

Chapter 1:

Google C++ Style Guide

Mijin An

meeeeeejin@gmail.com



Background

- **C++** is one of the main development languages used by many of Google's open-source projects
- C++ has many powerful features, but this comes with **complexity**
 - It can make code more bug-prone
 - It can make code harder to read and maintain
- For **readability**, the code convention is essential
- Its main goal is to manage the complexity by describing the dos and don'ts of writing C++ code

General Naming Rules

- Do **not abbreviate** by deleting letters within a word
- Filenames should be all **lowercase** and can include _ or -
- // is much more common comment style

```
1  class MyClass {
2      public:
3          int CountFooErrors(const std::vector<Foo>& foos) {
4              int n = 0;  // Clear meaning given limited scope and context
5              for (const auto& foo : foos) {
6                  ...
7                  ++n;
8              }
9              return n;
10         }
11         void DoSomethingImportant() {
12             std::string fqdn = ...;  // Well-known abbreviation for Fully Qualified Domain Name
13         }
14     private:
15         const int kMaxAllowedConnections = ...;  // Clear meaning within context
16 };
```

Don'ts of Writing C++ Code

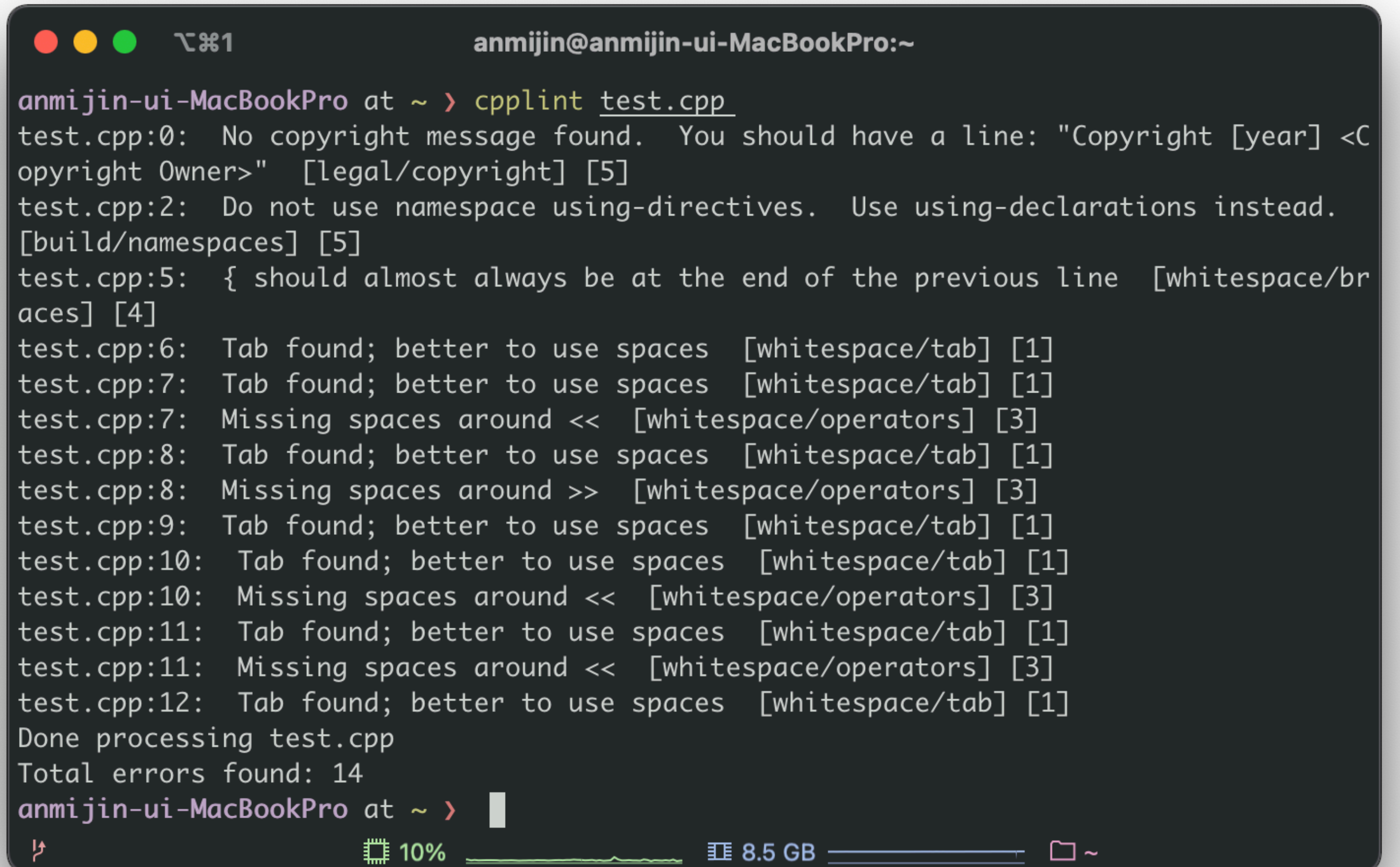
- Do **not** rely on **transitive inclusion**
- Do **not** use **using-directives** (e.g., `using namespace foo`)
- Do **not** define **implicit conversions** and **C-style casts**
- Do **not** add **new parameters** to the end of the function just because they are new
- Do **not** use C++ **exceptions**
- Do **not** use **unsigned types** to say a number will never be negative
- Do **not** use **tabs** in your code

