HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

**SCHOOL OF ELECTRICAL ENGINEERING**

-----o0o-----

**REPORT ON**

**CREATING A HEX DISPLAY PROGRAM**

**Course name: PROGRAMMING TECHNIQUES**

**Course code: EE3490E**

**Class code: 123135**

**Instructor: Nguyen Hong Quang**

**Student’s name (code): VO TRAN TRUNG HIEU - 20186104**

**Class:** [**Automotive Engineering**](http://ste.hust.edu.vn/wp-content/uploads/2014/11/Kỹ-thuật-ô-tô-Automotive-engineering-Tich-hop-Cu-nhan-thạc-sy.pdf) – **Elitech K63**

**HANOI 2021**

Contents

[A. Introduction 3](#_Toc74991897)

[B. Characteristics of The Program 4](#_Toc74991898)

[C. Flowchart and Explanations 6](#_Toc74991899)

[D. Some Captured Results of The Program 7](#_Toc74991900)

[E. Epilogue 7](#_Toc74991900)

A. Introduction

HexDisplay.exe is the program to display Intel Hex Format file (.hex).

According to Wikipedia, Intel hexadecimal object file format is a file format that conveys binary information in ASCII text form. It is commonly used for programming microcontrollers, EPROMs, and other types of programmable logic devices.

# I. Record structure of a hex source code:

A record is a line of text consists 6 parts:

1. Start code: “:”
2. Byte count: two digits – indicates the number of bytes in data field.
3. Address: four digits – indicates the memory address offset of the data.
4. Record type: from 00 to 05 – indicates different record types.
5. Data: a sequence of n data.
6. Checksum: two digits – a computed value to verify validity of the record.

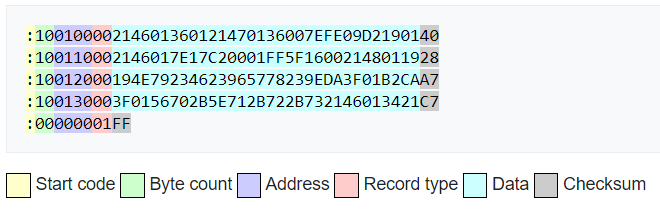


Figure 1. Intel Hex File Example.

# II. Requirements of HexDisplay.exe:

1. Taking the “.hex” file to start processing.
2. Check if this is a valid “.hex” file and send message to inform user.
3. Send the input data in the buffer.
4. Print address, data in hexadecimal and in character value.
5. Print 25 lines of data, then ask user if he wants to keep printing the next 25 lines. This process is repeated until the end of file.

B. Characteristics of The Program

# I. Program flow:

1. The program first asks user to enter the name of hex source file that he or she wants to display. If the hex file is in the same folder, user only have to type the name, otherwise, the exact directory to the hex file is required.
2. Then it shows user the first line of hex code that:

* Have invalid start code,
* Or have invalid checksum,
* Or have invalid byte count value,
* Or have invalid source code length,
* Or have the signal of the end of hex code.

1. The program starts display the data stored in the hex code that:
   * Have record type 00,
   * And have valid checksum,
   * And have valid start code.

The data is displayed in two forms:

* Hexadecimal,
* And character,

along with its corresponding address in the memory.

1. After each 25 rows of data displayed, the program asks the user whether or not to keep printing the next 25 rows. This process is repeated until the last data is displayed.
2. In the end, the program signals that from some point on, the memory is empty. This is the end of the program.

# II. What can the program do?

1. Verify validity of checksum, find the hex source line that has invalid checksum.
2. Verify validity of start code, find the hex source line that has invalid start code.
3. Verify validity of hex code length.
4. Inform user when buffer size is exceeded.
5. Filter the data that has record type 00, valid check sum and start code and stores it in the buffer.
6. Print the data in hexadecimal and character forms, along with its corresponding address in the memory.

# III. What kind of input file that disrupts the program input phase?

If the program detects one of these problem below, it will stop the input phase and proceed to display phase.

1. The length of hex source code is invalid:

For example, if the byte count value is 02, then the hex source code should have (11 + 2\*(02)) = 15 characters.

1. The checksum is invalid:

The checksum value of hex source code when sums up with all other value (byte count, address, data, …) should be a value that has LSBs equal to 00.

1. Start code is invalid:

The start code of all hex source code should be “:”.

1. End of file:

Either record 01 or the end of file is reached will stop the program input phase.

# IV. The important point that decide the performance of the program:

HexDisplay.exe is the combination of many sub-functions. Thus, the performance of HexDisplay.exe is relied on the performance of the most frequently-used sub-function and the most important sub-function.

1. The most frequently used functions is “hexStringToInteger()”:

This function returns an integer from one sub-string of the hex source code, that means it converts a string to integer.

Since the program has to deal with a lot of scenarios where the information needed is represented in character, which we cannot use directly, we have to create a function to convert the information to executable values for processing.

1. The most important and intricate sub-function is “dataSetUp()”:

This function does the most important works: filter data, find errors and stores data in the buffer.

The hardest thing for me when creating this function is the process of taking the address of the data and storing it in the proper position in the 2D buffer.

Performance of this function decides the performance of the program.

