

**Name: Shawn Ronaldo Grant**

**Cand. #: 1000162122**

**Center #: 100016**

**Center: Campion College**

**ZENCYPHER**

**Text Encryption**

**CAPE Computer Science Unit 1 Internal Assessment**

**TABLE of CONTENTS**

* Problem Definition
  + Background 2
  + Context Description 3
  + Problem Description 4
  + Objective Of Solution 5
* Structured Chart 6
* Narrative 7
* Record Definitions 10
* Pseudocode 11
* C code 19
* Code Test Plan 39
* Test Plan Results 43
* Conclusion 50
* Appendix 51

**PROBLEM DEFINITION**

**BACKGROUND**

Privacy of information has been an important part of history, from messages been sent secretly by birds to now, where we use the internet to send most of our information. Information sent over the internet can include: usernames, passwords, addresses and telephone numbers**.** With the rise of the internet, new threats to data security have arrived. All computer systems are susceptible to intrusion by hackers. New technology has been developed to combat the threat of hackers, but no system is hacker-proof yet.

Everyday people send sensitive data over the internet via social media or email. Although these networks may be strong against computer based attacks, no thought is put into the fact that once your phone or laptop is left open anyone can simply open your messages and view the data you’ve sent to others in plain text.

This is where local encryption is useful. Text and files can be encoded, so that even with access to the device it is impossible for anyone to make sense of the information. There are many different encryption algorithms, most popular being: MD5, RSA, SHA-1 and 3DES.

**CONTEXT DESCRIPTION**

Encryption is the process of encoding a message or information in such a way that only authorized parties can access it and those who are not authorized cannot.

When people send information over the internet, messages are encrypted and sent over a network to the recipient’s router, where it is decrypted and returned to the recipient’s device and represented as a plain text or file.

This works well for ensuring that external hackers cannot get intelligible data from information sent over the network.

This encryption, however, does not protect against anyone with access to the device from viewing messages as the data is only encoded while it is in transit.

If, say, you text a friend or relative your email password (for whatever reason), and either of you gets robbed and your phone stolen. There’s no way of protecting that message that was sent. The thief now has your email. Even if there is a lock on the device, hackers have always figured out ways of cracking those, since they are normally not very complex.

If the message sent was in an encoded form it would add an extra layer of protection on the information. There would be no way of determining what the original message was.

**PROBLEM DESCRIPTION**

* We share sensitive information regularly over plain text. This could be anything from a password to credit card information.
* Our information is only secure while it is being sent over a trusted network, by email or social media application.
* Many persons don’t have a simple screen lock on their phone, removing the first layer of security on a device
* The security of that information is comprised if someone simply takes your unlocked phone or computer (or you give it to them) and opens the email or messaging application you’ve already signed in to.
* A stolen device is the easiest way to get access to someone’s information.

**OBJECTIVE OF SOLUTION**

* Allow users to encode any text message to an unreadable form
* Allow users to create files with encoded messages inside
* Ensure that sensitive information stored on a device locally is protected against prying eyes
* Allow users to send an encrypted message and files to anyone without ever disclosing information.

**STRUCTURED CHART**

**DRIVER**

**SplashScreen**

**Decode**

**SaveToFile**

**Encode**

**MainMenu**

**LoginSignUp**

**strAppend**

**strEndsWith**

**strAppend**

**UpdateLogin**

**GoodBye**

**ShowHistorye**

**Clear**

**NARRATIVE**

**DRIVER**

1. Display a splash screen with the program name
2. Call Login()
3. If Login returns success:
4. Call MainMenu()
5. While menu option is not E:
   1. Call the function that corresponds to the choice using a switch
6. Call GoodBye()

**MAINMENU()**

1. Display Menu options  
   (A) Encode (B) Decode (C) Show History (D) Update Login Info (E) Exit
2. Accept menu option
3. Return the choice to menu

**LOGIN ()**

1. Open the userData.dat file
2. If the userData.dat is NULL or can’t be accessed
   1. user not logged in, Return login failed flag to driver
3. Read the text from the file
4. If file is empty:  
   - Prompt the user to create a username and password

* Read the username and password
* If username and password are valid:
  + Save username & password to userData.dat file
  + user is logged in, Return success flag to driver

1. If file is **not** empty  
   - Prompt the user for the saved username and password

* Read the username and password
* If username & password is **equal to** username & password from userData.dat file
  + user is logged in, Return success flag to driver

**UPDATELOGIN ()**

1. Open the userData.dat file
2. If the userData.dat is NULL or can’t be accessed
   1. Display error message
   2. Return to driver
3. Prompt the user for a new username and password
4. Read the username and password
5. Validate the username and password
6. Overwrite userData.dat file with new username & password

**ENCODE ()**

1. Create a public key by generating a single digit random number
2. Create a private key by performing a calculation on the public key
3. Prompt the user for any text to encode
4. Read the text
5. Iterate through each character in the text:
   1. Determine each letter’s ASCII integer value
   2. Add the private key to the ascii value
   3. Append the ascii value to the new encoded text
6. Set the public key as the first character in the encoded text
7. Display the encoded text
8. Update the history file with the new cypher
9. Call SaveToFile()

**DECODE ()**

1. Prompt the user to decode from a file or from text:
2. From file:
   1. Prompt the user for the filename
   2. Accept the filename
   3. Validate the file’s extension
   4. Open the file
   5. Read the text from the file to a string
3. From text:
   1. Prompt the user for cypher text
   2. Accept a string
4. Obtain the public key from the first character in the string read above
5. Obtain the private key by performing a calculation on the public key
6. Iterate through the text starting at the second character:
   1. Split the text into 3 digit asci strings
   2. Convert the 3 digit string to an integer
   3. Subtract the private key from the asci value
   4. Convert the 3 digit integer to character
   5. Append the character to the decoded text
7. Update the history file with the cypher
8. Display the decoded text

**SAVETOFILE()**

1. Accept a textcypher as parameter
2. Prompt the user for a file name
3. Accept the filename
4. Validate the file name
5. Add the ‘.zen’ extension to the file name
6. Create the file
7. Write the textcypher’s encoded text to the file

**ADDTOHISTORY()**

1. Accept a textcypher as parameter
2. Locate/create the local history.cyph file
3. Append the textcypher to the file

**SHOWHISTORY ()**

1. Locate the local history.cyph file
2. Open the file
3. Read the data from file into a textcypher array
4. Iterate through the array:
   1. Display each item
5. Prompt the user to clear the history
6. Accept the resonse
7. If the response is Y or y:
   1. Call Clear()