cpplint

- A static code checker for C++
- A command-line tool to check C/C++ files for style issues following Google's C++ style guide
- Developed and maintained by Google Inc.

Example: Before Code Checking

```
1  #include<iostream>
2  using namespace std;
3
4  int main()
5  {
6      int num1, num2, add;
7      cout<<"Enter Two Numbers: ";
8      cin>>num1>>num2;
9      add = num1+num2;
10     cout<<"\nResult = "<<add;
11     cout<<endl;
12     return 0;
13 }
```

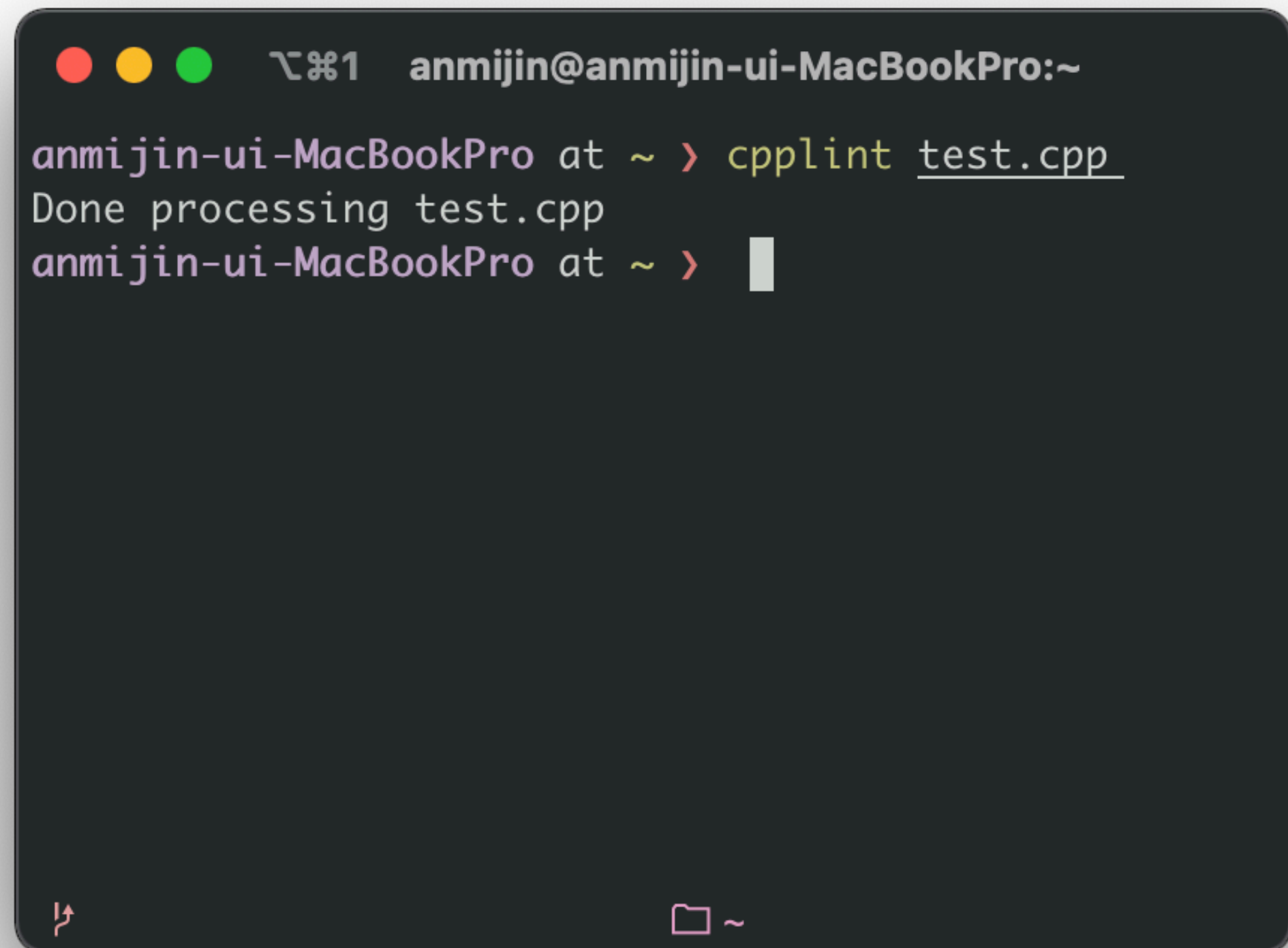


The image shows a macOS terminal window with a dark background. The title bar at the top has three colored window control buttons (red, yellow, green) and the text "anmijin@anmijin-ui-MacBookPro:~". The terminal content shows the command `cpplint test.cpp` being executed. The output lists 14 errors from the cpplint tool, including missing copyright, namespace usage, whitespace issues, and tab characters. At the bottom, it says "Total errors found: 14". The macOS status bar at the very bottom shows a 10% CPU usage indicator, 8.5 GB of memory usage, and a home directory icon.

```
anmijin-ui-MacBookPro at ~ > cpplint test.cpp
test.cpp:0: No copyright message found. You should have a line: "Copyright [year] <C
opyright Owner>" [legal/copyright] [5]
test.cpp:2: Do not use namespace using-directives. Use using-declarations instead.
[build/namespaces] [5]
test.cpp:5: { should almost always be at the end of the previous line [whitespace/br
aces] [4]
test.cpp:6: Tab found; better to use spaces [whitespace/tab] [1]
test.cpp:7: Tab found; better to use spaces [whitespace/tab] [1]
test.cpp:7: Missing spaces around << [whitespace/operators] [3]
test.cpp:8: Tab found; better to use spaces [whitespace/tab] [1]
test.cpp:8: Missing spaces around >> [whitespace/operators] [3]
