GROUP:
KHAIRULLAH KHALIQ (04072113003)
QASIM SHABBIR (04072113027)

CS-411

COMPILER CONSTRUCTION

Vortex Language Syntax with Corresponding CFG

Contents

1. Main Function

Syntax:

```
func main() {
    // program starts here
}
```

CFG:

```
<Function> ::= "func" <Identifier> "(" ")" <Block>
<Block> ::= "{" <StmtList> "}
<StmtList> ::= <Statement>
```

2. Variable Declaration

Syntax:

```
num age = 25;
const str name = "Ali";
```

CFG:

```
<Statement> ::= <VarDecl> ";"
<VarDecl> ::= ["const"] <Type> <Identifier> "=" <Expr>
<Type> ::= "num" | "str" | "bool" | "list"
```

3. Input

Syntax:

```
str name;
in(name);
```

CFG:

```
<Statement> ::= <InputStmt> ";"
<InputStmt> ::= "in" "(" <Identifier> ")"
```

4. Output

Syntax:

```
out("Hello, " + name + "!");
```

CFG:

```
<Statement> ::= <OutputStmt> ";"
<OutputStmt> ::= "out" "(" <Expr> ")"
```

5. Assignment

Syntax:

```
age = 30;
students[0] = "Ali";
```

CFG:

```
<Statement> ::= <Assignment> ";"
<Assignment> ::= <Identifier> "=" <Expr>
               | <Identifier> "[" <Expr> "]" "=" <Expr>
```

6. Function Call

Syntax:

```
greet("Ali");
```

CFG:

```
<Statement> ::= <FuncCall> ";"
<FuncCall> ::= <Identifier> "(" <ArgList>? ")"
<ArgList> ::= <Expr> ("," <Expr>)*
```

7. Function Definition with Parameters and Return

Syntax:

```
func add(num a, num b) -> num {
    give a + b;
}
```

CFG:

```
<Function> ::= "func" <Identifier> "(" <ParamList>
")" "->" <Type> <Block>
<ParamList> ::= <Param> ("," <Param>)*
<Param> ::= <Type> <Identifier>
```

8. Function with Default Parameter

Syntax:

```
func greet(str name = "Guest") {
    out("Hello, " + name);
}
```

CFG:

```
<Param> ::= <Type> <Identifier> [ "=" <Expr> ]
```

9. Return Statement

Syntax:

```
give a + b;
```

CFG:

```
<Statement> ::= <ReturnStmt> ";"
<ReturnStmt> ::= "give" <Expr>
```

10. Conditional Statement

Syntax:

```
when (x > 5) {
    out("High");
}
whenelse (x > 2) {
    out("Medium");
}
else {
    out("Low");
}
```

CFG:

```
<Statement> ::= <IfStmt>
<IfStmt> ::= "when" "(" <Expr> ")" <Block>
<WhenElseSeq>? ["else" <Block>]
<WhenElseSeq> ::= <WhenElse> <WhenElseSeq>?
<WhenElse> ::= "whenelse" "(" <Expr> ")" <Block>
```

11. repeat Loop

Syntax:

```
repeat (num i = 0; i < 5; i += 1) {
    out(i);
}
```

CFG:

```
<Statement> ::= <RepeatLoop>
<RepeatLoop> ::= "repeat" "(" <VarDecl> <Expr> ";"
<Assignment> ")" <Block>
```

12. cycle Loop

Syntax:

```
cycle (x < 3) {
    x += 1;
}
```

CFG:

```
<Statement> ::= <CycleLoop>
<CycleLoop> ::= "cycle" "(" <Expr> ")" <Block>
```

13. perform-cycle Loop

Syntax:

```
perform {
    x -= 1;
} cycle (x > 0);
```

CFG:

```
<Statement> ::= <PerformLoop>
<PerformLoop> ::= "perform" <Block> "cycle" "("
<Expr> ")" ";"
```

14. break and skip

Syntax:

```
break;
skip;
```

CFG:

```
<Statement> ::= "break" ";" | "skip" ";"
```

15. select / case / default

Syntax:

```
select (option) {
    case 1: out("One"); break;
    case 2: out("Two"); break;
    default: out("Invalid");
}
```

CFG:

```
<Statement> ::= <SelectStmt>
<SelectStmt> ::= "select" "(" <Expr> ")" "{" <CaseList>
[<DefaultCase>] "}"
<CaseList> ::= <Case>*
<Case> ::= "case" <Literal> ":" <StmtList> "break" ";"
```

```
<DefaultCase> ::= "default" ":" <StmtList>
```

16. try / catch

Syntax:

```
try {
    num x = 5 / 0;
} catch (err) {
    out("Error!");
}
```

CFG:

```
<Statement> ::= <TryCatch>
<TryCatch> ::= "try" <Block> "catch" "(" <Identifier> ")" <Block>
```

17. List Declaration

Syntax:

```
list values = [1, 2, 3];
```

CFG:

```
<List> ::= "[" <ExprList>? "]"
<ExprList> ::= <Expr> ("," <Expr>)*
```

18. List of Objects

Syntax:

```
list students = [
    {str name -> "Ali", num age -> 22},
    {str name -> "Sara", num age -> 23}
];
```

CFG:

```
<Object> ::= "{" <ObjectField> ("," <ObjectField>)*
"}"
```

```
<ObjectField> ::= <Type> <Identifier> "->" <Expr>
```