COS214 Practical Assignment 5 Team Standards

Members:

- **Aiden Govender - u22520458**
- **Mishka Dukhanti - u22617541**
- **Rethakgetse Manaka – u22491032**
- **Victor Igbojinna - u22492616**
- **Kamogelo Moeketse - u22623478**

Intro to Coding Standards Document

Code standards are rules, techniques, and guidelines to create cleaner, better readable, and more efficient code with minimal bugs and errors. Remember that coding standards are not personal opinions; they are concrete rules determining your code's programming style, procedures, and methods. Coding standards make sure all developers follow specified guidelines. As a result, every developer – even newcomers – can easily understand, debug, and maintain the code.

Coding Standards are necessary.

- They reduce security concerns and performance issues that might have resulted from poor coding practices.
- They help guarantee code quality, making your code easier to read, analyse, and work through. The code also becomes easier to maintain and extend.
- They lead to lower code complexity and more elegant design solutions.
- Any developer can examine any part of the code, understand it, and change it independently of when and who wrote it.

Advantages of Coding Standards

- **Enhancement of the efficiency of the software development process**: developers spend a large part of their time fixing code quality issues that could have been avoided. Implementing coding standards will help your team detect the problems early on or prevent them entirely.
- **Reduction in the code complexity and the number of bugs**: if your codebase is unnecessarily complex, the chances of being vulnerable to errors and bugs are higher. Coding standards help you develop less complex software and, therefore, reduce errors.
- **Improvement of the bug fixing process**: with coding standards, it becomes easier for developers to locate and correct bugs in the source code because it is written in a consistent manner.
- **Improvement of the code maintenance process**: if developers are following coding standards, the code is consistent and can be easily maintained. Anyone can understand it and modify it easily at any time.
- **Reduction of development cost and time by enabling reusability**: A clear code allows the developers to reuse the code whenever required. Reusability can radically reduce the cost and time of the development process.

- **Better team integration**: onboarding new developers become painless because the code is uniform and effortless to understand. New team members can quickly comprehend the codebase, and they know how to contribute to it in the same consistent way.

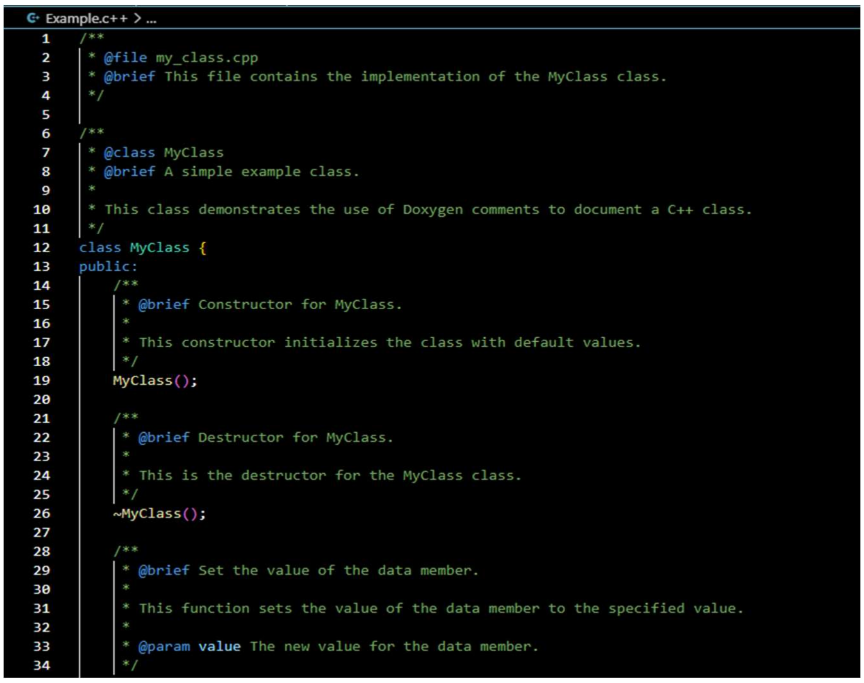
Our Coding Standards:

1. **Naming Standards**
   a) Use descriptive and meaningful names for variables, functions, classes, and objects.
   b) Use camelCase for variable and function names(e.g., ‘myVariable’, ’calculateValue()’).
   c) Use PascalCase for class and type names(e.g., ‘MyClass’, ‘MyStruct’).
   d) Use UPPERCASE for constants (e.g. ‘MAX_VALUE’).
   e) Prefix member variables with ‘m_’(e.g., ‘m_memberVariable’).

