Creative Software Programming Practice (week-4-1)

Every assignment will be announced on **Thursday** and should be sumitted by next **Tuesday**.

We have practice classes on Wednesdays and Thursdays.
The contents of the practice class are different from the assignments and aim to be completed on the same day.

Assignments will be publish on Thursday and must be submitted by the next Tuesday.

In this week **Handed out will be Sep 24, 2020, Due Sep 29, 2020**

Topics

1. Pointer and Const Review
2. C++ operators

1. Pointer and Const Review

Constants are expressions with a fixed value such as `const double PI = 3.14;`.
Literals are the most obvious kind of constants. They are used to express particular values within the source code of a program. We have already used some in previous chapters to give specific values to variables or to express messages we wanted our programs to print out, for example, when we wrote `a = 3;` In this case *3* is literal constant.

You cannot change the value of a variable through the pointer.

```
const int* ptr = &num;
*ptr = 30; // this may raise the compile error!
```

However, it does not make the variable itself a constant

```
num = 30;
```

You can make const pointer which cannot change value of pointer.

```
int num = 30;
const int* ptr1 = &num;
int* const ptr2 = &num;

*ptr2 = 40; // you can change the value of variable through the pointer
*ptr1 = 40; // Raise error! you can not change the value of variable through the
const pointer
ptr2 = &other_variable; // Raise error! raise compile error!
```

```
#include <iostream>

int main() {
    int num = 30;
    int *ptr = &num;
```

```
    std::cout << "The value of num: " << num << std::endl;
    std::cout << "The address of num: " << &num << std::endl;
    std::cout << "The value of ptr: " << ptr << std::endl;
    std::cout << "The address of ptr: " << &ptr << std::endl;
    std::cout << "The value of (pointer ptr): " << *ptr << std::endl;

    const int* const_ptr = &num;
    int* const ptr_const = &num;

    std::cout << "The pointer value of const_ptr: " << *const_ptr << std::endl;
    std::cout << "The value of const_ptr: " << const_ptr << std::endl;

    std::cout << "The pointer value of ptr_const: " << *ptr_const << std::endl;
    std::cout << "The value of ptr_const: " << ptr_const << std::endl;

    // Below lines raise error
    *const_ptr = 15;
    ptr_const = &const_ptr;

    return 0;
}
```

2. C++ Operators

Once introduced to variables and constants, we can begin to operate with them by using operators. What follows is a complete list of operators. At this point, it is likely not necessary to know all of them, but they are all listed here to also serve as reference.

There are mayn operators in C++.

- Increment(decrement): ++a, a++, --a, a--
- Arithmetic: +, -, *, /, %
- Relational: ==, !=, <, <=, >, >=
- Bitwise: a & b, a | b, a ^ b, ~a, a >> b, a << b
- Logical: a && b, a || b, !a
- ...

X	Y	X&Y	X|Y	X^Y	~(X)
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

```
#include <iostream>

int main() {
    int a = 3;
    std::cout << "The value of a: " << a << std::endl;
    std::cout << "The value of a: " << a++ << std::endl;
```

```
    std::cout << "The value of a: " << a << std::endl;
    std::cout << "The value of a: " << ++a << std::endl;

    bool cond = (a == 3);
    std::cout << "a == 3 is " << (cond ? "True" : "False") << std::endl;

    int x = 3; // 11
    int y = 2; // 10

    std::cout << "11 & 10 is " << (x & y) << std::endl;
    std::cout << "11 | 10 is " << (x | y) << std::endl;
    std::cout << "11 ^ 10 is " << (x ^ y) << std::endl;

    bool t = true;
    bool f = false;

    if (t && f) {
        std::cout << "t && f is true" << std::endl;
    }
    if (t || f) {
        std::cout << "t || f is true" << std::endl;
    }

    return 0;
}
```

Pratice

Make swap function which swap the input pointer value

So, The code below should run.

```
#include <iostream>
#include <cassert>

template <typename T>
void swap(T* first, T* second) {
    // TODO
    // fill the code here
}

int main() {
    int a = 3, b = 2;

    swap(&a, &b);

    // after swap a and b, a must be 2 and b must be 3.
    assert(a == 2);
    assert(b == 3);

    return 0;
}
```

```
```