

A SIMPLE CHABOT IN C

by Matthew Cumber (ll16m23c) submitted as
coursework 2 for the module Programming Project

CONTENTS

CONTENTS	2
1. SUMMARY OF PROJECT	3
2. SCHEDULE	4
2.1 19/03/18 - 25/03/18 (7 days)	4
2.2 26/03/18 - 1/04/18 (7 days)	4
2.3 16/04/18 - 27/04/18 (12 days)	4
3. DESIGN	5
3.1 Large Scale	5
3.1.1 Aim	5
3.1.2 Input, Process and Output	5
3.2 Medium Scale	5
3.2.1 Files	5
3.2.2 How The Program Will Work	6
3.3 Iterations	7
3.3.1 Version 0	7
3.3.2 Version 1	7
3.3.3 Version 2	7
3.3.4 Version 3	7
4. TEST PLAN	8
5. TEST OUTCOMES	11
5.1 Test 1 - Input	11
5.1.1 Case 1 - No Input	11
5.1.2 Case 2 - Maximum Input	11
5.1.3 Case 3 - Exceeding Limit Input	12
5.2 Test 2 - Unreadable Inputs	12
5.2.1- Case 1 - Only Unreadable Input	12
5.2.2 - Case 2 - Only Unreadable Input At End Of String	12
5.2.3 Case 3 - Mixed Unreadable Input	13
5.3 Test 3 - Specified Inputs	13
5.3.1 Case 1 - "I" Key Word	13
5.3.2 Case 2 - "What" Key Word	14
5.4 Test 4 - Input Text File	15
6. REFLECTION	16
6.1 What I Have Learned From Completing This Project	16
6.2 Strengths And Weaknesses	16
6.2.1 Strengths	16
6.2.2 Weaknesses	16
6.3 What I Can Use From This Project Later	16
7. REFERENCES	17

1. SUMMARY OF PROJECT

The project which I have chosen to do is project 1 - Chatbot. I chose this project because I wanted to see how easy or difficult it is to produce a piece of software that could behave in human like ways in this case having a conversation with another human. There have been a surplus of chatbots that have been made over the years some more successful than others. One in particular that has been successful is 'Mitsuki' which has won the Loebner prize for the last two years running in 2016 and 2017. The Loebner prize is an annual contest in which chatbot programs from all over the world are entered into a competition to see who can convince a select number of judges that the program is in fact human or displays enough characteristics of humans to be declared the winner. The goal of this competition is for one year to have a submission of a program that exhibits human responses to text, audio and visual input in which case a \$100,000 prize will be awarded to the creator marking the end of the competition.¹

A chatbot is define as follows, "A computer program designed to simulate conversation with human users, especially over the Internet."²

My project will only focus on text input from a user. The project will talk to a human user in a natural language. This means there will be no remembering of commands just simply entering text as if the program were a messaging service with another human user replying.

2. SCHEDULE

2.1 19/03/18 - 25/03/18 (7 days)

The first week of development will be used to create a version 0 and a version 1 of the project. Version 0 will be a Chatbot in it's most simplest form. At most two days will be spent on this version as it will only be used to get an understand of how a Chatbot should work and how the user will interact with the software in order to have conversation. The following 5 days will be used to create Version 1 which will include the various different modules implemented using a makefile and be able to have a basic conversation with the user.

2.2 26/03/18 - 1/04/18 (7 days)

The following week will be spent expanding on Version 1 to eventually create a Version 2. This second version will be able to rely to more inputs and remove any bugs found in the previous version. This period will also be used to test the software with real users. These users will be family and friends willing to help make the software better.

There will be a short break between Versions 2 and 3 as to account for a holiday and other subjects.

2.3 16/04/18 - 27/04/18 (12 days)

The final 12 days before submission will be split into final development and debugging/testing. Debugging and testing will take approximately 2-3 days. 1 day will be used to analyse past inputs that have been entered so the rest of the time will be spent adding smaller functionality to the software.

