Good Programming Practice

Many programmers make last character printed by a function a **\n** or **endl** to insert a new line. This ensures that the function will leave the screen cursor positioned at the beginning of a new line. Conventions of this nature encourage software reusability – a key goal in software development environments.

Indent the entire body of each function one level of indentation within the braces that defines the body of the function. This makes the functional structure of a program stand out and helps make programmers easy to read.

Some programmers prefer to declare each variable on a separate line. This format allows for easy insertion of a descriptive comment next to each declaration.

Place a space after each comma (,) to make programs more readable.

Choosing meaningful variable names helps a program to be "Self-Documenting," i.e. it becomes easier to understand the program simply by reading it rather than having to read manuals or use excessive comments.

Avoid identifiers that begin with underscore and double underscore because a C++ compiler may use names like that for its own purpose internally. This will prevent names you choose from being confused with the names the compiler chooses.

Always place a blank line before a declaration that appears between executable statements. This makes the declaration stand out in the program and contributes to program clarity.

If you prefer to place declarations at the beginning of a function, separate those declarations from the executable statements in that function with one blank line to highlight where the declarations end and the executable statements begin.

Place spaces on either side of a binary operator. This makes the operator stand out and makes the program more readable.

A lengthy statement may be spread over several lines. If a single statement must be spilt across lines, choose breaking points that make sense such as after comma in a comma separated list or often an operator like insertion or extraction operator in C++ in a lengthy expression. If a statement is spilt across two or more lines, indent all subsequent lines.

Refer to the operator precedence chart when writing expression containing many operators. Confirm that the operators in the expression are performed in the order you expect. If you are uncertain about the order of evaluation in a complex expression, use parenthesis to force the order of evaluation, exactly you would do in an algebraic expression. Be sure to observe that some operators such as assignment operators (=) associate right to left rather than left to right.