

C Language Reference Manual

- I. Introduction
- II. Lexical Conventions
 - A. Tokens
 - B. Comments
 - C. Identifiers
 - D. Keywords
 - E. Constants
 - F. Integer Constants
 - G. Character Constants
 - H. Floating Constants
 - I. Enum Constants
 - J. String Literals
- III. Syntax Notation
 - A. Meaning of identifiers
 - B. Storage Class
 - C. Basic Types
 - D. Derived Types
 - E. Type Qualifiers
 - F. Objects and Lvalues
 - G. Conversions
 - H. Integral Promotion
 - I. Integral Conversion
 - J. Integer and Floating
 - K. Floating Types
 - L. Arithmetic Conversions
 - M. Pointers and Integers
 - N. Void
 - O. Pointers to Void
- IV. Expressions
 - A. Pointer Generation
 - B. Primary Expressions
 - C. Postfix Expressions
 - 1. Array References
 - 2. Function Calls
 - 3. Structure References
 - 4. Postfix Incrementation
 - D. Unary Operators
 - 1. Prefix Incrementation Operators
 - 2. Address Operator
 - 3. Indirection Operator
 - 4. Unary Plus Operator
 - 5. Unary Minus Operator
 - 6. One's Complement Operator
 - E. Logical Negation Operator
 - F. Sizeof Operator
 - G. Casts
 - H. Multiplicative Operators
 - I. Additive Operators
 - J. Shift Operators
 - K. Relational Operators
 - L. Equity Operators
 - M. Bitwise AND Operator
 - N. Bitwise Exclusive OR Operator
 - O. Bitwise Inclusive OR Operator
 - P. Logical AND Operator
 - Q. Logical OR Operator
 - R. Conditional Operator
 - S. Assignment Expressions
 - T. Comma Operator
 - U. Constant Expressions
- V. Declarations
 - A. Storage Class Specifiers
 - B. Type Specifiers
 - C. Structure and Union Declarations
 - D. Enumerations
 - E. Declarators
 - F. Meaning of Declarations
 - G. Pointer Declarators
 - H. Array Declarators
 - I. Function Declarators
 - J. Initialization
 - K. Type Names
 - L. Typedef
 - M. Type Equivalence
 - N. Statements
 - O. Labeled Statements
 - P. Expression Statement
 - Q. Compound Statement
 - R. Selection Statements
 - S. Iteration Statements
 - T. Jump Statements
- VI. External Declarations
 - A. Function Definitions
 - B. External Declarations
 - C. Scope and Linkage
 - 1. Lexical Scope
 - 2. Linkage
- VII. Preprocessing
 - A. Trigraph Sequences
 - B. Line Splicing
 - C. Macro Definition and Expansion
 - D. File Inclusion
 - E. Conditional Compilation
 - F. Line Control;
 - G. Error Generation
 - H. Pragmas
 - I. Null Directive
 - J. Predefined Names
- VIII. Grammar

Stripped Language: AWK and LUA

Support Data Types: Double, Int, String, Set
Strongly Typed

Types inferred: 0.0, 0, "", []

Automatic Coercion: int -> double

Operators: + - * / == = && || []

string + string; string + int/double/set
highest precedence; returns string

set + set: concat sets

double + double: add

int + int: add

set[#] for access

Identifiers: name that is not reserved

Control Flow

```
if(expr){/*expr = 1 true, expr = 0*/
```

```
}else if(expr){
```

```
}else{
```

```
}
```

```
while(expr){
```

```
}
```

Expressions: id = expression;

```
print ""
```

General Structure

```
open "" /*Optionally opens file*/
```

```
{/*Begin*/}
```

```
[/Pattern*]/{/*Action*/}
```

```
    [/Subpattern*]/{/*Subaction*/}
```

```
}
```

```
[/Pattern*]/{/*Action*/}
```

```
    [/Subpattern*]/{/*Subaction*/}
```

```
}
```

```
{/*End*/}
```

Begin scope continues for entire pattern

Pattern scope continues for rest of program

End scope begins at start of end block and

ends at closing parentheses

Functions: think of a named pattern

```
name [/params*]/{
```

```
    return 0; //default return type
```

```
}
```

Defined in Begin and End Blocks

Patterns

```
["regex"]
```

```
["css selectors"]
```

Line by Line?

Access: this[#]

Open Questions

How to we handle multiple files?

Do we want to provide import statements?

What do we compile to? I suggest Java byte code since we know Java, there are good frameworks for parsing regex and xml, and it will be a lot faster than python.

Timeline

1. Determine target language (Mon)
2. Resolve open questions (Mon)
3. Fix syntax/Add + remove conventions (Mon)
4. Language Reference Manual Split (2 weeks)
 1. Introduction + Syntax
 2. Expressions
 3. Declarations
 4. Grammar