

# EGRE 246 Advanced Engineering Programming Using C++

## Homework #9 – C++ State-machines

This homework must be your own (individual) work as defined in the course syllabus and discussed in class.

Communication between electrical equipment in general happens by sending messages back and forth in a specific format. When you request a page in your internet browser, you are sending a message to a server, but this message is encapsulated in a standard format adding additional information. Even the wireless communication between the UAVs used by the VCU UAV research group has a standard format for sending messages over the air.

For this homework you will have to build a parser class that will be able to parse messages out of a binary file using a state-machine implementation. The parser should be implemented as a class and the definition of the class is given in “parser.h”. You will have to write the main function, the implementation of the parser class, and the “parser.h” needs to be updated to represent the correct states in your state-machine.

The messages given to you in the test files have the following format:

1. A Start byte (170)
2. A Type byte (0 = ASCII, 1 = Integer, 2 = float, 3 = double)
3. Low byte of the Size for the payload
4. High byte of the Size for the payload
5. Payload
6. Checksum byte

1 byte	1 byte	1 byte	1 byte	Size bytes	1 byte
Start (170)	Type	Size (low)	Size (high)	Payload	Checksum

The Start byte has a decimal value of 170 (0xAA) and represents the start of a new message.

The type byte indicates the type of the payload. When the value for the Type byte is 0, the payload is filled with ASCII characters. When the Type is 1, the payload represents an integer. When the Type is 2, the payload represents a float, and when the Type is 3, the payload is a double.

The next 2 bytes represent the size of the payload; the first Size byte is the lower half of the 16 bit Size representation and the second Size byte is the higher byte. For example, when the low Size byte has a value of 0x04 and the high Size byte has a value of 0x01,

the final value is 0x0104 (decimal value 260). This means that the size of the payload is 260 bytes. When the Type of the message is integer, float or double the size of the payload is fixed to either 4 (integer, float) or 8 (double).

The payload is the data that is of importance. In the example of the internet browser this might be the URL you typed for a specific website. The payload can be of variable size (Maximum size of 1024) for the ASCII type message. For the integer and float type messages the payload size will be 4 bytes and for the double type the payload size will be 8 bytes

The checksum byte is a running summation of all the previous bytes in the message. And since the type is of unsigned char it will overflow when reaching 256. Given the message:

0xAA	0x00	0x03	0x00	0x54	0x49	0x4D
Start	Type	Size	Size	T	I	M

The checksum byte will be:

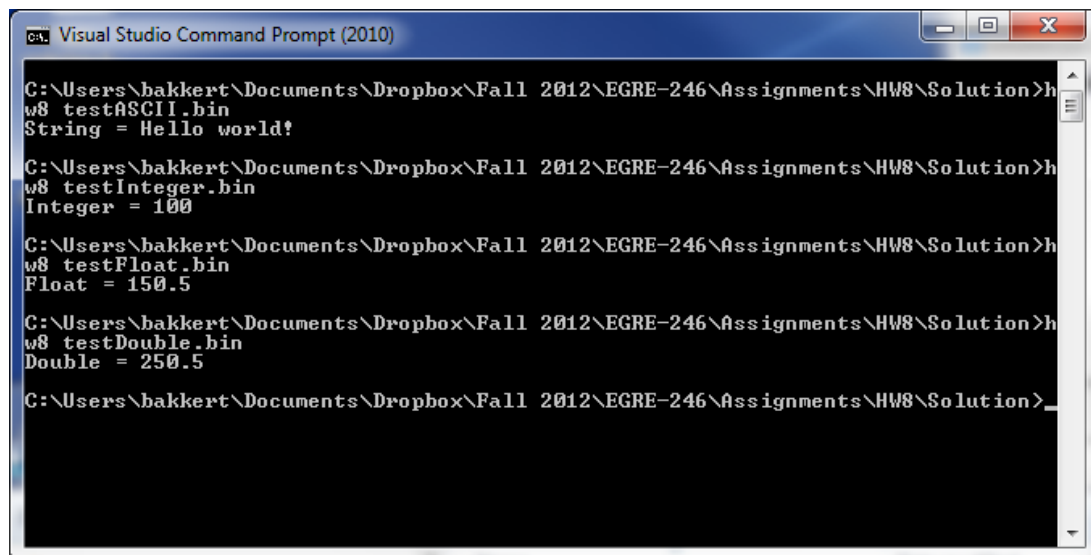
$$(0xAA + 0x00 + 0x03 + 0x00 + 0x54 + 0x49 + 0x4D) \% 0x100 = 0x97$$
$$(170 + 0 + 3 + 84 + 73 + 77) \% 256 = 151 = 0x97$$

The final message will look like:

0xAA	0x00	0x03	0x00	0x54	0x49	0x4D	0x97
Start	Type	Size	Size	T	I	M	Check

When the checksum is incorrect, the message should be dropped and the state-machine should be looking for the next start byte of the next message.