**CLEAR()**

1. Delete the local history file

**GOODBYE ()**

1. Display goodbye message
2. Exit the program

**RECORD DEFINITION**

1. **TextCypher** Record 🡪 Contains information about the encoding

**Record TextCypher**

**original: String**

**encoded: String**

**dateTime: String**

**EndTextCypher**

1. **User** Record 🡪 Stores the current logged in user

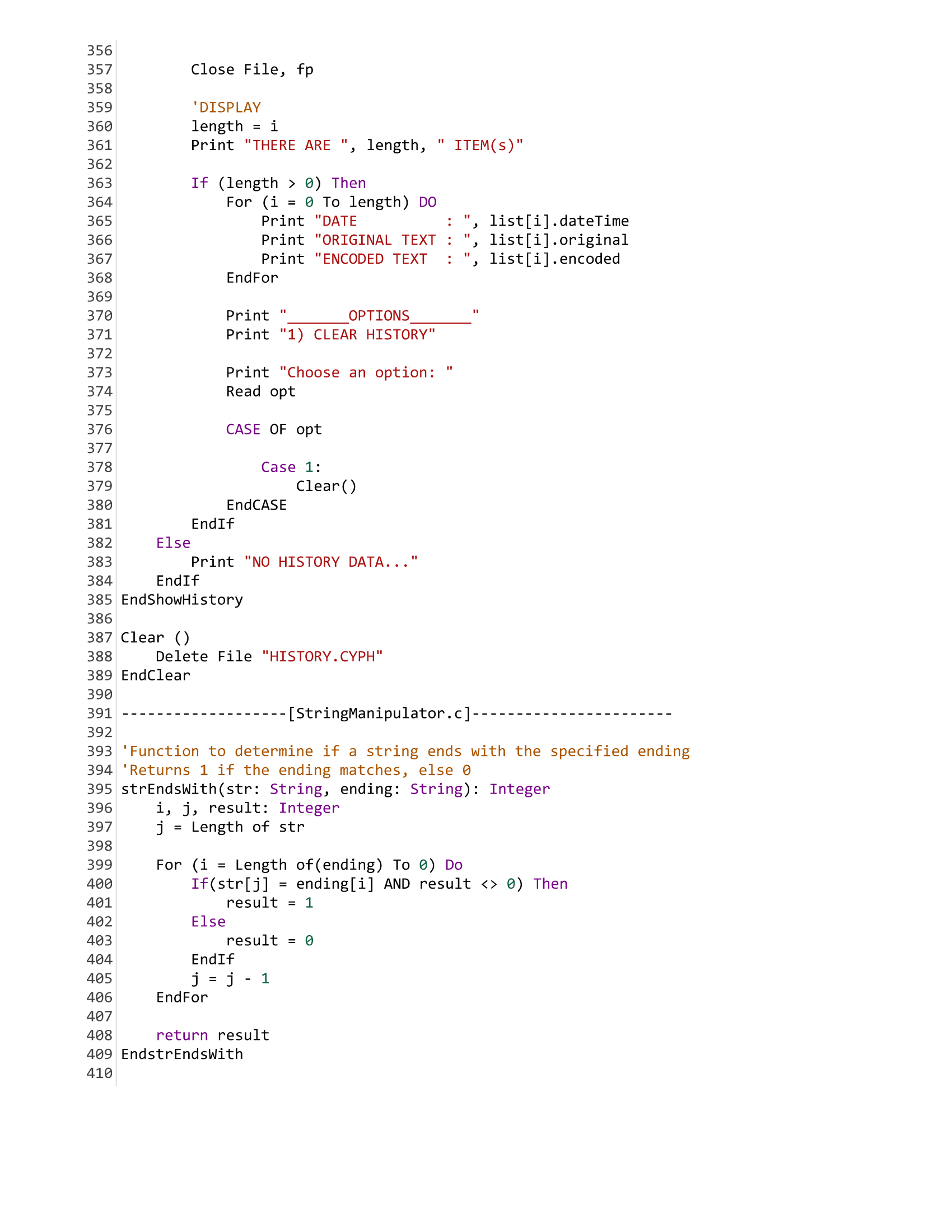
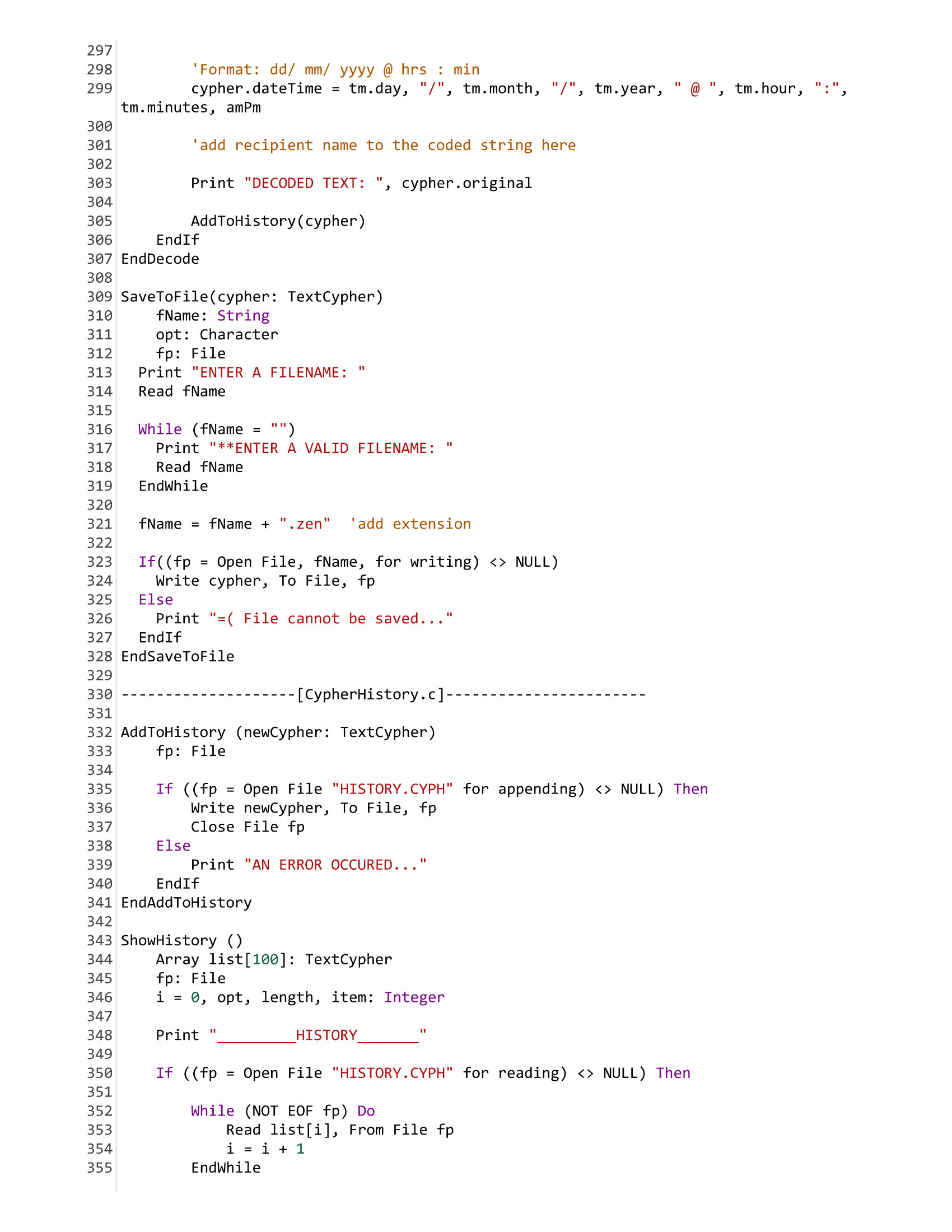
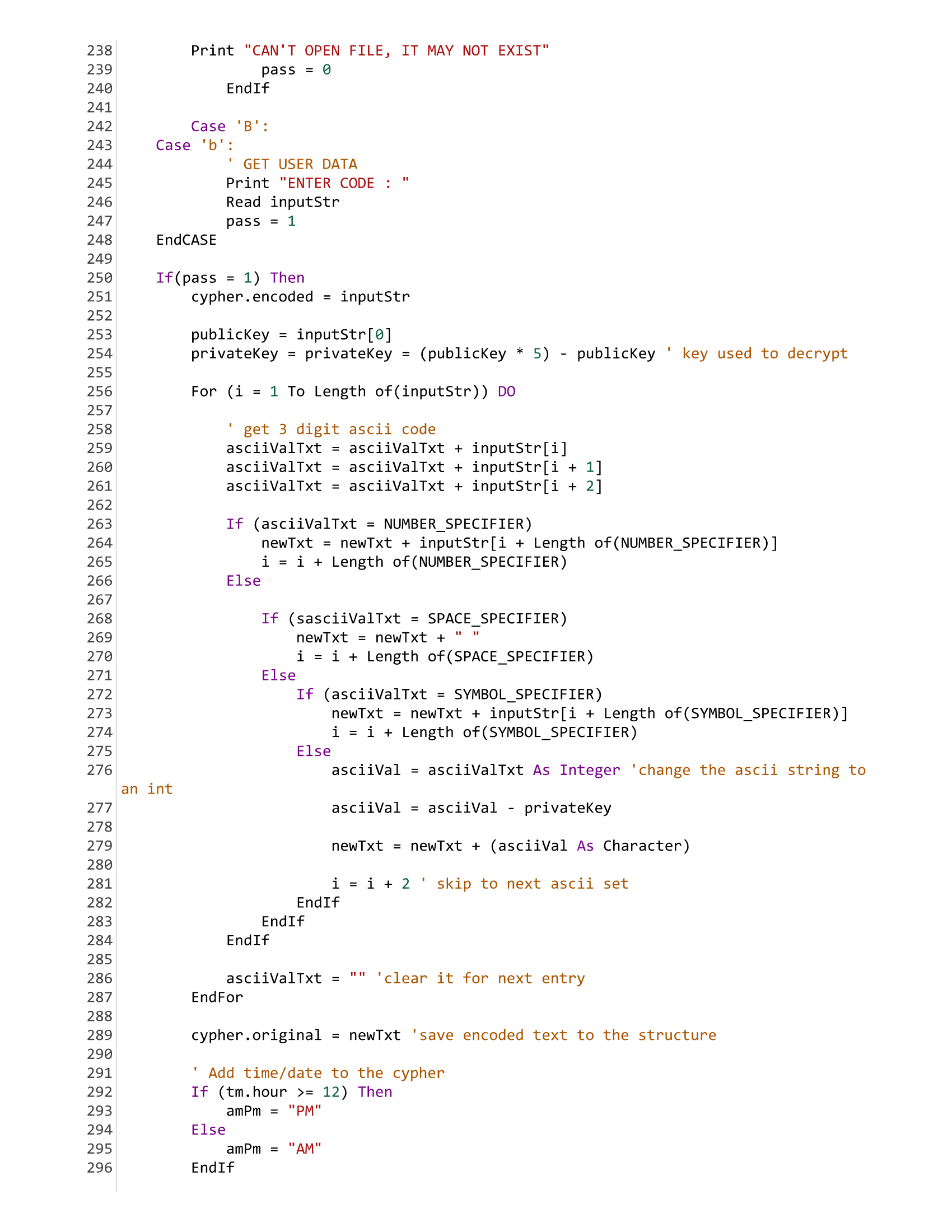
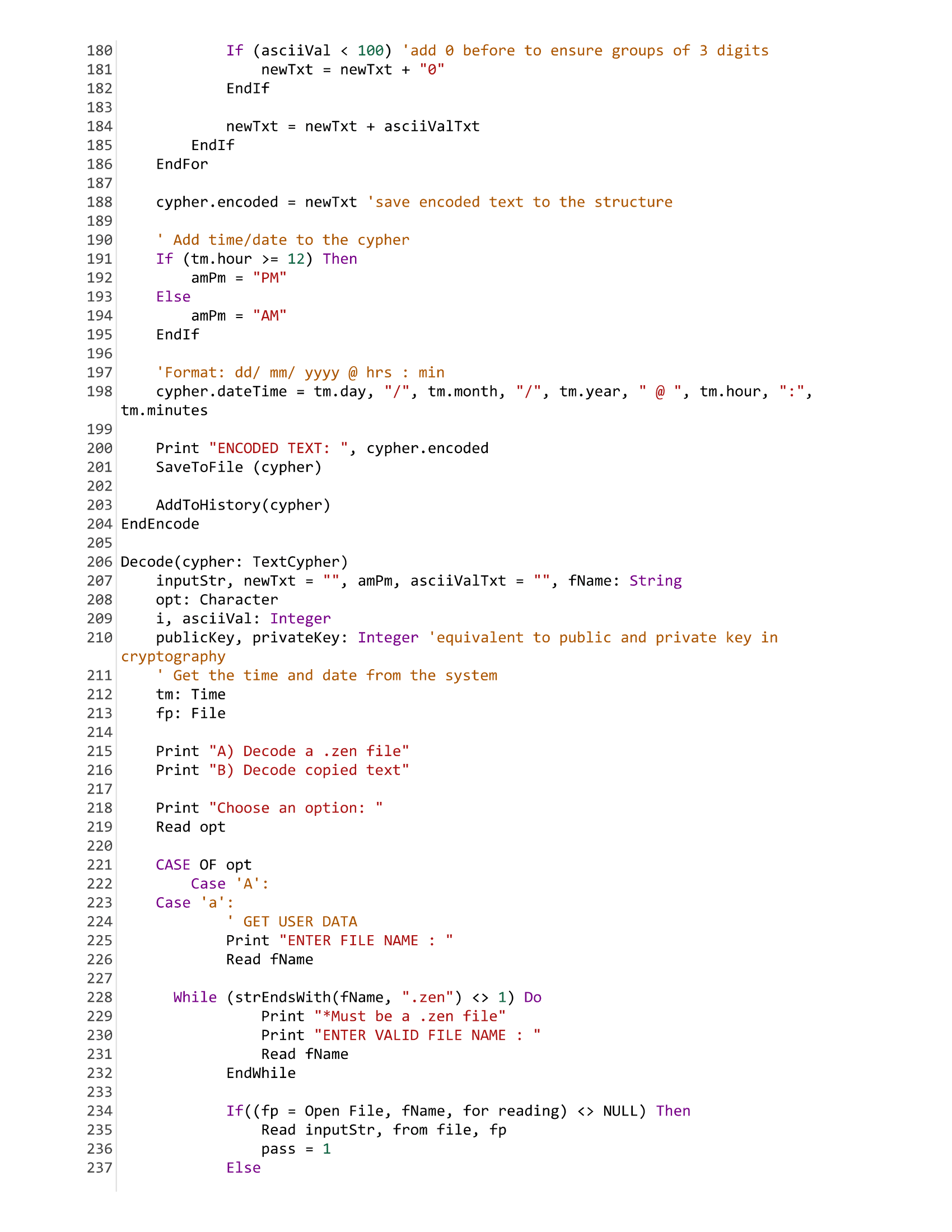
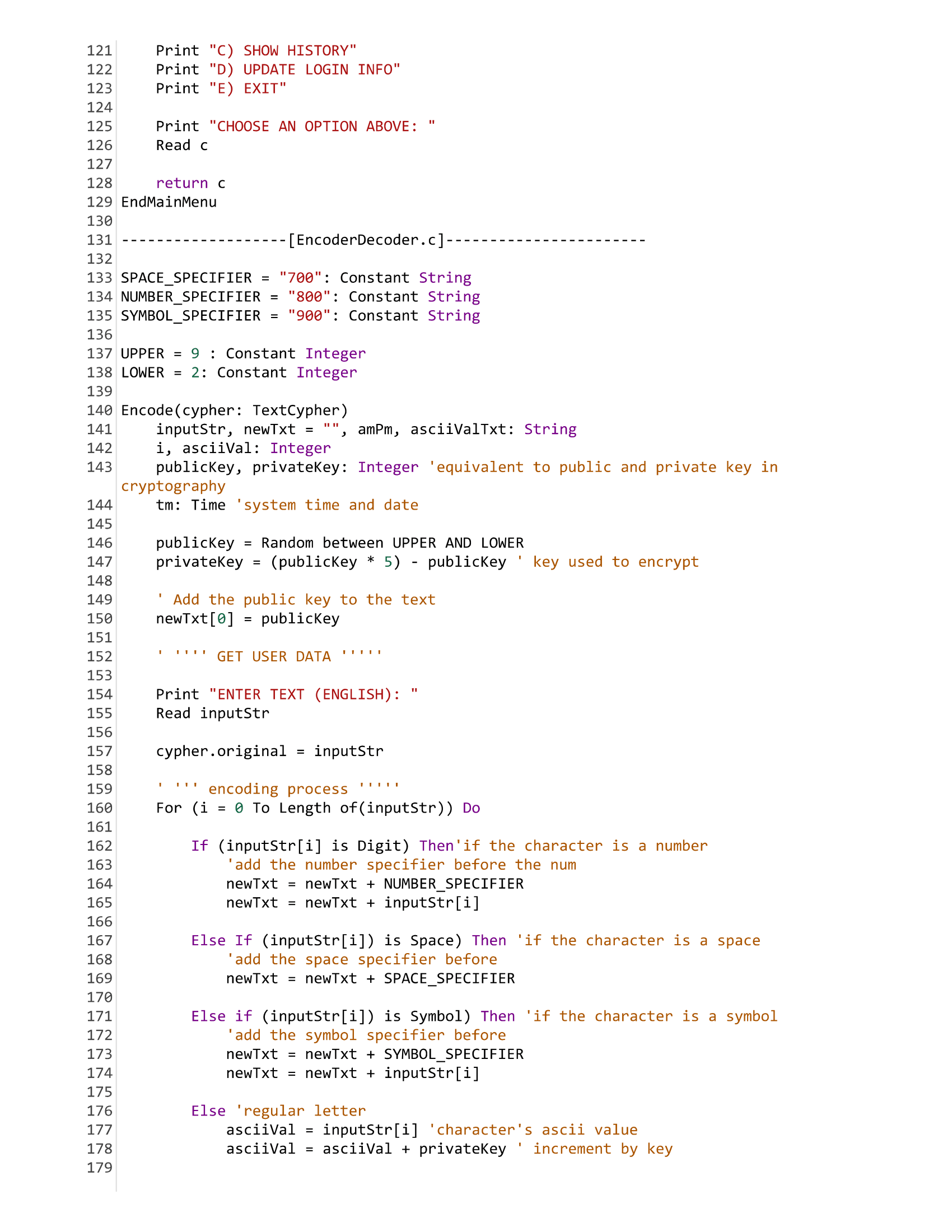
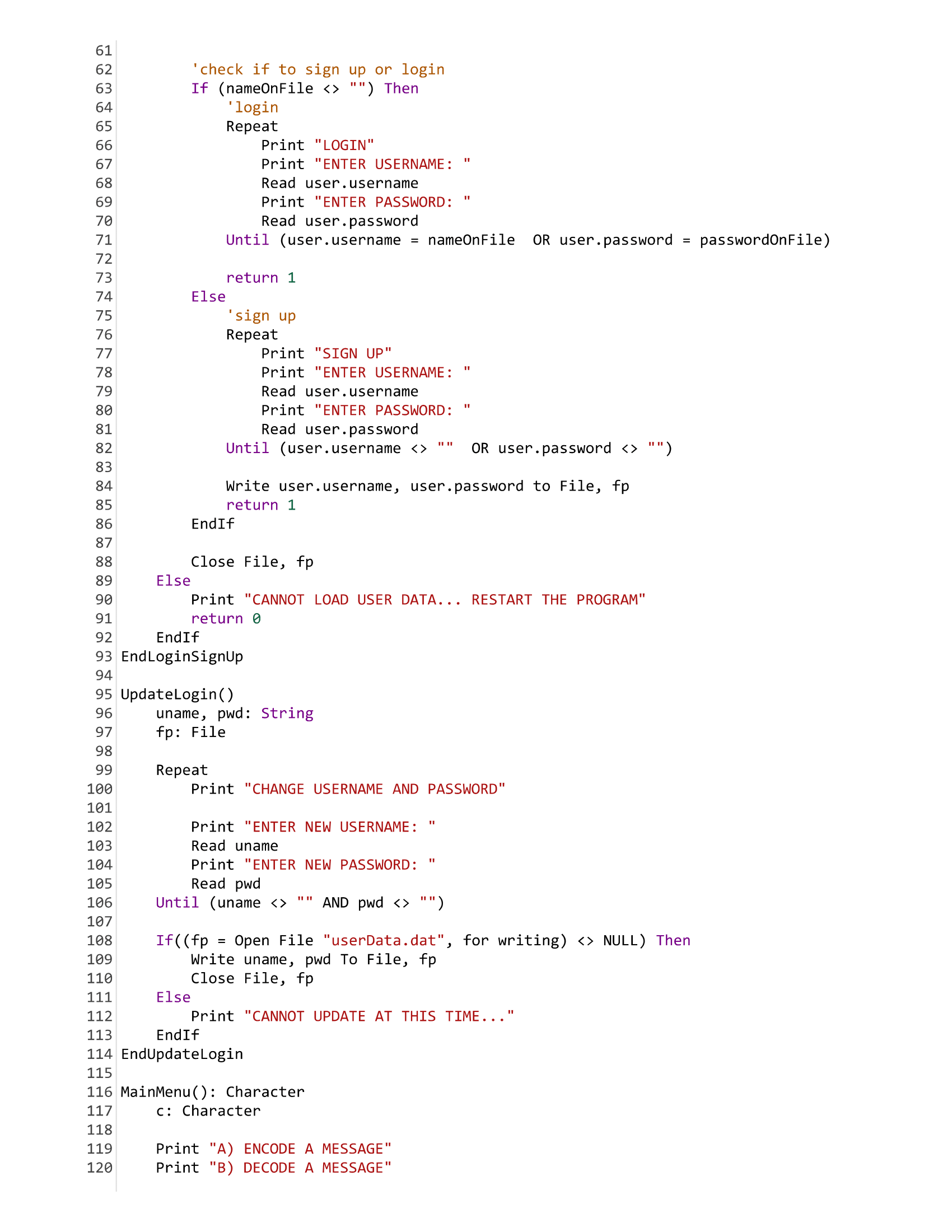
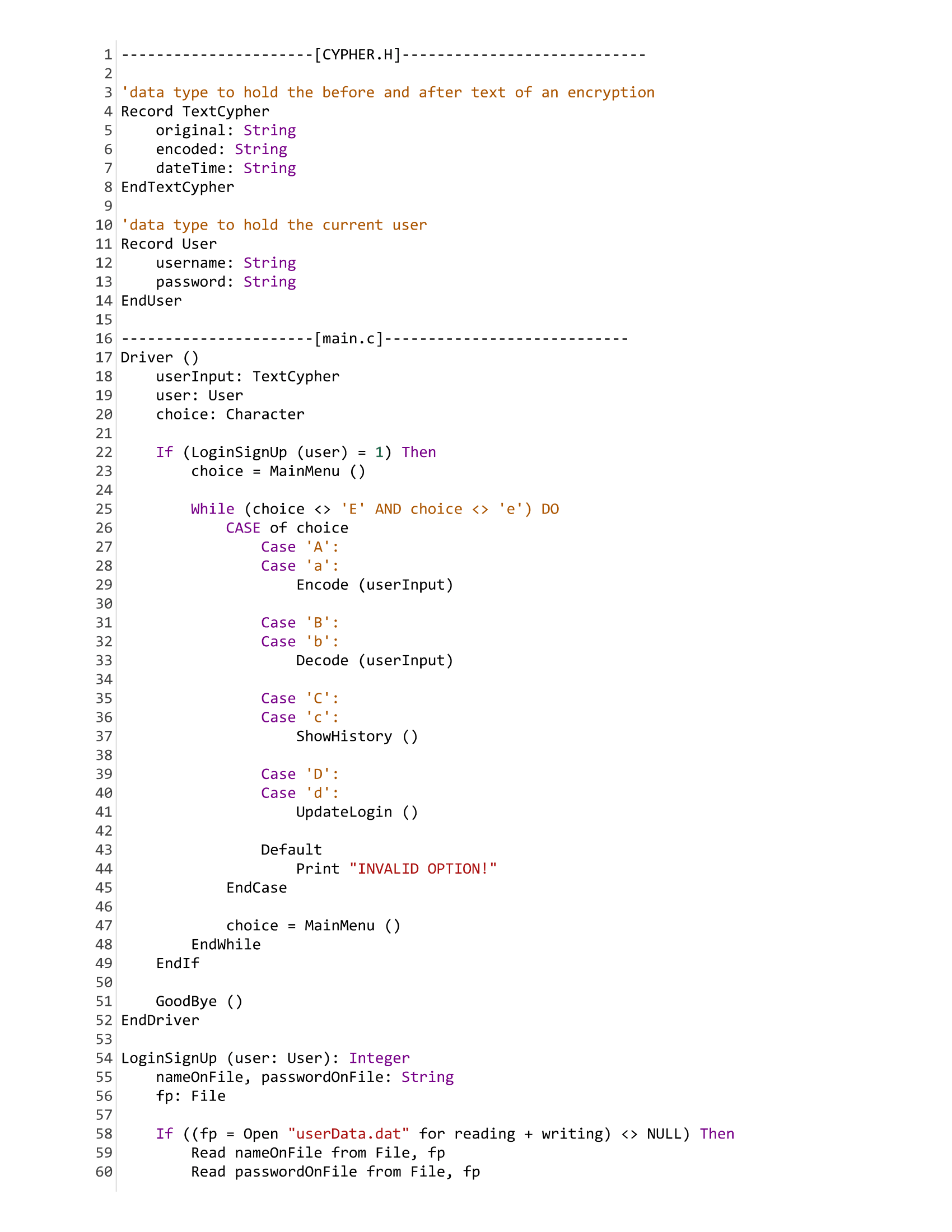
**Record User**

