Worksheet 9: Miscellaneous C

Updated: 29th July, 2019

Here we reinforce concepts discussed in lecture 9 — timing, random number generation, constants, enumerations, unions, bit manipulation and bit fields.

Pre-lab Exercises

1. Declarations

Explain the meaning of each of the following:

(a)
```
const float *const ptr;
```

(b)
```
float* const* ptr;
```

(c)
```
const int volatile * volatile const * const volatile * ptr;
```

(d)
```
void (*const func)();
```

(e)
```
enum GameActions {ROCK, PAPER = 5, SCISSORS};
```

(f)
```
enum {ASC, DESC} order(const char *str);
```

(g)
```
typedef union {
    enum GameActions ga;
    double *ptr;
} Things;
```

(h)
```
typedef struct Racer {
    const char *model;
    const char *identity;
    enum {ELECTRIC, PETROL} type;
    union {
        struct {
            double maxWatts;
            double ampDraw;
            double mah;
        } electric;
        struct {
            double fuelConsumption;
            double fuelLoad;
        } petrol;
    } stats;
    double (*calcFlightDuration)(const Racer*);
} Racer;
```

(i)
```
typedef struct {
    unsigned int id : 9;
    unsigned int age : 7;
    unsigned int married : 1;
    unsigned int kids : 4;
} Info;
```

2. Bit Manipulation

Determine the result of each of the following statements and expressions. Assume that `var` and `n` are both non-negative integers.

Note: Bits are numbered from right to left. The rightmost bit (in an integer, or anything else) is bit 0, the next-rightmost is bit 1, then bit 2, etc.

(a) `1 << n`

(b) `~(1 << n)`

(c) `bit = var & 1;`

(d) `var = var >> 1 << 1;`

(e) `var = var & ~1;`

(f) `var = var | 1;`

(g) `bit = (var >> n) & 1;`

(h) `var = var | (1 << n);`

(i) `var = var & ~(1 << n);`

(j) `var = var ^ (1 << n);`

3. Random Numbers

(a) How would you create a unit test for a function that relies on random number generation?

(b) The `rand()` function will return any integer (from 0 and `RAND_MAX`) with equal probability. Using simple arithmetic, how would you engineer things so that some outcomes are more likely than others?

Practical Exercises

1. Random Numbers

Write a function to return a random array element. Your function should accept two parameters — an array of `void` pointers and the array length. It should return a `void` pointer.

Your function should pick an element from the array at random and return it.

Your function does not need to seed the random number generator itself. You should do this in `main()` when testing the function.

2. Timing

Using `time()`, write a function to wait (doing nothing) for a given number of seconds.

Your function should accept an `int` — the number of second to wait. It should return nothing.

Note: This is effectively what the UNIX `sleep()` function does, but you should implement this yourself (purely for practice).

3. Bit Manipulation

Using bit manipulation, write a function that converts a positive number to its closest even number (either the number itself or the previous number)

Your function should accept an `int` — the number to convert. It is to return the closest even number. Whether that be the imported number of the previous number

For example. If the function imports a 7 it should return 6, where as if it imports a 4 then it will return 4.

Note: You shouldn't be using any if statements, division, modulus or anything similar to solve this

Now that youve done it for even numbers, how would you do it for an odd number. Import 5 return 5, import 6 return 7.

End of Worksheet