The <> Programming Language

<> = <D – Walter Bright, C++ - Lloyd Moore, Rust – Brad Gibson,
Carbon, Swift – Peter Williamson, Cpp2, Go, C#, Mojo, Hylo, Circle>

By: <Panelist Name>

Language History

Who:

What:

Where:

When:

Why:

How:

Status: {Experimental, Production}

Update Schedule: (minor and major updates)

Standards Body: {Yes, No}

Sample Code

Instructions for the sample code: Construct a piece of code written in your selected language that creates a variable by allocating it dynamically and then creates 100 instances of a thread that increment that variable 10000 times. Each instance should contain appropriate locks to handle race conditions. For help the above statement was roughly given to ChatGPT in several instances and generated reasonable code! In the final deck this block of text should be replaced with the sample code.

Memory Safety Features

Memory / Lifetime Management: {Manual, Ref Count, Garbage Collection, Other}

Automatic Bounds Checking: {Yes, No, Partial, Other}

Safe Arithmetic: {Yes, No, Configurable}

Pointers and References: {Supported?, How safe?}

Forced Initialization: {Yes, No}

Pointer Arithmetic: {Yes, No}

Managed Ownership: {None, Borrow Checker, Other}

Thread Safety Features

Sync Primitives: {Atomics, Locks, Memory Barriers}

Reenterant Safety: {Yes, No}

Thread Local Storage: {Yes, No}

Immutable Objects: {Yes, No}

Immutable by Default: {Yes, No}

Communication Primitives: {None, Message Passing, Pipes, etc}

Performance

Translation: {Compiled, Interpreted, JIT}

Deterministic @ Runtime: {Yes, No}

Bare Metal Programming: {Yes, No}

Other Features: {ie: Rust borrow checker allows for more aggressive optimizations, etc.}

Development Ecosystem

Available Tool Chains: {List available}

Library Support: {None, Some, Extensive}

Package Managers: {List available}

IDE / Editor Support: {List available}

Language Server Available: {Yes, No}

Community: {Small, Medium, Large}

Other Considerations

Easy or hard to learn?

Easy or hard to find developers?

Suitable for use in regulated environments?

Suitable for use in embedded environments?

Job Availability by career level: {Junior, Mid, Senior} x {Ample, Scarce}

Special features, etc?