 'A' << endl;
cout << (int) 2.35 << endl;
cout << (float) num2 / num1 << endl << endl;
```

```
#include <string>
 #include <sstream>
 {  string str1 = "20"; int num1 = 123647;
     cout << str1 << "  " << num1 << endl;
     stringstream ss;
      ss << num1;       // ss << str1;
    for (int i = 0; i < str1.length (); i++)
       {ss >> str1;}      // ss >> num1;
   cout << str1 << endl;
 }
```

Floating point precision

```
#include <iomanip>
cout << "\nThe average is: " << setprecision (4) << average << "\n\n";
```

Constants

```
  #include <iostream>
  #define PI 3.14;
int main ()
{
 const float PI = 3.14;
}
#include <cmath>
   area = M_PI * radius * radius;
```

String Concatenation

```
string st1 ("I am learning ");
     // st1 is being initialized to "I am learning"
```

cin reads only the first word of a string

"Yahya" + "Kemal" becomes "YahyaKemal"

Read and Write from Text Files

```
# include < fstream >
{
       ifstream fin ("numbers.in"); // open input file
       ofstream fout ("number.out"); // create and open output file

       int num1, num2;
       fin >> num1 >> num2; // read two integers from input file

         fout << "sum is: " << num1 + num2 << endl;

       fin.close (); // close the input file
       fout.close (); // close the output
 }
```

The Conditional Operator (? :)

```
cout << ((average > 60) ?"passed\n" : "failed\n");
```

The "switch" structure

```
switch (color)
{
 case 'r' :
    cout << "\nWAIT!\n\n";
    // break;
 case 'y' :
   cout << "\nGET READY!\n\n";
   break;
 case 'g' :
    cout << "\nGO!\n\n";
    break;
 default :
    cout << "\nWrong input!\n\n";
}
```

The 'while' Loop

```
while (i ≤ 10)
{
 cout << i << " ";
 i += 2;
}
 // while there is input to be read
while (fin >> mass >> quantity)
{
 totalMass += (mass * quantity);
}
```

```
int SENTINEL = 0;
while (studentMark != SENTINEL)
{
 cout << "Enter the mark from [1 – 5]: ";
 cin >> studentMark;
 totalMarks += studentMark;
}
while (binary > 0)
{
 decimal += ((binary % 10) * pow (2.0, i));
 binary /= 10;
 counter i++;
}

while (decimal > 0)
{
 binary += (decimal % 2) * pow (10.0, i);
 decimal /= 2;
 i++;
}
```

The 'do/while' Loop

```
do
{
 cout << i << " ";
   i++; /
}
while (i ≤ 10);
```

The 'for' Loop

```
for (int i = 1; i ≤ 10; i++)
{cout << i << " ";}
for (int i = 0; i < n; i++)
{
 cout << first << " ";
 swap (first, second);
 second += first;
}
```

The 'break' and 'continue' statements

```
while (1)
{
 cin >> letter;
 if (letter == '\n')
  break;
}
for (int i = 0; i < 20; i++)
{
 if ((i % 4) == 0)
  continue;
 cout << i << endl;
}
```

Nested Loops

```
for (int i = 1; i ≤ n; i++)
{
 for (int j = 1; j ≤ i; j++)
    {
      cout << j << " ";
    }
    cout << "\n";
}
int n;
bool isPrime = true;
cin >> n;

for (int i = 2; i < n; i++)
{
 for (int j = 2; j ≤ n; j++)
     {
      if ((i ≠ j) && (i % j == 0))
        {
         isPrime = false;
         break;
        }
     }
     if (isPrime)
      {
       cout << i << " ";
     }
      isPrime = true;
}
```

Generating random numbers

```
# include < time.h >
srand ((unsigned) time (NULL));
int number = rand () % 1001 + 1;
```

Functions

```
 // function prototype
int findSum (int, int);
 // funtion call
sum = findSum (a, b);
 // function definition
int findSum (int x, int y) {return x + y;}
```

Pass by Reference

```
 // function prototype
void function_B (int &, float &);
 // function call
function_B (x, y);
 // function definition
void function_B (int & a, float & b) {}
```

Static Arrays