**username: String**

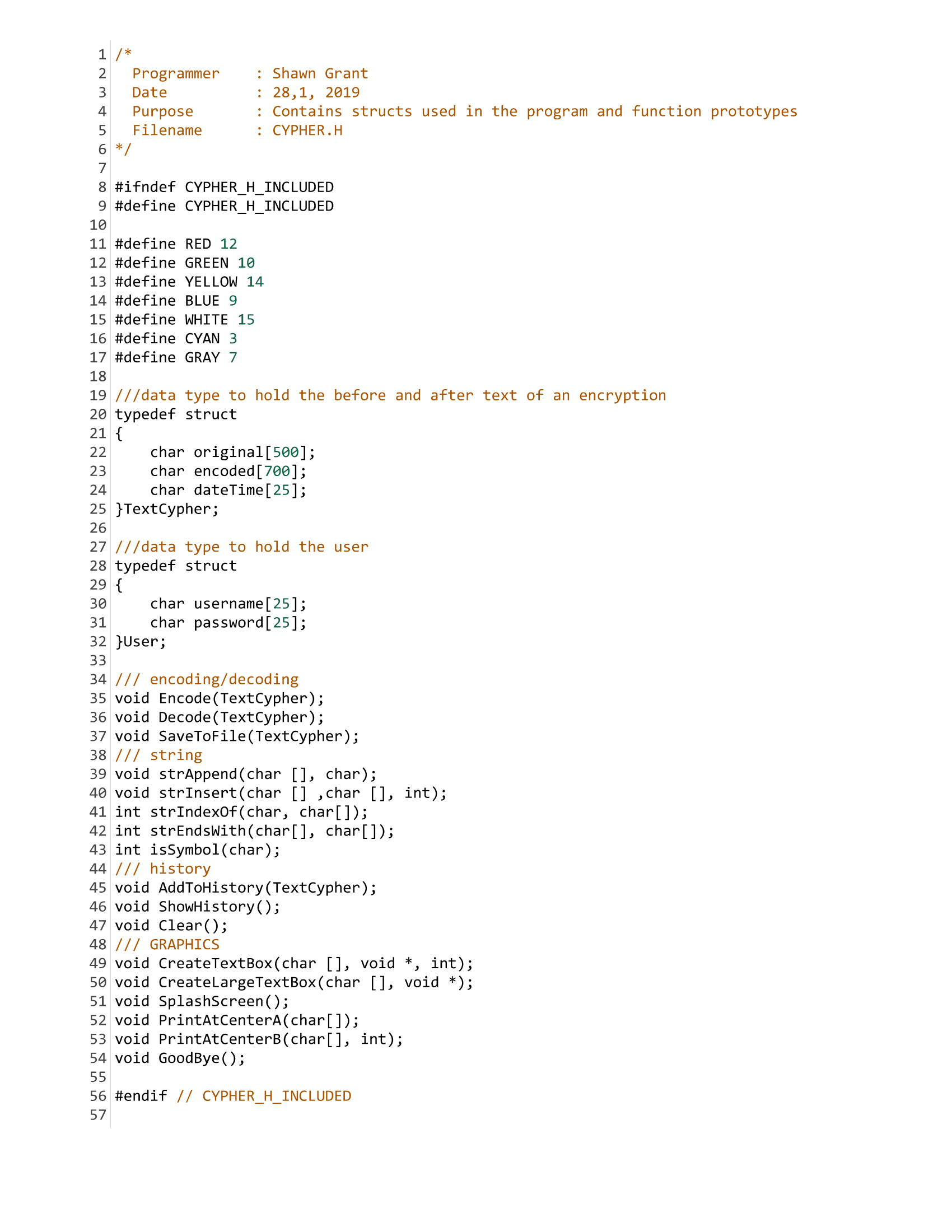
**password: String**

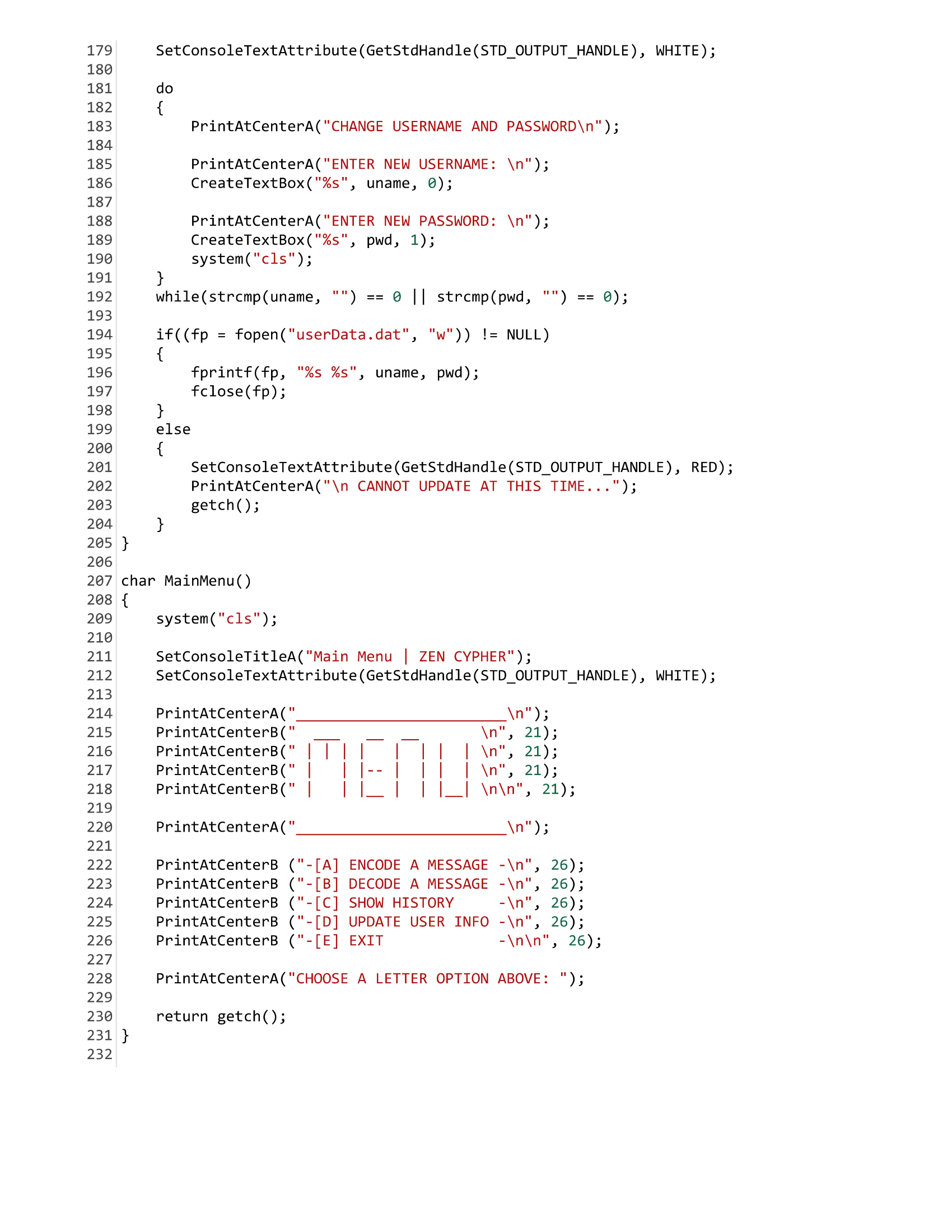
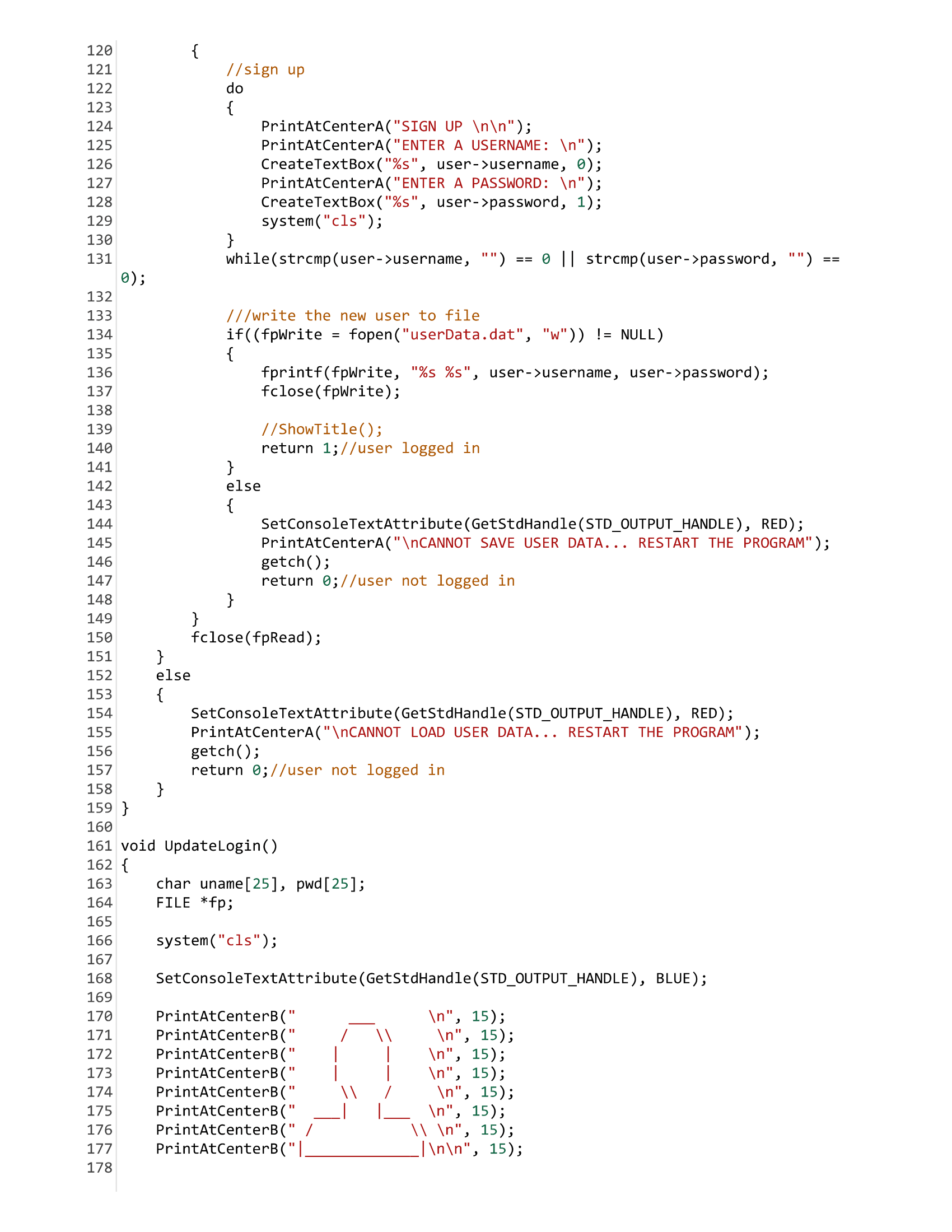
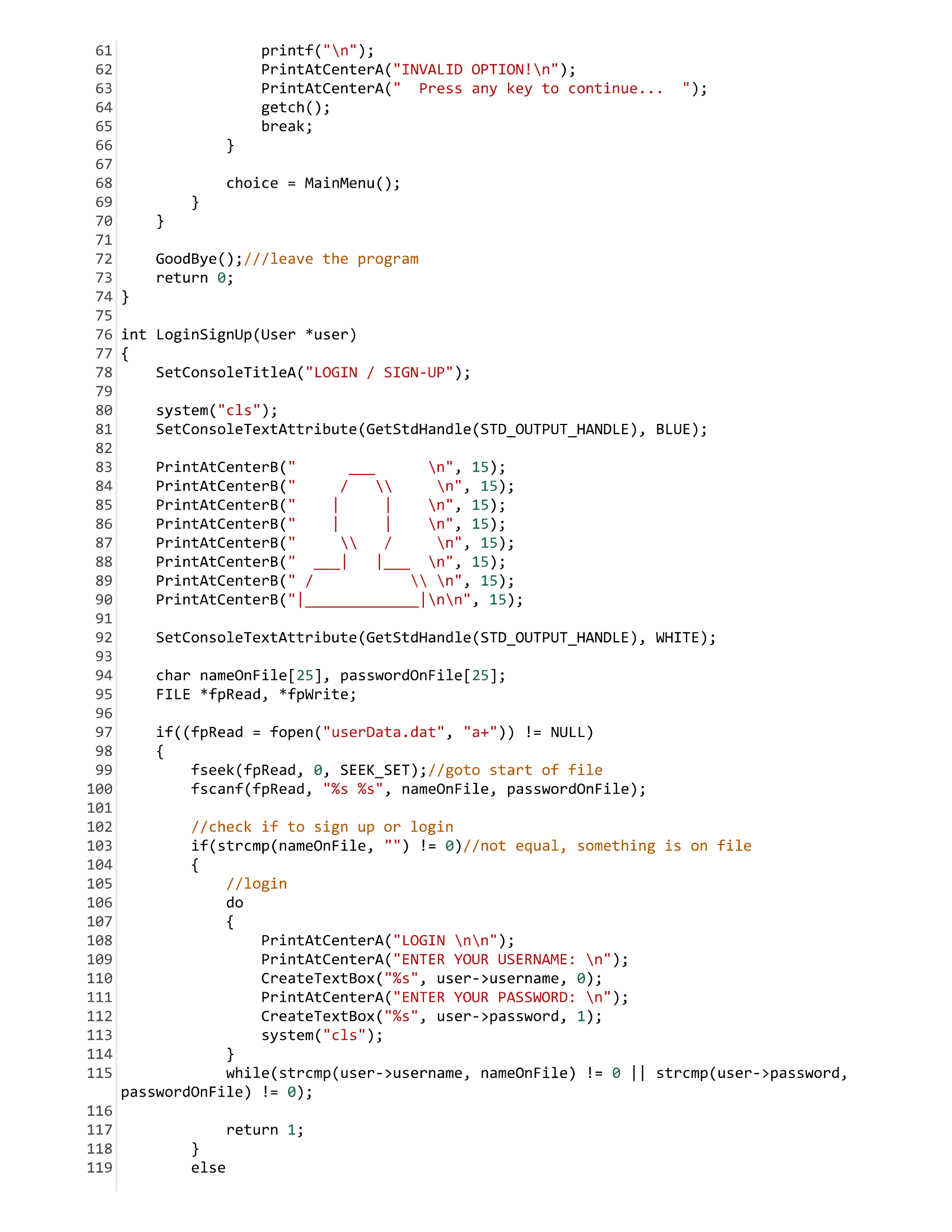
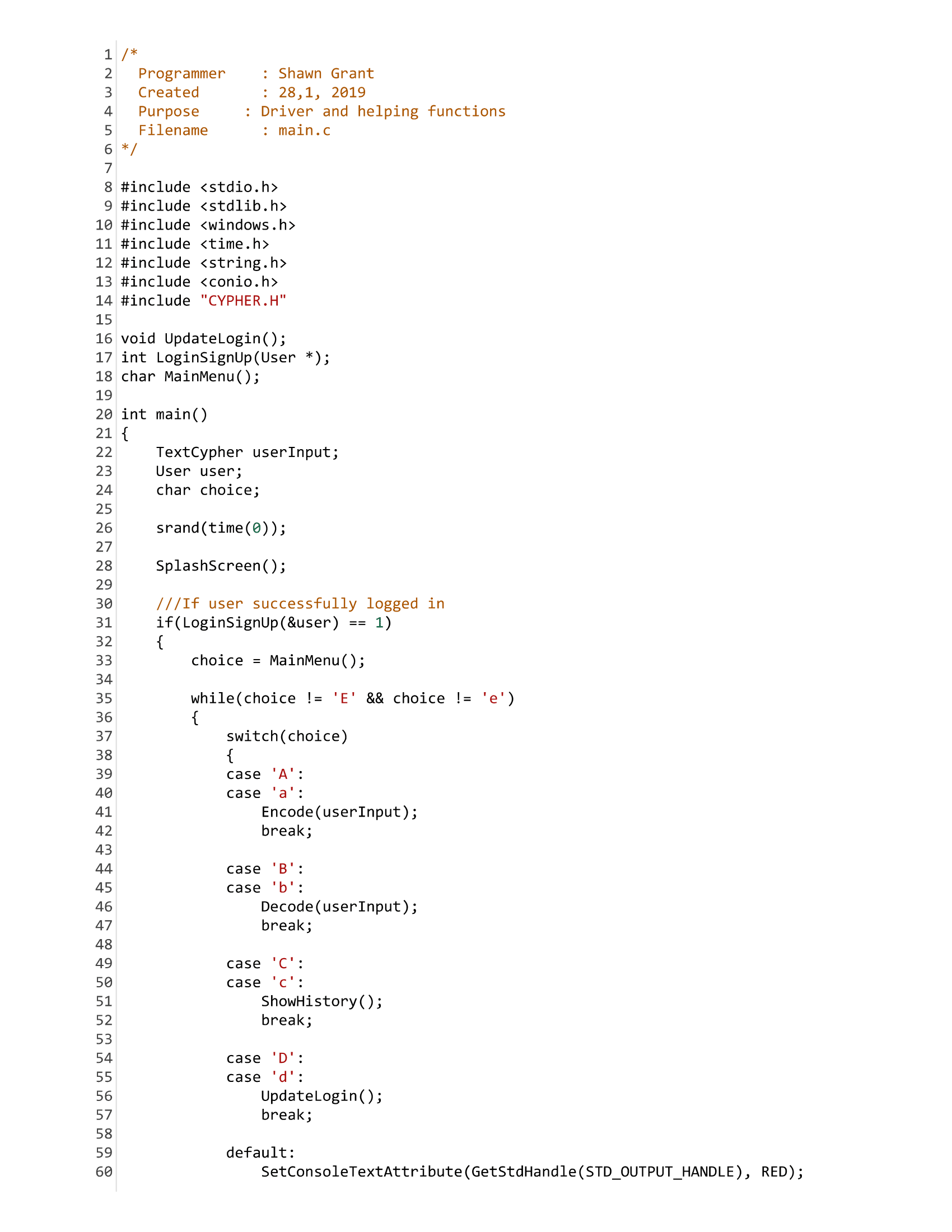
**EndUser**