C. Program Explanation

# I. Flowchart:

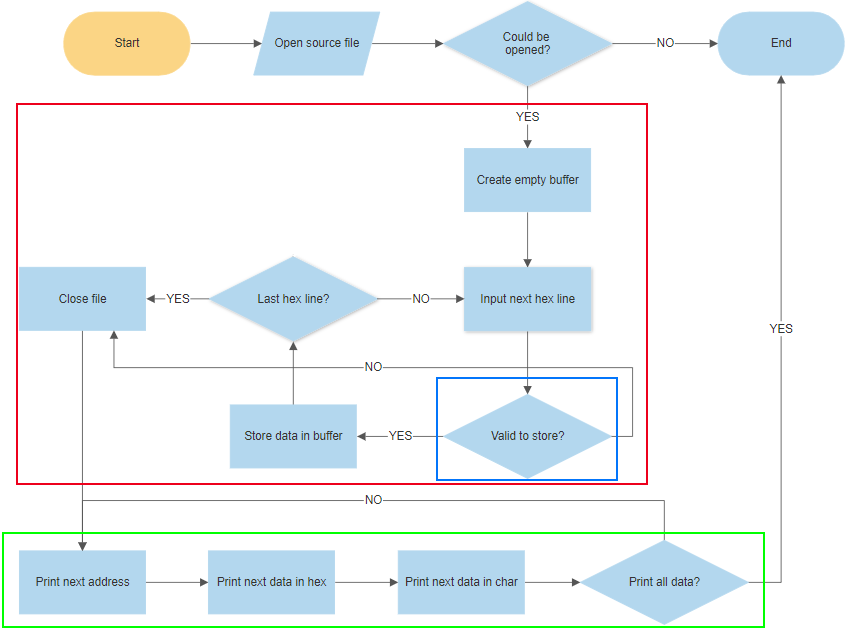


Figure 2. Flowchart of the program

Tasks that are in:

+ Red bracket: is executed by “int dataSetUp(FILE \*fpointer)”.

+ Blue bracket: is executed by “int dataFilter(char \*inputStr, int hexLine)”.

+ Green bracket: is executed by “void dataPrint(int influRange)”.

# II. Explanations:

1. File is taken, the program checks whether or not it could be opened:
   1. If not, then the program is immediately ended.
   2. Else, the program proceeds to the next step.
2. An empty buffer of size 4096x16 is created to store the input data. The buffer is initialized with the value 255 (equal to “FF” in hexadecimal, which denotes empty data address”.
3. The program recursively does this process:
   1. Input each line of the hex code in the input file.
   2. Check validity, if it is valid to store, then the program stores the data in the buffer; else, it does not store the data and proceed to (5).
   3. Check if all hex source lines are taken: if not, then redo (3a); else, proceed to the next step.
4. Close the file and proceed to the display phase.
5. The program recursively does this process:
   1. Print address of the first displayed data in the row.
   2. Print data in hexadecimal value.
   3. Print data in character value.
   4. Check if all data in buffer are printed: if not, then redo (5a); else end the program.

D. Some Captured Results of HexDisplay.exe

Some captured images of what HexDisplay.exe can do.

# I. For hex source file that has correct information

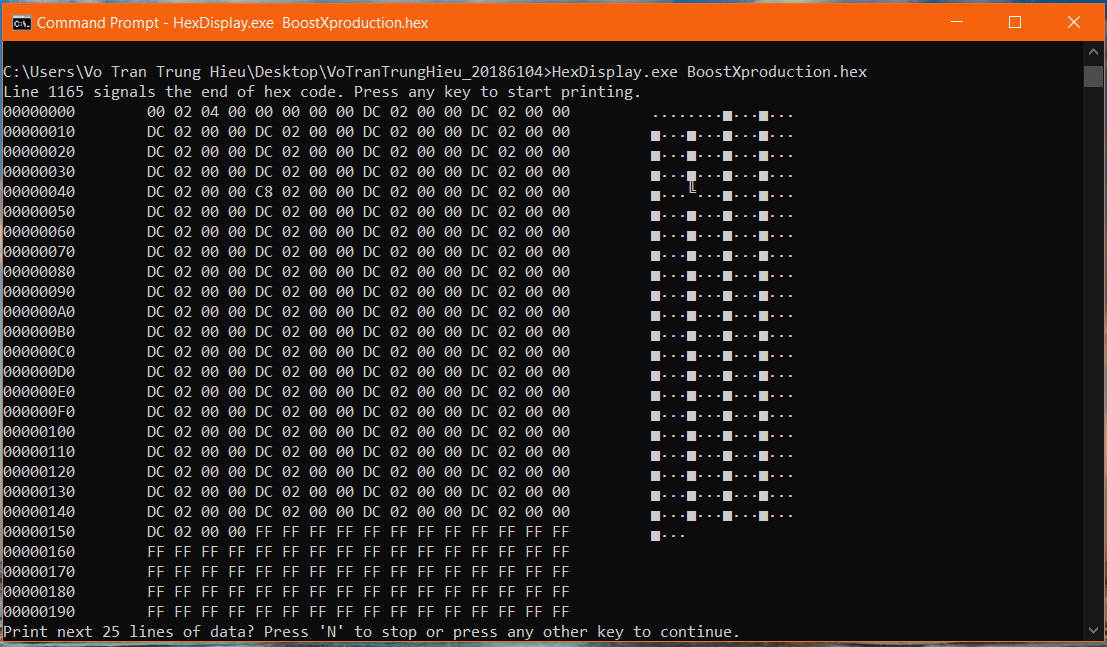


Figure 3. The beginning of the program.

