ITCS 4102 - Final Report

D Language

Group: Front Five - *Matthew Johnson, Matthew Smrzlick, David Newton, Daniel Cole, Luis Gomez*

Movie Library in D Language

I. D Language

A. History

The D programming language started being developed in 1999 by Walter Bright of Digital Mars. It was released in 2001 and reached version 1.0 by January 2007. The first version of D (D1) focused on object-oriented and meta-programming paradigms like C++ and Ruby. D's official standard library and documentation, known as Phobos, was unsatisfactory to the community at large. As a splinter group, Tango created a new set of runtime libraries and programming styles that focused more on object-oriented programming and modularity.

In June 2007, version 2 of D was released. D1 was transitioned to minor bug fixes and was eventually discontinued in December of 2012. In 2010, Andrei Alexandrescu released a book titled *The D Programming Language*, marking the stabilization of D2. From this point forward, it was simply referred to as D. In 2011, D development moved to GitHub, which has led to a huge increase in the number of contributions to the compiler, runtime, and standard libraries.

B. Paradigm

D was developed to support the following paradigms:

- Imperative (like C)
- Object-oriented (like C++)
- Meta-programing (code that creates other code)
- Functional (everything is evaluated as a function)
- Parallel (multiple CPUs)
- Concurrent (multithreading)

C. Elements of the Language (reserved words, primitive data types, etc.)

D has the following primitive types that come from C although the syntax may differ in some instances: bool, char, byte, ubyte, short, ushort, wchar, int (normal integer and long integer), uint (unsigned integer and unsigned long integer), long (long long), ulong (unsigned long long), float, double, real (long double), ireal (imaginary long double), creal (complex long double). Reserved words from C were carried over to D in addition to some new ones. New reserved words like `is`, `in`, and `typeof` come from Ruby.

D. Syntax

Libraries in D are grouped into modules by the task performed. The syntax for using D libraries is: *import std.stdio*, where stdio is the standard library for handling input and output. Similar to C, the D language requires a main function

as a starting point to determine the order in which other sections are executed.

Sequences of characters in D are referred to as tokens . A token may be a keyword, a constant, an identifier, a symbol, or a string literal.

The syntax for D comments are also similar to C. The D compiler ignores text surrounded by the /* */ markers. Single line comments are also allowed via // at the beginning of a statement.

As with most languages, D uses identifiers to identify variables, functions, and other user-defined items. Identifiers may not include certain punctuation characters such as @, $, and % since they are generally reserved for other things. D is a case sensitive language. So variable and Variable would identify two separate things.

The D language also has its share of reserved keywords that cannot be used as identifiers. There are too many of these to conveniently list. The D documentation provides an adequate list.

Additionally whitespace is used to define blanks, tabs, comments, and newline characters. The main purpose of whitespace in D is to separate statements into parts that allow the interpreter to understand them. Whitespace also contributes greatly to readability.

E. Control Abstractions (loops, conditionals)

The D programming languages provides many familiar control abstractions.

while loop: Checks condition before allowing body of the loop to be executed.

do...while Loop: Checks condition after loop body, thus will always be executed at least once.

for loop: Efficiently allows you to iterate a specific (previously defined) number of times.

The D programming language allows these loops to be combined with one another in a technique called *nesting*.

Similarly, D also has many recognizable means for dealing with decision making. D recognizes any zero or null as a "false" value. All other values are assumed true.

The following conditionals are provided in D:

if statements: Boolean expression followed by one or more statements to potentially be executed.

if/else statements: else statements can optionally follow if statement blocks and will be evaluated if the previous statements evaluate to false.

switch statement: lets a variable be tested against a list of values, with each value in the list referred to as a *case*.

Like control structures, these if, if/else and switch statements may also be nested inside one another to allow for a more refined and specific decision making process.

Finally, the D language provides the conditional operator ?. This can be used to succinctly replace if...else statements.

expression1 ? expression2 : expression3

This operator works by evaluating *expression1*. If expression1 is true then *expression2* is evaluated, becoming the value of the entire ? expression. If it is false *expression3* is evaluated, becoming the value of the entire ? expression.

Abstraction (functions, objects)

Functions in D should look familiar to anyone who has used C, C++, or even Java.

A function definition in the D programming language has three parts.

```
return-type function-name( params list)
{
        body...
}
```

Return Type: If a function needs to return a value, the data type of that value is declared in the return type. If no value needs to be returned the keyword **void** is used.

Function Name: The actual name of the function. Conventionally named something related to the task performed.

Params List: Parameters are any value(s) passed to the function for use in the function's body.

Body: The body of the function is generally a collection of statements that perform a task or calculation.

Since the D language supports the object-oriented paradigm, classes and objects are a core feature of D.