3. DESIGN

3.1 Large Scale

3.1.1 Aim

The aim of this project is to eventually produce software which is able to communicate with the user in a human like fashion about the users life in general which includes but is not limited to their family, likes and dislikes, and recent events that have happened to them. The chatbot which will be made with have not have the ability of answering general knowledge questions such as “what is a car?” or “how many days in a week?” as well as having no ability to carry out its own computations through the users inputs such as “what is 3 multiplied by 4?”. The software will mimic that of a first conversation between two people when they first meet. Ideally would start with introductions, occupation followed by family, the user’s interests and finally what they have recently done. This is a generalised structure which the chatbot will hopefully be able to follow to hold a meaningful understandable conversation with the user.

3.1.2 Input, Process and Output

The main input for this project will be strings which the user will enter via the keyboard of their computer. Another input will be a text file to be used as demonstration to show what the project will be able to do. The program must then read the string entered and figure out how to respond to the user in a way which makes sense and is a reasonable reply in comparison to that of a real human response. The program should not read commands and should accept text as if the user was talking to a friend. The program must read each word and try to respond minimising the chance of no response being given. The output will be text that replies to the user in a friendly and understandable way.

3.2 Medium Scale

3.2.1 Files

The initial versions of the project will stick to this specific design however if this design becomes impractical a new approach will be considered. The project will consist of the following files:

- MAIN:
The default file in which all the other files will be linked together to create the program which will be the chatbot. This file will include code for the user interface
- INPUT:
This file will be used to get an input from the user. The input will be the string a user types in. The file will only get the input exactly how the user typed it in and do nothing else with it. This file will also include code which inserts the input into a file so that past inputs can be analysed in order to make the bot better in the future.
- FORMAT:
This file will take the string from the input file and change the string to match a default style to outline below. This is for ease of processing and to account for different typing styles which should yield the same result from the program. Examples of this include “colour” and “color”, “Hello” and “hello” and “do not” and “don’t”. In each case they are different strings but the program must read them as the same as they have equivalent meaning in the English language.
- VALIDATE:
This file will ensure what the user types is itself natural language. This is to prevent undefined behaviour or crashing from happening if the string contains any special characters.
- HASHTABLE:

This file will be used to implement a hash table to be used for the program. The hash table for this application will be an array as a linked list is not needed as the hash table will not be large in size and it is not a problem if there are many collisions. If a collision does occur the new entry can be added in the next available memory location. This application will also not need to delete entries at run time so this will not be a full implantation of a tastable but sufficient for this project. The has function will not be my own. I will be using one created online for strings which is known as the DJB2 hash function.³

- OUTPUT:

This file will finally read the validated formatted string and determine an output which is appropriate and print this to the user.

3.2.2 How The Program Will Work

In order to read the input string, it must first be formatted. The format which each string will conform to is of the type "word,word,word". This is case sensitive. The strings will be converted to lower case as this will mean strings with both capital and lower case letters are readable. This is because capital letters carry no real additional meaning and for example, "HeLIO" is readable by a human so the program should be able to read this. The program should also be able to read some special characters which are used in plain text. These are , . ? ! " including the space character. In order to be able to read and produce the same response regardless if these are included in the response, each character should be removed and replaced with a comma. This means "Hello!!!" will be formatted to "hello,".

The program will also need to account for abbreviated words and equivalent words. This will be done by replacing words with a default word or words as to give the same response where applicable. So for example, "don't" would be replaced with "do,not,". There is also the case where a user does not include the abbreviation for example "dont". To account for this, during formatting the ' should be completely removed resulting in "don't" becoming "dont" and then replacing "dont" with "do,not". This way all possibilities of inputting a string will be able to be read and end up looking similar. A few example strings are shown below:

