Tabulate
Language Specification

Anshul Sangrame
CS21BTECH11004

Varun Gupta
CS21BTECH11060

Gautam Singh
CS21BTECH11018

Contents

***Abstract*—This document provides the language specification for *Tabulate*, a domain specific language (DSL) that provides programming constructs to automate spreadsheet processing efficiently.**

1. Introduction

1.1. Motivation

Spreadsheets are an integral part of our lives. Whether it comes to creating timetables, bookkeeping possessions or tabulating marks, it is difficult to imagine life without spreadsheets. Unfortunately, most spreadsheet softwares like Microsoft Excel and Google Sheets are *What You See Is What You Get* (WYSIWYG) editors. These softwares do not offer a very good programming interface, which in most cases can automate jobs much faster.

That's where Tabulate comes in. With high level programming constructs to abstract the implementation of seemingly complex operations, Tabulate makes it possible to *program* your spreadsheet.

1.2. Goals

Tabulate aims to be the go-to DSL for those who manage many spreadsheets in their everyday life. It aims to automate the tedious process of repetitive entries, updates, and formulae with high performance and efficiency. Built on top of C++, Tabulate aims to provide the programmer with more control over their spreadsheet.

2. Lexical Conventions

2.1. Comments

Tabulate has only one kind of comments; these are of the form `#...#`. Notice that this style of comments can be either single line or multiline.

2.2. Whitespaces

Whitespaces in Tabulate are useful only in separating tokens. Excess whitespaces are ignored.

2.3. Reserved Keywords

2.4. Identifiers

Identifiers in Tabulate can contain characters, digits and underscore. However, they must start with either a character or underscore. Identifiers are case-sensitive.

2.5. Punctuators

3. Data Types

3.1. Primitive Data Types

3.2. Non-Primitive Data Types

4. Operators and Expressions

5. Statements

`Main mdrchd hu`

5.1. Simple Statements

In Tabulate, the simplest type of statements are *simple statements*. These statements are delimited with a semicolon. Expression statements can be of the following types.

5.1.1. Declaration Statements. The syntax for declaration statements is given in . Note that Tabulate allows for multiple declarations in a single statement for the same data type, with the help of comma for separation.

5.1.2. Function Calls. The syntax for function calls is given in . Note that nested function calls are allowed in Tabulate. Further, note that there are no return statements in Tabulate as the variable and its type are declared at function declaration.

5.1.3. Expression Statements. Expression statements in Tabulate must contain a left and right hand side, as shown in .

5.1.4. Jump Statements. Tabulate supports the jump statements `break` and `continue`.

5.2. Compound Statements

In their simplest form, compound statements in Tabulate contain one or more expression statements nested within scope braces. However, compound statements can be nested in other compound statements.

5.3. Selection Statements

Tabulate also offers a construct for selecting statements to be executed based on one or more conditions. The syntax for such selection statements is illustrated in

5.4. Iteration Statements

Tabulate offers one iteration statement construct, illustrated in

6. Functions

6.1. Definition

The syntax for definition of a function is given in . Note that the return variable need not be declared in the function body.

6.2. Main Function

Programs that can be executed must contain a `main` function, as illustrated in .

7. Importing Code

Tabulate offers a powerful feature in the form of importing source code for use in other codes through the `import` directive. The following points regarding its use should be noted.

1) `import` directives occur at the top of the source code.
2) Source codes containing a `main` function cannot be imported in other codes.

8. Standard Library Functions

Tabulate contains an extensive standard library to handle your basic spreadsheet requirements without having to import anything. These requirements are implemented purely using standard C++ libraries.