PartialC

Language Reference Manual

Mingjie Lao (ml4545): System Architect, Tester

Jiaying Song (js5799): Manager, Language Guru

COMS W4115 Programming Languages and Translators

1. Introduction

PartialC is a language that implements part of the functions in C language, including array and pointers. With the functionalities provided by PartialC, many common algorithms like fibonacci can be implemented without difficulty.

2. Data Types

The data types are defined as below:

Data Types	Description	Syntax
int	integer	int x = 1
float	real number	float x=2.3
bool	boolean	bool x = true
string	string	string x = 'abcde'
ANY DATA TYPE[]	array, example: int[] string[]	int[] x = [1,2,3]
void	return type when no value is returned in function	void functionName()

3. Lexical Conventions

I. Key word

There is a list of tokens been reserved for PartialC:

Data type: int / float / bool / string / int[] / float[] / bool[] / string[] / void
Data value: true / false / null
Control flow: if / else / for / while / return
Function: main()

II. Identifier

An identifier is a sequence of letters of upper/lower case letters or underscore to represent variable names. It should be defined before using and the data type should be defined explicitly as well.

Example: int var_3 = 13

III. Literal

a. Integer:
A sequence of digits 0-9 with optional + or - character to indicate sign.

Example: 98, -4

b. Real number:
A sequence of digits 0-9 with period '.' to represent float number. It also has the optional + or - character to indicate sign. The part before the period represents the integer part while the part after represents the fraction part.

   Example: 2.3, -988.20

c. Logical literal:
Both true and false are reserved keywords to represent logical value.

d. String:
An immutable sequence of characters surrounded by single quotes.

   Example: 'abc'

e. Array:
Arrays are a list of elements of the same data type. Elements can be accessed by its position index ranging from 0, enclosed in a square bracket. This immutable single dimension data structure supports int, float, bool, or string as elements. The elements are separated by comma.

   Example: [1,2,3]

f. Array Element:
The specific element in the array is accessed by specifying the index (start from 0).
Example: we first declare int[] a = [1, 2, 3], then we have a[1] = 2

IV. Comment

Comment is any single-line sequence of characters that would be ignored by the language compiler. It is started with two consecutive front slashes //
Example: // This is a comment.

V. White space

White space is ignored by the compiler automatically. Newline \n, Return \r, Tab \t, and space are all considered whitespace.

VI. Separator

Separator type	Description
(	Left parenthesis for expression
)	Right parenthesis for expression

{	Left curly bracket for control flow and function
}	Right curly bracket for control flow and function
;	Semicolon to separate two expressions in control flow condition
,	Comma to separate array elements or arguments in function declaration

VII. Operator

Operator	Description	Example
+	binary arithmetic addition	1 + 5 1.0 + 5.2
-	binary arithmetic subtraction	1 – 5 1.0 - 5.2
*	binary arithmetic multiplication	1 * 5 1.0 * 5.2
/	binary arithmetic division	1 / 5 1.0 / 5.2
%	binary arithmetic modulus	1 % 5
>	binary relational greater than	a > 5
>=	binary relational greater than or equal to	a >= 5
<	binary relational less than	a < 5
<=	binary relational less than or equal	a <= 5
==	binary relational equal	a == 5
!=	binary relational not equal	a != 5
-	unary negation	-a
&&	binary logical AND for boolean expression	m && n
||	binary logical OR	m || n
!	unary logical NOT	!m

VIII. Assignment

The equal sign = is used to perform the assignment of right-hand expression to left-hand side expression.

Example: a = 3

4. Expression

Expression consists of the literals defined in Section 3.

I. Basic component of expression

The expression can be a single literal defined in Section 3. For example, any integer, float, bool, array, array element, string, or identifier.

II. Compound expression

The expression can also be composed of the format: expression OPERATOR expression. Implicit type conversion is supported.

III. Operator precedence

Expressions are evaluated based on the following precedence (from high to low):

Operator	Associativity
()	left
[]	left
!, - (unary)	right
*, /, %	left
+, -	left
>, >=, <, <=	left
==, !=	left
&&, ||	left
=	right
, ;	left

5. Statement

There are five types of statements in PartialC: declaration/assignment, if/else, for, while and return. The statements are terminated by semicolon.

I. Declaration/Assignment

The declaration statement is used to define an identifier.
Example: int a;

The assignment statement is used to assign value to an identifier.
Example: a = 3;

These two statements can also be combined to perform assignments while declaring.
Example: int a = 3;

II. If/Else
This is a compound conditional statement list that is enclosed by {} with no semicolon at the end. The structure is defined below:

```
if (logical expression) {
   statement1;
   statement2;
   ...
}else{
   statement3;
   statement4;
   ...
}
```

III. For
This is a compound statement list that is enclosed by {} with no semicolon at the end. The program starts to loop in the statement until the logical termination expression is reached. And for each iteration, the increment expression will be executed at the end. The structure is defined below:

```
for (<initialization statement>; <logical termination expression>; <increment
expression>){
   statement1;
   statement2;
   ...
}
```

IV. While
This is a compound statement list that is enclosed by {} with no semicolon at the end. The program starts to loop in the statement as long as the logical expression is true. And for each iteration, the increment expression will be executed at the end. The structure is defined below:

```
while ( <logical expression> ) {
   statement1;
   statement2;
   ...
}
```

V. Return

The return statement is used to define the value to be passed back from the current function.

Example: return a;

6. Function

There are two types of function: built in function and user-defined function.

I. Built-in function

There are two built-in function: print(<str>) and sizeOf(<array>)

print() is used to print the string in program output

sizeOf() is used to return the length of the array passed in. Since the array is immutable, this function is evaluated during compile time.

II. User-defined function

The function takes a list of arguments defined in the enclosed parenthesis (separated by comma) and returns a value of the type as defined in the function prototype.
The function with name main() will be executed first.

Function is declared in the following format:

```
ReturnDataType FunctionName (argument a, argument b, ...){
   statement1;
   statement2;
   …
   return expression;
}
```