

CMPS 150 Style Sheet (2003.10.09)

1. Constant Identifiers

- a. all upper case letters
- b. start with letter
- c. _ (underscore) used between words

examples: MAXIMUM, FIRST_DAY, MAX_LOAD

2. Variable Identifiers

- a. start with lower case letter
- b. each word after the first word starts with upper case letter
- c. body of words in lower case letters

examples: firstName, lastName, dateOfBirth

3. Function and Data Type Identifiers

- a. start with upper case letter
- b. each word begins with an upper case letter
- c. body of words in lower case letters

examples: NameListType, SumWeights, ReadCard

4. Indentation

- a. block (compound statement) markers ({ } - curly braces) are not indented
- b. all code inside a function body or compound statement (such as in a function, loop, if, else, switch, struct declaration, union declaration or class declaration) is indented 4 spaces
- c. all conditionally executed code is indented four spaces
- d. **exception** - *nested if-else statements are not indented but the conditionally executed statements internal to it are indented.*
- e. switch statements – each case should be indented four spaces; the first statement after a case colon (case # :) should begin after the colon and subsequent statements in a case should line up with the first (subject to normal indentation rules)

example of indention in function:

```
const int YEAR = 1984;

int main( )
{
    int currentYear;
    cout << "Enter Current Year: ";
    cin >> currentYear;
    if (currentYear - YEAR > 16)
```

```
        cout << "It must be at least 2001!";
    else
        cout << "Are you sure about the year?";
}
```

example of a nested-if-else:

```
if (weight > 100)
    cost = weight * .90;
else if (weight > 80)
    cost = weight * .94;
else if (weight > 60)
    cost = weight * .98;
else if (weight > 40)
    cost = weight * .99;
else
    cost = weight * 1.0;
```

example of a switch:

```
switch((int) (gpa+.05))
{
    case 0:  cout << "F";
            break;
    case 1:  cout << "D";
            break;
    case 2:  cout << "C";
            break;
    case 3:  cout << "B";
            break;
    case 4:  cout << "A";
            break;
    default: cout << "Invalid!";
}
```

5. Blank Lines

- a. each function is preceded by a blank line
- b. each section of code is preceded by a blank line

example:

```
const float PI = 3.14;

int main( )
{
    float a;
    float b;
    float c;

    cout << "Enter a value for side a: ";
    cin >> a;
    cout << "Enter a value for side b: ";
    cin >> b;
```

```
cout << "Enter a value for side c: ";
cin >> c;
```

6. Long Lines

When entering long lines of code (more than 75 characters), you must find a good “breaking point” on the line, and continue the statement on the next line. A line of code should never be longer than the paper is wide.

7. Documentation

- a. file header documentation as follows:

```
// *****
// Author      : --Your Name--
// CLID        : --Your LoginID--
// Class       : --CMPS 150 section x--
// Assignment  : --pa x--
// Date Assigned : --date assignment given--
// Due Date    : --due date--
// Due Time    : --due time--
//
// Program Description: --description of actions of
//                               program--
//
// --Certificate of Authenticity--
//
// *****
```

- b. each function or code section preceded by an inline comment

```
// *****
// ***** collect package information *****
// *****

cout << "Enter package width: ";
cin >> width;
cout << "enter package length: ";
cin >> length;
cout << "Enger package depth: ";
cin >> depth;
```

- c. declarations and other important code should be followed by a sidebar comment

example:

```
float width,
      length,
      depth,
```

```
weight; // package information variables
```