

Notexbook Documentation

C. H. Lay

August 4, 2018

Contents

1	Usage Principles	5
2	The Stylings	7
2.1	Terms	7
3	Pseudolanguage	9
3.1	Basic Syntax	9
3.2	Comments	9
3.3	Data Types	9
3.4	Control Flow	10
3.5	Functions	10
3.5.1	Iterative Process	10
3.5.2	Recursive Process	10
3.6	Standard Library	10
4	Algorithms	11

Chapter 1

Usage Principles

Any *ordinary page* is a standalone article. The article formats are optimized for computer science, mathematics and biology. The purpose is to write individual documents to refer back in the future without a need to reread from original sources. The base level for subject titles is *section*.

Because the articles are written in L^AT_EX, you can use any package that do not have conflicts with the commands defined here. However, if they do it is easier to conflicting ones in `Theme.sty` file.

Chapter 2

The Stylings

2.1 Terms

Many scientific texts seem to italicize a new term whenever it first occurs. We also follow that convention by italicizing and coloring the term.

Example 1. The *term style* is defined in `\term` command.

Chapter 3

Pseudolanguage

Code examples are written in a *pseudolanguage* which is a mix of Kotlin, Python and C. It also grows to support new programming paradigms whenever necessary. The core feature is to provide compact notations and readable syntax.

Also note that I have not designed or invented any of the features I am describing. This is just a way to organize and simplify the programmatic thought processes.

3.1 Basic Syntax

1. Every expression is ended by ';' character.
- 2.

3.2 Comments

Example 2. The language supports two kinds of comments.

```
# This is a single-line comment.
/*
 * The multi-line comment.
 */
```

3.3 Data Types

Example 3. The data types are declared similarly to UML inspired by Kotlin.

```
a: Integer = 1           # or Int.
b: Char = 'j'           # Is equivalent to Integer.
d: String = "example"   # or Str, is equivalent to Char[].
```

3.4 Control Flow

3.5 Functions

Example 4. The functions are inspired by Kotlin and Scheme.

```
define square(x: Integer)
{
    return (* x x);
}: Integer

# The function above can be simplified to
{def square(x: Int) (* x x)}: Int;
```

3.5.1 Iterative Process

3.5.2 Recursive Process

3.6 Standard Library

Lets fantasize that everything is accessible. The basic or common functions are defined in the `Theme.sty`.

Chapter 4

Algorithms

Algorithms pseudocode are written in mathematical style mixed with the pseudolanguage defined in previous chapter.

Example 5. An example of algorithm using algorithmicx.

Algorithm 1 Multiply two integers

Require: Integers a, b

Ensure: The product c of a and b .

```
function MULTIPLY( $a, b$ )  
   $c \leftarrow 0$   
  while  $b \neq 0$  do  
     $c = c + a$   
  return  $c$ 
```
