French Azerbaijani University
Arash Habibi
Second year
2017-2018

Advanced C Programming

Exercice session 3 : new data structures and numeration

Before you start this session, make sure you have read chapter 3 and 4.

1 Types and sizes and overflow

1.1 Types and sizes

1. Consider the following code :

```
char c, d, e;
printf("c : %p\n",&c);
printf("d : %p\n",&d);
printf("e : %p\n",&e);
```

The output of the first two instructions is :

```
c : 0x103c24017
d : 0x103c24018
```

What do you think would be the output of the third instruction ?

2. Consider the following code :

```
short c, d, e;
printf("c : %p\n",&c);
printf("d : %p\n",&d);
printf("e : %p\n",&e);
```

The output of the two first instructions is :

```
c : 0x101e5e012
d : 0x101e5e014
```

What do you think would be the output of the third instruction ?

1.2 Overflow

What would the the output of the following code ? Explain.

```
unsigned char a=255, b;
b=a+1;
if(a < b)  printf("This is OK.\n");
else       printf("This is strange.\n");
```

2 Vectors

2.1 Array fields

This subsection does not contain any question. It is simply an observation. You have already seen structures with different types of fields : integers, chars, floats. But you can also use arrays. For example :

```
struct person
{   int age;
    float size;
    char name[30];
};
```

In the person structure, the `name` is an array of 30 characters.

2.2 Array vectors

We seek to make a series of functions for vectors. We can represent vectors either by an array or by a structure. In this particular case, we need to be able to process 2D, 3D or 4D vectors. This is why we choose to represent vectors by an array of floats. And depending on `nb_dims` (the number of dimensions) : 2, 3 or 4, we process the correct number of components.

1. Write in C, a function with prototype `void V_allZero(float vector[], int nb_dims);` which sets all of the components of `vector` to 0.
2. Write in C, a function with prototype `void V_print(float vector[], int nb_dims);` which prints the value of the components of `vector`.
3. Write in C, a function with prototype `int V_belongsTo(float x, float vector[], int nb_dims);` which returns 1 if one of the components of `vector` is `x` and returns 0 otherwise.
4. Consider the following code :

   ```
   float v[2];
   //...
   V_allZero(v,3);
   V_print(v,3);
   ```

   This code outputs : `0.0000 0.0000` and then `abort trap: 6` (which is some sort of runtime error). Can you guess why ?

2.3 New vector structure

The problem is that, in C language, arrays are not necessarily associated with a size. They are only the address of the first element of the array (we will see that in depth in the following chapters). So it is the programmer's responsability to remember the number of elements in each array. Each function operating on arrays must get the array itself and also the number of elements, as in the functions above.

It would be great if we didn't have to do that. For example if each array itself could be characterized by a size. This size would be a property of the array.

4. Propose a new data structure in which the vector and the number of dimensions are associated. For simplicity, you may always use arrays with 4 elements and, depending on the number of dimensions, we will use only the 2, 3 first elements or all 4.
5. Write a function `V_new(int nb_dims);` which returns such a data structure with `nb_dims` dimensions ;
6. Write again the previous functions, but with the new data structure :

   ```
   void V_allZero2(struct vector v);
   void V_print2(struct vector v);
   int V_belongsTo2(struct vector v, int x);
   ```

3 Convert base b to decimal

Let us remind that :

Integer N is represented in base b by n digits $d_{n-1}, d_{n-2}, ..., d_2, d_1, d_0$ with $d_i \in \{0, ..., b-1\}$ if and only if :

$$N = d_{n-1}.b^{n-1} + d_{n-2}.b^{n-2} + ... + d_2.b^2 + d_1.b + d_0 \quad (1)$$

1. How would you write, in decimal, following positive binary numbers
   - 100
   - 101
   - 11110000
   - 11000010
2. How many numbers can you make with 4 binary digits ?
3. How would you write, in decimal, the ternay (base 3) number 110 ?
4. How many numbers can you make with 4 ternary digits ?
5. How would you write, in decimal, the octal (base 8) number 710 ?
6. How many numbers can you make with 2 octal digits ?
7. Trick question : how would you write, in decimal, the octal number 82 ?
8. How would you write, in decimal, the hexadecimal number 11 ?
9. How would you write, in decimal, the hexadecimal number 1f ?
10. How many numbers can you make with 2 hexadecimal digits ?

4 Convert decimal to base b

Now, for a given decimal number N, we wish to express it in base b and to find the digits. In other words, we want to find the d_i digits in equation (1). There are several possible algorithms.

4.1 Intuitive algorithm

At first, we do not want to implement anything on a machine. We only want to manually convert a decimal number and find its digits in base b.

1. At first, make sure you remember what the principle of the euclidian division and the modulo operator ;
2. Consider equation (1). What is the value of $N\%b$? What about N/b ? What about N/b^2 ? and N/b^{n-1} ?
3. Find an intuitive way to find the digits of a number N in base b. Test if your algorithm works for $b = 10$

4.2 Application of the previous algorithm

1. How would you write, in base 2, the decimal number 17 ?
2. How would you write, in base 3, the decimal number 17 ?
3. How would you write, in base 2, the decimal number 59 ?
4. How would you write, in base 3, the decimal number 59 ?