All of these naming standards make it easier to read and understand your code. It seems like a small thing but trust it will go a long way in understanding your code when another person reads it.

2. **Format Standards**
   a) Consistently use curly brackets({ }) to define code blocks, even for single statements.
   b) Use consistent and meaningful comments for documentation. Comments need to be doxygen friendly comments. An example will be posted below.
   c) Keep lines under 80 – 120 characters in length.
   d) Try to keep file sizes as little as possible. (i.e. 500 +- lines)

All of these format standards make it easier to read your code. Some of these rules are guidelines, but following these standards will be very helpful when it comes to compiling code. These format standard will help us maintain a good codebase as well as make debugging a little easier.

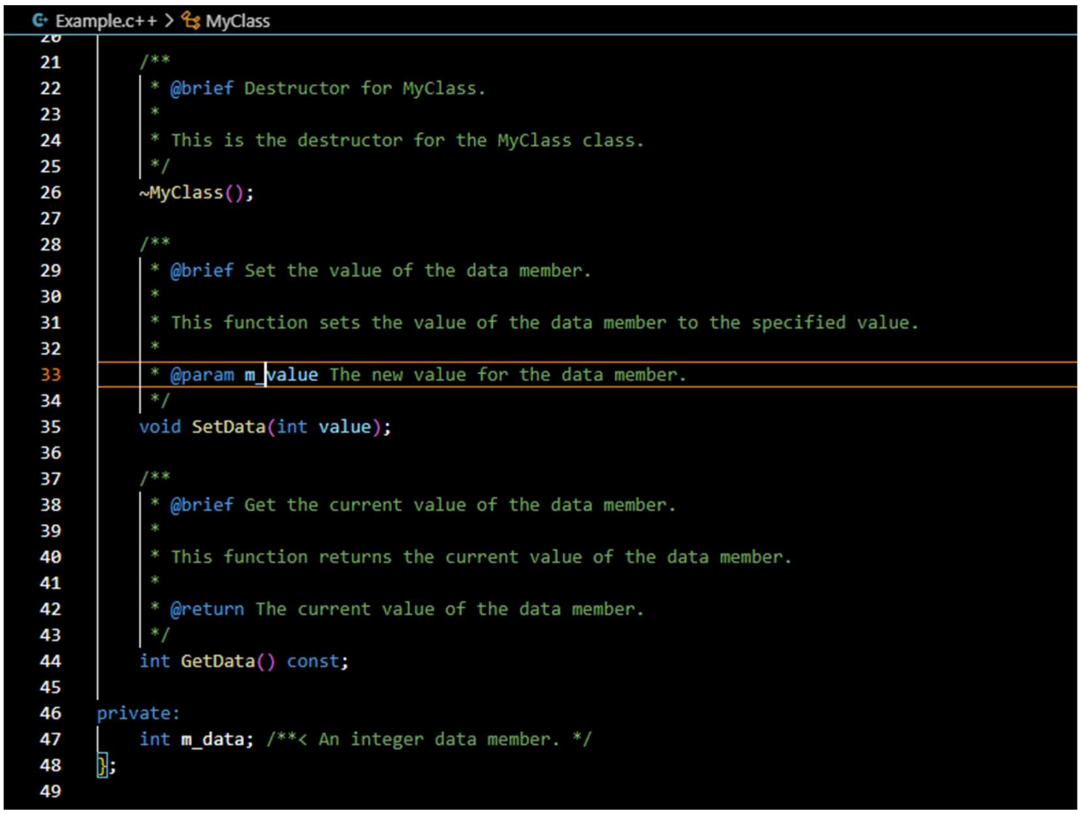


```cpp
/**
 * @file my_class.cpp
 * @brief This file contains the implementation of the MyClass class.
 */

/**
 * @class MyClass
 * @brief A simple example class.
 *
 * This class demonstrates the use of Doxygen comments to document a C++ class.
 */
class MyClass {
public:
    /**
     * @brief Constructor for MyClass.
     *
     * This constructor initializes the class with default values.
     */
    MyClass();

    /**
     * @brief Destructor for MyClass.
     *
     * This is the destructor for the MyClass class.
     */
    ~MyClass();

    /**
     * @brief Set the value of the data member.
     *
     * This function sets the value of the data member to the specified value.
     *
     * @param value The new value for the data member.
     */
```



```cpp
    /**
     * @brief Destructor for MyClass.
     *
     * This is the destructor for the MyClass class.
     */
    ~MyClass();

    /**
     * @brief Set the value of the data member.
     *
     * This function sets the value of the data member to the specified value.
     *
     * @param m_value The new value for the data member.
     */
    void SetData(int value);

    /**
     * @brief Get the current value of the data member.
     *
     * This function returns the current value of the data member.
     *
     * @return The current value of the data member.
     */
    int GetData() const;

private:
    int m_data; /**< An integer data member. */
};
```