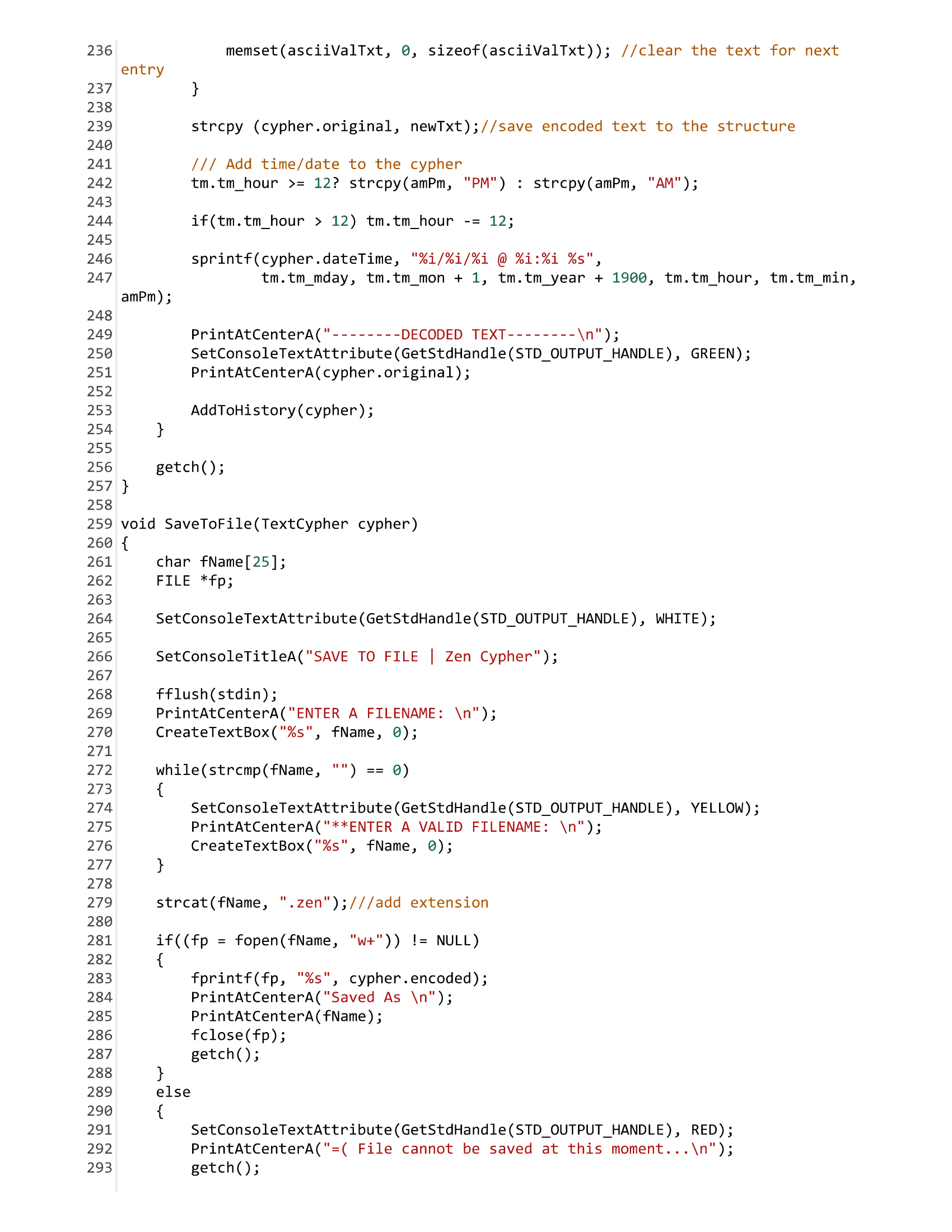
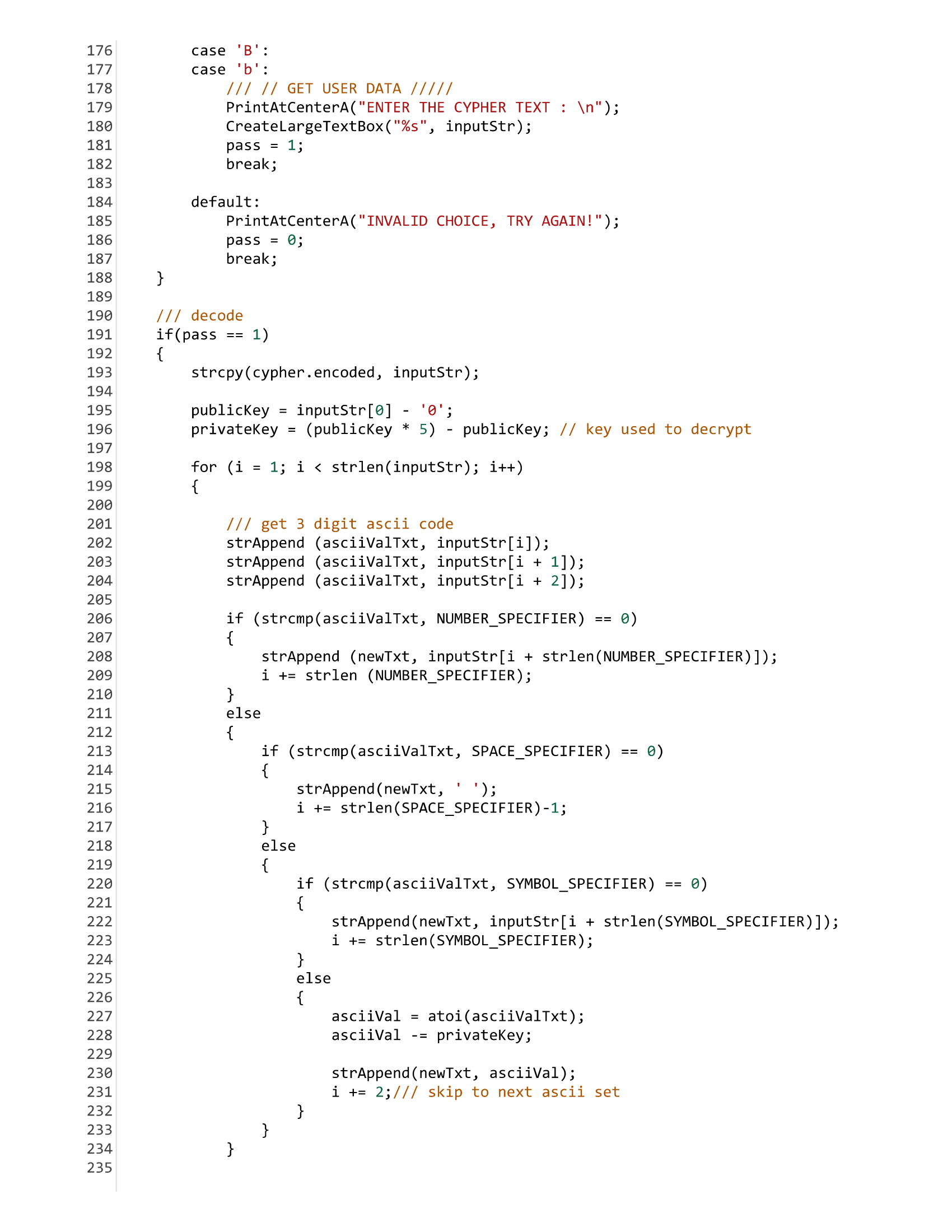
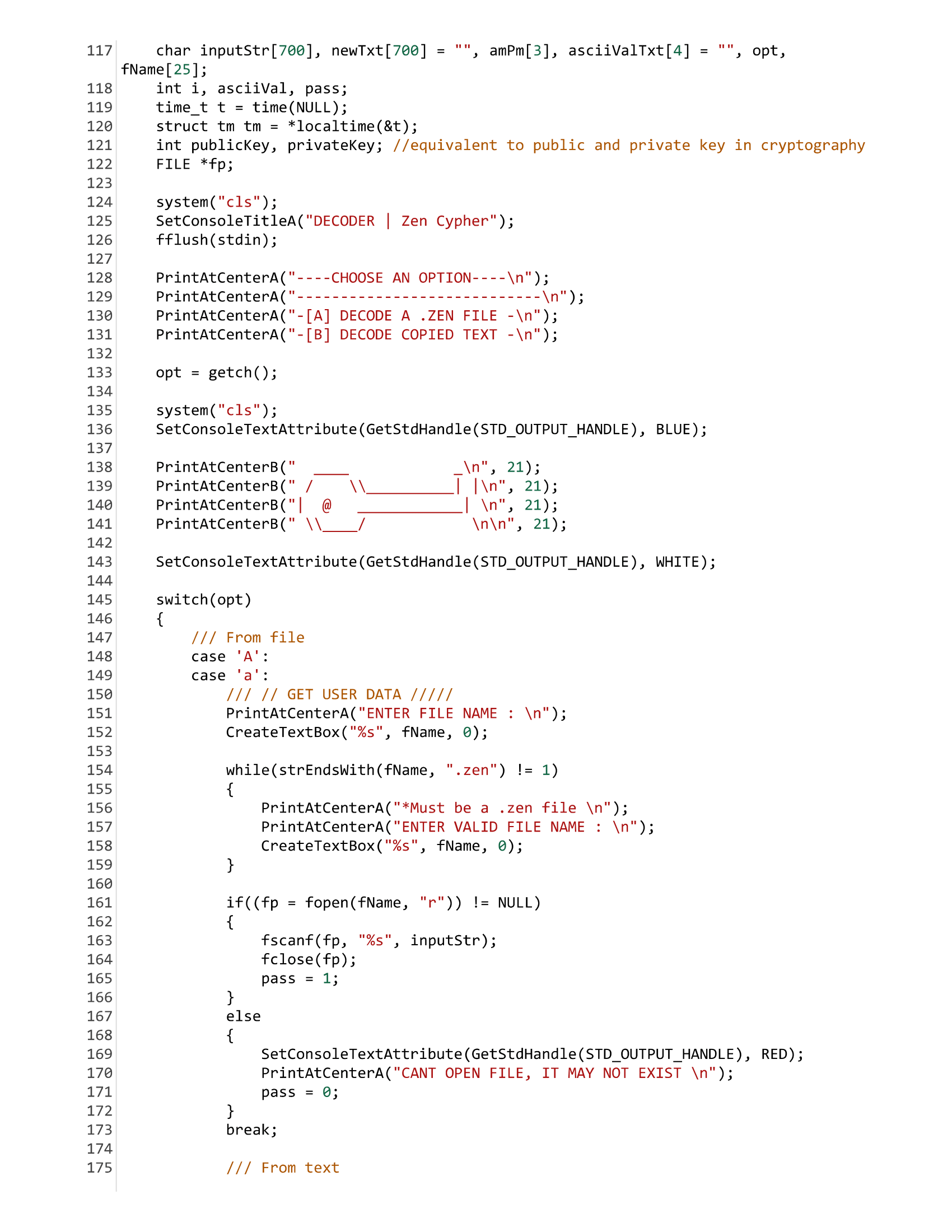
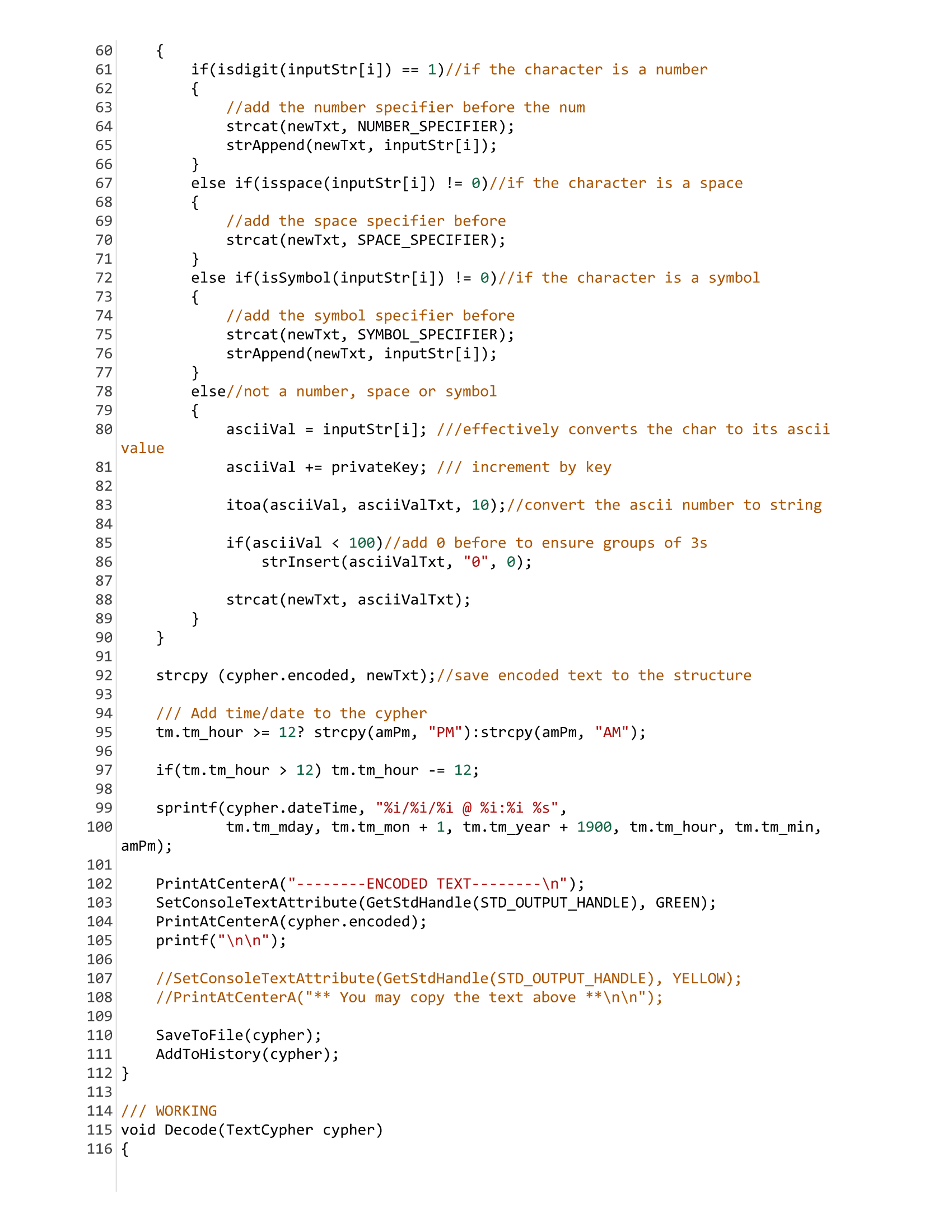
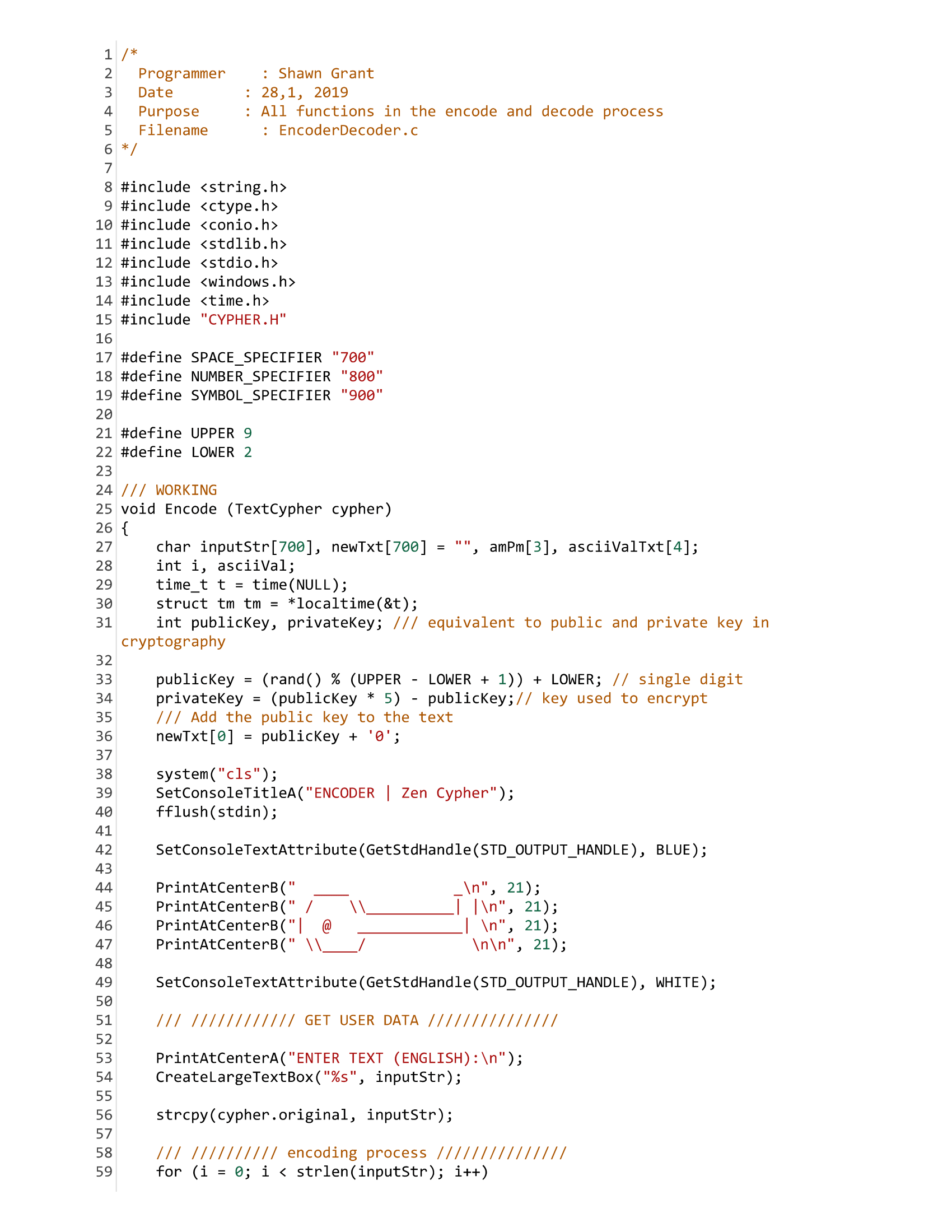
**PSEUDOCODE**

