7 Program Style

- A program written with attention to style
  - is easier to read
  - easier to correct
  - easier to change

7.1 Indenting

- Items considered a group should look like a group
  - Skip lines between logical groups of statements
  - Indent statements within statements

```
if (x == 0)
        statement;
```

- Braces {} create groups
  - Indent within braces to make the group clear
  - Braces placed on separate lines are easier to locate
  - IDE helps to find the corresponding open and close braces.

```
#include <iostream>

int main(int argc, const char * argv[])
{
    // cout is in the namespace 'std'
    std::cout << "Hello, World!\n";
    std::cout << "Hello, World!" << std::endl;

    { // a scope begins
        using namespace std;
        cout << "Hello, World!" << endl;
    } // a scope ends
    // 'using namespace std' is no longer good

    return 0;
}
```

7.2 Comments

- // is the symbol for a single line comment
  - Comments are explanatory notes for the programmer
  - All text on the line following // is ignored by the compiler
  - Example:

```
//calculate regular wages
gross_pay = rate * hours;
```

- /* and */ enclose multiple line comments
  - Example:

```
/* This is a comment that spans
multiple lines without a comment
symbol on the middle line */
```

7.3 Constants

- Number constants have no mnemonic value
- Number constants used throughout a program are difficult to find and change when needed
- Constants
  - Allow us to name number constants so they have meaning
  - Allow us to change all occurrences simply by changing the value of the constant

7.3.1 const

- const is the keyword to declare a constant
- Example:
```
const int WINDOW_COUNT = 10;
```
declares a constant named WINDOW_COUNT
  - Its value cannot be changed by the program like a variable
  - It is common to name constants with all capitals

7.3.2 macros

- Macro can define constants.
- Example:
```
#define WINDOW_COUNT 10
```
  - Pre-processor replace the string "WINDOW_COUNT" with 10 before compelling it.
    So be careful if you write arithmetic in macro
```
#include <iostream>

#define WINDOW_COUNT 10 + 20
#define WINDOW_PRICE 10.0

int main(int argc, const char * argv[]) {
    std::cout << "Cost = " << WINDOW_PRICE * WINDOW_COUNT << std::endl;
    return 0;
}
```
- Some Macros defined by systems's include file,
- Example: M_PI, M_PI_2, M_PI_4, M_E, etc....
```
#include <cmath>
:
:
        double rad = angle * M_PI / 180;
```
  - Note that M_PI etc. are not in Standard C++, but those are defined because those are taken over from C. For some system(VC++), you have to do:

```
#define _USE_MATH_DEFINES
#include <cmath>
```