<Booleanators>

<Boolean Logic Simulator in C++ Software Development Plan > Software Requirements Specifications

Version <1.0>

Revision History

Date	Version	Description	Author
03/21/2024	1.0	Filled out all information needed to turn in assignment	Everyone

Table of Contents

Software Requirements Specifications

1. Introduction

1.1 Purpose

The purpose of this software is to take arithmetic expressions as input, parse it, and then calculate the result according to the order of operations. The project will help us reinforce our understanding of parsing techniques, data structures, and algorithm design.

1.2 Scope

The program is a C++ program that demonstrates the functionality of logical operators like AND, OR, NOR, NAND, XOR, expression parsing, truth value input, evaluation and output, error handling and parenthesis handling.

1.3 Definitions, Acronyms, and Abbreviations

- AND (&): Returns True if both operands are True
- OR (|): Returns True if at least one operand is True
- NOT (!): Inverts the truth value of its operand
- NAND (@): Returns True only if both operands are False (opposite of AND)
- XOR ($): Returns True if exactly one operand is True
- T: True Statement
- F: False Statement

1.4 Overview

The rest of the Software Requirements Specifications document details interfaces used to create and use the program, the specific functionalities of the program and any supplementary requirements.

2. Overall Description

2.1 Product perspectives

2.1.1 User Interfaces

The user interface will feature text-based command lines input.

2.1.2 Software Interfaces

The software interface for the program will be C++.

2.1.3 Memory Constraints

No Memory constraints are known.

2.2 Product functions

The primary function of the product will be Boolean Logic Simulation.

2.3 User characteristics

The user will be able to enter complex logic circuits with multiple gates and input/output signals.

2.4 Constraints

The program must be made with C++ and the program must run on a Linux machine as well as Windows and OS.

3. Specific Requirements

3.1 Functionality

3.1.1 <Operator Support>

The system will have implemented logical operations for

- AND (&): Returns True if both operands are True
- OR (|): Returns True if at least one operand is True
- NOT (!): Inverts the truth value of its operand
- NAND (@): Returns True only if both operands are False (opposite of AND)
- XOR ($): Returns True if exactly one operand is True

3.1.2 <Expression Parsing>

The program will be able to parse user-provided Boolean expressions in infix notation, respecting operator precedence and parentheses.

3.1.3 <Truth Value Input>

The program will be able to Allow users to define truth values (True/False) for each variable represented by T and F.

3.1.4 <Evaluation and Output>

The program will calculate the final truth value of the entire expression and present it clearly (True or False).

3.1.5 <Error Handling>

The program will possess robust error handling for invalid expressions, missing parentheses, unknown operators, or other potential issues, and provide informative error messages.

3.1.6 <Parentheses Handling>

The program will be able to handle expressions enclosed within parentheses (including seemingly excessive but correctly included pairs of parentheses) to determine the order of evaluation.

3.2 Supplementary Requirements

The program will

- use object-oriented programming principles to structure the code using C++
- include comments and documentation to explain the logic and functionality of the program.
- provides clear and informative error messages for invalid input or situations.
- possess user-friendly interface (text-based or graphical) for interacting with the Boolean expression evaluator.

4. Classification of Functional Requirements

Functionality	**Type**
Logical operations for the boolean expressions AND, OR, NOT, NAND, and XOR	Essential
Mechanism to parse user-provided Boolean expressions in infix notation with respect to operator precedence and parentheses	Essential
Mechanism that allows users to define truth values for each variable represented by T and F	Essential
Mechanism that calculates the final truth value of the user's entire expression and presents the result clearly as True or False	Essential
Mechanism that handles errors for invalid expressions, missing parentheses, unknown operators, or other potential issues and provides informative error messages	Essential
Mechanism that allows the program to handle expressions enclosed in parentheses	Essential