The user is informed that which line of hex code is the end in the input file.

Each 25 rows of data, the user is asked whether or not he wants to keep printing.

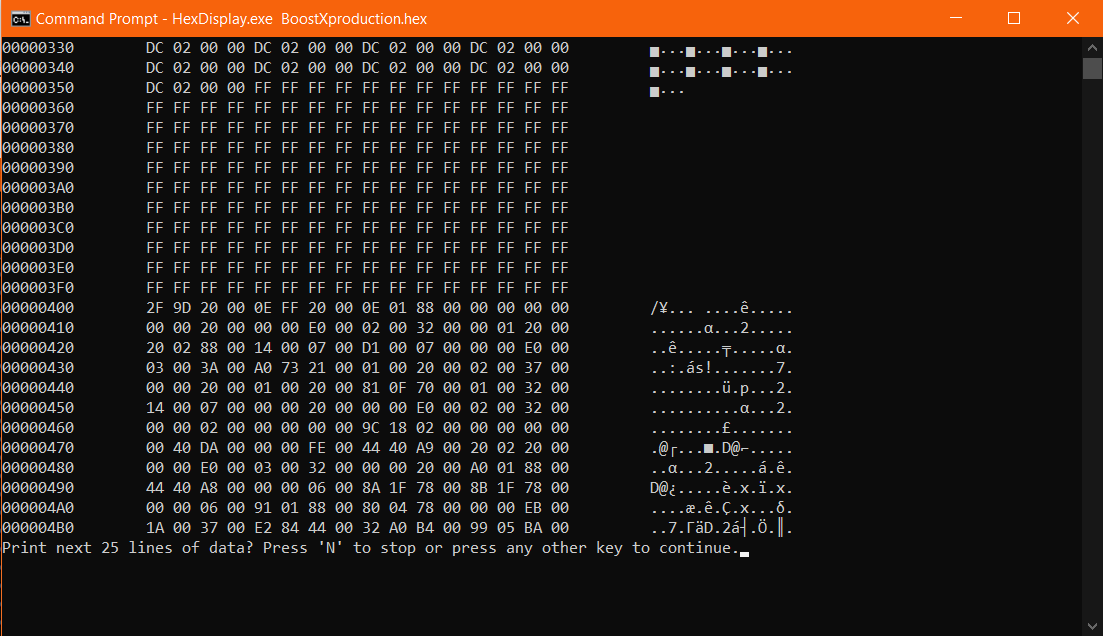


Figure 4. After pressing some keys to keep the program printing.

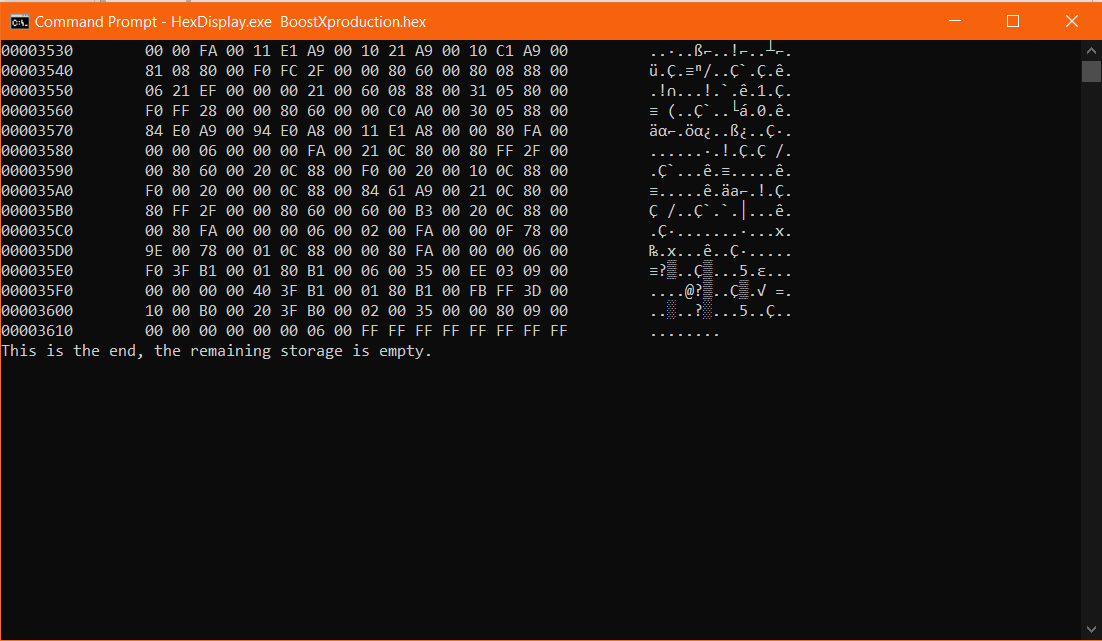


Figure 5. The end of the program.

The program notifies the end of data.