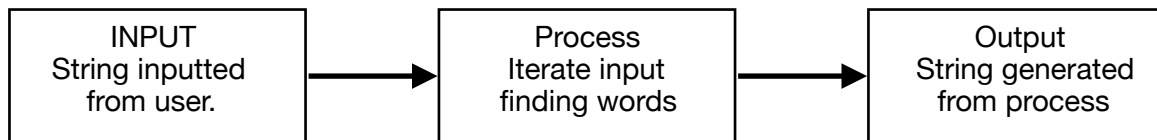
"Hello, How are you?"	->	"hello,how,are,you,"
"I can't read this!!!!"	->	"i,can,not,read,this,"
"I cant read this?!?!"	->	"i,can,not,read,this"
"GooD ByE!?',."	->	"good,bye,"

After being formatted the string must be validated. The program will produce an error message if special symbols are entered such as \$ or *. The program will also produce an error message if random text is entered such as "hsiuvbqvionb". Once validated the program will begin to reply to the string.

The program will split the input string into an array of words. This will be done by separating the string by commas as during formatting each word is followed by a comma. Once an array of words has been created, the program can begin to iterate through each word checking for various cases. These cases will first look if the word is in the hashtable. If it is then reply to this word with the data from the table and continue to the next word. The program will also check for key words which are labelled trigger words as they trigger an action to take place. These words are listed below:

- o do
- o going
- o how
- o i
- o my
- o went
- o what
- o who
- o why
- o you

Most of these trigger words will also include trigger phrases. These will be determined by comparing various elements in the words array with the trigger phrases using string compare functions. The trigger phrases will have their own functions to deal with generating a reply. These functions will then return and the main reply function will continue to iterate through the rest of the words. A successful reply will be determined if at least one word is responded to, otherwise an error message is printed.



3.3 Iterations

There will be four iterations of the project, each being a new version of the Chatbot itself. Every new version will have increased functionality expanding on the previous version with the exception of version 0.

3.3.1 Version 0

This will be the simplest version of the project consisting of one file which responds to a few strings with a basic response. There will be no advanced features and this will only respond to one or two words.

3.3.2 Version 1

This version will be considered the start of the project as here each module will be developed and used. The inputs will be formatted and validated and a tastable will be used to respond to the user.

3.3.3 Version 2

This version will be similar to version 1 but with added key words and phrases to which the project will respond to. More words added to the hashtable .

3.3.4 Version 3

The final version will be the most different in that here the project will begin to appear as a conversation between two people who have just been introduced to each other.

4. TEST PLAN

INPUT TEST PLAN :

Test	Test Case	Test Data	Expected Output
1. Keyboard Input	1.i No Text	“”	Error Message
	1.ii On The Limit Of Input Length	“hello,how are you? my name is matt. what do you like? i like pizza.i like cheese”	Response To The User
	1.iii Greater Than The Input Length	“hello, how are you? my name is matt. what do you like? i like pizza. i have an exam today”	Error Message
	1.iv Normal Input Not On Length Boundary	“Hello, how are you?”	Response To The User
2. File Input	2.i Reading A File With All Correct Format	tests.txt (shown later in report)	Responses To File

FORMAT TEST PLAN :

Test	Test Case	Test Data	Expected Output
1. To Lower Case	1.i All Upper Case	“HELLO”	“hello”
	1.ii All Lower Case	“hello”	“hello”
	1.iii Mixture Of Upper And Lower Case	“HeLlO”	“hello”
2. Punctuation	2.i Punctuation At End Of Input	“Hello!”	“hello,”
	2.ii Punctuation Mid Word	“Don’t”	“dont”
	2.ii Punctuation Mixed In Input	“?Hey! How?are You???”	“,hey,how,are,you,”
3. Comma Separated	3.i One Word	“Hello”	“hello,”
	3.ii Sentence	“The weather is good today”	“the,weather,is,good,today”

VALIDATE TEST PLAN :

Test	Test Case	Test Data	Expected Output
1. Special Characters	1.i Only Special Characters	"%#@€^"	Error Message
	1.ii Special Characters And Text	"He#\$%llo"	Error Message
	1.iii Punctuation Characters	"Hey, how are you? you're pretty good."	Response To The User
2. Meaningless Text	2.i Random Keyboard Input	"nehxywdj wcqwuibeviov"	Error Message

