

ECE 3220 Lab8
Morse Code Generator

Marshall Lindsay
mbllgh6@mail.missouri.edu
14211662

4/5/2017

Objective:

The objective of this lab was to become more familiar with classes, and inheritance, through by creating a morse code generator.

Discussion:

Overall this lab proved to be easier than some of the others. Three classes were made in total: Message, morseCodeMessage, and messageStack. The Message class was the base class of the morseCodeMessage. It contained one protected member, a string, that contained the ASCII representation of the message. The morseCodeMessage contained one protected member, a string, that contained the morse code representation of the message. The morseCodeMessage class also contained a public member, a morseCodeMessage pointer, used to implement the stack. Finally the messageStack class was a stand-alone class that contained only one private member, the pointer to the top of the stack.

The stack was implemented using a linked list with a LIFO architecture. The messageStack method “push()” is responsible for creating a new message and adding it to the stack while the “pop()” method is responsible for deleting the message on the top of the stack and adjusting the pointer to the top of the stack.

Figure1 and Figure 2 show the functionality of the program.

Conclusion:

Through the use of classes, inheritance and proper understanding of linked list, the morse code generator was created. By the end of this lab a greater understanding of these topics was obtained.

Figures:

```

what would you like to do?
1) Push message to the stack
2) Pop message off the stack
3) Print all messages on the stack
4) Quit
1
Please enter a message:
Message1

what would you like to do?
1) Push message to the stack
2) Pop message off the stack
3) Print all messages on the stack
4) Quit
1
Please enter a message:
Message2

what would you like to do?
1) Push message to the stack
2) Pop message off the stack
3) Print all messages on the stack
4) Quit
1
Please enter a message:
Message3

what would you like to do?
1) Push message to the stack
2) Pop message off the stack
3) Print all messages on the stack
4) Quit
3
Message3
-- . ... .- --. . ...--
Message2
-- . ... .- --. . ...--
Message1
-- . ... .- --. . ----

what would you like to do?
1) Push message to the stack
2) Pop message off the stack
3) Print all messages on the stack
4) Quit

```

Figure1: Screenshot showing the push functionality of the program

```

what would you like to do?
1) Push message to the stack
2) Pop message off the stack
3) Print all messages on the stack
4) Quit
2

what would you like to do?
1) Push message to the stack
2) Pop message off the stack
3) Print all messages on the stack
4) Quit
3
Message2
-- . ... .. - - - . . . ----
Message1
-- . ... .. - - - . . . ----

what would you like to do?
1) Push message to the stack
2) Pop message off the stack
3) Print all messages on the stack
4) Quit
2

what would you like to do?
1) Push message to the stack
2) Pop message off the stack
3) Print all messages on the stack
4) Quit
3
Message1
-- . ... .. - - - . . . ----

what would you like to do?
1) Push message to the stack
2) Pop message off the stack
3) Print all messages on the stack
4) Quit
2

what would you like to do?
1) Push message to the stack
2) Pop message off the stack
3) Print all messages on the stack
4) Quit
2

There is nothing on the stack!

what would you like to do?
1) Push message to the stack
2) Pop message off the stack
3) Print all messages on the stack
4) Quit

```

Figure2: Screenshot showing the pop, print, and error correction of the program

```

commit 4cc643af595f4738146eaf1bf7259f53191b29ad
Author: Marshall Lindsay <mbanks1050@gmail.com>
Date:   Wed Apr 5 18:37:16 2017 -0500

    Final commit

commit dfdc00bcf6fc3a9ab8206156bd8f3ea1f9fe5281
Author: Marshall Lindsay <mbanks1050@gmail.com>
Date:   Wed Apr 5 18:14:31 2017 -0500

    Completed

commit c69804b0ee5c658d2d9ce0ab7186745db3d40d02
Author: Marshall Lindsay <mbanks1050@gmail.com>
Date:   Tue Apr 4 18:47:29 2017 -0500

    Working message translation

commit ba5786a94cf92dffde6f396f1de07cbf4fac6144
Author: Marshall Lindsay <mbanks1050@gmail.com>
Date:   Tue Apr 4 15:18:08 2017 -0500

    Committing old changes

commit 38694fea78376fae81ee449c06f37472607f8cc9
Author: Marshall Lindsay <mbanks1050@gmail.com>
Date:   Thu Mar 23 10:20:33 2017 -0500

    Initializing

```

Figure3: Screenshot showing gitHub commit

Code:

```

/*
    Marshall Lindsay
    ECE 3220
    Lab8
    3/23/17
*/

#include <iostream>
#include <string>
#include <vector>

using namespace std;

class Message{
protected:
    string msg;

public:
    Message();

```

```

        Message(string);
        virtual ~Message();
        virtual void print();
};
/*
Default constructor;
Prompts the user for a new message and adds that message
to the msg string member
*/
Message::Message() {
    //cout<<"Constructor"<<endl;
    string userInput;
    cout<<"Please enter a message:"<<endl;
    getline(cin, userInput);

    //May need some error checking for special characters

    this -> msg = userInput;
}
/*
Parametric constructor;
Takes a string in as the message and adds that message to the
msg string member;
*/
Message::Message(string userInput) {
    //cout<<"Constructor"<<endl;
    this -> msg = userInput;
}

/*Destructor.
Nothing in this class needs to be deleted;
*/
Message::~~Message() {
    //cout<<"Destructor"<<endl;
}

void Message::print() {
    cout<<this ->msg<<endl;
}

//-----
//-----Extended Message Class for Morse Code-----
class morseCodeMessage: public Message{
protected:
    string morseMsg;
public:
    morseCodeMessage* next;
    morseCodeMessage();
    morseCodeMessage(string);
    ~morseCodeMessage();
    void translate(string);
    void print();
};
//Default constructor. Asks for a message and saves it as a string in

