04. Functions, Arrays, Strings and Parameter Passing - 02

Oritented Object Programming C++: Chapter 03

- Time: 120 Minutes
- Instructor: Thang, Nguyen Chien
- Email: chienthangplus@gmail.com
- Phone: 0349 688 571

Previous lessons

1. Introduction C++
   - C++ vs C
   - Compilers, IDEs
2. C++ Language Basics
   - Types: `int`, `float`, `double`
   - Control Structures: `if`, `for`, `while`, `switch-case`…
3. Functions, Arrays, Strings and Parameter Passing-1
   - `void func(int);`
   - `void func(int, int);`
   - `void func(int, int = 2);`
   - `int arr[10], brr[2][3];`

Contents

V. C-style string

- A string is a **sequences of characters**, called C-Style string in C++
- Example:
  - `char name[10];`

0	1	2	3	4	5	6	7	8	9

- The end of strings represented by the **null** character
  - Whose literal value can be written as **'\0'**

T	H	U	Y	\0					
0	1	2	3	4	5	6	7	8	9

Initialization C-style strings

- Initialization C-style strings:

```
char name1[10] = { 'T', 'H', 'U', 'Y', '\0' };
```

T	H	U	Y	\0					
0	1	2	3	4	5	6	7	8	9

```
char name2[] = { 'T', 'H', 'U', 'Y', '\0' };
char name3[] = "THUY";
```

The string literals

T	H	U	Y	\0
0	1	2	3	4

```
char name4[] = "Thu y";
```

T	h	u		y	\0
0	1	2	3	4	5

Initialization C-style strings (cont.)

- Example:

```
char name1[10] = { 'T', 'H', 'U', 'Y', '\0' };
char name2[] = { 'T', 'H', 'U', 'Y', '\0' };
char name3[] = "THUY";
char name4[] = "Thu y";
char name5[10];
char name6[10] = name1;                        // Error
char name7[];                                  // Error
name5 = { 'T', 'H', 'U', 'Y', '\0' };          // Error
name5 = "THUY";                                // Error
name5 = name2;                                 // Error
name5 = name3 + name4;                         // Error
```

C-style string Input/ouput

- Use cout to print a c-style string
  - `char name[10] = {'T','H','U','Y','\0'};`
  - `cout << name;`
- Get a word:
  - `char name[10];`
  - `cin >> name;`
- Get a line:
  - `char address[100]`
  - `cin.getline(address, 100);`
  - `cin.getline(address, 100, '\n');`

C-style string Input/ouput (cont.)

❖ Example 01:

```
#include <iostream>
using namespace std;
int main() {
    char name[10];
    char address[100];
    cout << "Address: ";
    cin.getline(address, 100);
    cout << "Name: ";
    cin >> name;
    cout << "- Address: " << address << endl;
    cout << "- Name: " << name << endl;
    return 0;
}
```

Output

```
Address: 254 Nguyen Van Linh
Name: Thuy
- Address: 254 Nguyen Van Linh
- Name: Thuy
```

C-style string Input/ouput (cont.)

Example 01:

```cpp
#include <iostream>
using namespace std;
int main() {
    char name[10];
    char address[100];
    cout << "Address: ";
    cin.getline(address, 100);
    cout << "Name: ";
    cin >> name;
    cout << "- Address: " << address << endl;
    cout << "- Name: " << name << endl;
    return 0;
}
```

Output

```
Address:
```

C-style string Input/ouput (cont.)

- Example 01:

```
#include <iostream>
using namespace std;
int main() {
    char name[10];
    char address[100];
    cout << "Address: ";
    cin.getline(address, 100);
    cout << "Name: ";
    cin >> name;
    cout << "- Address: " << address << endl;
    cout << "- Name: " << name << endl;
    return 0;
}
```

Output

```
Address: 254 Nguyen Van Linh
```

C-style string Input/ouput (cont.)

- Example 01:

```
#include <iostream>
using namespace std;
int main() {
    char name[10];
    char address[100];
    cout << "Address: ";
    cin.getline(address, 100);
    cout << "Name: ";
    cin >> name;
    cout << "- Address: " << address << endl;
    cout << "- Name: " << name << endl;
    return 0;
}
```

Output

```
Address: 254 Nguyen Van Linh
Name:
```

C-style string Input/ouput (cont.)

- Example 01:

```
#include <iostream>
using namespace std;
int main() {
    char name[10];
    char address[100];
    cout << "Address: ";
    cin.getline(address, 100);
    cout << "Name: ";
    cin >> name;
    cout << "- Address: " << address << endl;
    cout << "- Name: " << name << endl;
    return 0;
}
```

Output

```
Address: 254 Nguyen Van Linh
Name: Thuy
```

C-style string Input/ouput (cont.)

