Tutorial 4
Functions

Jiawei Li (Michael)
Office hours: Monday 1:00 – 3:00pm
Office: PMB426
Email: jiawei.li@Nottingham.edu.cn

Format of C code

2022 & 2023 Coursework:

- *Source code formatting (10%):* Your program should be correctly formatted and easy to understand by a competent C programmer. This includes, but is not limited to, indentation, bracketing, variable/function naming, and use of comments. See the textbook for examples of correctly formatting programs.

Testing C code

To test whether a program is correct, you need to verify that it meets the requirements and handles various situations as expected. According to the requirements and specifications, create a test plan and run the test cases. For each test case, provide the input, observe the output, and compare it with the expected results.

- Normal Test Cases:
- Edge Test Cases:
- Invalid Test Cases:

Functions

- Return value.
- Arguments.
- Function body.
- Declaration.

```
int func(int arg1, char arg2)
{
	int X;
	…

	return X;
}
```

Declare and call a function

- Declaration

```
int func(int arg1, int arg2);
```

- Function body

```
int func(int arg1, int arg2)
{
        int X = arg1 + arg2;
        return X;
}
```

- Function call

```
int main()
{
        int a = 1;
        int b = 2';
        int c = func(a, b);
        func(10, 5);
        func(func(a, b), 'L');
}
```

Pass by values

- Copies of arguments are passed to the function, so data at the caller side will be unchanged.

```
int func(int arg1, int arg2)
{
	arg1 = arg1 + arg2;
	return arg1;
}
```

```
int main()
{
	int a = 1;
	int b = 2';
	int c = func(a, b);
	printf(%d, %d, %d\n", a, b, c);
}
```

Pass by references

- When we want to change the values at the caller side (outside a function), we pass pointer(s) to the function.

```
void func(int* arg1, int* arg2)
{
	int value;
	value = *arg1;
	*arg1 = *arg2;
	*arg2 = value;
}
```

```
int main()
{
	int a = 1;
	int b = 2';
	func(&a, &b);
	printf(%d, %d\n", a, b);
}
```

Main function

- Command line arguments

```
#include<stdio.h>
int main(int argc, char* argv[ ])
{
	printf("argc=%d\n", argc);
	for(int i=0; i<argc; i++)
		printf("%s\n", argv[i]);
	return 0;
}
```

```
[z2017155@CSLinux ~]$ ./helloworld a b c
argc=4
./helloworld
 a
b
c
```

Frequent mistakes

- Pass an array to a function without giving the size

```
int func(int arr[ ]);
```

```
int func(int arr[ ])
{
        int size = sizeof(arr);
        printf("%d", size);
}
```

```
int main(void)
{
        int a[5]={1,2,3,4,5}
        func(a);
}
```

```
int func(int arr[ ], int size);
```

```
int func(int arr[ ], int size)
{
        …

}
```

```
int main(void)
{
        int a[5]={1,2,3,4,5}
        func(a, 5);
}
```

Frequent mistakes

- Passing by reference (copy of addresses passed)

```
void swap(int* arg1, int* arg2)
{
	int* value = NULL;
	value = arg1;
	arg1 = arg2;
	arg2 = value;
}
```

```
int main()
{
	int a = 1;
	int b = 2;
	swap(&a, &b);
	printf(%d, %d\n”, a, b);
}
```

```
void swap(int* arg1, int* arg2)
{
	int value;
	value = *arg1;
	*arg1 = *arg2;
	*arg2 = value;
}
```

```
int main()
{
	int a = 1;
	int b = 2;
	swap(&a, &b);
	printf(%d, %d\n”, a, b);
}
```

Exercise 1

Passing pointers to a function.

```
void swap(int* arg1, int* arg2)
{
	int* value = NULL;
	value = arg1;
	arg1 = arg2;
	arg2 = value;
}

int main()
{
	int a = 1, b = 2;
	printf(“%p, %p\n”, &a, &b);
	swap(&a, &b);
	printf(“%p, %p\n”, &a, &b);
}
```

Exercise 2

- Write two functions.
- The first function sorts an integer array in ascending order.

Example: input array 3, 5, 2, 8, 7

result array 2, 3, 5, 7, 8

- The second function prints all unique elements in an array.