```

```

//the base classes msg member. Then translates that msg into morse code
//and saves the translation to the extended morseMsg member
morseCodeMessage::morseCodeMessage() {
    translate(this -> msg);
    this -> next = NULL;
}

//Parametric constructor. Takes a string as input. Calls the base
//constructor with the same string. Then translates the string
//to morse code and saves it to the extended morseMsg member.
morseCodeMessage::morseCodeMessage(string userInput) : Message(userInput) {
    translate(this -> msg);
    this -> next = NULL;
}

morseCodeMessage::~~morseCodeMessage() {
    //cout<<"Destructor"<<endl;
}

void morseCodeMessage::translate(string input) {
    char test;
    for(int i = 0; i < input.length(); i++) {
        input[i] = tolower(input[i]);
    }
    for(int i = 0; i < input.length(); i++) {
        test = input[i];
        switch(test) {
            case 'a':
                this->morseMsg.append("-.- ");
                break;
            case 'b':
                this->morseMsg.append("-... ");
                break;
            case 'c':
                this->morseMsg.append("-.-. ");
                break;
            case 'd':
                this->morseMsg.append("-.. ");
                break;
            case 'e':
                this->morseMsg.append(". ");
                break;
            case 'f':
                this->morseMsg.append("..-. ");
                break;
            case 'g':
                this->morseMsg.append("--. ");
                break;
            case 'h':
                this->morseMsg.append(".... ");
                break;
            case 'i':
                this->morseMsg.append(".. ");
                break;

```

```
case 'j':
    this->morseMsg.append("... ");
    break;
case 'k':
    this->morseMsg.append("-.- ");
    break;
case 'l':
    this->morseMsg.append("-.. ");
    break;
case 'm':
    this->morseMsg.append("-- ");
    break;
case 'n':
    this->morseMsg.append("-. ");
    break;
case 'o':
    this->morseMsg.append("--- ");
    break;
case 'p':
    this->morseMsg.append("--- ");
    break;
case 'q':
    this->morseMsg.append("--- ");
    break;
case 'r':
    this->morseMsg.append("-. ");
    break;
case 's':
    this->morseMsg.append("... ");
    break;
case 't':
    this->morseMsg.append("- ");
    break;
case 'u':
    this->morseMsg.append("..- ");
    break;
case 'v':
    this->morseMsg.append("...- ");
    break;
case 'w':
    this->morseMsg.append("-- ");
    break;
case 'x':
    this->morseMsg.append("-..- ");
    break;
case 'y':
    this->morseMsg.append("-.- ");
    break;
case 'z':
    this->morseMsg.append(".... ");
    break;
case '0':
    this->morseMsg.append("----- ");
    break;
```



```

        case '1':
            this->morseMsg.append(".---- ");
            break;
        case '2':
            this->morseMsg.append("..--- ");
            break;
        case '3':
            this->morseMsg.append("...-- ");
            break;
        case '4':
            this->morseMsg.append("....- ");
            break;
        case '5':
            this->morseMsg.append("..... ");
            break;
        case '6':
            this->morseMsg.append("-.... ");
            break;
        case '7':
            this->morseMsg.append("---.. ");
            break;
        case '8':
            this->morseMsg.append("---.. ");
            break;
        case '9':
            this->morseMsg.append("----. ");
            break;
        case ' ':
            this->morseMsg.append(" ");
            break;
        default:
            break;
    } //Switch

}

}

void morseCodeMessage::print() {
    cout<< this->msg <<endl;
    cout<< this->morseMsg <<endl;
}

//-----
//-----Generic Stack Class-----
class messageStack{
private:
    morseCodeMessage* top;
public:
    messageStack();
    ~messageStack();
    void push();
    void pop();
    void printStack();
}

```

```

};

//Default constructor. Sets the pointer to the top of the
//Stack to NULL.
messageStack::messageStack() {
    this -> top = NULL;
}

//Destructor. This will delete each object in the linked list.
messageStack::~~messageStack() {
    morseCodeMessage* temp;
    while(top != NULL) {
        temp = this -> top;
        this -> top = this -> top -> next;
        delete temp;
    }
}

//Creates a new message and sets it to the top of the stack.
void messageStack::push() {
    morseCodeMessage* newNode = new morseCodeMessage();
    newNode -> next = this -> top;
    this -> top = newNode;
}

//Cycles down the stack and prints each message in turn.
void messageStack::printStack() {
    morseCodeMessage* msgPtr = this -> top;
    while(msgPtr != NULL) {
        msgPtr -> print();
        msgPtr = msgPtr -> next;
    }
}

//Deletes the top object from the stack. Sets the top of the
//stack to the next one down.
void messageStack::pop() {
    if(top == NULL) {
        cout<<"\nThere is nothing on the stack!"<<endl;
        return;
    }
    morseCodeMessage* retObj = this -> top;
    this -> top = this -> top -> next;
    delete retObj;
}

void optionMenu();
void errorMessage();

int main(int argc, char* argv[]) {
    messageStack stack;

```

```

string userInput;

while(1){
    optionMenu();
    getline(cin, userInput);
    if(userInput == "1"){
        stack.push();
    }else if(userInput == "2"){
        stack.pop();
    }else if(userInput == "3"){
        stack.printStack();
    }else if(userInput == "4"){
        return(0);
    }else{
        errorMessage();
    }
}

}

void optionMenu(){
    cout<<"\nWhat would you like to do?"<<endl;
    cout<<"1) Push message to the stack\n"
        <<"2) Pop message off the stack\n"
        <<"3) Print all messages on the stack\n"
        <<"4) Quit"<<endl;
}

void errorMessage(){
    cout<<"Invalid input!"<<endl;
}

```