Open Source Software Lab



Programming in C

Functions

- A function is a piece of code that receives a list of values as input and returns one value
 - identified by name and argument types
 - has an independent scope

Example

```
int getlenth(char * s) {
  int i = 0;
  for (; s != 0x0; s++) i++;
  return i;
}

int main() {
  char * s1, *s2;
  ...
  if (getlenth(s1) < getlenth(s2)) {
   ...</pre>
```

- function declaration
- function definition
- function name
- arguments
- return type
- call site

What Is Function For?

- Functions break large code into smaller pieces
 - function hides the details of a caller from a callee (and vice versa), which reduces complexity
- Function allows programmers to abstract code into highlevel operations
 - functions reduce redundancy in code
 - function allows programmers to write repeated executions of code, such as recursion
- Functions role as interfaces between two modules
 - the interface of a library in Unix (i.e., API) is a list of functions
 - e.g., main function

Function Declaration and Definition

- A function is declared as a triple of a return type, a function name and a list of argument types
 - a function may receive no argument
 - a function may return no value
 - there can be multiple function declarations sharing the same function name while having different lists of argument types (i.e., overloading)
 - a function may receive an arbitrary number of arguments
- A function is defined with a code block
 - each argument must be bonded with a specific variable name
 - the code block must return a value if the function has a return value

Execution Model

```
int get lenth ()
  char * s
 int i = 0;
 for (; *s != 0x0 ; s++)
  i++ ;
  return i ;
int main () {
 int i, max lenth;
  char * s[8];
  for (i = 0; i < 8; i++) {
   s = (char *) malloc(sizeof(char) * 8);
   scanf("%s", s[i]);
 \max lenth = 0;
  for^{-}(i = 0 ; i < 8 ; i++) {
   int r ;
   r = get length(s[i]);
   if (r > max length)
     max lenth = r;
 printf("%d", r);
  return 0;
```

Passing Arguments

- Call by value
 - the value given as an argument is copied to a new variable at a function call

Function

Recursion

- Recursion is to define a solution of a problem with the solutions of the sub-problems
 - a sub-problem shapes in the same form as the original problem, yet having a smaller input
 - definition
 - base case
 - recursion step
- A function is recursive when it calls itself in its body
 - recursion allows a finite logic to solve a problem with an arbitrary size of input

Enumerating Combinations

 Write a function combination that receives an integer array of unique numbers, and then print out all possible combinations of the given numbers

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 /* FIXME */
5
6 void
7 combination (int * list, int n list)
8 {
      /* FIXME */
10 }
11
12
13 int
14 main ()
15 {
       int a[4] = \{1, 2, 3, 4\};
16
       combination(a, 4);
17
18 }
```

```
1 2 3 4
1 3 4
1 3
1 4
1
2 3 4
2 3
2 4
2
3 4
3
```

Maze

- A map of maze is given as a 10x10 grid in maze.txt
 - the cell with '0' represents road
 - the cell with '1' represents wall
 - the cell with 'S' represents the start point
 - the cell with 'D' represents the destination point
 - there is no cycle in the maze
- Complete maze.c to find a route from the start point to the destination point, and print out the route to the screen

```
      S
      0
      0
      0
      0
      0
      0
      1
      1

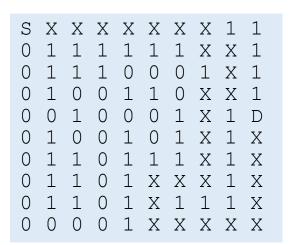
      0
      1
      1
      1
      1
      1
      0
      0
      1
      1

      0
      1
      1
      1
      0
      0
      0
      1
      0
      1

      0
      1
      0
      0
      1
      1
      0
      0
      1
      0
      1
      0
      1

      0
      1
      0
      0
      1
      0
      1
      0
      1
      0
      1
      0
      0
      0
      1
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
```

```
maze.txt
```



result on screen

Expression Evaluation

- An arithmetic expression is one of two cases:
 - an integer
 - (*exp op exp*)
 - expr is an arithmetic expression
 - *op* is either +, -, *, or
- Complete eval.c that reads an arithmetic expression and prints out the evaluation result

```
$./a.out "((1 + (2 * 3)) - (2 + 3))"
2
$
```