Coding Convention in C++

1. Comments

1.1 Every file that contains source code must be documented with an introductory comment that provides information on the file name and its contents.

1.2 Write some descriptive comments before every class and function.

1.3 Use // for comments.

1.4 Examples

1.4.1 Comment convention for classes

```
  // Class : Test
 // Description: This is a test class.
 //   ……..
 //   ……..
 // Created: 2007/5/10 12:00 pm
 // Author: Gildong Hong
 // mail: gdhong@cs.hongik.ac.kr
 //
 // Revsions :
 //     1. When & Who : 2007/05/15 13:12 pm by Ji-sung Park
 //         What : added bFlag,
 //                  modified calCosts
 //                    …..
 //     2. …….
 //
 //
```

1.4.2 Comment convention for operations

```
  // Function : int calCosts(int prevCost, int addedCost)
 // Description: This is a function that calculates the total costs by …
 //   ……..
 //   ……..
```

```
// Parameters :     int prevCost – the previous cost
 //                  int added Cost – newly added cost
// Return Value :   total cost value
//
// Created: 2007/5/10 12:00 pm
// Author: Gildong Hong
//
// Revsions :
//     1. When & Who : 2007/05/16 13:12 pm by Ji-sung Park
//        What : added other factors when calculating costs…
//                  …..
//     2. …….
//
//
```

2. Assigning Names

2.1 The names of variables, constants, and functions are to begin with a lowercase letter.

2.2 The names of abstract data types (class), structures, typedefs, and enumerated types are to begin with an uppercase letter.

2.3 In names which consist of more than one word, the words are written together and each word that follows the first is begun with an uppercase letter.

2.4 Do not use identifiers which begin with one or two underscores (`_' or `__').

2.5 Examples

2.5.1 Choice of names

```
int groupID; // instead of grpID
int nameLength; // instead of namLn
PrinterStatus resetPrinter; // instead of rstprt
```

2.5.2 Ambiguous names

```
void termProcess(); // Terminate process or terminal process?
```

2.5.3 Names having numeric characters can cause errors which are difficult to locate.

```
int l0 = 13; // Names with digits can be
int lO = l0; // difficult to read.
```

3. Classes

3.1 No member functions are to be defined within the class definition.

3.2 Never specify public or protected member data in a class.

3.3 Examples : No definitions of member functions within the class definition

```
// Instead of writing like this:
class String
{
    public:
    int length() const // No !!
    {
          return len;
    }
    // ...
    private:
        int len;
};

// Do it this way:
class String
{
    public:
    int length() const;
    // ...
   private:
     int len;
};
```