C++ style conventions enhance code readability and maintainability. Here are some common examples:

Naming Conventions

- **CamelCase**: for class names (e.g., MyClass).
- **snake_case**: for function and variable names (e.g., calculate_area).
- **ALL_CAPS**: for constants (e.g., MAX_VALUE).
- **m_prefix**: for member variables (e.g., m_name).

Indentation and Spacing

- Use 2 or 4 spaces for indentation; avoid tabs.
- Keep lines within 80-120 characters.
- Add a blank line between logical code blocks.

Comments

- Use // for single-line comments.
- Use /* ... */ for multi-line comments when necessary.
- Write clear, concise comments explaining the purpose of code sections.

File Structure

- Header files (.h or .hpp) for declarations.
- Source files (.cpp) for definitions.
- Include headers in a logical order, often grouped by project, library, and system includes.

Example

```
#include <iostream> // System header
#include "my_class.h" // Project header

const int MAX_SIZE = 100;

class MyClass {
public:
  MyClass(int value);
  int get_value() const;
  void set_value(int value);

private:
  int m_value;
};

MyClass::MyClass(int value) : m_value(value) {}

int MyClass::get_value() const {
  return m_value;
}

void MyClass::set_value(int value) {
  if (value >= 0 && value <= MAX_SIZE) {
    m_value = value;
  } else {
    std::cerr   << "Error: Value out of range." << std::endl;
  }
}

int main() {
  MyClass obj(50);
  std::cout << "Value: " << obj.get_value() << std::endl;
  obj.set_value(120); // This will print an error message
  return 0;
}
```