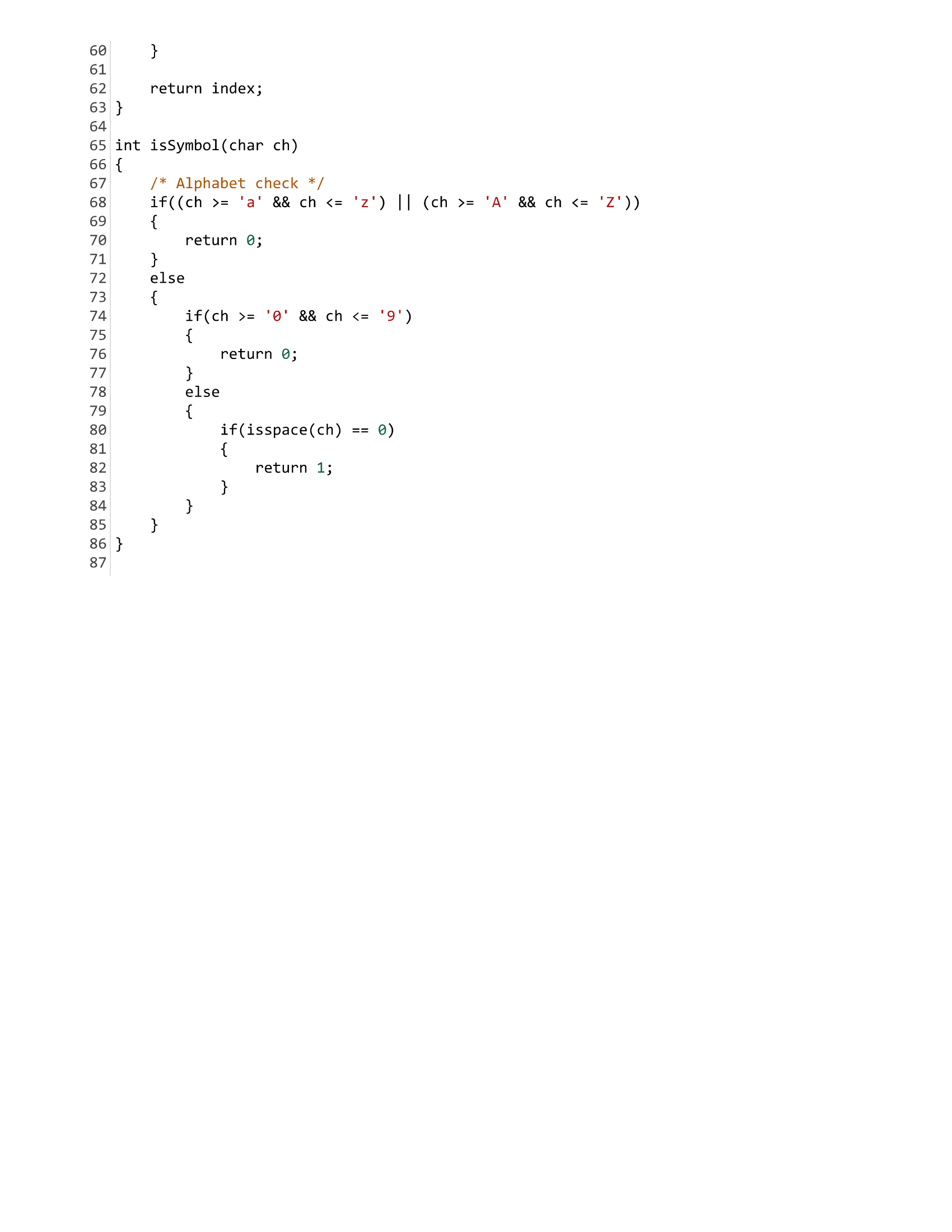
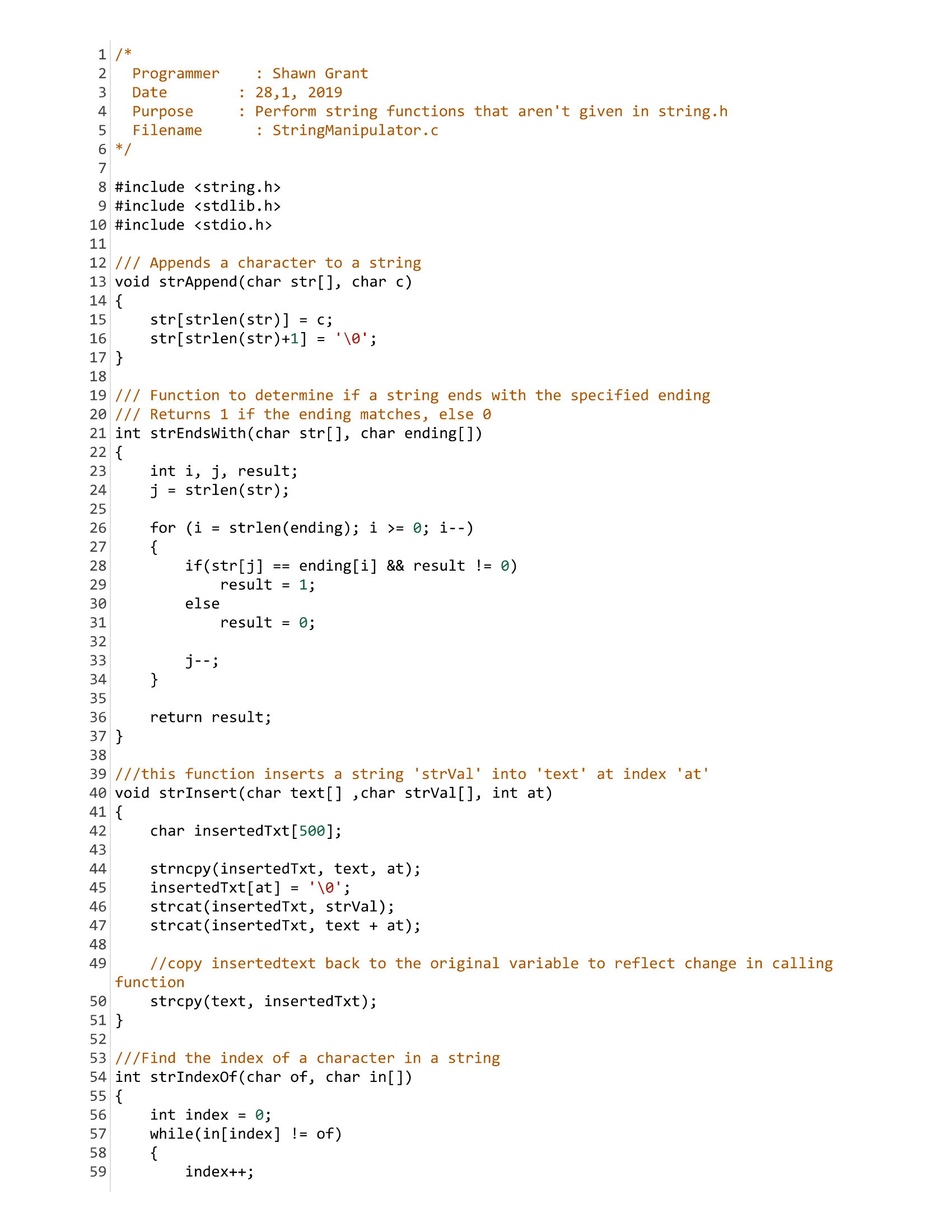


