

Format for Programming Projects

Items to submit:

- Report (turn in by hand during class)
- Submit zipped computer files in Blackboard. See the document entitled **Submitting a Project Using Blackboard**. Zipped files should include:
 - Copy of the project with all related files (including any data files)
 - Copy of the executable file (so that the instructor doesn't need to build your project to test it)
 - Copy of your report is optional

Report Format:

The report should contain the following sections:

- Title page
- Printouts
 - Main program, including all functions
 - Data files (if used)
 - Other files (such as Excel tables and graphs)
- Test Results
 - Results for any required test cases
 - Results for any extra credit features
 - Check results by hand when possible (not to be turned in - just for your benefit)
- Discussion – include three parts (label each)
 - Program performance – describe how the program performs and discuss any possible limitations
 - Extra credit features – discuss any extra credit features
 - Potential improvements – discuss any possible improvements that could be made to your program

Late Programs:

10 points per week is deducted for late program. Note that programs that do not work correctly will generally not receive a passing grade, so it is better to turn in a late working program than to turn in a non-working program on time.

Program Formatting Requirements:

- The main program should begin with:
 - TCC logo
 - Brief description of the program (1 to 3 lines)
 - List of inputs and outputs (in general, not one line for each variable)
- Each function should begin with:
 - Brief description of the program (1 to 3 lines)
 - List of inputs and outputs (in general, not one line for each variable)
- Declaring variables
 - Declare variables at the top of each function (including main)
 - Use clear variable names or provide comments with definitions of each variable
- Indentation – use indentation for all structures to make programs more readable

- Order of functions. Use the following ordering for your project:
 - Prototypes – list all
 - Main function
 - Other functions
- Functions
 - In general functions should be designed for wide application and maximum reusability.
- Global variables
 - Avoid using global variables in most cases.
 - Global variables may be used for constants, such as pi.
 - Inputs and outputs for functions should be passed as arguments, not using global variables.
- Comments
 - The key is to make the program readable to others
 - Comments should explain function, not syntax.
Example (poor comment): **Y++; // Add 1 to Y**
Example (good comment): **Y++; // Increment Y if leap year occurs**
 - Include comments as key structures are entered
Example: **do // loop to allow user to re-enter time if incorrect**
{
Statements(s);
 - Comments can be omitted when outputs explain the function.
Example (unnecessary comment):
cout << "Enter the weight of the object in lbf";
cin >> Weight; // Read the weight of the object in lbf (unnecessary comment)
- User-friendly programs. Programs should be clear and easy for the user to follow.
- Formatted output. Any outputs should be clear and well organized.

Cheating:

Programming projects must be your own work. You can ask other students and the instructor questions, but your work should be essentially your own. Programs vary tremendously so cheating is often easy to spot. If the instructor decides that two programs look too similar and that cheating was involved, both parties will receive a grade of 0 for the project.