CAAM420: Week 3 Friday Notes

Matt Delhey

09/13/13

1 Recursion (K&R 3.5)

Triangular numbers from last class:

$$T_n = \sum_{n=1}^{N} n$$

Example: Writing a program that prints the first 10 triangular numbers.

- $T_1 = 1$
- $T_2 = T_1 + 2$
- $T_N = T_{n-1} + N$

Every value of T can be found using only the previous value. In C, recursion is defined by **for**.

```
#include "cstd.h"
int main() {
  int T = 0;
  int n;
  for (n = 1; n < 11; n++) {
    T = T + n;
  }
  printf("T(10) = %d\n", T);

  return 0;
}
```

- In this recursion, T accumulates (and forgets) sums until $n < 11$ is false.
- Note that `n++` is equivalent to $n = n + 1$
- $n = 1$ initiates start of the loop, $n < 11$ initiates the end of the loop

The function `main` is called by the shell, and returns an integer value. The value of the last command to be executed can be determined by `echo $?`. All functions return something (except those with return type `void`). For functions that either run successfully or don't run successfully, we indicate successfully execution by returning 0.

2 Arrays (K&R 5)

What if we want to keep all the values of T? Previously we ended up throwing out all values of T except in the last loop. The solution is to store the values of T in an **array**.

An array of length 10 is declared using `int t[10]`. When this is executed the compiler sets apart a continuous block of memory that holds 10 integers.

The first element of the array has an index of 0, not 1! The reasoning behind this is that index represents the length (or offset) between the beginning of the array and the element. So the first element has 0 offset from the beginning of the array.

```
/* Store values of T in an array */
#include "cstd.h"
int main() {
  int t[10];
  t[0] = 1;
  int n;
  for (n = 2; n < 11; n++) {
    t[n-1] = t[n-2] + n;
  }
  printf("T(10) = %d\n", t[9]);
  return 0;
}
```

- Notice that because we start n at 1, in order to reference the first value of the array (index of 0) we must use $t[n-1]$.

```
/* Store values of T into an array that is more easily indexed */
/* Print results using for */
#include "cstd.h"
int main() {
  int t[10];
  t[0] = 1;
  int n;
  for (n = 1; n < 10; n++) {
    t[n] = t[n-1] + n+1;
  }

  for (n = 0; n < 10; n++) {
    printf("T(%d) = %d\n", n+1, t[n]);
  }
  return 0;
}
```

- C doesn't care about whitespace! Writing `n=1;n<10` is the same as `n = 1; n < 10`.

3 Functions (K&R 4.1)

Just as we can reuse `printf` in our files, we might want to craft our own functions that can be called in the same way.

Functions in C have a signature, starting with a standard format: what the function returns, the function's name, and a list of comma-separated arguments.

```
#include "cstd.h"

/*
  function decleration:
  (ret-val-type) (name)(type [arg1], type2 [arg2], ...);
```

```
*/

/*
  function definition:
  (ret-val-type) (name)(type1 arg1, type2 arg2, ...) {
  ...
  }
*/

int triangle(int m) {
  int t = 0;
  int n;
  for (n = 1; n < m+1; n++) {
    t = t + n;
  }
  return t;
}

int main() {
  int n;
  for (n = 0; n < 10; n++) {
    printf("T(%d) = %d\n", n+1, triangle(n+1));
  }
}
```

- Notice that a function must return a value of the declared type and that all variables must be declared.

A few more notes on functions:

- Functions must be *declared* and *defined*.
- Square brackets indicate that the argument names are optional in the function declaration. Instead, argument names are required in a function definition (note the lack of square brackets). All arguments must be given variable names so that they can be referenced in the body of the function by name.
- Function declaration tells you what a function does, function definition tells you how the function does it.
- Function definitions can go anywhere inside or outside the file, e.g. `printf`.
- But a function must be declared before it can be used.

```
/* Demonstration of some of the properties of functions. */
/* Should return the same output as previous code */
#include "cstd.h"
/* Declare triangle function so we can use it */
int triangle(int);

int main() {
  int n;
  for (n = 0; n < 10; n++) {
    printf("T(%d) = %d\n", n+1, triangle(n+1));
  }
  return 0;
}

/* Define triangle function after it's used */
```

```
int triangle(int m) {
  int t = 0;
  int n;
  for (n = 1; n < m+1; n++) {
    t = t + n;
  }
  return t;
}

/* If we we're to move #include "cstd.h" here, we would get an error */
```