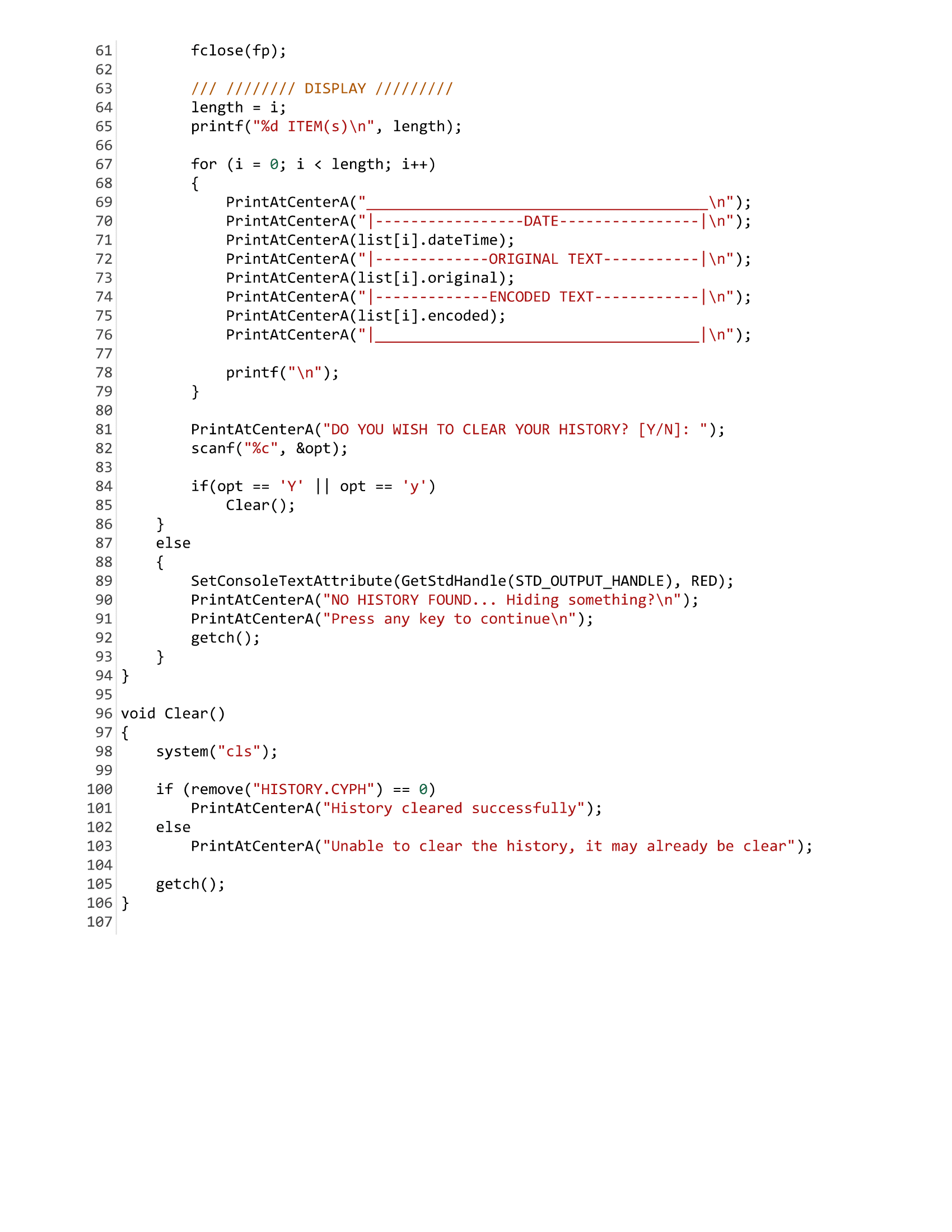
**C CODE**

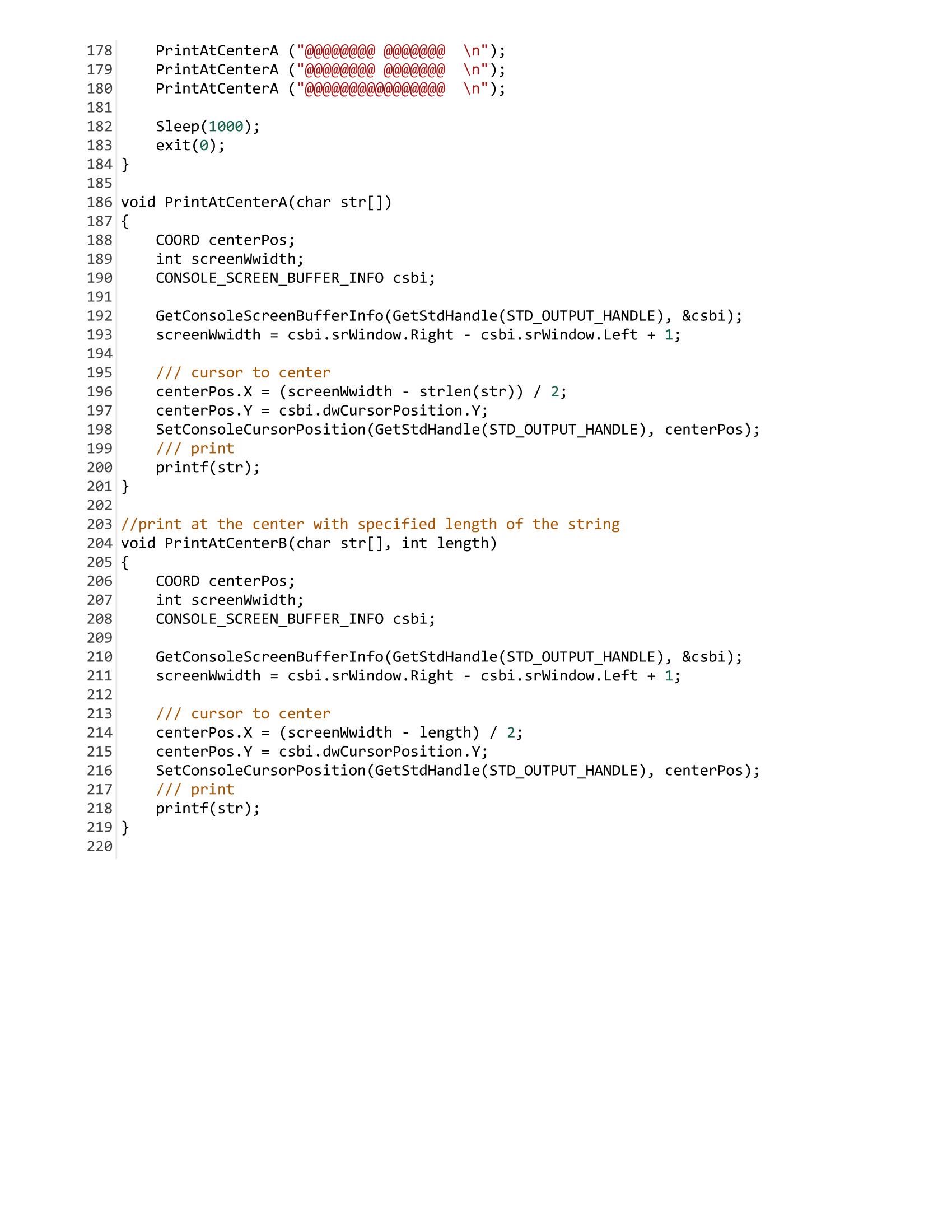
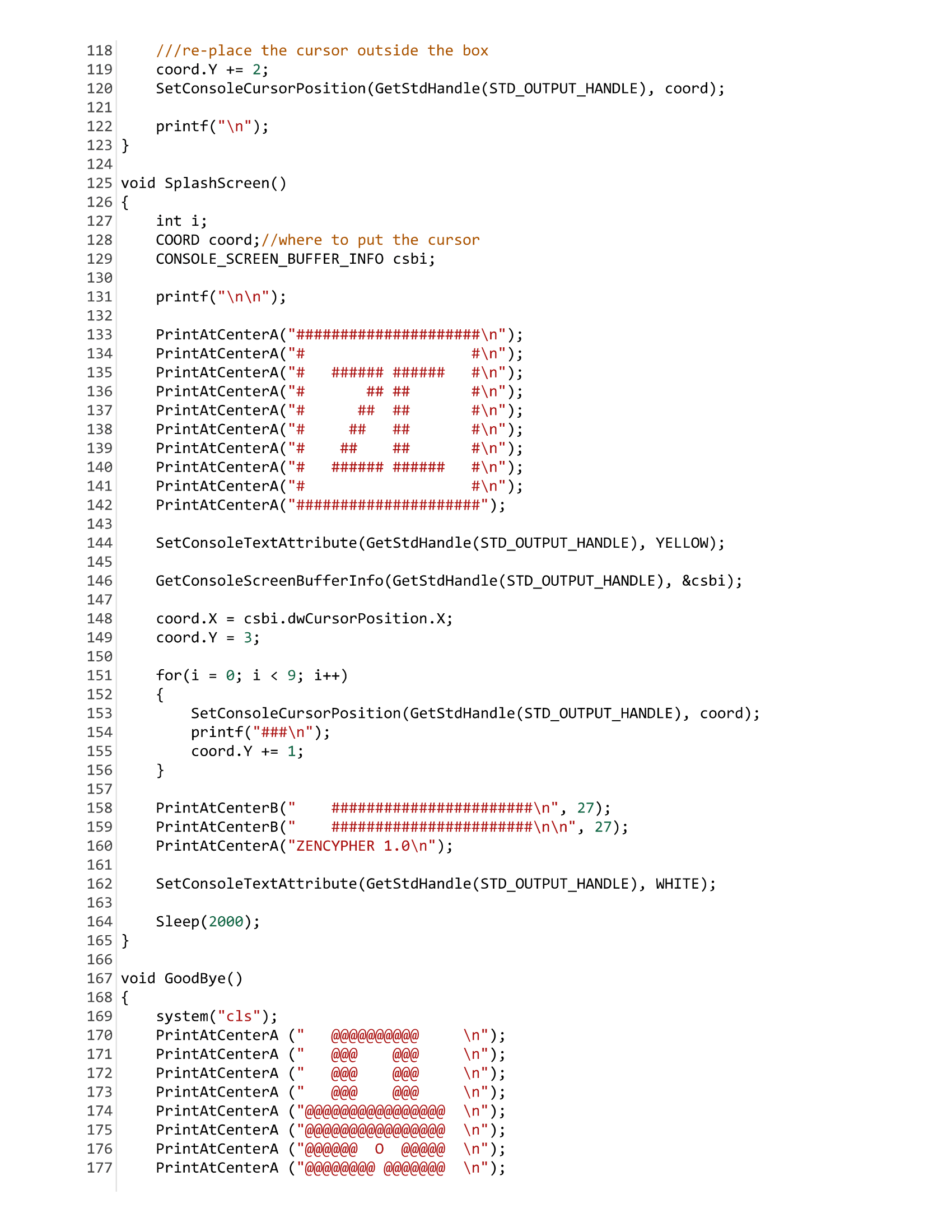
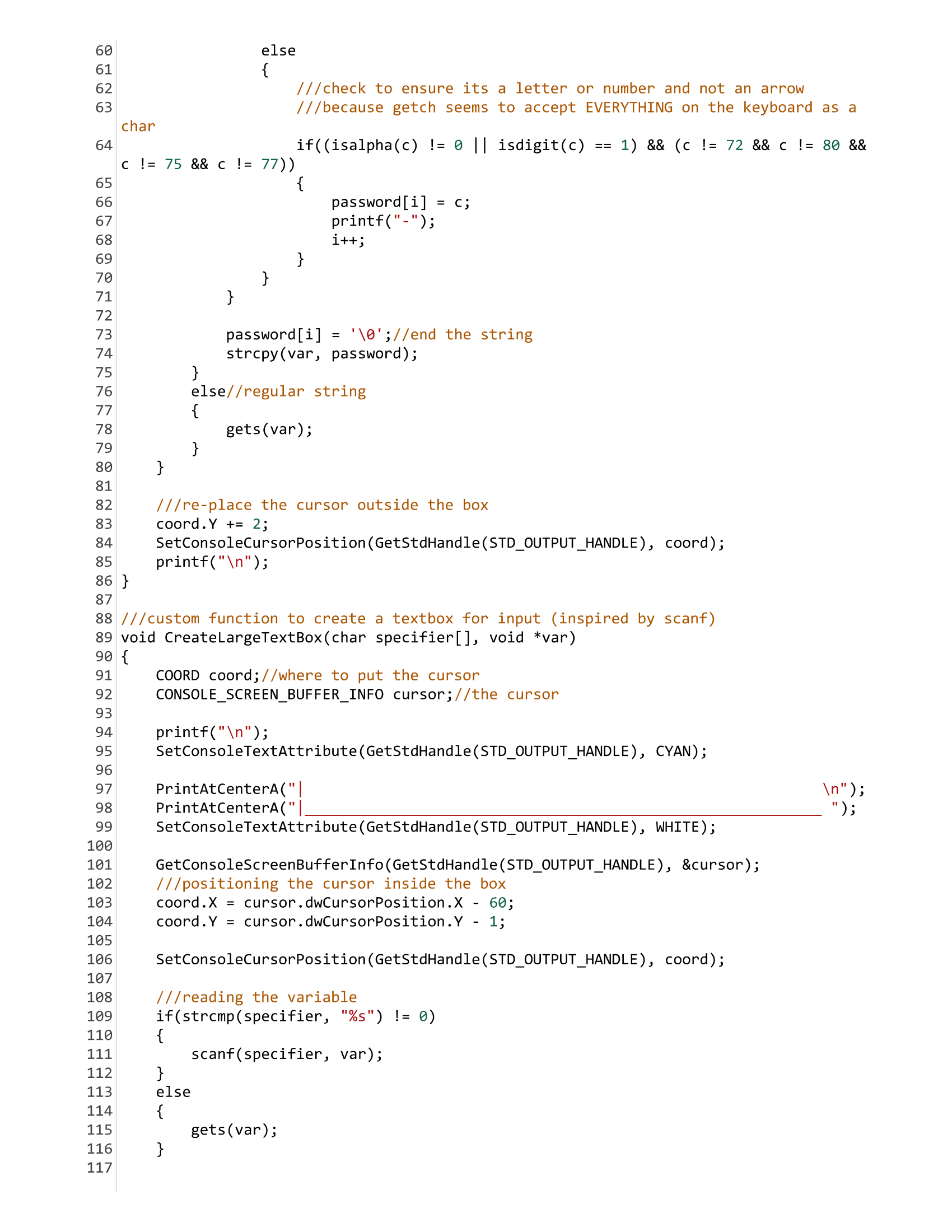
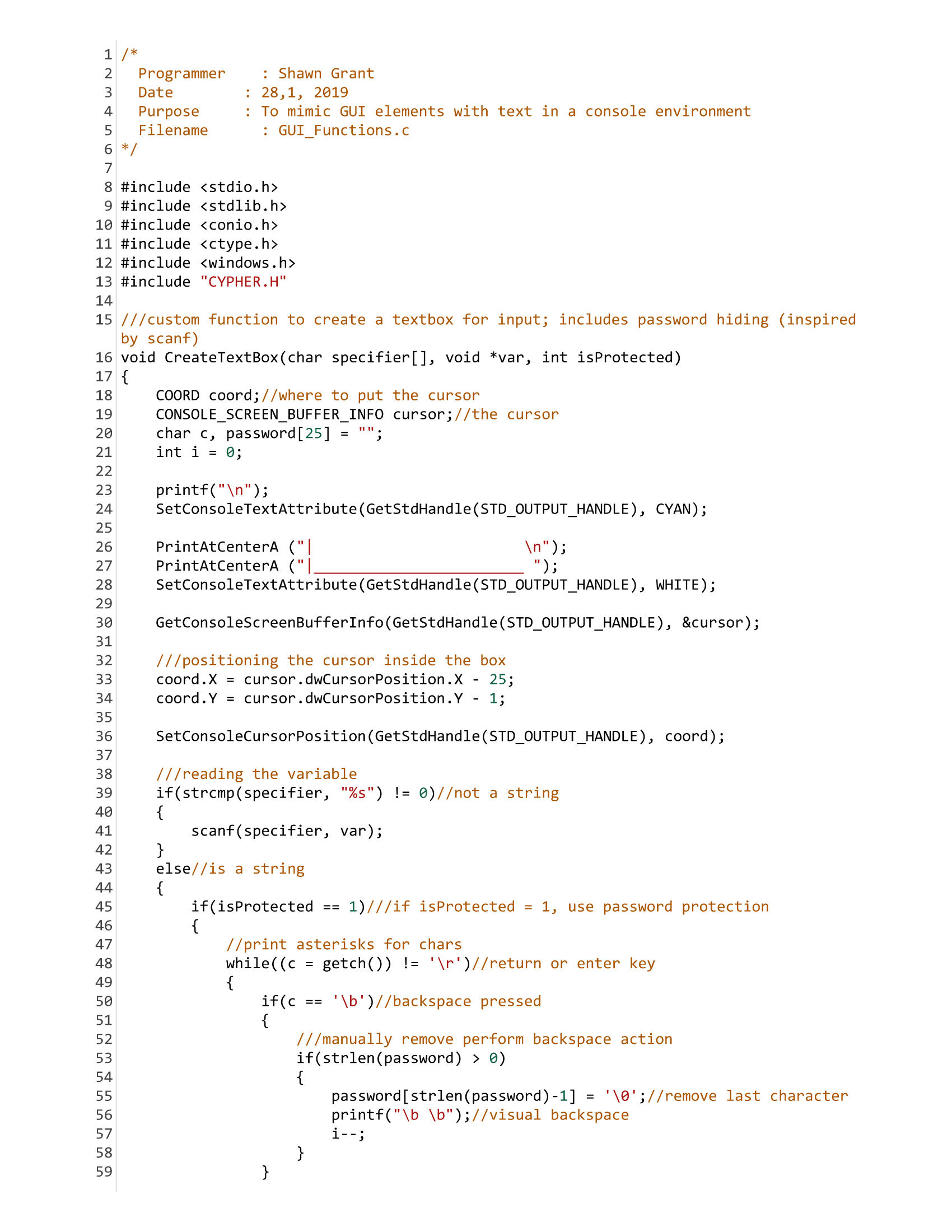












**CODE TEST PLAN**

Input values and their expected output results, to test that the program works.

**SignUp**

|  |  |
| --- | --- |
| Username | Password |
| shawn | pwd123 |

**Login**

|  |  |
| --- | --- |
| Username | Password |
| shawn | pwd123 |

**Encode**

|  |  |  |
| --- | --- | --- |
| Menu choice | English Text (input) | Encoded Text (output) |
| ‘A’ or ‘a’ | “Hello World” | 2080109116116119|095119122116108 |

**Encode**

|  |  |  |
| --- | --- | --- |
| Menu choice | Encoded Text (input) | English Text (output) |
| ‘B’ or ‘b’ | 2080109116116119|095119122116108 | “HelloWorld” |

**TEST PLAN RESULTS**

**CONCLUSION**

The ZenCypher program had many issues during development leading to delay in completion and certain features being excluded from the final build. The original cypher method had to be abandoned due to limitations of the C programming language in favour of an RSA-like system using public and private keys.

The program’s user interface (UI) was very exciting to build and provided a constant motivational challenge to think outside the box, in order to create GUI-like elements inside a console application. Originally it was intended for the user to have the ability to use the mouse to select items on screen, but this feature had to be discarded because of its complexity.

