Writing Functions in C

Test 2, Problem 5 b….

Write a function to allocate space for a new instance of your structure, as defined in part a. Write the C code for a function to get space from the heap using malloc for an instance of your structure, and return a pointer to that memory. You do not need to initialize any of the values in the structure.

```
struct vehicle {
        int time;
        enum lne { northbound=0,southbound=1} lane;
        enum wgt { compact=1, midsize=2, … } weight;
};
```

Step 0: Get the Big Picture

- What is your function supposed to do?
- Who is it doing this for?
- How will your function get called?
- How will it's return value be used?

Step 0: Get the Big Picture

- What is your function supposed to do?
  Allocate space for a new instance of the vehicle structure
- Who is it doing this for?
  Program to monitor bridge traffic
- How will your function get called?
  Whenever the monitor returns info about a vehicle
- How will it's return value be used?
  Program will fill in the structure and manage it

Step 1: Choose a Name

- What are you going to call the function?
- Name must follow the rules…
  - Start with a letter
  - Contains letters and numbers and underscores
  - Not a keyword
  - Not another function or variable name
- Choose a name that is easy to use
  - Easy to remember
  - Short to type

Step 1: Choose a Name

- What are you going to call the function? makeVehicle

Step 2: Identify Argument List

- What data will your function need to work?
- What data is your caller willing and able to provide to you?
- What is the data type of each argument?
- What name will you use for each argument?

- (Note… as your write your function, the answers to these questions may change… you may have to add or remove items from your argument list… but you should at least make a first pass at what is in the argument list up front.)

Step 2: Identify Argument List

- What data will your function need to work?
  Nothing… as long as I know the definition of the vehicle structure, I can allocate space for that structure. My caller specifically said that I don't have to fill in any of the fields in the structure (If I did, they would have to be arguments.)
- What data is your caller willing and able to provide to you?
  Nothing.

…. makeVehicle() or … makeVehicle(void)

Step 3: Identify Type of the Return Value

- First, what will this function return?
- Second, what is the data type of this value?

Step 3: Identify Type of the Return Value

- First, what will this function return?
  A pointer to an instance of structure vehicle
- Second, what is the data type of this value?
  struct vehicle *

struct vehicle * makeVehicle()

After step 3…

- You have a complete function prototype

  struct vehicle * makeVehicle()

- Use the prototype to declare the function:

```
struct vehicle * makeVehicle();
```

- Use the prototype to define the function:

```
struct vehicle * makeVehicle() {
    /* This is where the actual C implementation goes */
    …
}
```

Step 4: Implement the Function

- What C instructions and local variables are needed to perform the function?
- Is there more than one way to do this?
- If so, which is the easiest/ most straightforward way?

Step 4: Implement the Function

- What C instructions and local variables are needed to perform the function?
  Call malloc to get heap memory, return the result, cast to the right type.
- Is there more than one way to do this?
  In this case, other than minor variations, probably not.

  If so, which is the easiest/ most straightforward way?

```
struct vehicle * makeVehicle() {
    return (struct vehicle *) malloc(sizeof(struct vehicle)); …
}
```

Step 5: Test Your Function

- Need the higher level code to call your function.
- Need to think about what could go wrong in your function.
- Need to come up with ways to try out your function and evaluate whether it is working correctly.

New Problem

Write a function to count the number of times a specific letter (passed as the first argument) occurs in a null delimited text string (passed as the second argument). Return the number of occurances of the letter in the string.

Step 0: Get the Big Picture

- What is your function supposed to do?
- Who is it doing this for?
- How will your function get called?
- How will it's return value be used?

Step 0: Get the Big Picture

- What is your function supposed to do?
  Count the number of times a letter occurs in a string
- Who is it doing this for?
  ?
- How will your function get called?
  x=func('e',"This is a test");
- How will it's return value be used?
  ?

Step 1: Choose a Name

- What are you going to call the function?
- Name must follow the rules…
  - Start with a letter
  - Contains letters and numbers and underscores
  - Not a keyword
  - Not another function or variable name
- Choose a name that is easy to use
  - Easy to remember
  - Short to type

Step 1: Choose a Name

- What are you going to call the function? howMany
- Name must follow the rules…
  - Start with a letter
  - Contains letters and numbers and underscores
  - Not a keyword
  - Not another function or variable name
- Choose a name that is easy to use
  - Easy to remember
  - Short to type

Step 2: Identify Argument List

- What data will your function need to work?
- What data is your caller willing and able to provide to you?
- What is the data type of each argument?
- What name will you use for each argument?

- (Note… as your write your function, the answers to these questions may change… you may have to add or remove items from your argument list… but you should at least make a first pass at what is in the argument list up front.)

Step 2: Identify Argument List

- What data will your function need to work?
  1) a letter, and 2) a string
- What data is your caller willing and able to provide to you?
  Same as above
- What is the data type of each argument?
  1) char and 2) char *
- What name will you use for each argument?
  1) letter and 2) string

  …. howMany(char letter, char *string)

Step 3: Identify Type of the Return Value

- First, what will this function return?
- Second, what is the data type of this value?

Step 3: Identify Type of the Return Value

- First, what will this function return?
  The number of occurances of the letter in the string
- Second, what is the data type of this value?
  int

Step 4: Implement the Function

- What C instructions and local variables are needed to perform the function?
- Is there more than one way to do this?
- If so, which is the easiest/ most straightforward way?

Step 4: Implement the Function

- What C instructions and local variables are needed to perform the function?
  Need to walk through the string, checking each character to see if it matches letter.
  Need a variable to index into the string.
  Need a variable to count the number of hits.
- Is there more than one way to do this?
  For sure… e.g. pointers and vectors
- If so, which is the easiest/ most straightforward way?
  Let's use vectors

Example of Implementation

```
int howMany(char letter, char *string) {
        int answer=0; int j;
        for(j=0;j<strlen(string);j++) {
                if (letter==string[j]) answer++;
        }
        return answer;
}
```

Resources

- Programming in C, Chapter 7
- C Functions Tutorial: http://www.tutorialspoint.com/cprogramming/c_functions.htm
- C functions You Tube Tutorial: https://www.youtube.com/watch?v=iOS5sPivuJA