```
int wrongArray[n];
int myArray[5] = {16, 2, 77, 40, 12071};
myArray[1] = 44;
for (int i = 0; i < 5; i++)
{
 cout << myArray[i] << " ";
}
int myArray[3][5] = {{1, 2, 3, 4, 5}, {6, 7, 8, 9, 10}, {11, 12, 13, 14, 15}};
myArray[0][1] = 44;
for (int i = 0; i < 3; i++)
{
 for (int j = 0; j < 5; j++)
   { cout << myArray[i][j] << ", ";}
    cout << "\n";
}
```

Dynamic Vectors

```
#include < vector >
  vector < int > myDynamicArray;
   myDynamicArray.resize (5);

myVector[0] = 3;
cout << myVector[myVector[0]] << endl;

for (i = n - 1; i ≥ 0; i--)
{cout << myVector[i] << " ";}

 // add a new last element
myVector.push_back (1);
     // add a new last element
myVector.push_back (2);

 // delete last element
myVector.pop_back ();

 // insert at second position number 2
myVector.insert (myVector.begin () + 1, 2);
 // insert from second position, three times number 4
myVector.insert (myVector.begin () + 1, 3, 4);
 // insert at third position, range from first until fourth
myVector.insert (myVector.begin () + 2, myVector.begin (), myVector.begin () + 4);

 // resize vector and fill empty positions with value
myVector.resize (7, -99);
 // assign to vector A, seven times value 10
vectorA.assign (7, 10);
```

```
// assign the vector element with index 3 to a reference variable
   int & num1 = myVector.at (3);
    // a change in reference will change the vector element also
       num1 += 10;


 // erase first element
myVector.erase (myVector.begin ());
 // erase last element
myVector.erase (myVector.end () - 1);

 // erase position element
myVector.erase (myVector.begin () + 2);

 // erase from first until third
myVector.erase (myVector.begin (), myVector.begin () + 3);

 // sort this vector
sort (v.begin (), v.end ());
 // sort in descending order
sort (v.begin (), v.end (), greater < int > ());


 // reverse vector
reverse (v.begin (), v.end ());


# include < vector >
 # include < algorithm >
vector < int > v;
vector < int > ::iterator it;
 // find "4" in the vector and save its address at iterator
it = find (v.begin (), v.end (), 4);
 // show the actual memory address where 4 is saved
cout << & it << endl;
if (it ≠ v.end ())
{
  // we can show the actual value of the address that iterator points to
    cout << Value "<<(*it)<<" was found at "<<it-v.begin()<<" ith position ! \n ";
     }
      else {cout<<" Value not found ! \n ";
}
 // check if value is present
if (binary_search (v.begin (), v.end (), 4))
{
 cout << "The value was found!\n";
}
else
{cout << "Not found!\n";}


-----------------------------------------
```

String Class

```
cout << "Enter a string: ";
getline (cin, s2);   // reads a line of text
```

```
string str;
cout << "Enter a string: ";
getline (cin, str, ''); // read until empty space
cout << str << endl;
```

```
# include < string >
  # include < sstream >  // convert string to a stream of characters
    # include < vector >
```

```
string str = "0x0002,A5651QPR87GBZ094RTF52,D,A,000001,ABC ,10000.00 , EOT";
string word;
vector < string > stringVector;

 // convert string as a stream of characters
stringstream stream (str);

 // separate characters with delimiter ","
while (getline (stream, word, ', '))
{
  // cout << word << "\n";
 stringVector.push_back (word);
}

 // show saved vector
for (int i = 0; i < stringVector.size (); i++)
{cout << stringVector[i] << "\n";}
```

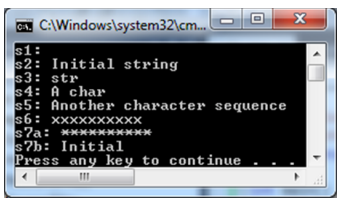
```
# include < iostream > # include < string > # include < sstream >
# include < vector >  using namespace std;

{string str = "1 2 3 4 5 6 7 8 9 10";
 string word;
 vector < string > stringVector;
 stringstream stream (str);
  while (getline (stream, word, ''))
    { stringVector.push_back (word);}
    int sum = 0;
 for (int i = 0; i < stringVector.size (); i++)
  {cout << stringVector[i] << " "; sum += atoi (stringVector[i].c_str ());}
   cout << "\nSum  = " << sum << endl;
```

- most of the vector class methods work with strings also.

Modifiers:

operator+=	Append to string (public member function)
append	Append to string (public member function)
push_back	Append character to string (public member function)
assign	Assign content to string (public member function)
insert	Insert into string (public member function)
erase	Erase characters from string (public member function)
replace	Replace part of string (public member function)
swap	Swap contents with another string (public member function)

