Casper

Project Proposal

Michael Makris

UNI: mm3443

COMS W4115 (CVN)

September 28, 2018

It is my intention to develop a rather limited in scope general-purpose imperative language, Casper, that resembles the C language, but with more emphasis on the high level than the traditional C low level capabilities. For example, I plan to include a String data type and library functions to manipulate strings. Also, if time permits, I might include dictionary and object structures in addition to arrays. At the same time, I will not be delving into memory manipulation or bitwise operations. In this respect, the language should be able to implement many of the usual algorithms for applications that are programmed in C, Java, and Python.

Language Features

Data Types

Type	**Description**	**Declaration syntax**
Integer	an integer depended on host machine	int x = 0;
Floating point	a floating point number	float x=3.14;
Boolean	reserved words true and false	bool x = true;
String	variable length sequence of characters	str x = “abc”; str x = ‘abc’;
Void	representing the empty set or no value	void x;

Variable Names

Strictly typed. Sequences of uppercase and lowercase letters, numbers and underscores except the reserved words. Global variables defined outside of any block specified by {} otherwise only visible within residing {}.

Reserved words

int float bool str void true false if else for while do until break return print input main

Operators

Operator	**Description**	**Syntax**
+	binary arithmetic addition, string concatenation	1 + 2 1.0 + 2.0 ‘a’ + “b”
-	binary arithmetic subtraction	1 – 2 1.0 - 2.0
*	binary arithmetic multiplication	1 * 2 1.0 * 2.0
/	binary arithmetic float division	1.5 / 2.5
%	binary arithmetic modulus	1 % 2
^	binary arithmetic exponentiation	2 ^ 2 2.0 ^ 0.5
>	binary relational greater than	1 > 2
>=	binary relational greater than or equal	1 >= 2
<	binary relational less than	1 < 2
<=	binary relational less than or equal	1 <= 2
==	binary relational equal	1 == 2
!=	binary relational not equal	1 != 2
++	unary increment (pre or post) an integer	int i = 0; i++; ++i;
--	unary decrement (pre or post) an integer	int i = 0; i--; --i;
=	binary assignment of right-hand expression to left-hand side	int i = 0; str x = “abc”;
+=	binary assignment of the sum of the two sides to the left-hand side	int i = 0; i += 1;
&&	binary logical AND	x && y
||	binary logical OR	x || y
!	unary logical NOT	!x

Precedence

As in C, will add later.

Comments

// for single-line comments after
/* for multi-line comments
inside delimiters */

(I hope to allow nested /**/ if time permits)

Control Flow

White space is ignored
Statements terminated by ;
Expressions defined by () with no ; after
Compound statements/blocks and scope defined by {} with no ; after
Conditional block

```
if (expression1) {statement1;}
else if (expression2) {statement2;}
else {statement3;}
```

Loops

```
for (optional initiation; optional termination; optional increment) { statement;}

while (test expression) { statement;}

do { statement;} until (test expression)
```

with break allowed in statement to exit loop

Functions

Declared as with variables with a type but include a block, optional arguments passed by value within (), and must return a value of declared type unless void.

```
int myFun(str x){
    if(x == 'hello') {return 1;}
    return 0;}
```

As in C, main () is the special function that executes first.

Arrays

Declared as with variables with a type and include multiple values of that type with count in [n]. Can be initialized with one value of same type or a comma-delimited list of same type and same count.

int x[5] = 0; int y[5] = [1,2,3,4,5];

I/O

print (variable) to output any variable to standard output
input (variable) to input from standard input to a variable of a certain type

I will try to implement some useful formatting syntax for I/O if time permits.

Example programs

GCD

```
int gcd(int x, int y) {
   if (y == 0) {
            return x;
   }
     return gcd(y, x % y);
}
```

Quicksort

```
void quicksort (int number[25], int first, int last) {
  int i, j, pivot, temp;

  if(first<last){
    pivot=first;
    i=first;
    j=last;

    while(i<j) {
      while(number[i]<=number[pivot]&&i<last)
        i++;
      while(number[j]>number[pivot])
        j--;
      if(i<j){
        temp=number[i];
        number[i]=number[j];
        number[j]=temp;
      }
    }
    temp=number[pivot];
    number[pivot]=number[j];
    number[j]=temp;
    quicksort(number,first,j-1);
    quicksort(number,j+1,last);
  }
}

int main(){
  int I; int count; int number[25];
  print("How many elements are you going to enter?: ");
  input(count);
  print("Enter " + str(count) + " elements: ");
  for(i=0;i<count;i++) { input(number[i]); }
  quicksort(number, 0, count-1);
  print("Order of Sorted elements: ");
  for(i=0; i<count; i++) { print(number[i]); }
  return 0;
}
```