Base64 Encoder Documentation

Thomas Corr – 40226499

# Introduction

This document is designed to provide information on the process, algorithm and code analysis of the base64 encoder. This document will consist of the following:

* Areas of research
* Algorithm
* Code Analysis
* Testing

# Areas of research

## Lifewire Website

This website was used to provide information on the process required to turn an ASCII text set into a base 64. The website helped to understand what the input and output of the encoding program will be. The information taken from this website helped to create an algorithm for later use.

<https://www.lifewire.com/base64-encoding-overview-1166412>

## Base64encode Website

This website was used in order to provide evidence for all my base 64 encoded calculations. Provides both encoding and decoding options upon the website with a brief description on how to go about encoding from ASCII into base 64.

## <https://www.base64encode.org/>

## Rene Nyffenegger Website

This website provided a source code for a base 64 encoder in C++. This source code helped to provide an understanding of how other individuals programmed a base 64 encoder efficiently. The code also provided an understanding of a base 64 encoder in practise.

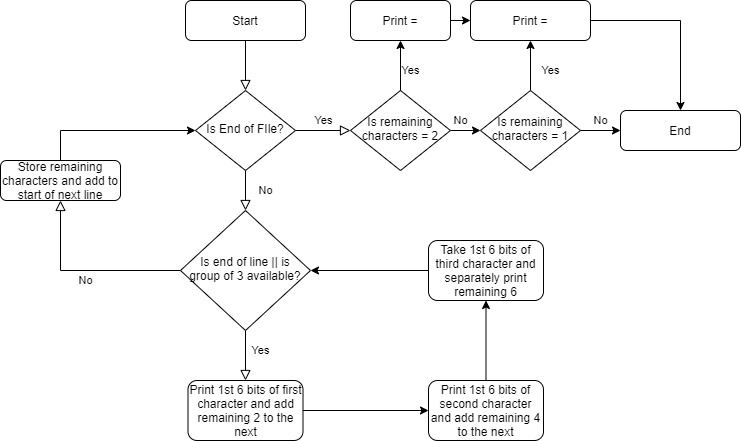
<https://renenyffenegger.ch/notes/development/Base64/Encoding-and-decoding-base-64-with-cpp>

## CPlusPlus Website

This website helped to provide an understanding of file handling using C++ that would be used to help take data from a text file.

<http://www.cplusplus.com/doc/tutorial/files/>

# Algorithm



# Code Analysis

For full code, see attached cpp file.

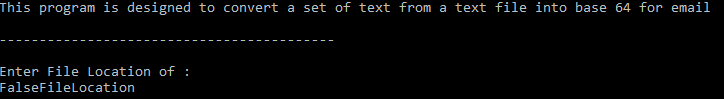
Upon commencing the program, the program will firstly request for a file path of a text file suitable for encoding. A while loop will commence until a valid file path is provided. If the file path is valid, a while loop will run through each line of the text file. From here a for loop will run through the line, going through each character in groups of three (passing any leftovers onto the next line). Each group will go through a process identical to the algorithm provided above. When all lines of the text file have been encoded, = signs will be printed for each leftover character from the process.

Once this process has been completed the program will ask if the user would like to encode another text file, if yes then the process will repeat, otherwise will exit.

# Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test No. | Description | Variables | Pass/Fail | Improvements |
| 1 | Incorrect File Location Test | “FalseFileLocation” | Pass | N/A |
| 2 | Correct File Location Test | “C:\Users\thoma\Desktop\Test.txt” | Pass | N/A |
| 3 | Encoding Single Line | “Man” | Pass | N/A |
| 4 | Encode Long Single Line | “This is a Man” | Pass | N/A |
| 5 | Encoding Multiple Lines | “This is a Man”  “This is a Woman”  “These are their Children” | Pass | N/A |
| 6 | Incorrect Restart Loop Request | “Incorrect Entry” | Pass | N/A |
| 7 | Restart Loop Request | “Y” | Pass | N/A |
| 8 | Reject Restart Loop Request | “N” | Pass | N/A |

Test 1





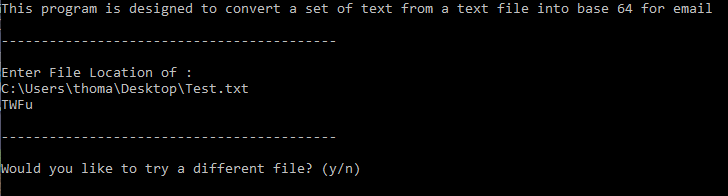
Test 2



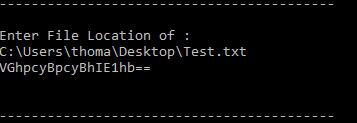


The above Base64 encoder is of a set of text contained within Test.txt at the given file location

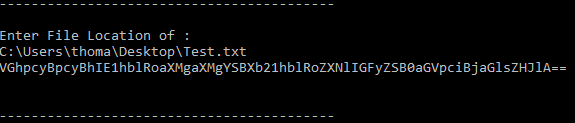
Test 3



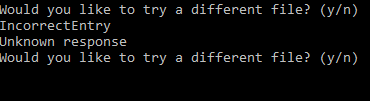
Test 4



Test 5



Test 6



Test 7



NOTE: when entering a file location after Test 7, the program crashed when encoding. This was because when the repeated program occurred, the global variables were not reset. This has been fixed as shown below



Test 8