# II. For hex source file that has errors

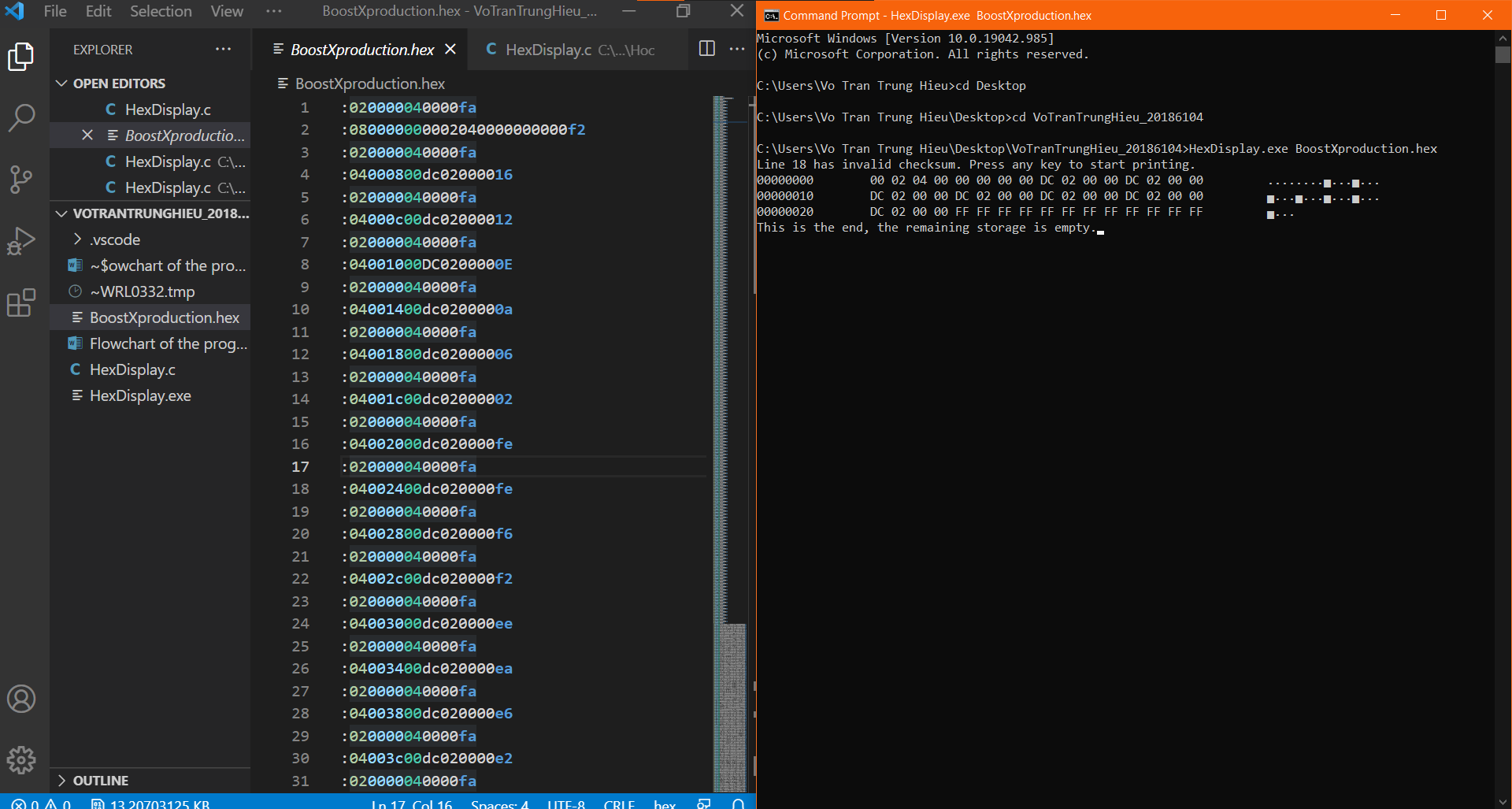


Figure 6. When a line of the hex source file has invalid checksum (line 18th)

When changing the checksum of hex line 18 (the true value is “fa”), the program informs user that at the line 18th, the checksum is invalid. It then stops input phase and only print the data of record type 00 from 17 previous lines.