test.cpp:9: Tab found; better to use spaces [whitespace/tab] [1]
test.cpp:10: Tab found; better to use spaces [whitespace/tab] [1]
test.cpp:10: Missing spaces around << [whitespace/operators] [3]
test.cpp:11: Tab found; better to use spaces [whitespace/tab] [1]
test.cpp:11: Missing spaces around << [whitespace/operators] [3]
test.cpp:12: Tab found; better to use spaces [whitespace/tab] [1]
Done processing test.cpp
Total errors found: 14
anmijin-ui-MacBookPro at ~ > 
```


Example: After Applying the Suggestions

```
1  // Copyright 2021 Mijin An
2  #include<iostream>
3
4  int main() {
5      using std::cout, std::cin;
6      int num1, num2, add;
7      cout << "Enter Two Numbers: ";
8      cin >> num1 >> num2;
9      add = num1 + num2;
10     cout << "\nResult = " << add;
11     cout << endl;
12     return 0;
13 }
```



A terminal window with a dark background and light-colored text. The window title bar shows three colored circles (red, yellow, green) and the text "anmijin@anmijin-ui-MacBookPro:~". The terminal content shows the command "cpplint test.cpp" being executed, followed by the output "Done processing test.cpp". The prompt "anmijin-ui-MacBookPro at ~ >" is visible at the bottom. The window has standard macOS window controls (red, yellow, green buttons) in the top-left corner and a small icon in the bottom-left corner.

```
anmijin@anmijin-ui-MacBookPro:~
anmijin-ui-MacBookPro at ~ > cpplint test.cpp
Done processing test.cpp
anmijin-ui-MacBookPro at ~ > 
```

Reference

[1] “Google C++ Style Guide”, Google, <https://google.github.io/styleguide/cppguide.html>

[2] “cpplint”, Google, <https://github.com/cpplint/cpplint>

[3] “C++ Program to Add Two Numbers”, CodesCracker, <https://codescracker.com/cpp/program/cpp-program-add-two-numbers.htm>