By completing the first two milestones, I learned how to use a C string to hold textual data so as to store the contact’s information such as name and postal code. I also learned to use the conversion specifiers for character string input. For example, I used “ %30[^\n]” to store the first name of the contact -- the empty space before the specifier discards any number of white space until the first non-whitespace character; the %30 indicates 30 characters to be stored; and the [^\n] reads all characters up to the newline character.

While working on this program, I thought that it would be more efficient to store each numeric and textual data field using two different functions. For instance, I would make a function called “numInput” to scan all the numeric information of the contact such as the street and apartment numbers. In this way, I would be able to simplify the program by calling that specific function when needed.

Similar to a primitive character array, a C string is also an array of characters but with a null terminator following the last meaningful character. This is why we must consider adding one more character to the string when declaring its size. Fortunately, the specifier “%s” in printf results in the same output for both a primitive character array and a C string. This is because this conversion specifier excludes the null terminator and displays all characters from them. For example, char arr[10] = { 'h', 'e', 'l', 'l','o' }; and char arr[10] = “hello” would display the same result, hello.