Class definitions are very simple and just start with the keyword class followed by a user-defined class name.

```
class Movie
{
        private string title;
        public int year;
}
```

In the simple example above, we have two class members. One declared public and one private. Like most object oriented languages, only public members can be directly accessed externally.

Other common object oriented concepts such as: Inheritance, overloading, encapsulation, interfaces and abstract classes are all supported by D.

F. Evaluation

1. Writability

The D language is relatively easy to write a program in, despite its lackluster documentation. The language has a heavy focus on object-oriented ideals, so abstraction is heavily utilized. Programmers can also utilize structs, similarly to the C language, another form of abstracting data. The language also features expressive operations such as dynamic typing with the keyword ‘auto’ and being able to iterate through an array in one line with the foreach iterator. What is normally a type of loop in other languages, foreach abstracts the iteration even further. Its heavy use of methods similar to Ruby allows the programmer to do lots of operations on data in few or one line of code.

2. Readability

The D programming language is a very readable and understandable language, especially for those with experience in any basic object-oriented language like C++ or Ruby. The only difficulty with being able to understand D programming is when it uses artifacts from the multiple languages it was derived from (C, C++, Ruby). For reading in numbers, D utilizes readf, similarly to C. When reading strings, it functions similarly to Ruby by reading lines of input (along with methods like chomp to remove whitespace). There seems to be no real distinction why certain features of the language use one kind of syntax, while the others use different syntax. For example, we had an issue when comparing a value to null (x == null) and had to alter it to more Ruby-like syntax of (x is null).

3. Reliability

D is also a reliable language as it does run type checking during compilation to reduce expense in runtime and cost of repairs. This is an improvement from an early version of C where the language made no attempt to make sure arguments being passed to a function were of the right data type.

G. Strengths

D has a few strengths that make it a compelling language. Since it is based on C and C was designed to be fast, programs in D run quickly as well. It it also

backward compatible with C and C++. Another strength it's support for multiple programming paradigms. D also supports multiple inheritance which some may find useful depending on the type of program that is being written.

H. Weaknesses

Something we noticed immediately is that the documentation for D leaves a lot to be desired. It is poorly organized and often vague as to how things are supposed to work. Another glaring issue is how it randomly goes from being C-like to Ruby-like depending on what you are doing. This is a blatant violation of generality and caused us some issues along the way.

II. Movie Library Project

A. Overview

For our sample programs, we created something simple program that does a few different things to highlight features of the D language. A very simple movie database was created to illustrate arrays, structs, and writing data to the console. D has a data structure called an associative array which allows you to bind array values to keys. A simple example binding "one" to 1 and so on was created and the values and their keys were printed to showcase how this works. A simple statement asking the user how they are doing and responding with a message was

used to show how keyboard input and console output work. Finally, since D supports multithreading a short example was created for that a well. It displays the thread ID that main() is executing in, creates a new thread, prints the new thread's ID and a message, then returns to main(). Multithreading was the only difficult thing to implement. It works differently in D than it does in C and C++. The main difficult was figuring out how to make main wait on the child thread to execute and exit before continuing.

For our term project we created a movie database. The database supports the following functions: add a new movie, read information about a movie, update a movie's information, delete a movie, and rating a movie in the database as well as seeing a list of all movies in the database. Since D supports object oriented programming, we decided to implement movie as a class instead of a struct. Movies are sorted according to their title in alphabetical order. The Movie class has the following fields:

- title: the title of the movie
- genre: the genre of the movie
- director: who directed the movie
- yearReleased: the year that the movie debuted
- duration: the running time of the movie in minutes
- rating: ratings from user on a one to five scale
- ratingCount: used to average the ratings from all users

The Movie class contains the following functions:

- getTitle and setTitle: used when creating and updating a movie
- getGenre and setGenre: used when creating or updating a movie
- getDirector and setDirector: used when creating or updating a movie
- getYearReleased and setYearReleased: used when creating or updating a movie
- getDuration and setDuration: used when creating or updating a movie
- addRating: used to update the movie's rating when a user rates it
- opCpm: overrides the opCpm function and is used when sorting the database

The program begins by reading the database from a file. Movie objects are stored in an array for simplicity. Next the user is prompted to select the action they would like to perform. Regular expression are used for input validation here as well as when entering running time, release year, and rating. Regular expressions are also used when searching the database.

When searching for a movie, the user can enter the full title of the movie or a partial string. The program looks for an returns all possible matches given what the user enters.

When creating a new movie, the user is asked to enter the title, director, release year, runtime, and genre. Input validation is performed where necessary. The user is alerted if they attempt to add a movie already in the database. The same is true

when editing an existing movie's information, but users additionally have to enter the title as it appears in the database in order to delete it.