HASHTABLE TEST PLAN :

Test	Test Case	Test Data	Expected Output
1. Entries	1.i Entries Insert Into Table To NULL	key -> "bye" data -> "Good Bye"	Entry Inserted Into Table
	1.ii Entries Insert Into Table Collision	key -> "bye" data -> "See ya"	Entry Inserted Into Table In One Place Greater Than Calculated Key
2. Data	2.i Get Data For Existing Entry	key -> "bye"	Print Entry Data
	2.ii Get Data For Non-existing Entry	key -> "hello"	Error Message

OUTPUT TEST PLAN :

Test	Test Case	Test Data	Expected Output
1. Single Hashtable Word	1.i Word Exists In Hashtable	"hi"	"hello"
	1.ii Word Does Not Exist In Hashtable	"table"	Error Message
2. Key Words	2.i Key Word "do"	"do" Followed By Phrases	Response To Each Phrase
	2.i Key Word "going"	"going" Followed By Phrases	Response To Each Phrase
	2.i Key Word "how"	"how" Followed By Phrases	Response To Each Phrase
	2.i Key Word "i"	"i" Followed By Phrases	Response To Each Phrase
	2.i Key Word "my"	"my" Followed By Phrases	Response To Each Phrase
	2.i Key Word "went"	"went" Followed By Phrases	Response To Each Phrase
	2.i Key Word "what"	"what" Followed By Phrases	Response To Each Phrase
	2.i Key Word "who"	"who" Followed By Phrases	Response To Each Phrase
	2.i Key Word "why"	"why" Followed By Phrases	Response To Each Phrase
	2.i Key Word "you"	"you" Followed By Phrases	Response To Each Phrase

VERSION TESTING :

Test	Test Case	Test Data	Expected Output
1. Hashtable Size	1.i Inserting Data Does Not Exceed Hashtable Size	Series Of Data Entries Inserted	Program Terminates Before Attempting To Add Data To Unallocated Memory

5. TEST OUTCOMES

The software produced should be robust in the sense that it can handle and deal with all errors which may arise during use without crashing

5.1 Test 1 - Input

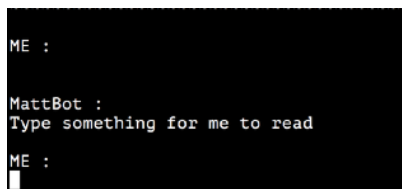
5.1.1 Case 1 - No Input

Test Input String : ""

Expected Output :

Prints error message asking user to type something.

Actual Output :



```
ME :  
  
MattBot :  
Type something for me to read  
ME :  
█
```

The test has passed as the program does not crash and is able to identify when the user does not enter any characters.

5.1.2 Case 2 - Maximum Input

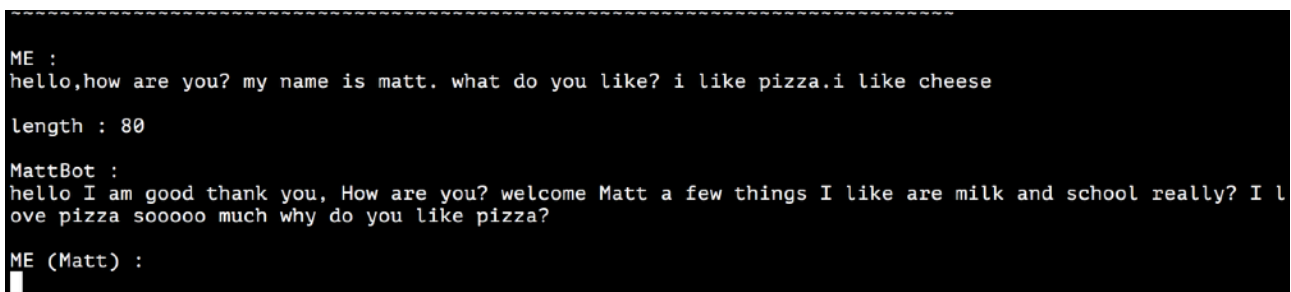
The test input is 80 characters long which is the maximum number of characters the program will read at once.

