Alan Thomas

CSI 130-01CA  Ch3-27:Project#1



Applying the 5 Steps of Problem Solving to Encryptinator

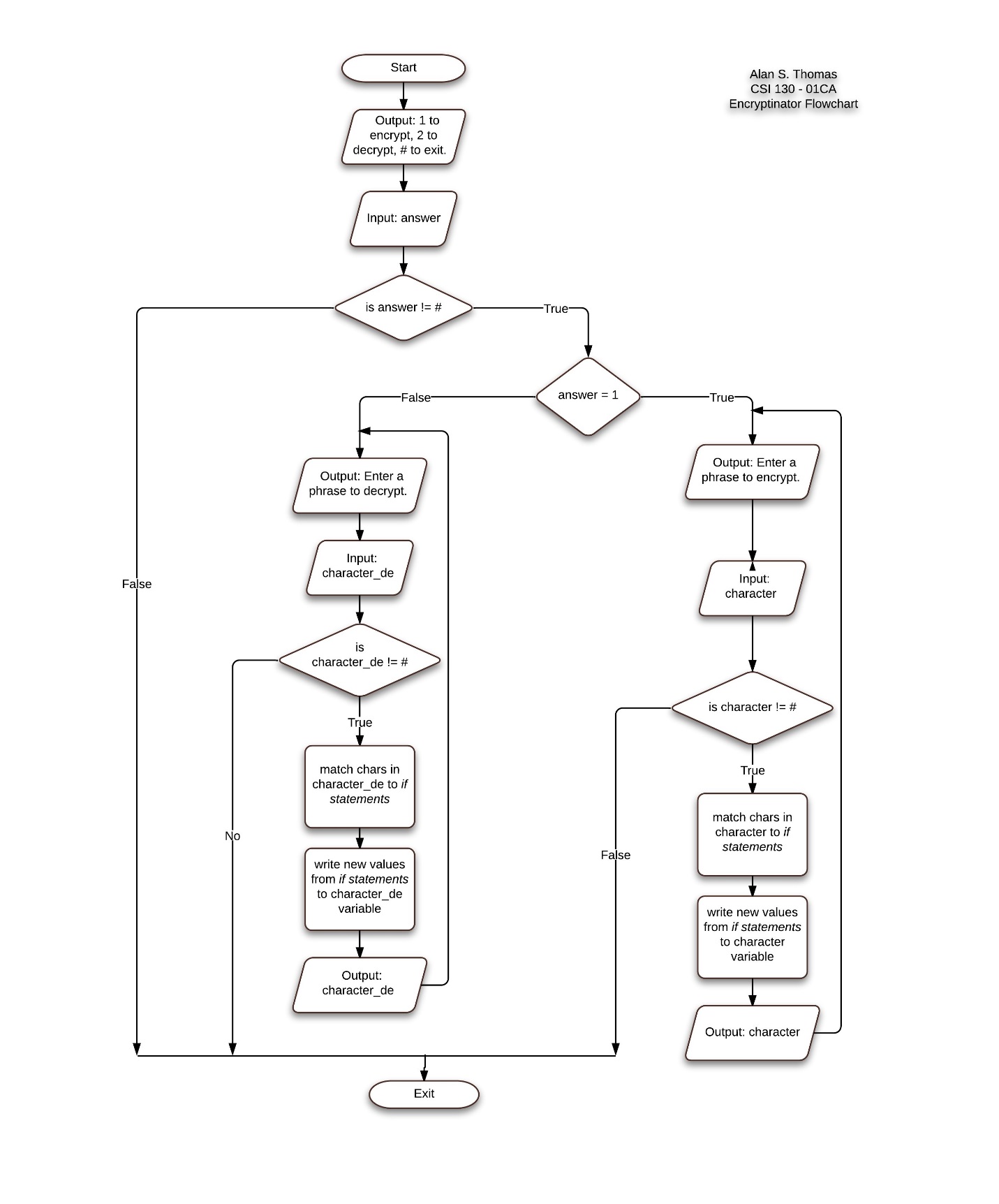
Step #1) Understand the Problem

Project 1 presents us with a scenario where a user can input a character or characters and have an encrypted output. The problem dictates we take characters as input and add seven to its numerical position in the alphabet. For example, the character “a” would hold the value 1. When using this program we will add 7 to the value of “a” resulting in 8. The letter “b” once encrypted would have a value of 9, and so forth. There are also several other rules in regards to punctuation. A period will hold the value of A, commas hold the value of B, question marks hold the value of C, and exclamation marks are represented as D. With the information given, there are several approaches we could take to implement a solution.

We know in order to solve this problem we have to be able to take input from a user than replace that input with a numerical value or a value designated by the criteria. Once the values have been modified they need to be printed to the screen in order for the user to be able to view the results.

