Creative Software Programming Practice (week-3-1)

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 17, 2020, Due Sep 22, 2020**

Topics

1. C string and std::string
2. Namespace
3. Function overloading

1. Cstring and std::string

Strings are defined as an array of characters. The difference between a character array and a string is the string is terminated with a special character '\0'.

```
char str[] = "String";

str[0] = 'S';
str[1] = 't';
str[2] = 'r';
str[3] = 'i';
str[4] = 'n';
str[5] = 'g';
str[6] = '\0'; // this is end of string
```

Also you can use below syntax for initializing a string.

```
char str[] = "String";
char str[32] = "String";
char str[] = {'S', 't', 'r', 'i', 'n', 'g', '\0' };
const char* str = "String";
```

Since'\0' is a special character that indicates the end of a string, c treats it as the end of a character when it encounters'\0'. Therefore, if you put'\0' in the middle of a string, it is recognized as a string only until then.

```
char str[] = "String";
str[3] = '\0';
std::cout << str << std::endl; // will print only "Str"
std::cout << &(str[4]) << std::endl; // will print remain parts "ng"
std::cout << (str+4) << std::endl; // it's same as above
```

For a more seamless use of strings, C++ has **std::string**.
It has many features such as size, length, clear, ...

```
#include <iostream>

int main() {
    char str1[] = "String";
    char str2[32] = "String";
    char str3[] = {'S', 't', 'r', 'i', 'n', 'g', '\0'};

    std::cout << str1 << std::endl;
    std::cout << str2 << std::endl;
    std::cout << str3 << std::endl;

    char str4[] = "String";
    str4[3] = '\0';
    std::cout << str4 << std::endl; // will print only "Str"
    std::cout << &(str4[4]) << std::endl; // will print remain parts "ng"
    std::cout << (str4+4) << std::endl; // it's same as above

    std::string str5 = "String";
    std::cout << str5 << std::endl;
    std::cout << "length of str5: " << str5.size() << std::endl; // std::string
has many features.
    std::cout << str5.c_str() << std::endl; // return char pointer for Backward
Compatibility
}
```

2. Namespace

Namespaces provide a method for preventing name conflicts in large projects.

Symbols declared inside a namespace block are placed in a named scope that prevents them from being mistaken for identically-named symbols in other scopes.

Multiple namespace blocks with the same name are allowed. All declarations within those blocks are declared in the named scope.

```
// file: namespace.cc
#include <iostream>

namespace a {
    void function() {
        std::cout << "function in namespace a" << std::endl;
    }
}

namespace b {
    namespace c {
        void function() {
            std::cout << "function in namespace c in namespace b" << std::endl;
        }
    }
    void function() {
        std::cout << "function in namespace b" << std::endl;
    }
}
```

```
using namespace a;
// using namespace b; raise overloaded error

int main() {
    a::function();
    b::function();
    b::c::function();
    function();

    return 0;
}
```

3. Function overloading

In C++, two or more different functions can have the same name if their parameters are different; either because they have a different number of parameters, or because any of their parameters are of a different type. For example:

```
void function (int a) {
}
void function (int a, int b) {
}
void function (double a) {
}
```

In this example, there are three functions called `function`, but one of them has one parameter of type int, while the other has them of type double or two of type int. The compiler knows which one to call in each case by examining the types passed as arguments when the function is called. If it is called with two int arguments, it calls to the function that has two int parameters.

```
// file: overloading.cc
#include <iostream>

void function(int a) {
    std::cout << "function with int a called!" << std::endl;
}
void function(int a, int b) {
    std::cout << "function with int a, int b called!" << std::endl;
}
void function(double a) {
    std::cout << "function with double a called!" << std::endl;
}

int main() {
    function(3);
    function(4, 5);
    function(2.0);
    return 0;
}
```

```
```