COS333 Practical 1

Dewald Colesky u23536030

2 March 2026

Contents

1 Research Questions

1. A computer programming language designed to experiment with weird ideas. They may be difficult to program in and are often created with a specific goal in mind (As a joke, to explore concepts, to be 'artistic', etc.)

2. Useful contributions of esoteric programming languages:

   - Test the boundaries and possibilities of a programming concept
   - Acts as a challenge for programmers
   - Can teach programming concepts minimally (like Turing completeness)
   - Exploring language design and minimalism

3. Chosen Esoteric Languages (a.Whitespace, b.Intercal)

   (a) Whitespace is an imperative, stack-based language (similiar to Forth), created by Edwin Brady and Chris Morris in 2002. Whitespace defines a command as a sequence of whitespace characters (Tab, Space, Linefeed). Whitespace was created as a joke about code readability and the absolute minal visible footprint.

   (b) Intercal ("Compiler Language With No Pronounceable Acronym") is deliberately unconventional and parodic, aiming to diverge from all major programming languages of the era (FORTRAN, COBOL, ALGOL).Intercal was designed by Donald R. Woods and James M. Lyon in 1972. Programs consist of statements optionally labelled in parantheses and each statement must begin with 'politeness qualifers' such as "DO", "PLEASE", "PLEASE NOTE" (Over or underuse of "PLEASE" can lead to compiler rejection for being too polite or impolite). The semantics emphasize absurdity and (minimal) similiarity to conventional languages.

Hello world in Whitespace (The following source code is for a Whitespace "Hello, world!" program. For clarity, it is annotated with S, T and L before each space, tab, and linefeed.)

```
S S S T S S T S S S L:Push_+1001000=72='H'_onto_the_stack
T L
S S :Output_'H';_S S S T T S S T S T L:Push_+1100101=101='e'_onto_the_stack
T L
S S :Output_'e';_S S S T T S T T S S L:+1101100=108='l'
T L
S S S S S T T S T T S S L:+1101100=108='l'
T L
S S S S S T T S T T T T L:+1101111=111='o'
T L
S S S S S T S T T S S L:+101100=44=','
T L
S S S S S T S S S S S L:+100000=32=Space
T L
S S S S S T T T S T T T L:+1110111=119='w'
T L
S S S S S T T S T T T T L:+1101111=111='o'
T L
S S S S S T T T S S T S L:+1110010=114='r'
T L
S S S S S T T S T T S S L:+1101100=108='l'
T L
S S S S S T T S S T S S L=+1100100=100='d'
T L
S S S S S T S S S S T L:+100001=33='!'
T L
S S :Output_'!';_L
L
L:End_the_program
```

The following is a "Hello, world!" program in INTERCAL.

```
DO ,1 <- #13
PLEASE DO ,1 SUB #1 <- #238
DO ,1 SUB #2 <- #108
DO ,1 SUB #3 <- #112
DO ,1 SUB #4 <- #0
DO ,1 SUB #5 <- #64
DO ,1 SUB #6 <- #194
DO ,1 SUB #7 <- #48
PLEASE DO ,1 SUB #8 <- #22
DO ,1 SUB #9 <- #248
DO ,1 SUB #10 <- #168
DO ,1 SUB #11 <- #24
DO ,1 SUB #12 <- #16
DO ,1 SUB #13 <- #162
PLEASE READ OUT ,1
PLEASE GIVE UP
```

4. SALT was developed at the Institut für Informatik, Technische Universität München. It is a domain-specific specification and assertion language designed for expressing temporal properties. Advantages for programmers is a higher level of abstraction which makes coding concise. Intuitive temporal specifications are more readable compared to raw LTL formulas while remaining fully translatable to standard temporal logics.

5. Vibe coding is letting an LLM code for you while you just provide prompts and dont understand (or care) what code it gives you.

   - adv. Vibe coding can increase productivity by implementing mundane and routine code (Some algorithms, class headers, boilterplate, etc.) which allows the programmer to have a skeleton almost immediately which can be changed to suit their needs.
   - adv. Vibe coding can also make coding easier for non-technical people allowing them to build functional apps without knowing how to code.
   - disadv. AI can create and add onto small codebases (with questionable quality) but in larger codebases inneficiencies start to add up and will need refactoring.
   - disadv. AI can pose a security risk by having security keys in plain text or even in code. It also doesn't understand low level concepts which can cause security flaws and stability/performance problems.

2 Implementation Questions

References

[1] Jeremy Singer and Steve Draper. Let's Take Esoteric Programming Languages Seriously. In *Proceedings of Onward! '25*, 2025. https://dl.acm.org/doi/10.1145/3759429.3762632

[2] Daniel Temkin. *Forty-Four Esolangs: The Art of Esoteric Code.* MIT Press, 2025. https://mitpress.mit.edu/9780262553087/forty-four-esolangs/

[3] Daniel Temkin. Esoteric Programming Languages: A Unique Challenge. IEEE Spectrum, September 2025. https://spectrum.ieee.org/esoteric-programming-languages-daniel-temkin

[4] Wikipedia contributors. Whitespace (programming language). *Wikipedia, The Free Encyclopedia.* [Online]. Available: https://en.wikipedia.org/wiki/Whitespace_(programming_language) [Accessed: February 2026].

[5] Whitespace examples. University of Durham CompSoc. [Online]. Available: https://web.archive.org/web/20150717115008/http://compsoc.dur.ac.uk/whitespace/examples.php [Archived Jul. 17, 2015].

[6] Wikipedia contributors. INTERCAL. *Wikipedia, The Free Encyclopedia.* [Online]. Available: https://en.wikipedia.org/wiki/INTERCAL [Accessed: February 2026].

https://www.pspace.org/a/publications/icfem06.pdf