Step #2) Design a Solution

Typically with any problem presented there are multiple solutions which can resolve the problem at hand. I felt as though *if statements* would serve this project the best. This provides an easy to code solution - If the input is ‘a’ than output this. I contemplated utilizing a dictionary list. I’m familiar with them in Python however, I’ve never coded in C++ so I stuck with what we’ve learned thus far. A dictionary could have reduced the amount of coding used. When less coding is used it inherently minimizes the amount of mistakes that can be made. However, I think the pros and cons of using either a dictionary or *if statements* was negligible*.* On the following page a flowchart can be viewed to see the algorithm used in the design of the solution. The solution called for encryption, I also implemented a decryption function in the program. Assuming a program like “Encryptinator” was being developed for a client they would assumedly require the ability to decrypt as well.



Step #3) Verify the Solution

To verify the design of my solution would perform as intended I conducted code tracing. Using the values; AZ, amzM, dog, is, barking.,?!+># I conducted the trace. The results can be seen below. As you can see the output is printed to the user up until the final phrase. This phrase contains a ‘#’. The guidance provided in this problem dictated if a ‘#’ is detected in the input the program exit.

|  |  |
| --- | --- |
| character  (passed through *if statements*) | Output |
| ~~AZ~~  ~~amzM~~  ~~dog~~  ~~is~~  barking.,?!+># | *Please enter a phrase to encrypt:*  AZ  *|8|33|*  amzM  *|8| 20| 33| 20|*  dog  *|11|23|14|*  is  *|16|26|*  barking.,?!+>#  *|9|8|25|18|16|21|14|A|B|C|D| Unsupported value*  *exit* |

Step #4) Implement the Best Solution

As I mentioned in the first step (Understand the Problem) we had certain variables of this project dictated. As with all subsequent projects, we implemented this code in C++. Deciding which language to use is part of the fourth step of problem solving. Having decided to use C++ we can than decide which solution will work best in carrying out the pseudo coding we did in the trace, we also want to duplicate as best as possible the solution we provided in the flowchart. With this in mind the following code was written relying heavily on *if statements*. Once actually coding the program it was evident it was going to be quite lengthy considering the requirements. As this is part of the learning process, I’m sure once the semester progresses and I become more fluent in C++ I could implement a more efficient algorithm. With the considerable amount of code used to implement this solution it will be included in the last pages of this text.

Step #5) Testing the Solution

I ran test every few minutes as I wrote the code. During the process of continuous testing I was able to resolve errors before the code became cumbersome and more difficult to sift through. However, there were several issues I encountered which ultimately I was unable to resolve. More specifically I had a lot of trouble detecting whitespace in my code. For instance, when I typed a phrase in the program that contained a space it would go undetected and would output only the encrypted characters and not output a space. I did a lot of reading and felt like I learned quite a bit during this project. But, this issue was something I couldn’t resolve. I tried using ‘noskipws’ and ‘get’ but, neither of these provided the results I was looking for. I included a decrypt function in the code as well. This presented another problem. Admittedly I didn’t spend a whole lot of time researching and trying to resolve the issue as it wasn’t part of the intended project. To elaborate though, the idea was to be able to type in the characters that were output from the encrypt function of the program into the decrypt function. Simple encrypt and decrypt. However, it wasn’t working as intended. The output would only take one batch of numbers at a time. So, to decrypt “test” I would have to type 27, 12, 26 and 27 again all individually. Does it break the program? No, but it certainly affects how user-friendly it is.

Another issue I had was when characters that weren’t supported were passed to the character variable the program would exit. This was resolved by adding an *else* statement. Ultimately the program works. It doesn’t work as well I’d like for it to. But, it provided a venue to learn and some research on our own to help troubleshoot the problems we did encounter.

Encryptinator Code on Following Pages

//Alan Thomas CSI130-01CA Pg.3-27:Project1 09/08/2014

#include <iostream>

#include <string>

#include <cstdlib>

using namespace std;

//Start of program, get en/decrypt option from user

int main()

{

char answer = 0;

int encrypt();

int decrypt();

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" <<endl;

cout << "\* \*" << endl;

cout << "\* Encryptinator \*" << endl;

cout << "\* v0.1 \*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" <<endl;

cout << "\* '1' to encrypt | '2' to decrypt | '#' to exit \*" <<endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" <<endl;

cout << ">";

cin >> answer;

//if statement for answer

if (answer == '1'){

encrypt();

}

else if (answer == '#'){

exit(1);

}

else if (answer == '2'){

decrypt();

}

else{

cout << "Input valid response please." << endl << endl;

main();

}

}

//start of encrypt function

int encrypt()

{

cout << "\n\n\n==Follow the prompts to begin==\n\nPound '#' will exit the program.\nThe tilda '~' will bring you to the main menu." << endl << endl << endl << endl;

cout << "Please enter a phrase to encrypt: " << endl;