- Example 01:

```
#include <iostream>
using namespace std;
int main() {
    char name[10];
    char address[100];
    cout << "Address: ";
    cin.getline(address, 100);
    cout << "Name: ";
    cin >> name;
    cout << "- Address: " << address << endl;
    cout << "- Name: " << name << endl;
    return 0;
}
```

Output

```
Address: 254 Nguyen Van Linh
Name: Thuy
- Address: 254 Nguyen Van Linh
- Name: Thuy
```

C-style string Input/ouput (cont.)

- Example 02:

```
#include <iostream>
using namespace std;
int main() {
    char name[10];
    char address[100];
    cout << "Name: ";
    cin >> name;
    cout << "Address: ";
    cin.getline(address, 100);
    cout << "- Name: " << name << endl;
    cout << "- Address: " << address << endl;
    return 0;
}
```

Output

```
Name: Thuy
Address: - Name: Thuy
- Address:
```

C-style string Input/ouput (cont.)

Example 03: Solve the problem on Example 02

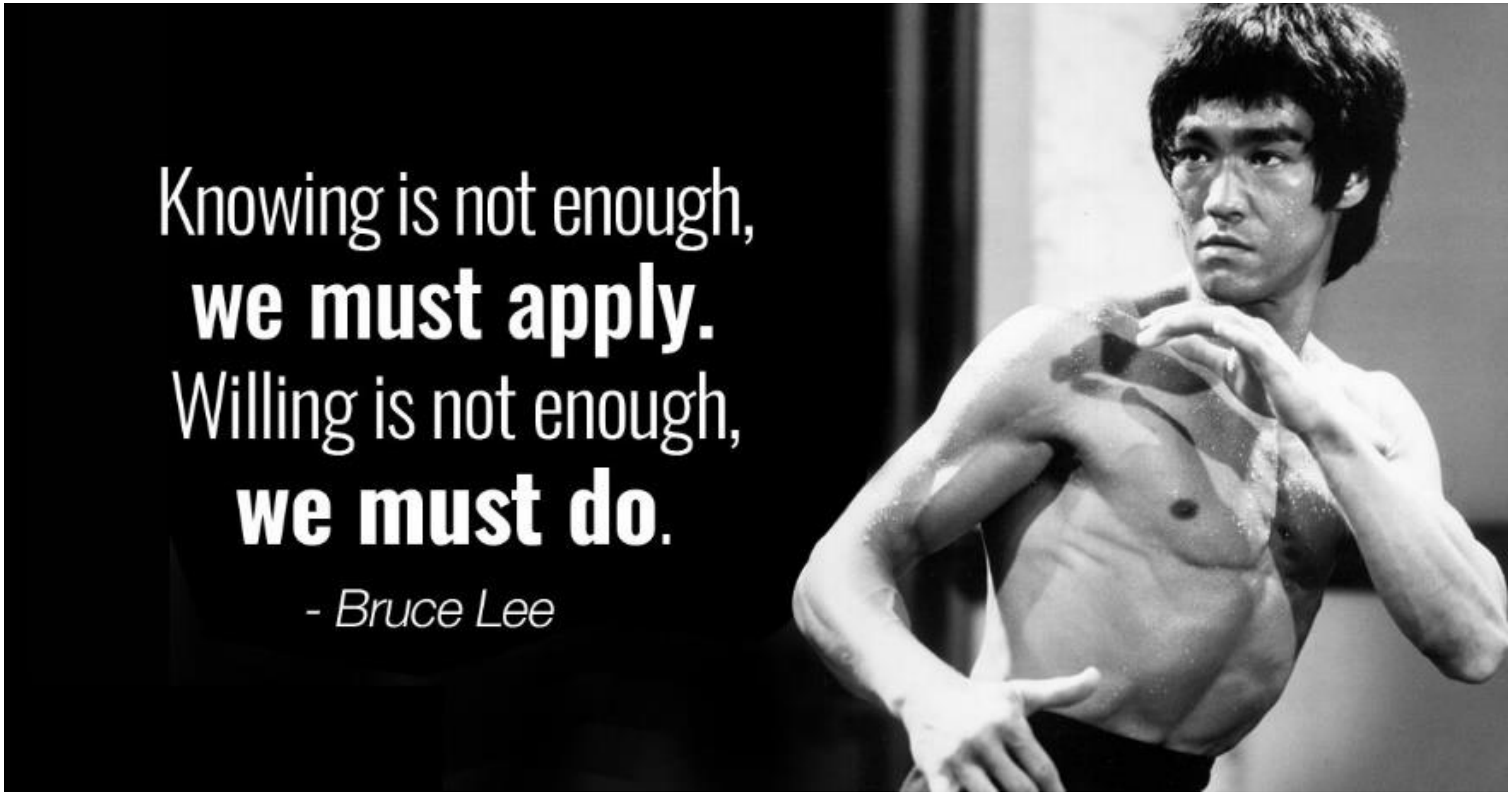
```
char name[10];
char address[100];
cout << "Name: ";
cin >> name;
cout << "Address: ";
do {
    cin.getline(address, 100);
} while (address[0] == '\0');
cout << "- Name: " << name << endl;
cout << "- Address: " << address << endl;
```

