CSC180 tutorial notes – week of Sept. 15

This tutorial covers variables and functions in C.

Variables

Initial explanation

Explain what variables are, and that the idea is that they can be changed during runtime; walk through the following, keeping track of the variables’ values on the board. The idea is to emphasize that variables can be used as scratch space. (Note: when you go through the assignment lines, remind the students that it’s always the value of the right-hand side variable that’s assigned to the left-hand side variable)

```c
double temp, answer;
double scrt_num = 15; /*secret number*/
temp = scrt_num + 8;
temp = temp * 2;
temp = temp / 4;
answer = temp - scrt_num / 2; /*always 4, regardless of the initial value of secret_number*/
```

Point out the declaration (type then name), initializations, and assignments in the code above.

Review of variable types

Go through the different types, including strings. (It’s probably not necessary to write all of this on the board)

<table>
<thead>
<tr>
<th>Type</th>
<th>Comments</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>Integer (whole number). The range is different for different architectures. \ -32,768 to +32,767 if the implementation uses 16 bits to represent int, \ -2,147,483,648 to +2,147,483,647 if it uses 32 bits</td>
<td>int number = -5;</td>
</tr>
<tr>
<td>unsigned int</td>
<td>Non-negative (i.e., positive or 0) whole number. The range is 0 to 4,294,967,295 for 32-bit machines.</td>
<td>unsigned int positive_number = 4000000;</td>
</tr>
<tr>
<td>char</td>
<td>ASCII characters/number from -128 to 127. The literals are written in single</td>
<td>char c = ‘a’; char question_mark = ‘?’;</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td>Examples</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>double</td>
<td>floating-point number (i.e., not necessarily a whole number)</td>
<td>double a = 1.2; double b = 1.2e6;</td>
</tr>
<tr>
<td>char [200]</td>
<td>A string (i.e., sequence (“array”) of chars) of length no more than 199</td>
<td>char str[200] = “hello, world”;</td>
</tr>
<tr>
<td></td>
<td>(Mention that a string is a character array that ends with ‘\0’. No string type in C.)</td>
<td></td>
</tr>
</tbody>
</table>

Practice question for variable declaration: declare variables that would store the following:

(a) A student’s age (answer: `int age;` or `unsigned int age;`)
(b) A student’s name, assuming it is no longer than 199 characters (answer: `char name[200];`)
(c) A student’s UofT student number (note: student numbers consist of 9 digits) (answer: `char student_number[10];` note that the answer `int sin;` isn’t correct, since if `int` is 16-bit the range of `int` would be too small.)
(d) The temperature outside, in degrees Celsius (answer: `double temp;`)
(e) The (non-special) last key that was pressed on the keyboard (answer: `char key;`)

**Functions**

Review the concept of functions (probably w/out mentioning side-effects). Examples of usage:

```
double square_root_of_two = sqrt(2);
```
(Mention that calling `sqrt` requires us to link the C math library explicitly. Use `gcc a.c –lm` when using the math library)

**Examples of functions**

```c
int square(int x)
{
    return x*x;
}
```
point out the meaning of the header, and where the inputs and outputs are. Ask how to use this to get the square of 2.

    double divide(double x, int y)
    {
        return x/y;
    }

Point out that the order matter when calling the function (i.e., divide(2,3) and divide(3,2) are different). Point out why it’s important that the return type be double.

Discuss how to get the roots of a quadratic equation given the coefficients. Given what they know, ignoring the possibility that the discriminant is < 0 (they didn’t do conditionals yet), we can declare

    double root1(double a, double b, double c);
    double root2(double a, double b, double c);