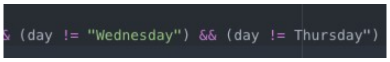
# Style Guide (for early CPSC 120)

Based on Kevin Wortman's October 7, 2019 "Style Guide (for early CPSC 120)"

| # | Guideline | Good Example | Bad Example |
|---|---|---|---|
| **Variables** | | | |
| 1 | Names use snake_case (lower-case letters and underscores) | `int input;`<br>`string user_name;` | `int Input;`<br>`string UserName;` |
| 2 | Names accurately describe the meaning of their contents (not data type) | `int player_guess;`<br>`double total_fines;` | `int x;`<br>`int number;`<br>`int first;`<br>`double second;`<br>`string a_string;` |
| 3 | Use the simplest data type that is appropriate; int for whole numbers, double for fractional numbers, string for text, bool for true/false | `int days_late = 3;`<br>`double tax_rate = .0875;`<br>`string name = "Ada";`<br>`bool is_valid = true;` | `double days_late = 3;`<br>`string days_late = "3";`<br>`int tax_rate = .0875;`<br>`int is_valid = true;` |
| 4 | Declare constant parameters `const`, and in ALL_CAPS | `const double PI = 3.14159265358979;`<br>`const int DAYS_PER_WEEK = 7;` | (doesn't declare constants) |
| 5 | Do not use global variables, except for constants | `int main(int argc, char** argv) {`<br>`  int input;`<br>`  ...` | `int input;`<br>`int main(int argc, char** argv) {`<br>`  ...` |
| 6 | Delete variables that are never used | `int x;`<br>`… (y is never used)` | `int x, y;`<br>`… (y is never used)` |
| **Comments** | | | |
| 7 | The first line of your program should be a comment with your name and lecture section number | `// Ada Lovelace, 120-05` | (no such comment) |
| 8 | Delete TODO comments after | `int score;` | `// TODO declare a variable for score` |

| | | | |
|---|---|---|---|
| | you’ve done what they talk about | | int score; |
| 9 | Add a descriptive comment before any line of code that is not self-explanatory | // generate a random letter A-Z<br>char letter = ‘A’ + (rand() % 26); | char letter = ‘A’ + (rand() % 26); |
| 10 | Delete commented-out code | (no commented-out code)<br>srand(time(NULL)); | (obsolete commented-out code)<br>//srand(0);<br>//srand(NULL);<br>srand(time(NULL)); |
| **Whitespace and Braces** | | | |
| 11 | Put each statement on its own line | int input;<br>cin >> input; | int input; cin >> input; |
| 12 | Indent the contents of { } scopes by 2 spaces | if (n > 0) {<br>  cout << “positive”; | if (n > 0) {<br>cout << “positive”; |
| 13 | Always indent exactly 2 spaces, so nested scopes align vertically | if (n > 0) {<br>  cout << “positive”;<br>  if ((n % 2) == 0) {<br>    cout << “and even”;<br>    bool was_even = true; | if (n > 0) {<br>    cout << “positive”;<br> if ((n % 2) == 0) {<br>       cout << “and even”;<br>  bool was_even = true; |
| 14 | Indent a close-brace } the same amount as the code with the corresponding open-brace { |   if (n == 0) {<br>    cout << “zero”;<br>  } |   if (n == 0) {<br>    cout << “zero”;<br>} |
| 15 | Use 1TBS-style braces, with an open-brace on the same line as the corresponding if statement or main declaration | if (n == 0) {<br>  cout << “zero”;<br>} | if (n == 0)<br>{<br>  cout << “zero”;<br>} |
| 16 | Put exactly one space between an open brace { and the close-paren ) before it | if (n == 0) { | if (n == 0){<br>or<br>if (n == 0)     { |
| 1 | Separate logical | int choice;<br>cout << “enter choice: | int choice;<br>cout << “enter choice: |

| | | | |
|---|---|---|---|
| 7 | sections of code with a blank line | “;<br>cin >> choice;<br><br>if (choice == 0) {<br>  pay = 0;<br>else {<br>  pay *= 2;<br>}<br><br>cout << “pay = “ << pay; | “;<br>cin >> choice;<br>if (choice == 0) {<br>  pay = 0;<br>else {<br>  pay *= 2;<br>}<br>cout << “pay = “ << pay; |
| 18 | Limit lines to 80 columns wide (the vertical line shown in Atom) | if ((day != "Monday") &&<br>    (day != "Tuesday") &&<br>    (day != "Wednesday") &&<br>    (day != "Thursday") &&<br>    (day != "Friday")) {| <br>& (day != "Wednesday") && (day != Thursday")<br>(all on one line past the vertical 80-column mark) |
| **Expressions** | | | |
| 19 | In complicated expressions, use parentheses to clarify the order of operations | if (((n == 1) \|\| (n == 2)) && (k > 0) {<br>... | if (n == 1 \|\| n == 2 && k > 0) {<br>... |
| 20 | Never use the assignment operator = inside an expression | if (count == 0) { | if (count = 0) { |
| 21 | Never use the comma operator , inside an expression | if ((n == 0) \|\| (n == 1)) { | if (n == 0, 1) { |
| **if statements** | | | |
| 22 | Always enclose the body of an if/else with braces { } | if (n == 0) {<br>  cout << “zero”;<br>} | if (n == 0)<br>  cout << “zero”; |
| 23 | Do not write redundant logic in an if/else if; use an if/else instead | if (days <= 0) {<br>  cout << “negative”;<br>} else {<br>  cout << “positive“;<br>} | if (days <= 0) {<br>  cout << “negative”;<br>} else if (days > 0) {<br>  cout << “positive“;<br>} |

| **Command-Line Arguments** | | | |
|---|---|---|---|
| 24 | Assign argv[1] etc. into a variable with a meaningful name | `string name = argv[1];`<br>`cout << “Your name:” << name;` | `cout << “Your name:” << argv[1];` |

Thanks for Michael Shafae for putting this together.