Project

Teams: None, please work individually on this project

References:
1. Pointers and memory lectures and projects
2. Command line arguments lecture and projects
3. Valgrind lecture and project
4. Embedded systems: https://en.wikipedia.org/wiki/Embedded_system#:~:text=An%20embedded%20system%20is%20a,larger%20mechanical%20or%20electrical%20system.&text=Embedded%20systems%20control%20many%20devices%20in%20common%20use%20today.
5. Effective C++ in an Embedded Environment: https://www.artima.com/shop/effective_cpp_in_an_embedded_environment

---

Starter Project:
See starter project (256_pointer_project_SKELETON) on course website, projects tab

Topics covered:
- Pointers and dynamic memory
- Command line arguments
- Low memory environments

Intro:
An embedded device is a combination of a processor, memory and input/output (IO) peripheral devices. An embedded device can be purpose built and designed to perform 1 task, with just enough memory, computing power and IO for that task. This configuration allows embedded devices to be physically small and to consume minimal power. The tradeoff these limitations impose is that embedded devices often have challenging hardware characteristics including:

1. Limited memory
2. Limited CPU
3. May or may not have an operating system
4. May or may not have a hard drive
5. May be difficult to upgrade

In short, embedded systems often trade flexibility for power efficiency and reduced size.

This project will focus on the limited memory aspect of embedded systems. What happens when you do not have a large heap? What happens when you cannot allocate enough memory to meet the current application demand?

If an application does not have access to a large heap then it runs the risk of throwing an out of memory exception whenever memory is allocated. A possible solution to this problem is for the application to allocate the maximum allowable amount of memory at application startup, and use only this memory as the program runs. This means the application can use no other APIs that allocate heap, particularly it cannot use standard library containers (vector, list, string, etc.) and cannot dynamically allocate memory using ‘new’ or ‘malloc’ outside of the application creating its initial allotment.

This project will simulate a memory challenged application.

Tasks

You are given a starter project. Its purpose is to; read an input file into a list of data structures, capitalize all alphabetic characters, and write all characters to an output file. This application operates using a limited amount of memory. The following lists the content that you are responsible for;

In file '256_pointer_project.cpp'
Modify the main function to harvest the following command line arguments;

1. Input file
2. Amount of heap memory the application can use (in bytes)
3. Output file

In file 'utilities.cpp'
Manage all includes.
Provide an implementation for the function defined in utilities.h.

In 'transform1.cpp'
Manage all includes.
Provide an implementation for the function defined in transform1.h

In file 'memorymanager.cpp'
Manage all includes.
Provide an implementation for the functions defined in memorymanager.h

Requirements

- Please submit the following files (please do not zip them):
  256_pointer_project.cpp
  utilities.cpp
  transform1.cpp
  memorymanager.cpp

- Please ensure that you do not use strings or any standard library containers to hold the chars read from input file. Please only use the list implemented in your memory manager.

Grading:
100% as per program output
10% penalty if you do not follow correct submission format

Stuff to consider
Look at the tests, particularly the test_system() function to see how the list is used.