Output

```
Name: Thuy
Address: 254 Nguyen Van Linh
- Name: Thuy
- Address: 254 Nguyen Van Linh
```

Contents

Practices by Your-Self in 5 minutes



Example 01

- Enter a string, print the string length.

```
int main() {
    char str[100];
    cout << "Enter a string: ";
    cin.getline(str, 100);
    int length = 0;
    while (str[length] != '\0') {
        length++;
    }
    cout << "The string length: " << length;
    return 0;
}
```

Example 02

- Write a function to copy a c-style string to another

```
void myStrcpy(char destination[], char source[]);
int main() {
    char strA[100];
    char strB[] = "Hello DTU";
    myStrcpy(strA, strB);
    cout << "strA: " << strA << endl;
    cout << "strB: " << strB << endl;
    return 0;
}
```

Output

```
strA: Hello DTU
strB: Hello DTU
```

Example 02 (cont.)

- The solution:

```
void myStrcpy(char destination[], char
source[]) {
    int i = 0;
    while (source[i] != '\0') {
        destination[i] = source[i];
        i++;
    }
    destination[i] = '\0';
}
```

VI. cstring functions

- `#include <cstring>`
  - **cstring** or **string.h** defines functions to manipulate C-style strings and arrays
  - http://www.cplusplus.com/reference/cstring/
- Get string length:

```
unsigned int strlen(const char* str);
```

- Copy string:

```
char* strcpy(char* destination, const char* source);
```

- Copy characters from string:

```
char* strncpy(char* destination, const char* source,
unsigned int num);
```

VI. cstring functions (cont.)

- Concatenate strings:

```
char* strcat(char* destination, const char* source);
```

- Append characters from string:

```
char* strncat(char* destination, const char* source,
unsigned int num);
```

- Compare two strings:

```
int strcmp(const char* str1, const char* str2);
```

- Compare characters of two strings:

```
int strncmp(const char* str1, const char* str2,
unsigned int num);
```

- **Note:** some compilers required: strcat_s, strncat_s,..

Example 01:

```cpp
#include <iostream>
#include <cstring>
using namespace std;
int main() {
  char str1[100] = "Hello DTU";
  char str2[100] = "123456789", str3[100];
  cout << "str1: " << str1 << endl;
  cout << "strlen(str1): " << strlen(str1) << endl;
  strcpy(str3, str1); // Copy str1 to str3
  cout << "strcpy(str3, str1): " << str3 << endl;
  strncpy(str2, str1, 4);//Copy 4 chars to str2
  cout << "strncpy(str2, str1, 4): " << str2;
  return 0;
}
```

Output

```
str1: Hello DTU
strlen(str1): 9
strcpy(str3, str1): Hello DTU
strncpy(str2, str1, 4): Hell56789
```

Example 02:

```
#include <iostream>
#include <cstring>
using namespace std;
int main() {
    char str1[100] = "Hello";
    char str2[100] = "CMU-CS OOP C++";
    char str3[100] = "Students";
    strcat(str1, str2);
    cout << "strcat(str1, str2): " << str1 << endl;
    strncat(str3, str2, 3);
    cout << "strncat(str3, str2, 3): " << str3;
    return 0;
}
```

Output

```
strcat(str1, str2): HelloCMU-CS OOP C++
strncat(str3, str2, 3): StudentsCMU
```

Example 03:

```
#include <iostream>
#include <cstring>
using namespace std;
int main() {
  char str1[100] = "Hello";
  char str2[100] = "Hallo";
  if (strcmp(str1, str2) == 0) {
    cout << "str1 and str2 are the same" << endl;
  } else if (strcmp(str1, str2) > 0) {
    cout << "str1 > str2" << endl;
  } else if (strcmp(str1, str2) < 0) {
    cout << "str1 > str2" << endl;
  }
  return 0;
}
```

