The Intentio Language Reference

Anna Bukowska, Marek Kaput

May 17, 2018

Contents

Preface ii

I The Intentio Language 1

1 Introduction 2

1.1 Notation 2

1.2 Program structure 3

1.3 Values and types 3

1.4 Namespaces 3

2 Lexical structure 4

II The Intentio Standard Library 5

3 Introduction 6

A Influences 7

Preface

We live in times of rapid emerging of new, modern programming languages. Some of them, like Rust[7], Go[6] or Swift[1], have proved that programming language styles have not settled down and there is still room for new ideas, especially for merging existing paradigms. Fundamentally, one can observe a shift from imperative programming to functional programming.

Despite all these changes, not all ideas get a chance to shine. Some of them are becoming forgotten and treated as esoteric. One of these is goal-oriented evaluation, with Icon[10] being one of its most *iconic* implementers. Authors of this document believe that Icon exposes some very interesting ideas and they made an attempt to recreate them in a new programming language: *Intentio*, named after Latin for *intention*.

This document is the primary reference for the Intentio programming language. It consists of three parts:

- Chapters that *semi-formally*[1] describe each language construct and their use.
- Chapters that *semi-formally* describe runtime services and standard library that are core part of the language.
- Various appendix chapters.

This document does not serve as a beginner-friendly introduction to the language.

[1]This document tries to maintain reasonably formal description of all items, but there are no guarantees all cases are described. As a fallback, the *intentioc*[2] compiler can be used as secondary reference.

Goals

The primary goals when designing Intentio was for language to satisfy following constraints:

1. It should feature core concepts of Icon language: goal-oriented evaluation and generators
2. It should support Unicode character set
3. It should be simple, source code should be easy to read and understand
4. It should be fast and easy to build prototype applications, the language should be *ergonomic* from developer perspective **and** *IDE friendly*
5. It should feature rich capabilities in processing textual data

Our main goals were **not**:

1. The language should be general purpose language
2. Compilation time should be short
3. Memory usage of Intentio programs should be low
4. It should be easy to integrate with other languages

Acknowledges

The layout and parts of content of this document are inspired by two existing language specifications which we believe are good examples to follow: The Rust Reference[8] and Haskell 2010 Language Report[4].

Part I

The Intentio Language

Chapter 1

Introduction

Intentio is domain specific, imperative programming language oriented for processing textual data. Intentio provides goal-oriented execution, generators, strong dynamic typing with optional type annotations, and a rich set of primitive data types, including Unicode strings, lists, arrays, maps, sets, arbitrary and fixed precision integers, and floating-point numbers. Intentio tries to incorporate ideas of the Icon[10] programming language into modern programming patterns.

This part defines the syntax of Intentio language and informal abstract semantics for the meaning of such programs. We leave as implementation detail how Intentio programs are manipulated, interpreted, compiled, etc. This includes all steps from source code to running program, programming environments and error messages. This also means that this document do not describe the reference compiler for Intentio language - *intentioc*[2].

1.1 Notation

Throughout this document a BNF-based notational syntax is used to describe lexical structure and grammar:

$$nonterminal ::= \texttt{term} | \texttt{anotherterm} + nonterm2 ::= nonterminal*$$

Following conventions are used for presenting productions syntax:

($pattern$)	grouping
[$pattern$]	optional
$pattern*$	zero or more repetitions
$pattern+$	one or more repetitions
$pattern_a \| pattern_b$	choice
`token`	terminal symbol (in fixed-width font)

Unicode productions

A few productions in Intentio's grammar permit Unicode[9] code points outside the ASCII range. These productions are defined in terms of character properties specified in the Unicode standard, rather than in terms of ASCII-range code points. Intentio compilers are expected to make use of new versions of Unicode as they are made available.

1.2 Program structure

1.3 Values and types

1.4 Namespaces

Chapter 2

Lexical structure

This chapter describes the lexical structure of Intentio. Most of the details may be skipped in a first reading of the reference.

In this chapter all whitespace is expressed explicitly in syntax descriptions, there is no implicit space between juxtaposed symbols. Terminal characters represent real characters in program source code.

Part II

The Intentio Standard Library

Chapter 3

Introduction

This part defines the Intentio Standard Library (shortly *stdlib*), its contents, semantics and the *prelude* which is automatically imported in each Intentio program. This library provides essentials for building proper Intentio programs, some of which are used by the compiler through compiler intrinsics. It also provides for common convenience utilities which ease application development and make a standard for interoperation within code. The standard library must be distributed with each implementation of the Intentio language.

Appendix A

Influences

Intentio is not particularly original language, having the language Icon as the main source of inspiration, but also borrowing design element from wide range of other sources. Some of these are listed below:

- Icon[10]: goal-oriented execution, generators
- Rust[7]: syntax
- Erlang[3]: syntax
- Python[5]: syntax

Bibliography

[1] Apple Inc. The Swift Programming Language. `https://swift.org/`.

[2] A. Bukowska and M. Kaput. *intentioc* - The reference Intentio compiler. `https://github.com/intentio-lang/intentio`.

[3] Ericsson AB. Erlang programming language. `http://www.erlang.org/`.

[4] S. Marlow. Haskell 2010 language report. `https://www.haskell.org/definition/haskell2010.pdf`.

[5] Python Software Foundation. Python language reference, version 3.6. `https://docs.python.org/3.6/reference/`.

[6] The Go Authors. The Go Programming Language. `https://golang.org/`.

[7] The Rust Project Developers. The Rust Programming Language. `https://doc.rust-lang.org/book/`.

[8] The Rust Project Developers. The Rust Reference. `https://doc.rust-lang.org/reference/`.

[9] The Unicode Consortium. The Unicode Standard. Technical Report Version 6.0.0, Unicode Consortium, Mountain View, CA, 2011.

[10] University of Arizona. The Icon Programming Language. `https://www2.cs.arizona.edu/icon/`.