Southern New Hampshire University

# IT 312 Programming Guidelines and Rubric

**Overview:** In this course, you will be responsible for completing a number of programming-based assignments. Learning to program in C++ requires developing an understanding of general programming concepts and learning the syntax of the C++ programming language. These programming exercises will build on each other and help you cultivate you programming knowledge. It is recommended that students do not limit their practice to just that which is graded. The more you write your own code, the more proficient you will become with the tools and techniques available to you in C++.

**Prompt**: Your submission for this assignment should include your source code, complete with a brief description of the file in a commented "header" section.

The following **critical elements** should be addressed in your project submission:

- **Code Description**: Give a brief explanation of the code and a brief discussion of any issues that you encountered while completing the exercise.

- **Functioning Code**: A source code must meet its specifications and behave as desired. To develop proper code, you should **produce fully functioning code** (produces no errors) that aligns with accompanying annotations. You should write your code in such a way that the submitted file actually executes, even if it does not produce the correct output. You will be given credit for partially correct output that can actually be viewed and seen to be partially correct.

- **Code Results:** Properly generated results establish that your source code does the following:
    A. Generates accurate output.
    B. Produces results are streamlined, efficient, and error-free.

- **Annotation/Documentation**: All code should also be well commented. This is a practiced art that requires striking a balance between commenting everything, which adds a great deal of unneeded noise to the code, and commenting nothing. Well-annotated code requires you to:
    A. Explain the purpose of lines or sections of your code detailing the approach and method the programmer took to achieve a specific task in the code.
    B. Document any section of code that is producing errors or incorrect results.

- **Style and Structure**: Part of the lesson to be learned in this course is how to write code that is clearly readable and formatted in an organized manner. To achieve this, you should:
    A. Develop logically organized code that can be modified and maintained
    B. Utilize proper syntax, style, and language conventions/best practices

**Guidelines for Submission:** For each exercise, include the source code file containing code should start with a header comment containing a title (name, course, date, project number) and a discussion of the code.

Southern New Hampshire University

| Critical Elements | Exemplary (100%) | Proficient (85%) | Needs Improvement (55%) | Not Evident (0%) | Value |
|---|---|---|---|---|---|
| **Code Description** | Meets "Proficient" criteria and the explanation is provided and includes elements of scholarly research | A complete explanation of the code is provided and difficulties are discussed | The code description is incomplete | No explanation is provided | 10 |
| **Functioning Code** | Meets "Proficient" criteria and aligns with the accompanying annotated code | Program is fully functioning and includes code to meet all specifications | Program is not fully-functioning or does not include all of the specifications of the given problem | Program does not run or significant details of the specifications are violated or omitted | 20 |
| **Code Results: Accurate Output** | Meets "Proficient" criteria and the code is capable of producing accurate results beyond the specifications of the given problem | Data results are accurate in regard to the given problem | Code produces incorrect results for the given problem | Code does not produce results for the given problem | 15 |
| **Code Results: Efficiency** | Meets "Proficient" criteria and includes sophisticated techniques such as error handling or reference to user created functions | Written source code generally uses the most appropriate statements, functions, and/or methods, and contains minimal extraneous elements, steps, and/or procedures | Written code is minimally inefficient (i.e., multiple minor occurrences of convoluted code or alternative code element would achieve results in a simpler manner) | Code significantly inefficient with multiple examples of overly complex or improper approaches to achieve results | 20 |
| **Annotations / Documentation: Explanation of Purpose** | Meets "Proficient" criteria and clarity of annotations facilitates code navigation for a varied audience, and is written in a concise manner | Code annotations fully explain the code and facilitate navigation of the code | Comments provide little assistance with understanding the code. Code annotations do not fully explain the code or do not facilitate navigation of your code | Code is not fully or logically annotated | 5 |
| **Annotations / Documentation: Errors** | Meets "Proficient" criteria and sections that might have preferable or alternative solutions are clearly annotated | Sections of code that produce errors are clearly annotated and include a summary of the issue | Sections of code producing errors or incorrect results are identified, but no further explanation of the issues is provided | Sections producing errors or incorrect results are not documented | 5 |
| **Style and Structure: Logically Organized Code** | Meets "Proficient" criteria and the script is stylistically well-designed. The code is well-organized and presented in a way that it can be modified and maintained | The code is logically organized | The code contains portions that are not logically organized | Code is poorly organized or very difficult to read | 5 |

| | | | | | |
|---|---|---|---|---|---|
| **Style and Structure: Syntax** | Meets “Proficient” criteria and demonstrates deliberate attention to spacing, whitespace, and variable naming | Code follows proper syntax and conventions | Code contains variations from established syntax and conventions | Contains significant variations from established syntax and conventions | 10 |
| **Style and Structure: Best Practices** | Meets “Proficient” criteria and demonstrates an understanding of why certain techniques are considered “best practice” | Best practices were used in designing and writing the code, but some errors are present | Best practices were used in designing and writing the code, but many errors are present | Best practices were not used in designing and writing the code | 10 |
| | | | | **Total** | **100%** |