Output

```
str1 > str2
```

Contents

VII. string class

- `#include <string>`
  - Defines **std::string class** to represent sequence of characters as an object of class.
  - string class defined functions to operate on strings.
  - **www.cplusplus.com/reference/string/string/**
- Initialization string objects:
  - `string s0;`
  - `string s1 = "Hello DTU";`
  - `string s2("Hello DTU");`
  - `string s3 = s1;`
  - `char str[] = "DTU students";`
  - `string s4 = str;`

VII. string class (cont.)

string class <string>	C string <cstring>

`string s1 = "Hello";`
`string s2 = "DTU";`
`string s3;`
	`char c1[100] = "Hello";`
	`char c2[] = "DTU";`
	`char c3[100];`

`cin >> s3;`
`getline(cin, s3);`
	`cin >> c3;`
	`cin.getline(c3, 100);`

`s1.length();`	`strlen(c1);`
`s3 = s1;`	`strcpy(c3, c1);`
`s3 = s1.substr(0, 2);`	`strncpy(c3, c1, 2);`
`s3 = s3 + s2;`	`strcat(c3, c2);`
`s3 += s2.substr(0,2);`	`strncat(c3, c2, 2);`
`s1>s2; s1==s2; s1<s2;`	`strcmp(c2, c3);`

Example 01

Output

```
First name: Thuy
Last name: Tran
Class room: 702 NVL
Thuy Tran
Learn in 702 NVL
```

```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
    string firstName, lastName, classRoom;
    cout << "First name: "; cin >> firstName;
    cout << "Last name: "; cin >> lastName;
    cout << "Class room: ";
    do {
        getline(cin, classRoom);
    } while (classRoom.length() == 0);
    cout << firstName + " " + lastName << endl;
    cout << "Learn in " << classRoom;
    return 0;
}
```

Example 02

```
void toUpperCase(string str) {
    for (int i = 0; i < str.length(); i++) {
        if ('a' <= str[i] && str[i] <= 'z')
            str[i] = str[i] - 32;
    }
}
int main() {
    string s = "Viet Nam";
    toUpperCase(s);
    cout << s << endl;
    return 0;
}
```

Output

```
Viet Nam
```

Example 03

```
string toUpperCase(string str) {
    for (int i = 0; i < str.length(); i++) {
        if ('a' <= str[i] && str[i] <= 'z')
        str[i] = str[i] - 32;
    }
    return str;
}
int main() {
    string s = "Viet Nam";
    s = toUpperCase(s);
    cout << s << endl;
    return 0;
}
```

Output

```
VIET NAM
```

Contents

VIII. Vector class

- `#include <vector>`
  - Defines the vector container class
  - Vectors are sequence containers representing arrays that can change in size.
- Declaration for an vector: `vector<type> name;`
- Example:
  - `vector<int> intVt;`
  - `vector<float> floatVt;`
  - `vector<string> stringVt;`

VIII. Vector class (cont.)

- Initializing vectors:
  - `vector<int> vt1;`
  - `vector<int> vt2(4, 8);`

	0	1	2	3
vt2	8	8	8	8

  - `vector<int> vt3(vt2);`

	0	1	2	3
vt3	8	8	8	8

VIII. Vector class (cont.)

- Add element at the end
  - `void push_back(const value_type& val);`
- Delete last element
  - `void pop_back();`
- Get the number of elements in the vector
  - `unsigned int size() const;`
- Test whether vector is empty
  - `bool empty() const;`
- Removes all elements from the vector
  - `void clear();`

VIII. Vector class (cont.)

```
vector<int> vt;
vt.push_back(9);
vt.push_back(1);
vt.push_back(7);
cout << "size(): " << vt.size() << endl;
vt.pop_back();
cout << "size(): " << vt.size() << endl;
cout << (vt.empty() ? "Empty" : "Not empty")<< endl;
vt.clear();
cout << "size(): " << vt.size() << endl;
cout << (vt.empty() ? "Empty" : "Not empty")<< endl;
```

Contents

Summary

- C-style string is a **sequences of characters**, likes C
- **cstring** library defined functions for C-style string
- **string** class likes C-string, but easy to using
- **vector** class likes array, but better to using

Problems (cont.)

1. Cstring library has functions: strlen, strcpy, strcmp. Re-write the functions by your-self,
2. Write countWords(…) to count how many words have in a string an return the quality.
3. Write displayBinary(…) to display the binary of an integer.
4. Write functions to get two big numbers (the length > 100), and add the number together, show the result.