The main function should read in the file byte for byte and for every byte the parse() method in the parser class should be called. The binary file for reading should be passed into the program as a command-line argument. When the user does not use the program correctly a message should be displayed giving the user instructions on how to use this program.



```
C:\Users\bakkert\Documents\Dropbox\Fall 2012\EGRE-246\Assignments\HW8\Solution>h
w8 testASCII.bin
String = Hello world!

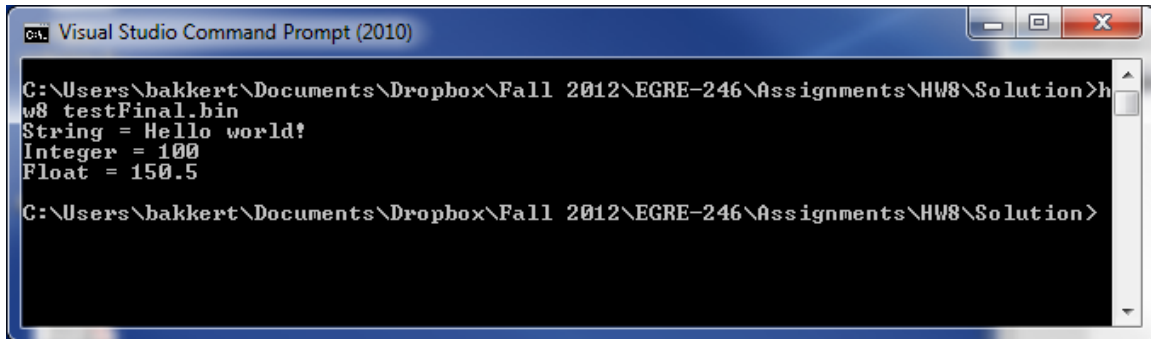
C:\Users\bakkert\Documents\Dropbox\Fall 2012\EGRE-246\Assignments\HW8\Solution>h
w8 testInteger.bin
Integer = 100

C:\Users\bakkert\Documents\Dropbox\Fall 2012\EGRE-246\Assignments\HW8\Solution>h
w8 testFloat.bin
Float = 150.5

C:\Users\bakkert\Documents\Dropbox\Fall 2012\EGRE-246\Assignments\HW8\Solution>h
w8 testDouble.bin
Double = 250.5

C:\Users\bakkert\Documents\Dropbox\Fall 2012\EGRE-246\Assignments\HW8\Solution>
```

You are given 4 practice files- testASCII.bin that contains a message of the ASCII type, a testInteger.bin file where the payload represents an integer, a testFloat.bin file where the payload consists out of a float, and a testDouble.bin file where the payload is of a type double.



```
Visual Studio Command Prompt (2010)

C:\Users\bakkert\Documents\Dropbox\Fall 2012\EGRE-246\Assignments\HW8\Solution>h
w8 testFinal.bin
String = Hello world!
Integer = 100
Float = 150.5

C:\Users\bakkert\Documents\Dropbox\Fall 2012\EGRE-246\Assignments\HW8\Solution>
```

For this homework you will have to implement the following:

1. main function, reading in the file byte for byte and calling the parse member function in the parser class. The command-line argument to the program should be the test file that you want to load and parse. Make sure you have sufficient checks and a help message is displayed when the program is executed incorrectly.
2. Regular constructor and destructor for the parser class.
3. bool parse(unsigned char): which will take in a byte as an argument and will run the parser state-machine. The return Boolean should be set to true if and only if the checksum is correct.
4. void process(void): this method will be called when the parse() method returns true, and the message can be processed. In this method the payload should be converted to the correct type and printed to the console.

Deliverables:

main.cpp, parser.cpp, parser.h

Included in submission:

main.cpp, parser.cpp, parser.h, testASCII.bin, testDouble.bin, testFloat.bin, testInteger.bin

## BONUS:

Where in the original homework assignment you read the data from a file, for the bonus part the data will be send over TCP. You are given two files `tcp.h` and `tcp_client.cpp`. These two files contain a TCP client class that you can use to connect to a TCP server. The class works with callback functions, these are functions that you pass to the class and whenever an event has happened these callback functions are executed from the TCP client object.

Additional to the above files you are also given a `bonus.cpp`, this file includes all the necessary information and functions to connect to the server. The server sends periodically (every 2 seconds) another message in the same format as described above but now with transmission errors included. The server is located at IP address 128.172.167.75 at port 5999 and can be only accessed from the VCU campus or through a VPN connection.

Once you have received the message the message should be parsed using the parser build in the original assignment and should the messages should be displayed correctly.