Include Files in C++

`#include` is a C++ "preprocessor" directive to read in the content of a text file provided as argument

```
#include  "xyz.h" // reads in the content of the file xyz.h
                  // starting at this point
```

To be able to use a class (e.g. creating object of class type) you need to include the class' interface file (file with a .h extension) with `#include`

there are three types of include files - c++ standard
C standard , and others

Include Files in C++

the c++ standard include files

`<iostream>, <iomanip>, <fstream>`

`- stream input/output`

`<vector>, <list>, <deque>, <queue>, <stack>, <map>, <set>, <bitset>, <functional>, <memory>, <iterator>, <algorithm> - stl includes`

`<exception>, <stdexcept> - exceptions`

`<string>, <sstream> - strings`

`<locale>, <limits>, <typeinfo> misc.`

Typical C++ programs

Interface files (also known as header files):
used to define interfaces of classes (usually the filename has a .h extencion)

Implementation files (also known as source files):
used to implement member functions (i.e., defnitions of definition of member functions) of classes.

Client programs:
application-oriented programs, using `#include` preprocessor directive to include the contents of the header and gaining access to the class definitions. A `main()` function, the start point of the program, is defined in the client program.

A "dice" class

```
#ifndef _DICE_H
#define _DICE_H

//  class for simulating a die (object "rolled" to generate
//  a random number)
//  File name: dice.h
//
```

```
//  Dice(int sides) -- constructor, sides specifies number of
//                     "sides" for the die, e.g., 2 is a coin,
//                     6 is a 'regular'   die
//
//  int Roll() -- returns the random "roll" of the die, a
//                uniformly distributed random number between
//                1 and # of sides
//
//  int NumSides() -- access function, returns # of sides
//
//  int NumRolls() -- access function, returns # of times
//                    Roll called for an instance of the
//                    class
```

"dice" class interface

```
#include "rando.h"            // for random number generator

class Dice{
  public:
    Dice(int sides);           // constructor
    int Roll();                // return the random roll
    int NumSides();            // how many sides this die has
    int NumRolls();            // # times this die rolled
  private:
    RandGen myGenerator;       // random number generator
    int myRollCount;           // # times die rolled
    int mySides;               // # sides on die
};

#endif    /* _DICE_H not defined */
```

```
#include "dice.h"
// Implementation of dice class
// File name: dice.cc
Dice::Dice(int sides)
// precondition: sides is nonnegative
// postcondition: all private fields initialized
{
   myRollCount = 0;
   mySides = sides;
}
int Dice::Roll()
// precondition: none
// postcondition: number of rolls updated;  random 'die' roll returned
{
   myRollCount = myRollCount + 1; // update # of times die rolled
   return myGenerator.RandInt(1,mySides); // // in range [1..mySides]
}
```

```
int Dice::NumSides()
// precondition: none
// postcondition: return # of sides of die
{
   return mySides; }
```

```
int Dice::NumRolls()
// precondition: none
// postcondition: return # of times die has been rolled
{
     return myRollCount;
   }
```

Using the dice class

```
// A test program (client program) for testing the dice class
//  implementation
// File: testDice.cc
#include <iostream>
#include “dice.h”
using namespace std;
void main()
{
     Dice d(3) // we will be using 3 dice
     int value, numOfOnes, numOfTwos, numOfThrees = 0;
     //Out of 100 rolls of 3 dice how many time will we roll
     // 1’s, 2’s, and 3’s?
```

Using the dice class

```
  for (int i = 1; i <= 100; i++){
           value = d.roll();
           if (value == 1)
               numOfOnes++;
           else if (value == 2)
                numOfTwos++
           else numOfThrees++
  }
   cout<<”Total number of times we rolled 1’s, 2’s, and 3’s are”;
   cout<< numOfOnes << “ 1s<< numOfTwos <<“ 2s” <<numOfThrees<< “ 3s”;
}
```