//name variable and initializing

char character;

character = 0;

//loop until # sign

while (character != '#'){

character = 0;

cin >> character;

// if else statements for alphabet

if (character == 'a' || character == 'A' )

cout << "8";

else if (character == 'b' || character == 'B')

cout <<"9";

else if (character == 'c' || character == 'C')

cout <<"10";

else if (character == 'd' || character == 'D')

cout <<"11";

else if (character == 'e' || character == 'E')

cout <<"12";

else if (character == 'f' || character == 'F')

cout <<"13";

else if (character == 'g' || character == 'G')

cout <<"14";

else if (character == 'h' || character == 'H')

cout <<"15";

else if (character == 'i' || character == 'I')

cout <<"16";

else if (character == 'j' || character == 'J')

cout <<"17";

else if (character == 'k' || character == 'K')

cout <<"18";

else if (character == 'l' || character == 'L')

cout <<"19";

else if (character == 'm' || character == 'M')

cout <<"20";

else if (character == 'n' || character == 'N')

cout <<"21";

else if (character == 'o' || character == 'O')

cout <<"22";

else if (character == 'p' || character == 'P')

cout <<"23";

else if (character == 'q' || character == 'Q')

cout <<"24";

else if (character == 'r' || character == 'R')

cout <<"25";

else if (character == 's' || character == 'S')

cout <<"26";

else if (character == 't' || character == 'T')

cout <<"27";

else if (character == 'u' || character == 'U')

cout <<"28";

else if (character == 'v' || character == 'V')

cout <<"29";

else if (character == 'w' || character == 'W')

cout <<"30";

else if (character == 'x' || character == 'X')

cout <<"31";

else if (character == 'y' || character == 'Y')

cout <<"32";

else if (character == 'z' || character == 'Z')

cout <<"33";

else if (character == '.')

cout <<"A";

else if (character == ',')

cout <<"B";

else if (character == '?')

cout <<"C";

else if (character == '!')

cout <<"D";

/\* Had difficulty detecting whitespace, read on Google. Tried noskipws and get. Neither provided a satisfactory result

else if (character == '\n')

cout <<"SPACE";

\*/

else if (character == '#')

exit(1);

else if (character == '~')

main();

//error handling to let user know to input a different value

else{

cout << "\*\*UNSUPPORTED VALUE\*\*" <<endl << "Please enter a phrase to encrypt: " << endl << endl;

}

cout << " | ";

}

system("PAUSE");

return 0;

}

int decrypt()

{

cout << "\n\n\n==Follow the prompts to begin.==\n\nPound '#' will exit the program.\nThe tilda '~' will bring you to the main menu." << endl << endl << endl << endl;

cout << "Enter your phrase to be decyphered(One set of characters at a time): " << endl;

//name variable and initializing

string character\_de = "";

//loop until # sign

while (character\_de != "#"){

character\_de = "";

cin >> character\_de;

// if else statements for alphabet

if (character\_de == "8")

cout << "A";

else if (character\_de == "9")

cout <<"B";

else if (character\_de == "10")

cout <<"C";

else if (character\_de == "11")

cout <<"D";

else if (character\_de == "12")

cout <<"E";

else if (character\_de == "13")

cout <<"F";

else if (character\_de == "14")

cout <<"G";

else if (character\_de == "15")

cout <<"H";

else if (character\_de == "16")

cout <<"I";

else if (character\_de == "17")

cout <<"J";

else if (character\_de == "18")

cout <<"K";

else if (character\_de == "19")

cout <<"L";

else if (character\_de == "20")

cout <<"M";

else if (character\_de == "21")

cout <<"N";

else if (character\_de == "22")

cout <<"O";

else if (character\_de == "23")

cout <<"P";

else if (character\_de == "24")

cout <<"Q";

else if (character\_de == "25")

cout <<"R";

else if (character\_de == "26")

cout <<"S";

else if (character\_de == "27")

cout <<"T";

else if (character\_de == "28")

cout <<"U";

else if (character\_de == "29")

cout <<"V";

else if (character\_de == "30")

cout <<"W";

else if (character\_de == "31")

cout <<"X";

else if (character\_de == "32")

cout <<"Y";

else if (character\_de == "33")

cout <<"Z";

else if (character\_de == "A")

cout <<".";

else if (character\_de == "B")

cout <<",";

else if (character\_de == "C")

cout <<"?";

else if (character\_de == "D")

cout <<"!";

else if (character\_de == "#")

exit(1);

else if (character\_de == "~")

main();

//error handling to let user know to input a different value

else{

cout << "\*\*UNSUPPORTED VALUE\*\*" <<endl << endl << endl;

}

cout << endl << "Next set: ";

}

system("PAUSE");

return 0;

}