Test Input String : "hello,how are you? my name is matt. what do you like? i like pizza.i like cheese"

Expected Output :

The program is able to respond to everything that is typed in and does not print an error message.

Output :



```
ME :  
hello,how are you? my name is matt. what do you like? i like pizza.i like cheese  
length : 80  
MattBot :  
hello I am good thank you, How are you? welcome Matt a few things I like are milk and school really? I l  
ove pizza sooooo much why do you like pizza?  
ME (Matt) :  
█
```

The program passed this test as the program replies to everything the user said without behaving in an undefined way.

5.1.3 Case 3 - Exceeding Limit Input

The maximum limit of an input is 81 characters including the final return character. Entering more than this many characters could result in undefined behaviour or the program crashing.

Test Input String is : "hello, how are you? my name is matt. what do you like? i like pizza. i have an exam today"

Expected Output :

The program prints an error message identifying the user has entered too much text.

Output :

```
ME :  
hello, how are you? my name is matt. what do you like? i like pizza. i have an exam today  
MattBot :  
I can't read all that  
ME :  
█
```

The program passed the test as the input was recognised as being greater than the maximum input and an error message was printed to stop any undefined behaviour occurring such as reading the input stream when the user did not type anything.

5.2 Test 2 - Unreadable Inputs

5.2.1- Case 1 - Only Unreadable Input

Test Input : "%#@€^"

Expected Output :

The program does not read or attempt to reply to the text and prints an error message.

Output :

```
ME :  
%#@€^  
MattBot :  
I'm sorry but I am unable to read what you said  
ME :  
█
```

The program passed this test as it recognised there were no readable characters and printed an error message.

5.2.2 - Case 2 - Only Unreadable Input At End Of String

Test Input : "Hello %"

Expected Output :

The program does not read or attempt to reply to the text and prints an error message.

Output :

```
ME :  
Hello %  
  
MattBot :  
I'm sorry but I am unable to read what you said  
  
ME :  
█
```

The program passed the test as it recognised the unreadable character even though there is a readable word in the input and still printed the error message.

5.2.3 Case 3 - Mixed Unreadable Input

Test input : "He#\$%llo"

Expected Output :

The program does not read or attempt to reply to the text and prints an error message.

Output :

```
ME :  
He#$%llo  
  
MattBot :  
I'm sorry but I am unable to read what you said  
  
ME :  
█
```

The program passed this test as it was able to recognise unreadable characters within a word and print an error message.

5.3 Test 3 - Specified Inputs

The program looks for key words which it will then identify and proceed to look for common phrases in an attempt to respond to the user. The key word's are outlined in the design section of this report. Each word has been tested. Below shows how two key words have been tested.

5.3.1 Case 1 - "I" Key Word

The key word I has several phrases which can follow such as "i like" and "I am". These have been tested below.

Test String : "I like cars"

Expected Output :

The program should print a random word which shows an interest to the user. It should also inform the user that the program will remember what they said and ask the user why they like cars as no reason was given as part of the input string.

Output :

```
ME :  
i like cars  
  
MattBot :  
Intriguing I'll take note of that why do you like cars?  
  
ME :  
█
```

Test String : "I am good"

Expected Output :

The program should reply to the input string with the data from the hash table. the data is "that's good".

Output :

```
ME :  
i am good  
  
MattBot :  
that's good  
  
ME :  
█
```

The program passed this test as a response was given to each input regarding the key word "I".

5.3.2 Case 2 - "What" Key Word

The key word I has several phrases which can follow such as "what do you like" and "what is your favourite". These have been tested below.

Test String : "what do you like?"

Expected Output :

The program should randomly choose two things in which it is defined to like/have an option about and print these to the user.

