C Programming

Tutorial 2

1. What are the types used by main()?

int

2. switch statement vs if/else chain. Which one do you use and why?

switch is cleaner, and perhaps more efficient. Use switch when you can. However, case labels are limited to single, constant, integral expressions. If you want to cover arbitrary ranges or non-constant expressions, you'll have to use an if/else chain.

3. What does the following declaration mean?

```
int* p1, p2;
```

```
int *p1, *p2;
```

* is part of the declarator. The whitespace between * and int is not necessary, but can avoid the confusion created by int* - in the first statement p2 is an int not a pointer.

4. We have the following code:

```
int a = 1, b = 2, c = 11, d = 22;
f(...); /* pass appropriate actual parameters */
```

Implement function f which will change a's value to c's value (11) and b's value to d's value (22).

```
void f(int *ptra, int *ptrb, int ac, int ad);
```

```
f(&a, &b, c, d);
```

5. What's the difference between const char *p, char const *p, and char * const p?

The first two are interchangeable; they declare a pointer to a constant char – you cannot change the char. char * const p declares a constant pointer to char – you cannot change the pointer.

6. int f1(const int *p1) vs int f2(const int p2). Do you really need const here?

The first function declares a parameter which is a pointer to const int. This is very common, because you do not want to accidently change the value in the int.

The second function declares a parameter which is a const int p2. p2 is a local variable, it is not necessary or unimportant to make it a const.

7. int a[10] = {0, 1, 2, …, 8, 9}; What is the output of sizeof(a)? Is a treated as a pointer?

On a 32-bit machine, the size is 40bytes. With sizeof operator, a is NOT treated as a pointer.

8. NULL pointer vs. uninitialised pointer. What is the difference between the two?

A null pointer points nowhere. An uninitialized pointer points somewhere you do not know. It could be unused memory or used by important data.

9. See the following code. What is the value of m?

```
int *p, m, n = 10;

p = &n;

m = *p++;
```

The postfix ++ and -- operators have higher precedence than the prefix unary operators. Therefore, *p++ is equivalent to *(p++). To increment the value pointed to by p, use (*p)++.

10. See the following code. Function f is supposed to initialise pointer p. Can you implement function f to initialise p with a NULL pointer?

```
int *p;

f(p);


...
```

```
int *p;
f(&p);
...

void f(int **p)
{
  *p = NULL;
}
```