When rating a movie, the user enters a title to search for. They must enter the title exactly as it appears in the database. If the movie is found they are prompted to enter their rating on a scale of 1.0 to 5.0, and then their rating is added.

When deleting a movie, the user must also enter the title exactly as it appears.

When the user is finished with the actions they want to perform and select the option to exit, the database is first saved then the user is prompted to press any key to exit.

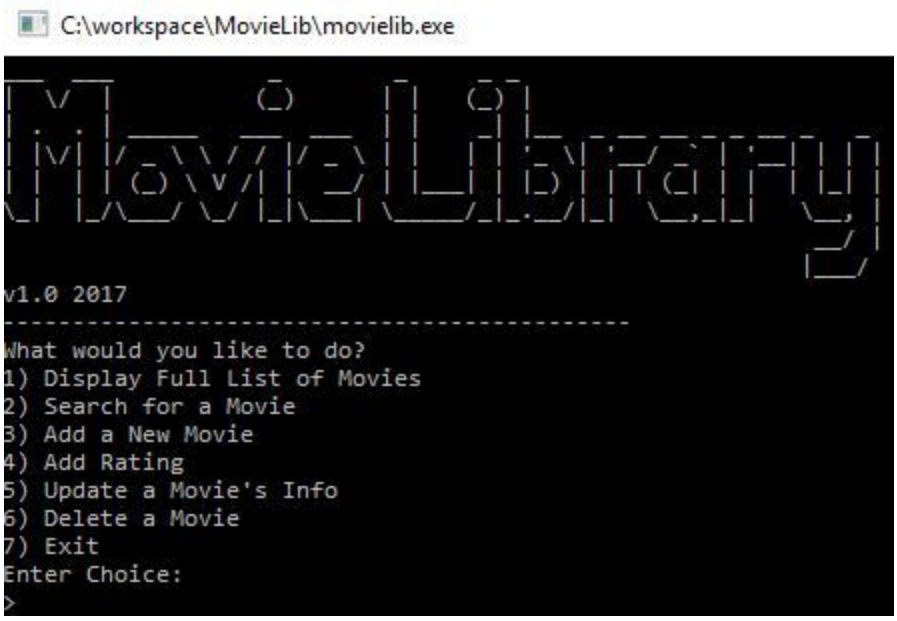
B. Difficulty

In many ways D is similar to C and C++ which made it relatively easy to learn. However, there were a few things we ran into that made things difficult. The first such thing is fallthrough of case statements. In C, when code in a case is done executing, the program then jumps to the end of the case statement and continues executing. Case statements do not work the same way in D. Multiple break statements needed to be inserted to break out of the case statement in the way we wanted it to. This method was clunky and also deprecated so we replaced it with a simpler to implement nested if-else if statement.

Another problem we ran into were instances when D diverged from C and C++ syntax to Ruby syntax. Initially, when we removed a movie from the database

(array) we set that element to null. When writing the array to the file at the end of the program, we wanted to check if an element was null so that it could be skipped. At first we tried ‘element == null’, but we got an obscure error. What the compiler was looking for was ‘element is null’. We found this to be completely random and a blatant violation of generality in the language. That method of deletion didn’t work correctly and was scrapped once we discovered that D has a remove function for arrays.

C. Sample Run



Display all Records:

```
Enter Choice:
>1


All Movies:
-----------------------------------------------------
2001: A Space Odyssey(1968) - Directed By Stanley Kubrick - Genre: Sci-Fi - Runtime: 142 minutes - Rating: 4
Alien(1979) - Directed By Ridley Scott - Genre: Horror - Runtime: 121 minutes - Rating: 3.55
Dawn of the Dead(1979) - Directed By George A. Romero - Genre: Horror - Runtime: 127 minutes - Rating: 2.9
Interstellar(2014) - Directed By Christopher Nolan - Genre: Sci-Fi - Runtime: 169 minutes - Rating: 5
Office Space(1999) - Directed By Mike Judge - Genre: Comedy - Runtime: 89 minutes - Rating: 4
RoboCop(1987) - Directed By Paul Verhoeven - Genre: Action - Runtime: 101 minutes - Rating: 3.5
Shaun of the Dead(2004) - Directed By Edgar Wright - Genre: Horror/Comedy - Runtime: 99 minutes - Rating: 4.1
Star Wars: A New Hope(1977) - Directed By George Lucas - Genre: Sci-Fi - Runtime: 121 minutes - Rating: 5
Star Wars: Return of the Jedi(1983) - Directed By Richard Marquand - Genre: Sci-Fi - Runtime: 131 minutes - Rating: 4
Star Wars: The Empire Strikes Back(1980) - Directed By Irvin Kershner - Genre: Sci-Fi - Runtime: 124 minutes - Rating: 5
The Matrix(1999) - Directed By The Wachowski Brothers - Genre: Sci-Fi - Runtime: 136 minutes - Rating: 4.5
-----------------------------------------------------
```

