CRC Calculation Program

The Program has three primary functions, to calculate and append the CRC to a text file containing hex characters, to check and strip this CRC value, and to scan for a hex signature in a file.

The Program starts simply by calling up the MenuDriver function, which displays a menu asking which task you want done, and calls the specific function for the selected task.

The first task is to Calculate the CRC, so the CalculateCRC\_Process function is called, which drives the CRC calculation task. The process works likes follows:

* First, the OpenFile function is called, which takes a file pointer as an argument, prompts for a file name, opens the file, checks the validity of the file, and ends.
* Next, the ReadInput function is called, which takes the file pointer and a vector of ints as arguments. The function puts each character into its own spot in the vector of chars, then calls the ConvertInput function, which converts the vector of chars into a vector of ints, and checks for invalid characters. It ignores white space, and will close the program if an illegal character is in the input.
* ConvertToBinaryArray is called next, which takes a vector of ints and a vector of bools as arguments. It takes each int, and pushes four bools onto the vector of bools, corasponding to the ints binary value.
* Next, is the PrintStartNumberValue function. This takes a vector of bools and a vector of ints as arguments, and displays the hex and binary values in the console.
* Then SetUpCRC\_Constant is called next. It takes a vector of bools as input, and pushes onto it the value of the CRC Constant.
* After, PadBinaryData is called, which takes a vector of bools as an argument, and pushes 16 zeroes onto it,
* Then, CalculateCRC is called, which takes in two vectors of bools as arguments, and returns a string. If the input is over a certain size, the function will ask if the user wishes to display the calculations to the console or not. Over a certain size, displaying to the console will cause the program to run 100’s of times slower then if printing to the console was skipped. The program XORs the input with the CRC, displays the value, and then moves the index of the input over by one. It then takes the last 16 bools of the input, and returns them as a string.
* Then the program prints the value of the CRC in Binary, using the PrintPropplySpacedBinary function to space it correctly.
* The program then converts the binary string into a string of Hex characters, using the ConvertCRC function, and then prints this string to the console
* The program then clears the input files flags, and appends the CRC to the end of the input file. Then it displays the input files input again, and closes the input file.

The next process does is the VerifyCRC\_Process. Functions reused from the CalculateCRC\_Process will be named, but not described again.

* First, OpenFile is called.
* Then ReadInput
* Then ConvertToBinaryArray
* Then PrintStartNumberValue
* Then SetUpCRC\_Constant
* Then CalculateCRC
* And finally VerifyCRC, which checks to see if all the values in the vector of bools are 0 or not. If they are all zero, it returns true, that the CRC was Verified.

The next process is the StripCRC process. This first verifies the CRC, then if it is correct, it strips the value. Functions reused from earlier process will be named, but not described again.

* Instead of calling OpenFile, it manually does this within the function, so to save the filename for use later.
* ReadInput is called.
* Then ConvertToBinaryArray
* Then PrintStartNumberValue
* Then SetUpCRC\_Constant
* Then CalculateCRC
* Then VerifyCRC.
* If the CRC checks out, it will rewrite the input file, except with out the last 4 characters. If the CRC check fails, it will say so and do nothing to the input file.

The final process is the VirusCheck process. This will take an input file and a virus signature, and check for the virus signature in the input file.

* First, it prompts for and opens the input file to be checked, with the GetInput\_forVirusCheck function.
* Next, it prompts for the Virus Signature, with the GetSig function. This will prompt the user if they want to manually input the Signature or to open up a file.
* The function will then check to see if the Signature is a valid size. The Signature must at least have 2 characters, and be smaller then the input file. If not, the function exits.
* The function then sets up nested for loops. With the input file being the outer loop, i, and the signature the inner loop, j. The inner loop will check if input[i+j] == signature[j]. If it is, nothing happens. If it does not, the WildCardCount will be incremented by one. If WildCardCount get to two, the inner loop will break. Once the inner loop is done, info will be displayed in the console if a signature has been found. Then the outer loop will be incremented by one, and the search done again. If no Virus Signatures are found, a line stating so will be outputted to the console.

Functionality Problems and notes:

* All of the CRC related process will not behave correctly if the input file is smaller then 3 or less hex characters, which would be smaller then the CRC value it is being checked against.
* The program will not create additional files, it will write directly to the input file.

Programing Info:

The program was written in C++, using Microsoft Visual C++ 2010 Express, has been designed for Microsoft Windows, and has been tested on the 64-bit edition of Windows 7.