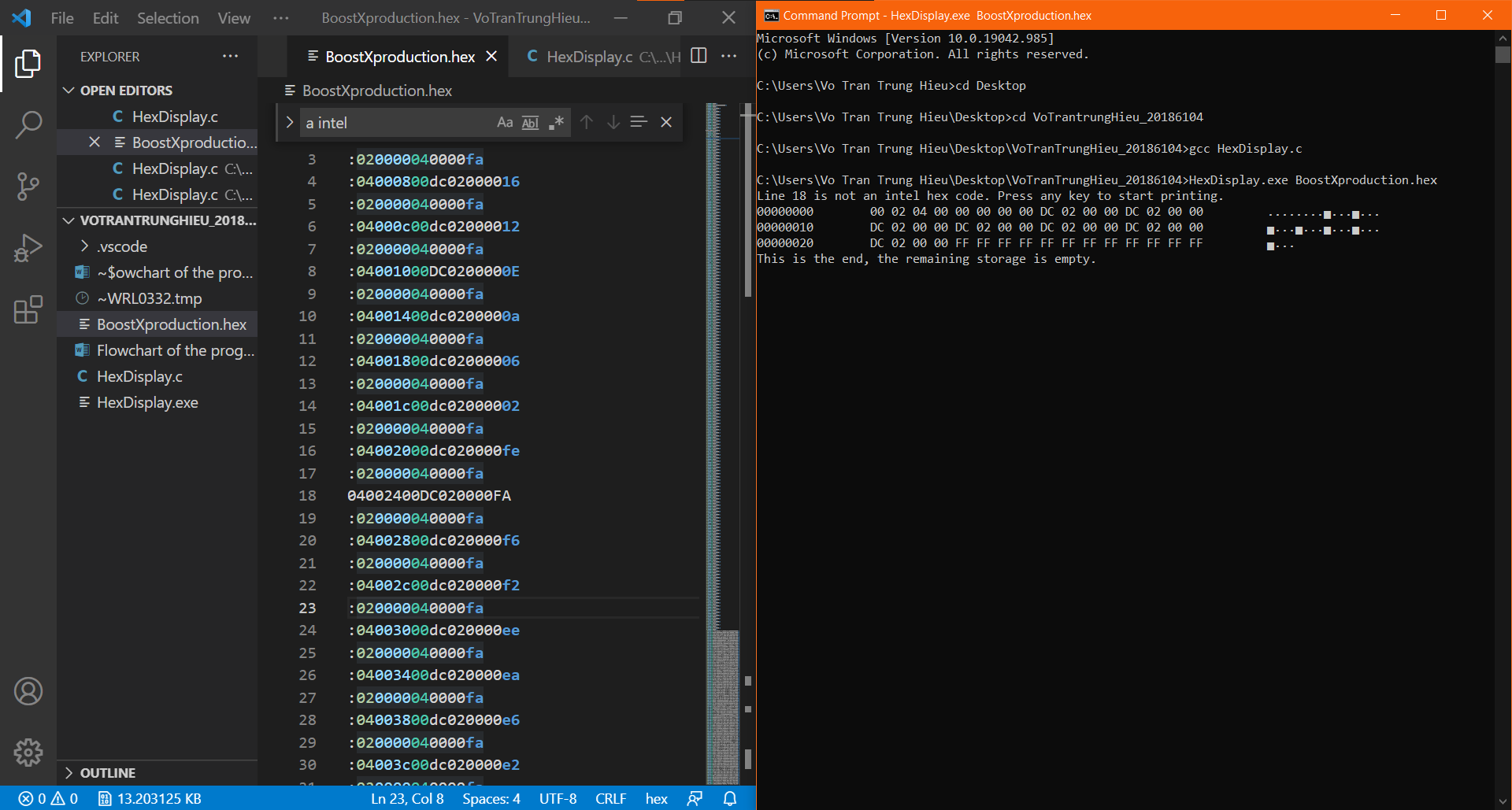


Figure 7. When a line of the hex source file is not an intel hex code (line 18th)