Searching:

```
Enter Choice:
>2

Search by:
1) By title
2) By genre
>1

Search For:
>star wars

Results Found: 3
-----------------------------------------------------
Star Wars: A New Hope(1977) - Directed By George Lucas - Genre: Sci-Fi - Runtime: 121 minutes - Rating: 5
Star Wars: Return of the Jedi(1983) - Directed By Richard Marquand - Genre: Sci-Fi - Runtime: 131 minutes - Rating: 4
Star Wars: The Empire Strikes Back(1980) - Directed By Irvin Kershner - Genre: Sci-Fi - Runtime: 124 minutes - Rating: 5
-----------------------------------------------------
```

Create New:

```
Enter Choice:
>3

Enter the title:
>Stalker

Enter the director:
>Andrei Tarkovsky

Enter the release year:
>1979

Enter the runtime:
>163

Enter the genre:
>Sci-Fi

Successfully Added: Stalker(1979) - Directed By Andrei Tarkovsky - Genre: Sci-fi - Runtime: 163 minutes - Rating: 0
```

Add Rating:

```
Enter Choice:
>4

Enter the title you want to rate:
>stalker
What is your rating?[1.0-5.0]
>3.5

What would you like to do?
1) Display Full List of Movies
2) Search for a Movie
3) Add a New Movie
4) Add Rating
5) Update a Movie's Info
6) Delete a Movie
7) Exit
Enter Choice:
>2

Search by:
1) By title
2) By genre
>1

Search For:
>stalker

Results Found: 1
-----------------------------------------------------
Stalker(1979) - Directed By Andrei Tarkovsky - Genre: Sci-fi - Runtime: 163 minutes - Rating: 3.5
-----------------------------------------------------
```

Update Information:

```
Enter Choice:
>5

Enter exact title of movie to edit:
>stalker

What field would you like to alter?
1) Title
2) Genre
3) Director
4) Release Year
5) Runtime
>5

Enter the updated runtime:
>161

Successfully Updated: Stalker(1979) - Directed By Andrei Tarkovsky - Genre: Sci-fi - Runtime: 161 minutes - Rating: 3.5
```

Delete Record:

```
Enter Choice:
>6

Enter the title you want to delete:
>stalker

Deleted: Stalker(1979) - Directed By Andrei Tarkovsky - Genre: Sci-fi - Runtime: 161 minutes - Rating: 3.5

What would you like to do?
1) Display Full List of Movies
2) Search for a Movie
3) Add a New Movie
4) Add Rating
5) Update a Movie's Info
6) Delete a Movie
7) Exit
Enter Choice:
>1


All Movies:
-----------------------------------------------------
2001: A Space Odyssey(1968) - Directed By Stanley Kubrick - Genre: Sci-Fi - Runtime: 142 minutes - Rating: 4
Alien(1979) - Directed By Ridley Scott - Genre: Horror - Runtime: 121 minutes - Rating: 3.55
Dawn of the Dead(1979) - Directed By George A. Romero - Genre: Horror - Runtime: 127 minutes - Rating: 2.9
Interstellar(2014) - Directed By Christopher Nolan - Genre: Sci-Fi - Runtime: 169 minutes - Rating: 5
Office Space(1999) - Directed By Mike Judge - Genre: Comedy - Runtime: 89 minutes - Rating: 4
RoboCop(1987) - Directed By Paul Verhoeven - Genre: Action - Runtime: 101 minutes - Rating: 3.5
Shaun of the Dead(2004) - Directed By Edgar Wright - Genre: Horror/Comedy - Runtime: 99 minutes - Rating: 4.1
Star Wars: A New Hope(1977) - Directed By George Lucas - Genre: Sci-Fi - Runtime: 121 minutes - Rating: 5
Star Wars: Return of the Jedi(1983) - Directed By Richard Marquand - Genre: Sci-Fi - Runtime: 131 minutes - Rating: 4
Star Wars: The Empire Strikes Back(1980) - Directed By Irvin Kershner - Genre: Sci-Fi - Runtime: 124 minutes - Rating: 5
The Matrix(1999) - Directed By The Wachowski Brothers - Genre: Sci-Fi - Runtime: 136 minutes - Rating: 4.5
-----------------------------------------------------
```

Exit Confirmation:

```
>7
Updating Libary and Shutting down...
Press any key to exit.
```

III. Works Consulted

Alexandrescu, Andrei. The D programming language. Upper Sadle River, NJ: Addison-Wesley, 2010. Print.

"Phobos Runtime Library." *Phobos Runtime Library - D Programming Language*. N.p., n.d. Web. 20 Apr. 2017. <https://dlang.org/phobos/index.html>.

D Language Website, www.dlang.org