Output :

```
ME :  
what do you like  
  
MattBot :  
a few things I like are water and chips  
  
ME :  
█
```

Test String : "what is my favourite food"

Expected Output :

The program should print to the user what they have previously stated is their favourite otherwise it should state it does not know yet.

Output :

```
ME :  
my fave food is chicken  
  
MattBot :  
Fascinating I'll take note of that  
  
ME :  
what is my fave food  
  
MattBot :  
Your favourite food is chicken  
  
ME :  
█
```

The program passed this test. Each input produced a valid response which made sense to the user. In the second case the program was able to correctly identify the users favourite

5.4 Test 4 - Input Text File

The aim of the project was to “produce software which is able to communicate with the user in a human like fashion about the users life in general which includes but is not limited to their family, likes and dislikes, and recent events that have happened to them”. This test case will attempt to evaluate the softwares effectiveness at matching this aim. This will be done by reading a series of inputs from a file and replying to each one. At the end of the test there will be a series of responses made to each input from the file. The input text file is shown here:

```
1 hello,  
2 how,are,you,  
3 my,name,is,don,  
4 very,good,thanks,  
5 i,like,sunny,days,  
6 because,it,makes,me,happy,  
7 do,i,like,sunny,days,  
8 i,like,chips,  
9 you,suck,  
10 how,are,you,  
11 i,love,you,  
12 how,are,you,  
13 what,do,you,like,  
14 what,is,your,favourite,film,  
15 my,favourite,sport,is,football,  
16 bye,  
17
```

The program proceeds to answer each question. After analysing the conversation between the file and the program, it is clear that the program can read a text file and answer a wide range of different inputs.

6. REFLECTION

6.1 What I Have Learned From Completing This Project

I have learnt the importance of designing a project and having a goal to aim for is essential before starting to make the project itself. This is because with no clear direction code will be added which has no real functionality and does not help to achieve the goal. Also, knowing what the project will consist of is important especially when designing and choosing modules as linking each one together could be challenging without knowing which ones need to interact with each other.

I have also learnt that having a schedule and sticking to it is important. It is easy to leave a project until last minute and rush the project to get it completed in time. However this leads to more bugs in the code which can yield devastating consequences depending on the project.

A final idea I have learnt is iterative design and how this can be useful. Having iterations helps to see progress in the project and adds the ability to add functionality in stages rather than all at once. This method of development also enables you to see what worked well in previous iterations and always have the option of going back to the previous iteration if the new code is too bug prone.

6.2 Strengths And Weaknesses

6.2.1 Strengths

One of my strengths is that I managed to stick to the schedule and meet my own internal deadlines for progress in the project. I managed to spend a few hours on the project each day of the week which was plenty of time to be able to get the project done. However, without the deadlines I set myself I would have not spent time daily on the project and would rather spend time in the final week rushing to completion.

6.2.2 Weaknesses

A weakness of mine is not thinking about tests which could be made during development and instead thought of tests towards the end of the project. Thinking of them whilst developing will save time and also result in better testing and tests which you may not think of at the end of a project could be added if thought of when developing.

6.3 What I Can Use From This Project Later

I am studying the High Performance Graphics and Games Engineering degree so I will hopefully work on producing video games in a few years. Aspects of this AI can be taken and used in a video game for example when talking to NPC's.

I can also re-use some code such as the hashtable implementation. By adding additional features this code could be reused in future projects.

Again I could also use the layout of this report on future projects. For example the test plan is transferrable as different tests can be added to the same table layout.

7. REFERENCES

1. Loebner Prize - Wikipedia. 2018. Loebner Prize - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/Loebner_Prize. [Accessed 25 April 2018].
2. Oxford Dictionaries | English. 2018. chatbot | Definition of chatbot in English by Oxford Dictionaries. [ONLINE] Available at: <https://en.oxforddictionaries.com/definition/chatbot>. [Accessed 25 April 2018].
3. www.cse.yorku.ca. 2018. No page title. [ONLINE] Available at: <http://www.cse.yorku.ca/~oz/hash.html>. [Accessed 25 April 2018].