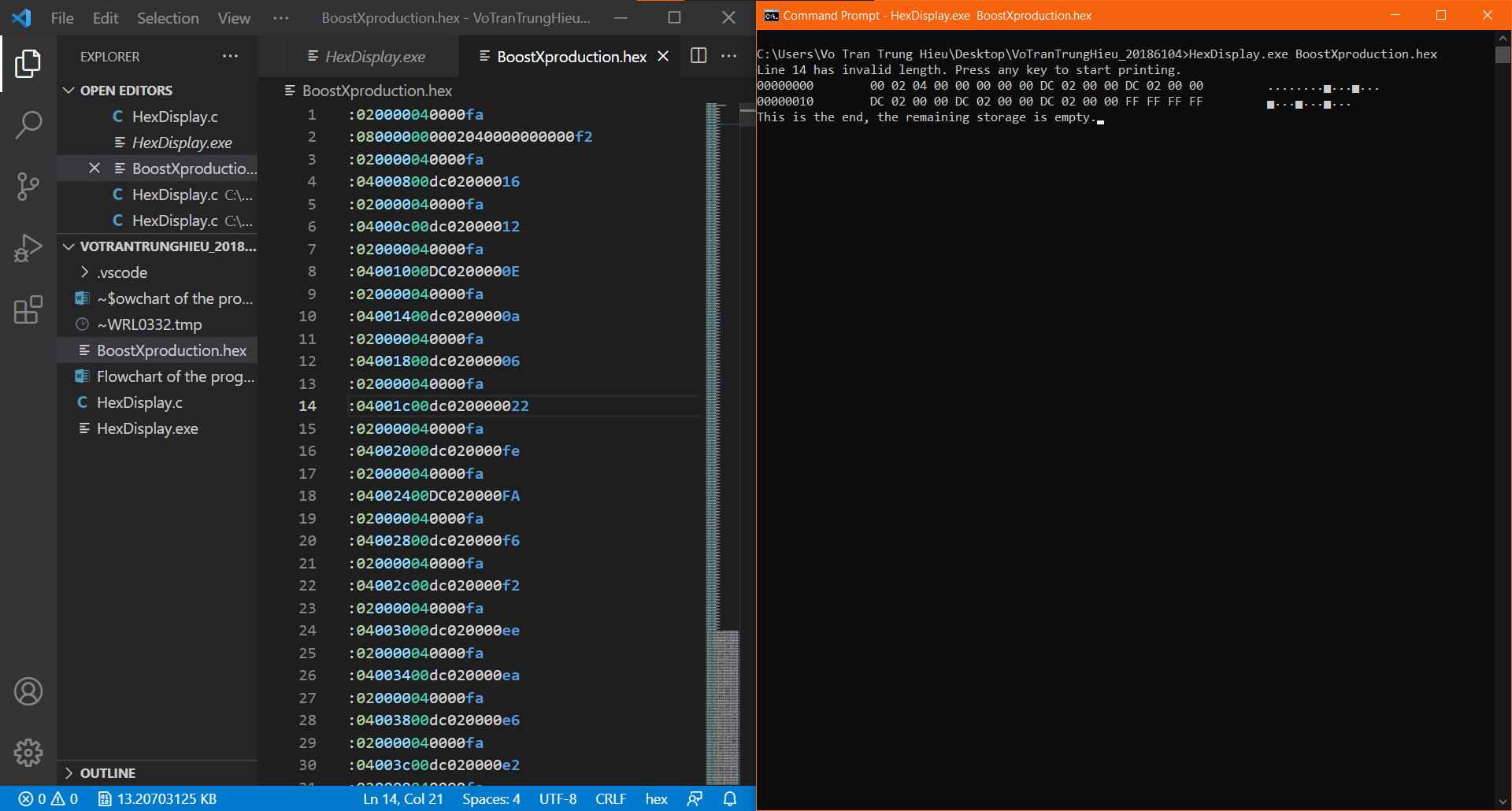


Figure 8. When a line of the hex source file has invalid length (line 14th)

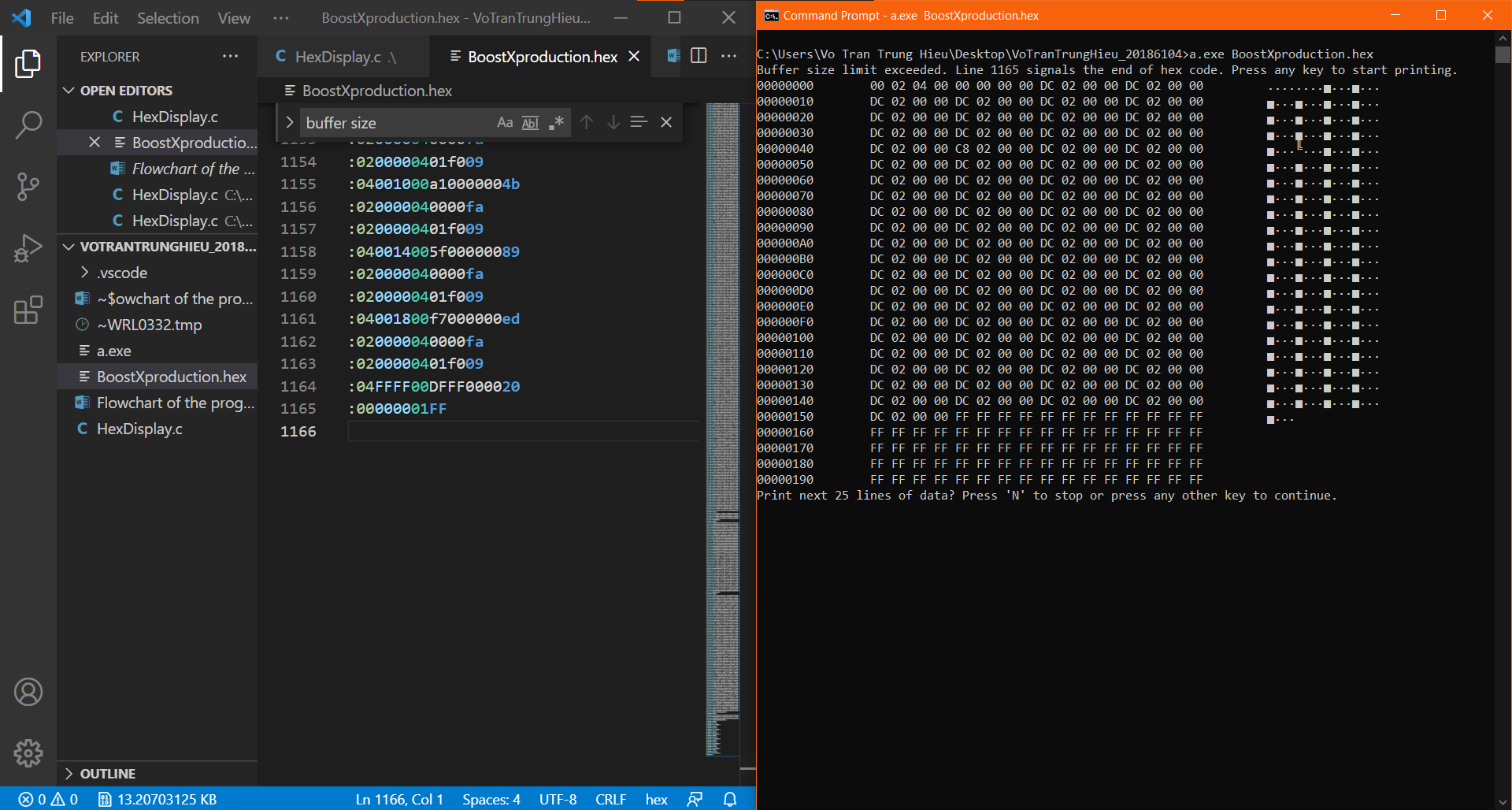


Figure 9. When buffer is exceeded

Line 1164th indicates the program to store the data “DF” in “FFFF”, so the next 3 bytes of data which are “FF”, “00” and “00” will be stored in the address 10000, 10001 and 10002. This means that the buffer limit is exceeded. (See figure 10).