A loading animation was also to be implemented but was removed because it was not consistent with the subtle UI of the rest of the program.

In future updates to the software, more features will be added such as:

* A networking component to allow for sending of encrypted files/text over the internet or Local Area Network(LAN)
* The ability to choose different encryption schemes
* The option to ecrypt files already on the computer

**APPENDIX**

Research References

* What is Encryption - <https://searchsecurity.techtarget.com/definition/encryption>
* Types of Encryption - <https://www.toptenreviews.com/software/articles/encryption-algorithms/>
* Public Key Cryptography –   
  <https://www.globalsign.com/en/ssl-information-center/what-is-public-key-cryptography/>
* RSA Algorithm -   
  <https://www.geeksforgeeks.org/rsa-algorithm-cryptography/>

Programming References

* ASCII Character codes - <https://ascii.cl/>
* Convert char to ASCII in C - <https://stackoverflow.com/questions/1472581/printing-chars-and-their-ascii-code-in-c>
* Password Hiding - <https://www.engineersgarage.com/c-language-programs/user-authentication-access-secured-resources>
* C Color Codes - <https://www.programmingsimplified.com/c/graphics.h/colors>

Other References

* Creating auto-run files - <https://msdn.microsoft.com/en-us/library/windows/desktop/cc144206(v=vs.85).aspx>
* Install Creator - <https://install-creator.en.uptodown.com/windows>