```
string s0 ("Initial string");
string s1;
string s2 (s0);
string s3 (s0, 8, 3);
string s4 ("A character sequence", 6);
string s5 ("Another character sequence");
string s6 (10, 'x');
string s7a (10, 42);
string s7b (s0.begin (), s0.begin () + 7);
```



* string methods

Capacity:

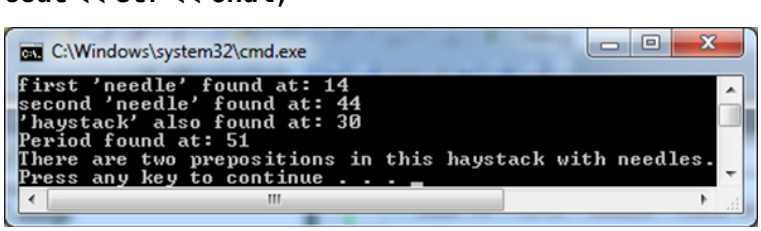
size	Return length of string (public member function)
length	Return length of string (public member function)
max_size	Return maximum size of string (public member function)
resize	Resize string (public member function)
capacity	Return size of allocated storage (public member function)
reserve	Request a change in capacity (public member function)
clear	Clear string (public member function)
empty	Test if string is empty (public member function)

Element access:

operator[]	Get character in string (public member function)
at	Get character in string (public member function)

```
// add two more positions and populate them with character
str.resize (sz + 2, ' + ');

// treate a string as an array
string str ("Test string");
int i;
for (i = 0; i < str.length (); i++)
{cout << str[i] << "";}

// return character at position of string
string str ("Test string");
for (size_t i = 0; i < str.length (); i++)
{cout << str.at (i) << endl;}

// assigns new content to the string replacing its current content.
string str;
string base = "The quick brown fox jumps over a lazy dog.";

str.assign (base);
cout << str << endl;
str.assign (base, 10, 9);
cout << str << endl; // "brown fox"
str.assign ("pangrams are cool", 7);
cout << str << endl; // "pangram"
str.assign ("c-string");
cout << str << endl; // "c-string"
str.assign (10, '*');
cout << str << endl; // "**********"
```

```
// string content is extended by inserting some additional content at a specific location
string str = "to be question";
string str2 = "the ";
string str3 = "or not to be";
str.insert (6, str2); // to be (the) question
str.insert (6, str3, 3, 4); // to be (not) the question
str.insert (10, "that is cool", 8); // to be not (that is) the question
str.insert (10, "to be "); // to be not (to be) that is the question
str.insert (15, 1, ' : '); // to be not to be ( : ) that is the question
str.insert (str.end (), 3, '.'); // to be not to be : that is the question (...)


// erases a part of the string content, shortening the length of the string.
string str ("This is an example phrase.");
string::iterator it;

str.erase (10, 8);
cout << str << endl; // "This is an phrase."

it = str.begin () + 9;
str.erase (it);
cout << str << endl; // "This is a phrase."

str.erase (str.begin () + 5, str.end () - 7);
cout << str << endl; // "This phrase."

cout << str << endl;


// search the string for the content specified,
and returns the position of the first occurrence in the string.
string str ("There are two needles in this haystack with needles.");
string str2 ("needle");
size_t found;

found = str.find (str2);
if (found ≠ string::npos)
cout << "first 'needle' found at: " << int (found) << endl;

found = str.find ("needles are small", found + 1, 6);
if (found ≠ string::npos)
cout << "second 'needle' found at: " << int (found) << endl;

found = str.find ("haystack");
if (found ≠ string::npos)
cout << "'haystack' also found at: " << int (found) << endl;

found = str.find ('.');
if (found ≠ string::npos)
cout << "Period found at: " << int (found) << endl;

str.replace (str.find (str2), str2.length (), "preposition");
cout << str << endl;
```



```
// string compare
string str1 ("green apple");
string str2 ("red apple");

if (str1.compare (str2) ≠ 0)
cout << str1 << " is not " << str2 << "\n";

if (str1.compare (6, 5, "apple") == 0)
cout << "still, " << str1 << " is an apple\n";

if (str2.compare (str2.size () - 5, 5, "apple") == 0)
cout << "and " << str2 << " is also an apple\n";

if (str1.compare (6, 5, str2, 4, 5) == 0)
cout << "therefore, both are apples\n";
```