601.220 Intermediate Programming

Writing functions, command-line arguments

Outline

- Functions we write ourselves
- Exercise 2-3

Same idea as Java methods, Python functions

We've called functions (printf, scanf, isalpha, ...), but haven't defined any of our own

```
int foo(char c, int i) {
    return i;
}
int foo
  • foo is the name
  • int is the return type
(char c, int i)
  • This is the parameter list
```

char c and int i are parameters.

Everything inside { curly braces } is the body

Communication between functions

- For functions to communicate, we use function parameters and/or function return values
 - External (global) variables are another option; though use of these is discouraged except in specific circumstances

Function definitions and function calls

```
float func1 (int x, float y) {
   return x+y;
Calls to above function definition:
int a = 7;
float b = 2.5;
float c = func1(a,b);
float d = func1(a,c);
float e = func1(a+2,3.5);
```

 Different calls to function match different arguments with the formal parameters x and y

A function returning nothing has return type void, and does not need a return statement

```
void say_hello() {
    printf("Hello, World!\n");
}
```

If there are no parameters, parameter list written as () or (void)

Function examples

```
int add(int lhs, int rhs) {
   int sum = lhs + rhs;
   return sum;
}
double convert(double fahrenheit) {
   return 5.0 / 9.0 * (fahrenheit - 32.0);
}
```

Checkpoint Question!

Given the function defnition, which function call is invalid in C?

```
int abs(int a) {
        if (a < 0)
            return -a;
        return a;
    }
A. abs(2 / 5);
B. int c = 0; abs (c);
C. abs('%'):
D. abs (-2.2);
```

F. None of the above

Checkpoint Question!

Which function name(s) below is/are guaranteed to be valid in C?

- A. function_092
- B. fun\$12
- C. 5fun2
- D. func68
- E. A and D

"Experience has shown that the best way to develop and maintain a large program is to construct it from smaller pieces or modules, each of which is more manageable than the original program."

- Deitel & Deitel Chapter 5 intro
- Rule of 30: https://dzone.com/articles/rule-30-%E2%80%93when-method-class-or

Factoring your code into functions — instead of putting everything in main — has major advantages:

- Keeps you concentrating on smaller problems, one at a time
- Makes code more readable
- Helps with testing
 - Can test functions one by one
 - Tests are easy to write; call function with certain inputs, assert something about return value
- Easier to collaborate
 - "I'll write function X, she writes function Y, you write everything else assuming you have X and Y."

Putting a chunk of code in a new function is appropriate when:

- The code has a clear, distinct goal
- The number of variables used in the snippet is not large
- The code has a single result (or not very many)

Parameter passing

- Argument values in C are passed by value
 - function is given argument values in temporary variables (not originals)
 - therefore, function can not directly modify an argument variable back in the calling function
 - [... MORE ON THIS NEXT CLASS...]

Command-line arguments

When you type mkdir cs220, you're running a program called mkdir and passing it a single command-line argument, cs220

C programs can take arguments similarly, but you have to declare your main function in a special way...

Command-line arguments

```
// args_eq_1.c:
#include <stdio.h>
int main(int argc, char* argv[]) {
    printf("argc = # arguments + 1: %d\n", argc);
    for(int i = 0; i < argc; i++) {
        printf("argv[%d] = %s\n", i, argv[i]);
    }
    return 0;
$ gcc args_eg_1.c -std=c99 -pedantic -Wall -Wextra
$ ./a.out rosebud
argc = # arguments + 1: 2
argv[0] = ./a.out
argv[1] = rosebud
```

Command-line arguments

int main(int argc, char* argv[])

- int argc is a parameter that equals the number of command-line arguments. The program name (e.g. ./a.out) counts as the first.
- char* argv[] is an array of strings. The strings are the command-line arguments.
 - Might also see char argv[][] or char **argv; these all mean the same thing

In-Class Exercise

- Work on Exercise 2-3
 - Complete it before next meeting