3. **Header Files and Inclusion Guards**
   a) Use inclusion guards in header files to prevent multiple inclusions (e.g., '#pragma_once' or '#ifndef HEADER_H').
   b) Include what is only necessary in header files. Minimize dependencies. Include files only that file will need to use.

These standards will make it easy when it comes to compiling your code. It will reduce linking errors such as undefined reference and etc.

4. **Classes and Objects**
   a) Keep class definitions in separate header and source files
   b) Try to keep inheritance to a minimum. This will help reduce coupling.
   c) Don't use any recursive functions
   d) Use normal C++ features (i.e. use a while instead of a do while)

These standards will make it easier to structure code in a uniform manner as well as allow unit testing and other testing procedures to happen.

5. **Data Structures**
   a) Where possible try using data structures already provided by C++. This is not 132 or 110. (.e.g., STL libraries)

This will help us by actually allowing us to spend time on the important development as well as minimize bugs in our code.

6. **Testing**
   a) Each member will have to set up unit tests for the section of code they coded/worked on. There are exceptions to the rule of course. You can use Google Test, CTest and other available testing frameworks.
   b) Where possible practice test-driven development.

These standards will make it easier for you to verify that your code is working and bug-free, as well as make it easier to integrate it to other portions of the project. Test-driven development helps you code in such a manner that you anticipate bugs before you code thus making you code more robust code. Research on testing Frameworks is advised.

7. **Research and Reporting**
   a) Rethakgetse Manaka will be responsible for compiling the report.
   b) All research conducted(including assumptions, alterations, and design decisions) must be accompanied by a reference and sent to this email address u22491032@tuks.co.za .

This has been included to make every team member aware of how and where to submit research for their portion of the project. It is assumed that research may include implementations of similar projects before this one, as well as a reason must be stated as to why the implementation and design decision was made that way.

8. **Git Standards and Github Standards**
   a) When working on different features work on a different branch different to main/master branch.
   b) Squash your commits before merging your branch. This means that commit to latest point and then merge.
   c) Use short descriptive sentences for commit messages.
   d) Use tags as for this will be able to "rollback" a specific version if things go wrong.
   e) One person is in charge of the repository. They are responsible for setting up protections and rules that will protect the repository from "dangerous" edits to the repository.

Coding Standards : https://blog.codacy.com/coding-standards-what-are-they-and-why-do-you-need-them/

Git Standards : https://www.freecodecamp.org/news/how-to-use-git-best-practices-for-beginners/

NASA Coding Standards : https://www.youtube.com/watch?v=GWYhtksrmhE&t=14s
Git Best Practices : https://opensource.com/article/20/7/git-best-practices