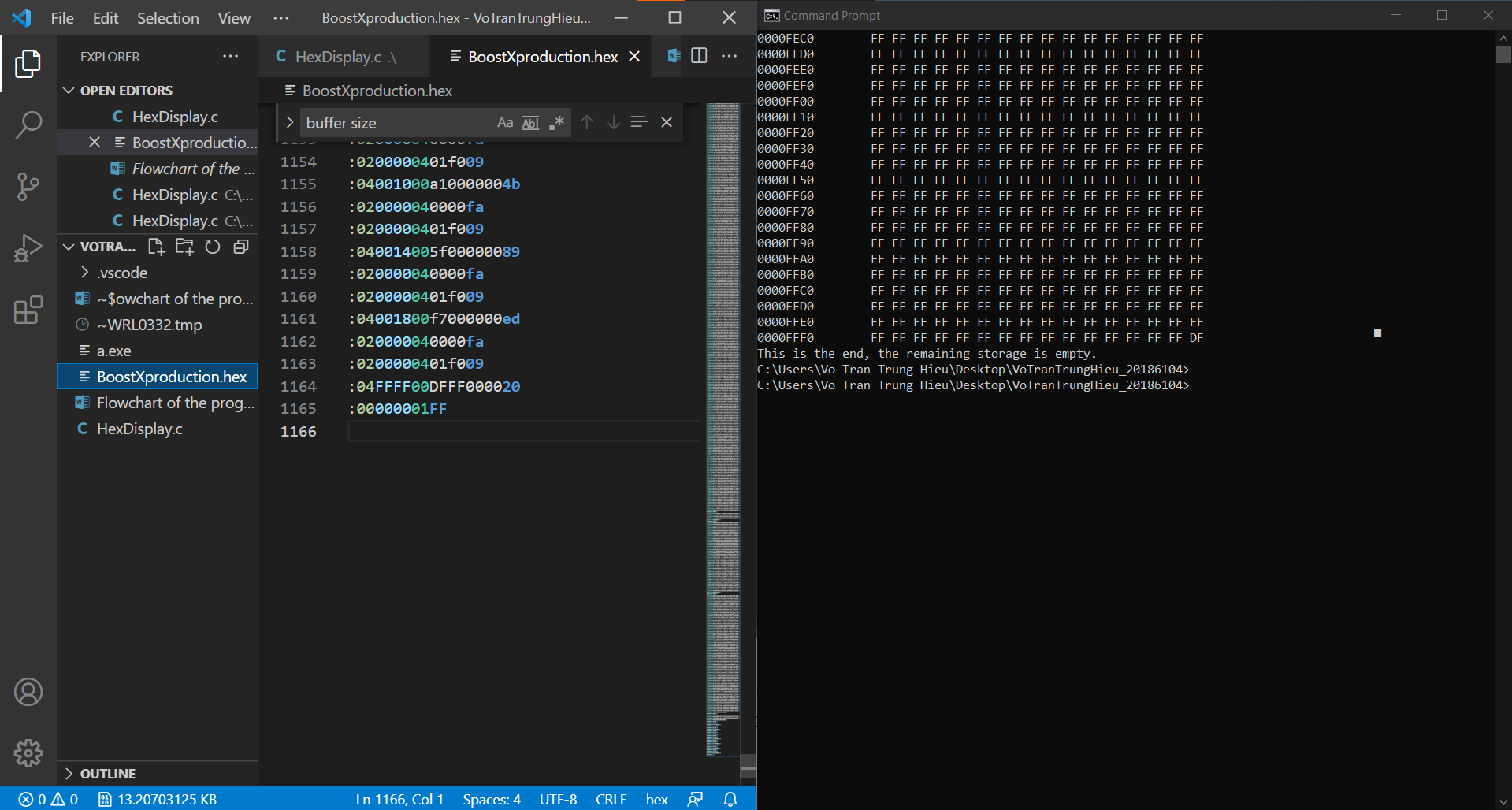


Figure 10. The buffer limit is exceeded displayed in the end

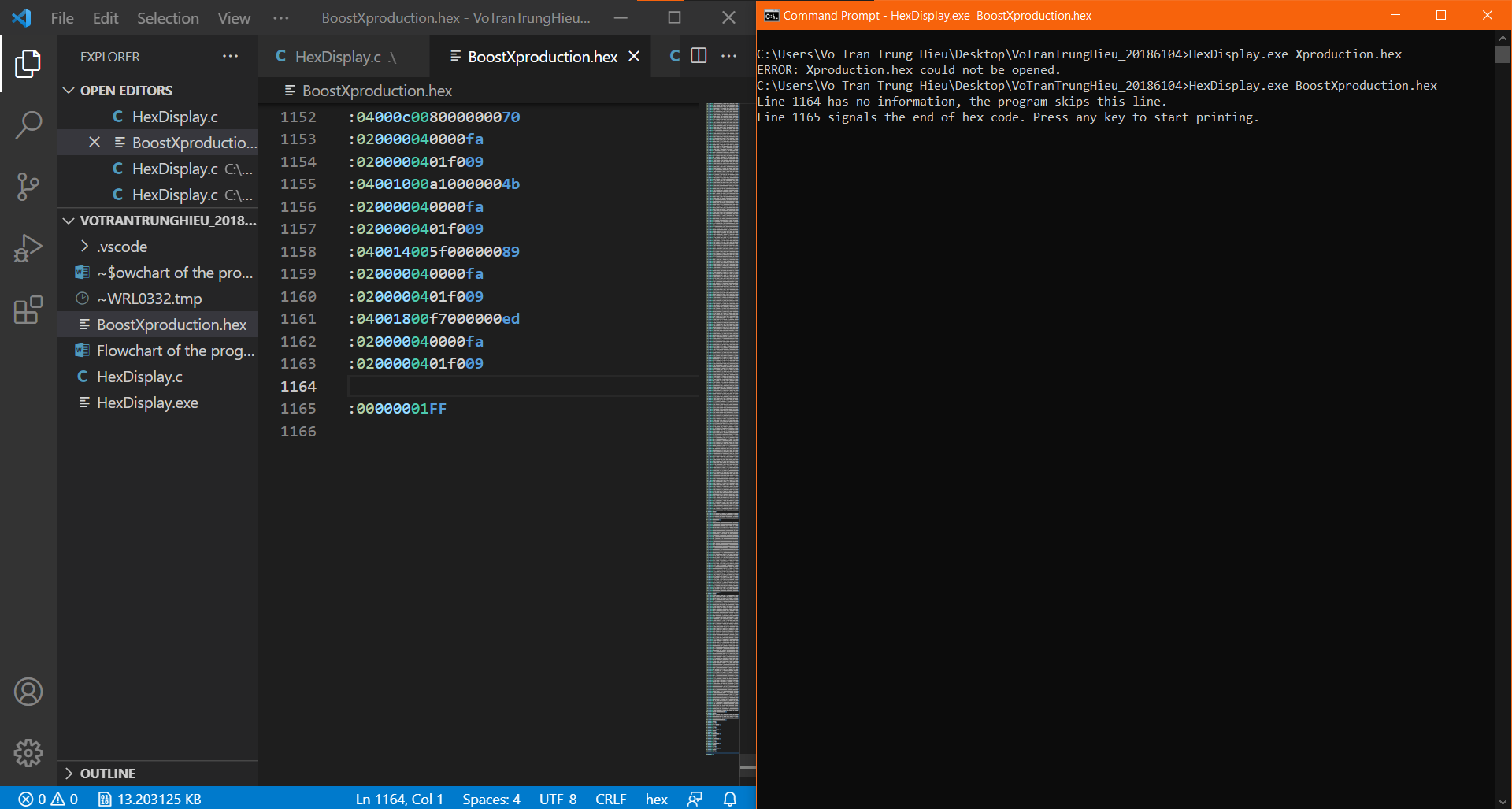


Figure 11. When a line of hex code has no information

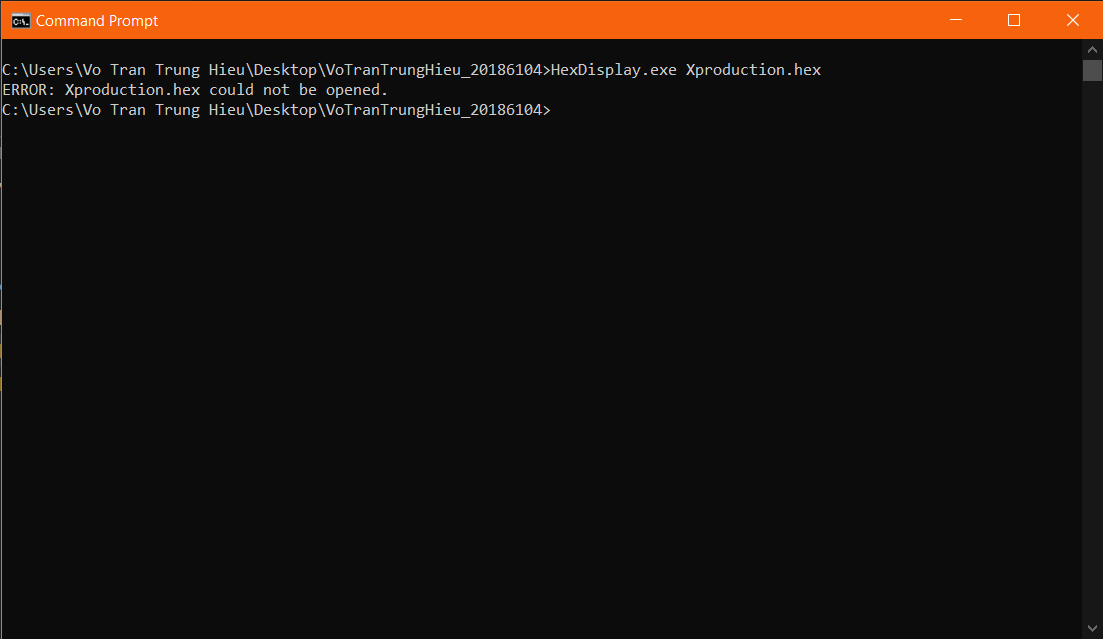


Figure 12. When the file could not be opened

E. Epilogue

Over the course of Programming Techniques, I gained a lot of knowledge on C programming and got the mindset of a programmer.

The biggest step on the ladder to be a proficient programmer, so far, is through this exercise. I strengthened my skills of working with files, transforming data representation in ASCII table, manipulating index in the array and basic shell programming.

In the last words, I want to thank you – instructor Prof. Nguyen Hong Quang, for being enthusiastic, devoting time and energy to give us special care in this course.

*(This is the end of my report)*