CSC 372 Language Documentation

# CSC 372 Language Documentation

## Let's discover our language.



## Getting Started

Get started by navigating to our **GitHub Repository**, containing all of our code and documents.

## What you'll need

- Python and your favorite IDE!

## Basic Structure

```
    print("Parameter",a,b,c);

    string name = input();
    int y = int(input());

    print("Values inputted:");
    print(name,y);

    print("Loop");
    for(int x = 10;x<15 && x>=10 && x!=20;x+=1){
        if(x==12){
            for(int y=100;y<200;y+=10){
                if(y>110 && y<=190)
                {
                    print(y);
                }
            }
        }
        else { print(x); }
    }
    return 0;
}

print("Calling function");
int ret = asd(1,"asd",false);
print("Function returned",ret);
```

Programs written in our language follow an easy structured format. A program is defined as a series or list of statements. Statements include declarations, assignments, expressions and evaluation, function defition and calls.

```
<program> ::= <statement_list>
<statement_list> ::= <statement> <statement_list> | <statement>
```

## Data Types

Our language supports three basic data types:

- Integers, digits 0 - 9
- Booleans, true/false
- Strings, collection of characters

```
<identifier> ::= <letter> (<letter> | <digit>)*
<integer> ::= <digit>+
<string> ::= "\"" <string_helper> "\""
<boolean> ::= "true" | "false"
<letter> ::= "a"–"z" | "A"–"Z"
<digit> ::= "0"–"9"
```

## Declarations

```
// Declare an integer
int myInt = 10;

// Declare a boolean
bool myBool = true;

// Declare a string
string myString = "Hello, World!";
```

Variables are strongly types and must be declared. For declarations, they must supply a name with a data type when initializing.

```
<variable_declaration> ::= <int_declaration> | <bool_declaration> | <string_declaration>
<int_declaration> ::= "int " <identifier> " = " <basic_expression>
<bool_declaration> ::= "bool " <identifier> " = " <boolean_expression>
<string_declaration> ::= "string " <identifier> " = " <string>
```

## Expressions and Operators

```
int a = 5;
int b = 10;

// Arithmetic expression
int sum = a + b;
```

```
// Comparison expression
bool isGreater = a > b;

// Boolean expression
bool result = (a < b) && (sum > 10);
```

Expressions can be arithmetic, boolean, or comparison-based. The following are supported:

- Arithemetic: addition '+', minus '-', multiplication '*', divide '/', and modulus '%'
- Boolean: and '&&', or '||', not '!'
- Comparison: less than, greater than, equals, less than or equal to, greater than or equal to

```
<expression> ::= <basic_expression> | <boolean_expression> | <comparison_expression>
<basic_expression> ::= <integer> ("+" | "-" | "*" | "/" | "%") <basic_expression> | <integer>
<boolean_expression> ::= <boolean_expression> ("&&" | "||") <boolean_expression> | "!" <boolean_express
<comparison_expression> ::= (<identifier> | <integer>) <comparison_operator> (<identifier> | <integer>)
```

## Conditional Statments

```
bool isSunny = true;
if (isSunny) {
    print("It's a sunny day!");
} else {
    print("It might rain today.");
}
```

Allows for standard conditional logic statements which execution is based on boolean expressions.

```
<conditional_statement> ::= "if" "(" <boolean_expression> ")" "{" <statement_list> "}"
                          | "if" "(" <expression> ")" "{" <statement_list> "}" <conditional_statement>
                          | "if" "(" <expression> ")" "{" <statement_list> "}" "else" "{" <statement_li
```

## Loops

```
// Print numbers from 1 to 5
for (int i = 1; i <= 5; i = i + 1) {
    print(i);
}
```

For loops are supported, they require initialization of looping variable, boolean condition, and iteration expression to progress the loop.

```
<loop_statement> ::= "for" "(" <assignment> ";" <boolean_expression> ";" <assignment> ")" "{" <statemen
```

## Functions

```
// Function declaration
function int add(int x, int y) {
    return x + y;
}
```

Functions can be declared with a return type, a list of valid paramters, followed by a block of exectuable statements.

```
<function_declaration> ::= "func " <data_type> <identifier> "(" <parameters_list> ")" "{" <statement_li
                         | "func " <data_type> <identifier> "(" <parameters_list> ")" "{" <statement_li
<return_statement> ::= "return " (<identifier> | <expression>) ";"
```

## Input / Output

```
print("CSC 372");
String x = input("is awesome");
```

Our language supports the basic operations of input and output. `print` statements will be used for output, while we can capture user input with `input`, which reads a string that can be converted to other types.

```
<print_statement> ::= "print" "(" <string> ")" ";"
<input_statement> ::= <data_type> <identifier> "=" "input" "(" ")" ";"
```

Edit this page