Computer Systems & Low-Level Programming

C: headers, control structures, loops, functions

Marija Stanojevic
Spring 2019

Important functionalities

- ++x vs x++
- x += 1 is equivalent to x = x + 1 and to ++x
- getc (stdin) or getchar() - get character from standard input
- putc(c, stdout) or putchar(c) - put character c in standard output
- #define PI 3.14 - defines constants
- #include<stdio.h> - includes libraries
- **Famous libraries:** <stdio.h>, <stdlib.h>, <ctype.h> (character handling), <math.h>, <time.h>, <string.h>, <limits.h>, <float.h>
- ‘\0’ - denotes end of string

What are .h files

Header	Explanation
`<stdarg.h>`	Defines macros for dealing with a list of arguments to a function whose number and types are unknown.
`<stddef.h>`	Contains common type definitions used by C for performing calculations.
`<stdio.h>`	Contains function prototypes for the standard input/output library functions, and information used by them.
`<stdlib.h>`	Contains function prototypes for conversions of numbers to text and text to numbers, memory allocation, random numbers and other utility functions.
`<string.h>`	Contains function prototypes for string-processing functions.
`<time.h>`	Contains function prototypes and types for manipulating the time and date.

`<assert.h>`	Contains information for adding diagnostics that aid program debugging.
`<ctype.h>`	Contains function prototypes for functions that test characters for certain properties, and function prototypes for functions that can be used to convert lowercase letters to uppercase letters and vice versa.
`<errno.h>`	Defines macros that are useful for reporting error conditions.
`<float.h>`	Contains the floating-point size limits of the system.
`<limits.h>`	Contains the integral size limits of the system.
`<locale.h>`	Contains function prototypes and other information that enables a program to be modified for the current locale on which it's running. The notion of locale enables the computer system to handle different conventions for expressing data such as dates, times, currency amounts and large numbers throughout the world.
`<math.h>`	Contains function prototypes for math library functions.
`<setjmp.h>`	Contains function prototypes for functions that allow bypassing of the usual function call and return sequence.
`<signal.h>`	Contains function prototypes and macros to handle various conditions that may arise during program execution.

- **Arrays**: int n[5] = {32, 27, 64, 18, 95};
  - Get i-th element: n[i]; *n+i; - those two are equivalent
- **Matrices**: float x[2][3] = {{1, 2, 3}, {4, 5, 6}};
  - Get element: n[i][j]; *n + i*3 + j; - those two are equivalent
- Make sure you are within the bounds!

- **const** int a = 5; // value 5 can't be changed
- **const** int *const ptr = &x; // const ptr to const value
- **static** int a; - initialized as 0 (not happening with int a). Remains in memory until program is running. Static global variables and functions scope is file.

Control structures

```
#include <stdio.h>
#define TRUE “Very much”
#define SO_SO “So-so”
#define FALSE “Not at all”
int temp; // global variable
int main() {
        if (temp >= 50) {
                printf(“How cold? ”, FALSE);
        } else if (temp >= 40) {
                printf(“How cold? “, SO_SO);
        } else {
                printf(“How cold? ”, TRUE);
        }
```

```
switch(temp) {
        case 100:
                printf(“Insanely hot”);
                break;
        case 70:
                printf(“Great weather!”);
                break; // what if we forget it?
        case 30:
                printf(“Freezing!”);
                break;
        default:
                printf(“Can’t tell you”);
}
```

Conditional expression: e1 ? e2 : e3:

```
(temp < 50) ? printf(“Cold”) :
```

Loops

```
for (unsigned int i = 0; i < 20; i++) { printf(“%d ”, ++i);} // what will be printed?

double grade = 2.5;
while (grade < 2.5) {puts(“Study”); grade += 0.2;}
printf("%10.2f", grade); // what is printed? => 2.5

do {puts(“Study”); grade += 0.2;} while (grade < 2.5);
printf("%-10.2f", grade); // what is printed? => 2.7
```

break: exits from for, while-do, do-while loop immediately
continue: skips the rest of the lines in the for, while-do, do-while loop for current value of runner (i, grade); increases runner and goes back into the loop
goto location: jumps to the location and runs from there

Functions and variable scope

- **Variable scope**: says where the variable can be accessed from. Variable has scope in block where it is defined and all of its children blocks.
- Block is everything that starts with { and ends with }
- Static before variable name can change this behavior (check slide 5)
- **Global variable** can be accessed from all parts of that program
- **Local variable** is defined in some smaller block (function, loop) and can be accessed only inside that block
- **Function declaration**: int max (int n1, int n2); has return type, parameters' list and their types. Required if function is defined after call.
- Function: int max (int n1, int n2) { if (n1 > n2) return n1; else return n2;}
- Function call: max (3, 5); or int a = 5, b = 3; max(a, b);