# Effective C++

## Performance Assignment

### This summative assessment is designed to test your ability to develop solutions to real-world problems. It assesses your skills in applying the techniques covered in the module and the wider course, i.e. performance considerations, multi-threading and use of C++ generally.

This assignment is individual work and contributes **30%** of the overall coursework mark for this module.

### Assignment Overview

The skeleton application provided (see BlackBoard) provides some basic timing code.

You are to:

1. Enhance this application so that it performs image processing tasks (see below) on a number of JPG files.
2. Optimise your application so that it executes as fast as possible. Your application MUST process the images as fast as possible and in real time.
3. Supply an explanation of the **code structure/architecture**, **its operation** and a **reflection on which elements of your course (and not just this module) have informed your choice of solution for this assessment**. For this reflective piece, describe any relevant modules and why they have been useful in helping you arrive at a solution to this problem. For this, you may describe any software engineering principles, techniques or approaches you have learned and used. This should be augmented with an architectural description your solution, i.e., the major functions and classes and what these contribute to the application and the relationships between them. Also describe any major language features that have contributed. Lastly describe any algorithms (not STL “algorithms”) employed and major design decisions. Both text and diagrams are expected.

### Additional Image Processing Tasks

As it stands, the application applies a simple brightening function to each image. The application must perform this as well as the following tasks:

1. Rotate each image clockwise by 90 degrees.
2. Scale each image to double its size - in both horizontal and vertical dimensions. To do this, apply the **bilinear interpolation algorithm** to produce a resulting image that is a good quality interpretation of the original.
3. Convert each image to greyscale (i.e., remove the colour information to make the image appear as though it was shot with black-and-white film).

### Additional Information

The source files are JPGs and the destination files (after processing) must be PNGs. Assume the images are of different dimensions (width and height) and this needs to be taken into account when processing.

The sample JPG files are for testing purposes. Your application should be capable of scanning the contents of a folder and identifying all JPG files, irrespective of filename and use these for processing. You will not know the number of final images used for marking purposes, but the same images will be used for marking every student submission. The order in which the images are processed and the order in which the image processing tasks are performed is up to you. When marking the assessment, you can assume the files will be different to the samples used for testing.

You can apply any C++ optimisation strategies to improve the performance of your application. You can also use assembly language or GPU processing, as well as C++, if you wish.

The skeleton source code contains some performance measuring routines that will be used to measure your application's execution time. Do not alter this part of the program.

You may use third-party libraries. If you do this, you MUST provide references and describe where and when your application depends on these. Missing references will be treated as plagiarism.

### Performance Mark Allocation

If your application correctly and successfully implements the four required image processing tasks, then it will be subjected to a performance test. The execution time will be compared to other assignment submissions (which are equally correct and successful) and marks will be awarded according to a banding system. The top 5 highest performing applications will receive 15 marks. The next 5 will receive 12 marks, and so on.

Please make sure your code is commented and easy to read – if the structure of your code is incomprehensible then you may also be disqualified.

### Overall Mark Allocation

|  |  |
| --- | --- |
| Correct image rotation | 2 |
| Correct greyscale conversion | 2 |
| Correct scaling and bilinear interpolation for image size increase | 6 |
| Maximum mark available for the performance evaluation | 15 |
| Explanation of code structure and operation  Marks will be awarded for depth of explanation in terms of algorithms used, general structure of source code, major functions/classes, function call hierarchy, use of STL and run-time issues. | 5 |
| Total | 30 Marks |

### Submission Check List

Please check you have submitted the following:

|  |  |
| --- | --- |
| Cleaned source code and project files, compilable on Visual Studio 2017 or 2019 on. Ensure this compiles and works. |  |
| Summary of code structure (functions, classes, and algorithms). |  |

There is no need to include the image files with your submission.

Clean your project and delete any unnecessary files before submitting (.vs folder, etc).

MAKE SURE THE APPLICATION COMPILES AND EXECUTES WITHOUT ERROR.

**If you use third-part libraries, make sure ALL required files are included.** Failure to submit the required files, or if your application does not compile on University machines, will result in zero marks being awarded.

### Hand-in Date

Thursday 22nd April 2021

# Effective C++

## Performance Assignment

## Marking and Feedback

### This formative assessment is designed to test your ability to develop solutions to real-world problems. It assesses your skills in applying the techniques covered in the module so far, i.e. performance considerations, multi-threading and object-oriented design.

|  |  |
| --- | --- |
| **Student Name:** | **Mark** |
| Correct image rotation | 2 |
| Correct greyscale conversion | 2 |
| Correct image scaling | 2 |
| Implementation of bilinear interpolation | 4 |
| Marks available for the performance competition | 15 |
| Description of code structure and operation | 5 |
| **Total** | **/ 30** |