Language Reference Manual

CS18B002
CS18B033

Lexical Elements

Identifiers:

- Identifiers are sequences of characters used for naming variables, functions. You can   include letters, decimal digits, but first character cannot be a digit
- Lowercase and uppercase letters are distinct.

Keywords:

- Keywords are special identifiers reserved for use as part of the programming language itself. You cannot use them for any other purpose.

void	int	float	char	print
if	double	return	getInt	const
else	block	while	unsigned	static

Separators

( ) [ ] { } ; ,

White Space:

- White space is the collective term used for several characters: the space character, the tab character, the newline character, the vertical tab character, and the form-feed character.
- White space is ignored (outside of string and character constants), and is therefore optional, except when it is used to separate tokens.

Data Types

Primitive Data Types

Integer

- Sequence of digits are assumed to be decimal base (decimal 10)
- The integer data type is of size 32 bits (4 bytes)

Char

- Stores a single character
- Char data type is of size 1 byte

Arrays

- An array can be declared by specifying its data type, name and length. It can either be 1-Dimensional or 2-Dimensional. The length of the array along each dimension should be positive.
- Arrays can also be initialized by declaring initializing values separated by commas separated by commas in a set of braces.
  Example int arr[5] = {0, 1, 2, 3, 4};
- All array elements need not be initialized explicitly, for example the below code initializes values at indices 3, 4 to zero.
  Example int arr[5] = {13, 14, 14};
- If you initialize every element of an array, then no need to specify it's size, it is determined by the number of elements in the array.
  Example int twoArr = { {1, 2}, {5, 6}, {8, 9} }; (3 rows and 2 columns)

Strings

- An array of characters can be used to construct a string
- It can be declared by specifying a string literal enclosed in double quotation marks or as a comma-delimited list of characters.
- After initialization, you cannot assign a new string literal to an array using the assignment operator.
  Example : char str[5] = {'p', 'q', 'r', 's', '\0'};
  str = "abc";

Scope

- Variables can be declared ***"static"*** to use outside of its regular scope.

Expressions and Operators

Arithmetic operators

- Standard arithmetic operators are addition, subtraction, division and division
- Modular arithmetic is defined on natural numbers

Logical operators

- Precedence is
  (LT, LTE, GT, GTE) >> (EQL, NEQL) >> (Logical AND) >> (Logical OR)

Assignment operator (=)

- Used to assign values to a variable

Statements:

If:

You can use the if statement to conditionally execute part of your program, based on the truth value of a given expression.
Executed if logical expression is TRUE

Else:

Optional, executed if value of logical expression in “If” statement is FALSE

While:

Loop is executed until the looping test fails. General format is

```
while( Logical_Expression ) :
        Statements
```

Functions

Function definition

- A function definition specifies what a function actually does
- A function definition consists of information regarding the function’s name, return type, and types and names of parameters, along with the body of the function. The function body is a series of statements enclosed in braces

return-type function-name (parameter-list);

Function Declaration

- A function is a declaration to specify the name of a function, a list of parameters, and the function's return type. A function declaration ends with a semicolon. Here is the general form:

```
return-type function-name (parameter-list);
function-name(parameter-list){
        logic
}
```

- In our language, *values are passed by value, not passed by reference*
- Example:

```
Int twice(int);
twice(n)
{
        return n+n
}
```

Calling Function

- A function can be called by using its name and supplying any needed parameters. Here is the general form of a function call:

function-name (parameters);

- In our language, *all the specified parameters in function declaration and definition should be specified in function call, variable length arguments are not supported.*
- Function can only return one value
- Equality of functions is not supported

Recursive Functions

- Recursive functions are supported, that is, a function can call itself
- Example of fibonacci function implemented recursively :

```
int fibonacci(int);
fibonacci(n)
{
        if(n<2)  { return n; }
        else { return fibonacci(n-1)+fibonacci(n-2); }
}
```

Main Function

- Every program requires at least one function, called ‘main’. This is where the program begins executing. You do not need to write a declaration or prototype for main, but you do need to define it.
- This function does not support keyword arguments

Sample Programs

1.

```
main()
{
   int a;
   a = 5;
   int b;
   b=4;
   int c;
   c = a+b;
   print(c);
}
```

2.

```
int adder(int);
adder(n)
{
        n+n
}

main() {
        int a;
        getInt(a);
        print(adder(a));
}
```