Functions

2024 Winter APS 105: Computer Fundamentals
Jon Eyolfson

Lecture 10
1.0.0

We Can Define Our Own Functions

The idea is to break your program into re-usable pieces

It also makes your code easier to read

To Define a Function, Use Curly Brackets After the Prototype

Recall: a function prototype tells you:
the types of the inputs,
and the type of the output

If you name an argument, you can use it in the function

Example:

```
int addTwo(int x) {
    return x + 2;
}
```

The Order Matters

The compiler reads your program from top to bottom
  Your code runs starting at `main`

The function definition (or prototype) must come before where you use it

addTwo Must Come Before main

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    printf("Result: %d\n", addTwo(4));
    return EXIT_SUCCESS;
}

int addTwo(int x) {
    return x + 2;
}
```

Will give you an error, or warning, such as:

```
warning: implicit declaration of function 'addTwo' is invalid in C99
[-Wimplicit-function-declaration]
```

The Previous Warning Means C Doesn't Know What `addTwo` Is

An implicit declaration means you did not declare
(provide a prototype for) the function

`-Wimplicit-function-declaration` is the compiler flag for this warning
You can ignore the compiler flag

First Option is to Move `addTwo` Before `main`

```c
#include <stdio.h>
#include <stdlib.h>

int addTwo(int x) {
    return x + 2;
}

int main(void) {
    printf("Result: %d\n", addTwo(4));
    return EXIT_SUCCESS;
}
```

Otherwise, Write a Function Prototype for `addTwo` Before `main`

```c
#include <stdio.h>
#include <stdlib.h>

int addTwo(int x);

int main(void) {
    printf("Result: %d\n", addTwo(4));
    return EXIT_SUCCESS;
}

int addTwo(int x) {
    return x + 2;
}
```

Let’s Write a Program to Print a Mirrored Triangle of Stars

```
    *
   **
  ***
 ****
*****
```

Previous Solution

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    const int N = 5;
    for (int row = 1; row <= N; ++row) {
        for (int col = 1; col <= N; ++col) {
            if ((N - col) >= row) {
                printf(" ");
            }
            else {
                printf("*");
            }
        }
        printf("\n");
    }
    return EXIT_SUCCESS;
}
```

Let’s Write a Program to Print Rows of Stars Using Functions

Input: an integer representing the number of rows to print

Output: rows of stars

Example:

```
Number of rows: 7
*
**
***
****
*****
******
*******
```

The Start of Our Solution

```
#include <stdio.h>
#include <stdlib.h>

void printRow(int row) {
    for (int count = 1; count <= row; ++count) {
        printf("*");
    }
    printf("\n");
}
```

The Remaining Part of Our Solution

```
void printTriangle(int maxRow) {
    for (int row = 1; row <= maxRow; ++row) {
        printRow(row);
    }
}

int main(void) {
    printf("Number of rows: ");
    int n = 0;
    scanf("%d", &n);

    printTriangle(n);

    return EXIT_SUCCESS;
}
```

C Copies Function Arguments

Specifically, in programming languages we refer to this as pass by value

For example, if the variable is an `int`, C will create a copy of the `int` value

We'll See `x: 3` Both Times, Because `addTwo` Gets a Copy of `x`

```c
#include <stdio.h>
#include <stdlib.h>

int addTwo(int x) {
    int result = x + 2;
    x = 42;
    return result;
}

int main(void) {
    int x = 3;
    printf("x: %d\n", x);
    printf("Result: %d\n", addTwo(x));
    printf("x: %d\n", x);
    return EXIT_SUCCESS;
}
```