CS 240 Programming in C

Functions

September 18, 2019

Homework 1

- For this first time, I'm generous for the different submissions of your homework, which I do not subtract any score.
- But later on I will stick to the requirements made in the homework instructions.
- The first homework is meant to get you started, it is easy score. But you still need to do everything asked in the homework instructions to get the full score.
- For those typescripts which do not show source code or your object code runs correctly, are deducted points by 10, even you wrote your code correctly. It is only fair to other students who paid more attention.

- All the correct compiling and running of your programs will be determined by if they will be correctly compiling and running on server.
- So each time if you were not writing code on server directly, please compile and run your code on server again to make sure you will get the full score.
- Thanks for all your understanding.

Schedule

- Last class, we introduced increment and decrement operators, array and char array. Today let's first do some demos about them.
- Then we will get into functions.

Functions

- Functions provide a convenient way to encapsulate some computation, which can then be used without worrying about its implementation

```
#include <stdio.h>
int power(int m, int n);
/* test power function */
int main(void) {
  int i;
  for (i = 0; i < 10; ++i)
    printf("%d %d %d\n", i, power(2, i), power(-3, i));
  return 0;
}

/* power: raise base to n-th power; n >= 0 */
int power(int base, int n) {
  int i, p;
  p = 1;
  for (i = 1; i <= n; ++i)
    p = p * base;
  return p;
}
```

Function Declaration and Definition

- Function declaration

```
int power(int m, int n);
```

  - Declared before main, says that power is a function that expects two int arguments and returns an int
  - This declaration is called a function prototype
  - If the definition or any use of this function do not follow the pattern in the prototype, you will get an error
  - Note: parameter names are optional in the prototype, but they are often used for clarity

- Function definition

```
returnType funcName(parameter declarations, if any) {
  declarations

  statements
}
```

Parameters vs Arguments

- Formal parameter
  - The names given to the arguments in the function definition
  - Often referred to as parameters
- Actual parameter
  - The names supplied in a function call
  - Often referred to as arguments

Call by Value vs Call by Reference

- In C all function arguments are passed "by value"
- The called function is given the values of its arguments in temporary variables, distinct from the originals
- This means that anything you do to a variable inside a function has no effect on that variable outside of the function
- When call by reference is used, we can alter an argument outside of the function with actions inside the function

Function Example: maxLine, 1/2

```
#include <stdio.h>
#define MAXLINE 1024      //maximum input length
                          //function declarations
int getline(char line[], int maxline);
void copy(char to[], char from[]);

int main(void) {          //print the longest input line
  int len;                //current line length
  int max;                //maximum length seen so far
  char line[MAXLINE];     //current input
  char longest[MAXLINE]; //longest line

  max = 0;
  while ((len = getline(line, MAXLINE)) > 0)
    if (len > max) {
      max = len;
      copy(longest, line);
    }
  if (max > 0)            //there was at least one line
    printf("%s", longest);
  return 0;
}
```

Function Example: maxLine, 2/2

```
//getline: read a line into s, return length
int getline(char s[], int lim) {
  int c, i;

  for (i = 0; i < lim - 1 && (c = getchar()) != EOF && c != '\n'; i++)
    s[i] = c;
  if (c == '\n') {
    s[i] = c;
    i++;
  }
  s[i] = '\0';
  return i;
}

// copy: copy 'from' into 'to'; assume to is big enough
void copy(char to[], char from[]) {
  int i;

  i = 0;
  while ((to[i] = from[i]) != '\0')
    i++;
}
```

Notes on maxLine

- The `char s[]`, `char to[]`, and `char from[]` arguments all have unspecified length
- This is okay as the lengths of those arrays are set in `main()`
- Operator precedence
  `i < lim - 1 && (c = getchar()) != EOF && c != '\n'`
- How do we know who takes precedence?
- Consult the precedence table – look it up when in doubt. Page 53. There are also lots on line. We will do demos after covering bit-wise operations.
- An array name is actually an address – passed by value as a pointer
- This is how copy is able to make changes to an array that is passed to it as an argument

Associativity

- Determines order if the operators have the same precedence
- `x = y += z -= 4`
- Right to left
- `x = (y += (z -= 4))`
- If x, y, z are 2, 3, 5 before the assignments, what are their values after the assignments?

C Standards

- A standard or specification is a document that defines a programming language in some way (e.g., listing the syntax and semantics of the language, describing the behavior of a compiler for the language, etc.)
- K&R C: Everything before standardization, 1972-1989
- C89 (ANSI C): ANSI 1989 standard
  ANSI also ratified C99, but as generally used, ANSI C refers to C89 (which is the same as C90), 1989-1990
- C90: ISO's 1990 standard, 1990-1999
- C99: ISO's 1999 revision of the standard, 1999-2011
- C11: ISO's 2011 revision of the standard
  The current definition of the C language
- ANSI C is the best-supported standard, and as such is the best choice for writing portable code
  It is the standard we use in this course
- `gcc -ansi -Wall yourProgram.c -o executable`