A Code by Any Other Name
<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>03</td>
</tr>
<tr>
<td>Esoteric Language</td>
<td>05</td>
</tr>
<tr>
<td>Coding Vocabulary</td>
<td>08</td>
</tr>
<tr>
<td>Stacks as a Data Structure</td>
<td>09</td>
</tr>
<tr>
<td>SPL Grammar</td>
<td>10</td>
</tr>
<tr>
<td>Outputting Input Reversedly</td>
<td>15</td>
</tr>
</tbody>
</table>
Instructor | Gamer | Bookish | Programmer

Computer Science by Way of the Liberal Arts

Computer Science instructor teaching beginning to advanced topics for undergraduate students at NMT. Also the Women in Computer Science Coordinator and dedicated to encouraging young women to pursue careers in computer science. Believes that computer science is more than just coding and analytics as it requires quite a bit of creativity and problem solving.

https://blueberryboson.com/about-me/
Bored late on a February evening while avoiding their homework assignment, Karl Hasselstrom and Jon Aslund developed the Shakespeare Programming Language. Their goal – to write a language with source code that looked like a Shakespearean play. This language contains nothing fancy, just your everyday arithmetic and goto statements. In their own words, they have "combined the expressiveness of BASIC with the user-friendliness of assembly language."
es·o·ter·ic

adjective  |  e-sə-ter-ik

1. a: designed for or understood by the specially initiated alone
   // a body of esoteric legal doctrine
   – B. N. Cardozo

   b: requiring or exhibiting knowledge that is restricted to a small group
   // esoteric terminology

2. a: limited to a small circle
   // engaging in esoteric pursuits

   b: Private, Confidential
   // and esoteric purpose

A Code by Any Other Name

*Merriam-Webster Dictionary
Why Esoteric?

Languages designed to test the boundaries of computer programming language design, as a proof of concept, as software art, as a hacking interface to another languages, or even as a joke.

Usability?

Usability is rarely a goal for esoteric programming language designers -- often the design leads to quite the opposite. Their usual aim is to remove or replace conventional language features while still maintaining a language that is *Turing-complete*, or even one for which the *computational class* is unknown.

Cultural Context

Esoteric languages can be seen as similar to *code art* and *code poetry*. They can be used to expose the inherent conflict between human thinking and computer logic.

A variety of events and websites allow the general public to present or publish code poetry, including Stanford University’s *Code Poetry Slam*, the PerlMonks Perl Poetry Page, and the International Obfuscated C Code Contest.

---

LOLCODE

Designed to resemble the speech of lolcats. Although frequently criticized for not being terribly esoteric, as it is really just a procedural language with an interesting vocabulary, LOLCODE is a prime example of cultural influence on programming.

MALBOLGE

Malbolge is the eighth circle of hell in Dante’s Inferno. The esoteric language was designed to be the most difficult esoteric programming language. Among other features, code is self-modifying by design and the effect of an instruction depends on its address memory.

PIET

In Piet, programs are bitmaps that look like abstract art. There are 20 colours for which behaviour are specified—18 colourful colours, and black and white. This esoteric language was named after the Dutch painter Piet Mondrian, as Mondrian was already taken.

BRAINF*@K

This language, with its rather aggressive name, is designed for extreme minimalism and leads to obfuscated code. Programs only contain eight simple commands and an instruction pointer. While it is Turing-complete, it is intended to challenge and amuse programmers.
We will be using a number of programming specific language during this demonstration. The words below probably mean something slightly different in everyday use.

- **stack** - a data structure used to store a collection of objects.
- **comment** - a line of code used for informational purposes by programmers.
- **variable** - a container used to store values that can be altered during program execution.
- **goto** - a statement that tells the program which line of code to run next.
- **parser** - takes input data and builds a data structure that it uses to define what the input means in the context of the program.
- **program counter** - contains the address of the next instruction to be executed.
A stack is a data structure used to store a collection of objects. Individual items can be added and stored in a stack using these two operations:

- **PUSH** - adds an element to the top of the collection
- **POP** - removes the most recently added element that has not yet been removed.

The image above shows a simple representation of a stack runtime with PUSH and POP operations. The order in which elements come off a stack gives rise to its alternative name, LIFO or last in, first out. The name stack for this type of structure comes from the analogy to a set of physical items stacked on top of each other. This structure makes it easy to take an item off the top of the stack, while getting to an item deeper in the stack may require taking off multiple other items.

https://en.wikipedia.org/wiki/Stack_%28abstract_data_type%29
The next few slides will act as a primer on writing a program in The Shakespeare Programming Language. With that in mind, let's take a look at the design for an SPL program.

- **Title** - everything up until the first period of an SPL program is the title. This title is merely aesthetic, and serves no real purpose in the code except to comment what the program will be about.

- **Dramatis Personae** - These are the characters, or variables, in your play. The SPL language recognizes virtually any of the myriad characters available within the works of Shakespeare. Characters are listed with their name, followed by a comma, and then a comment about them.

- **Acts and Scenes** - These divide our play (or program) into smaller parts. Acts and Scenes are numbered using Roman numerals, begin with the word Act or Scene, followed by a number, and then a description.
SPL Grammar Cont.

Enter - followed by a list of one or more characters

Exit - followed by exactly one character

Exeunt - followed by a list of at least two characters to leave the stage, or no characters, in which all characters will leave the stage.

In order to interact with other, characters must be on-stage. Enter, Exit, and Exeunt cause the characters to get on and off the stage.
Lines consist of a character name followed by a colon and typically contain one or more data sentences. In this workshop we will use six kinds of sentences:

- **input** - cause the character to require information from the user.
- **output** - causes output to the screen
- **goto** - cause the characters to return to a previous scene or act.
- **conditional statement** - cause the characters to make a choice.
- **pushing/popping** - cause a character to store/remove information.
- **statements** - cause the specified character to assume a certain value.
Assigning Values

THE LEXICON OF PROGRAMMING IN SHAKESPEARE

... the difference between the square of the difference between my little pony and your big hairy hound and the cube of your sorry little goat.

\[
(\text{my little pony} - \text{big hairy hound})^2 - (\text{sorry little goat})^3
\]

\[
\text{noun} \times 2^{\text{number_of_adjectives}} = \text{value}
\]

\[
( (1 \times 2^1) - (1 \times 2^2))^2 - ((-1) \times 2^2))^3 = 68
\]
ASCII, or the American Standard Code for Information Interchange, is a character encoding standard for electronic communication. A numerical value (0-127) is assigned to each letter, digit, character, and non-printing characters, since computers only understand numbers — not English.
Our First SPL Program

Outputting Input Reversedly

This program will read in any number of characters (remember that ASCII I mentioned a few slides back), and then spit them back out in reverse order.
WE KNOW WHAT WE ARE, BUT KNOW NOT WHAT WE MAY BE.

~ WILLIAM SHAKESPEARE

Let's get CODING!