Contents

A title for the code goes here

```
SUB def // hello
```

1 Overview

We will walk the AST generated in A3 to create code in A5.

1.1 Visitor Pattern

When building the AST (A3), we chose to implement an `ast_visit` method for each type of node within the class for that node. For A5, we will instead choose to use a visitor pattern. This provides cleaner code that can be easily swapped and adjusted. We will have a `CodeGenVisitor` class that defines how code is generated for each type of AST node. We will also create a `LHSVisitor` class to generate code for the left-hand side of assignment statements (`:=`).

2 Storage

2.1 Variables

Variables are either free or bound in any particular scope. We say a variable is bound if it is declared in that scope, otherwise it is free.

Bound vs Free

```
{
  var x:Integer // x bound
  {
    var y:Integer // y bound
    x:= 1 /* x free */
  }
}
```

Because our AST checks that every free variable has been declared, all free variables are declared *somewhere*. We will augment our symbol table from A3 to now include offsets.

2.1.1 Main

2.1.2 Procedures and Functions

2.1.3 Minor Scopes

2.2 Integers and Booleans

2.3 Text

3 Expression

3.1 Constants

3.2 Scalars

3.3 Array elements

3.4 Arithmetic Operators

3.5 Comparison Operators

3.6 Boolean Operators

3.7 Conditionals

4 Functions and Procedures

4.1 Activation Record

4.2 Entrance Code

4.3 Exit Code

4.4 Parameter passing

4.5 Function Calls and Value Return

4.6 Procedure Calls

4.7 Display Management

5 Statements

5.1 Assignment

5.2 If

5.3 While and Repeat

5.4 Returns