Scientific programming with C++

INTRODUCTION SLIDES

Jere Lavikainen
jere.lavikainen@uef.fi

Course organization and contents

- ~9 topics over ~8 weeks
- Contact sessions on Monday, Tuesday & Thursday 14:15-16:00
  - Brief theory about each topic
  - Help available
  - Voluntary attendance
- All mandatory assignments in UEF eLearn (Moodle)
  - Programming assignments
    - 2 weeks to turn in
    - First set available on September 4th

Course grading

- Points from programming assignments
  - 50% of all points: grade 1
  - 60%: 2
  - 70%: 3
  - 80%: 4
  - 90%: 5
- No exam – easy credit points?
  - Work is in the programming assignments

Course topics

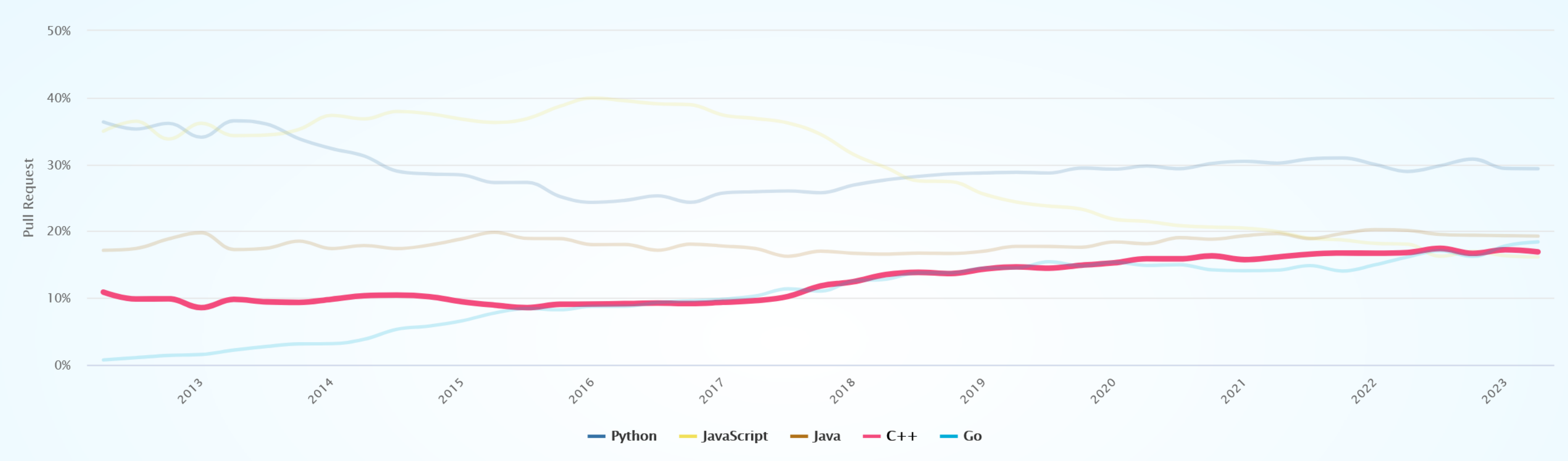
1. Introduction
2. Input and output, variables, functions
3. Objects and classes
4. Containers and iterators
5. Reading and writing data
6. Dynamic memory management and exception handling
7. Multiple inheritance
8. Multithreading
9. Use of 3rd party libraries in research

Course objectives

- Programming
  - "The purpose of the course is that the student will learn the syntax of the C++ language, can design object-oriented programs using the C++ language and are familiar with the C++ standard library."
    - Translation: **You have the necessary skills and confidence to use C++ and learn more on your own**
- Scientific
  - Examples of C++ in academic research
    - Reading and processing research data
    - Using APIs of research software
  - Hands-on programming assignment at the end of the course
  - **You'll have some idea of how to use C++ for research purposes**

Why C++?

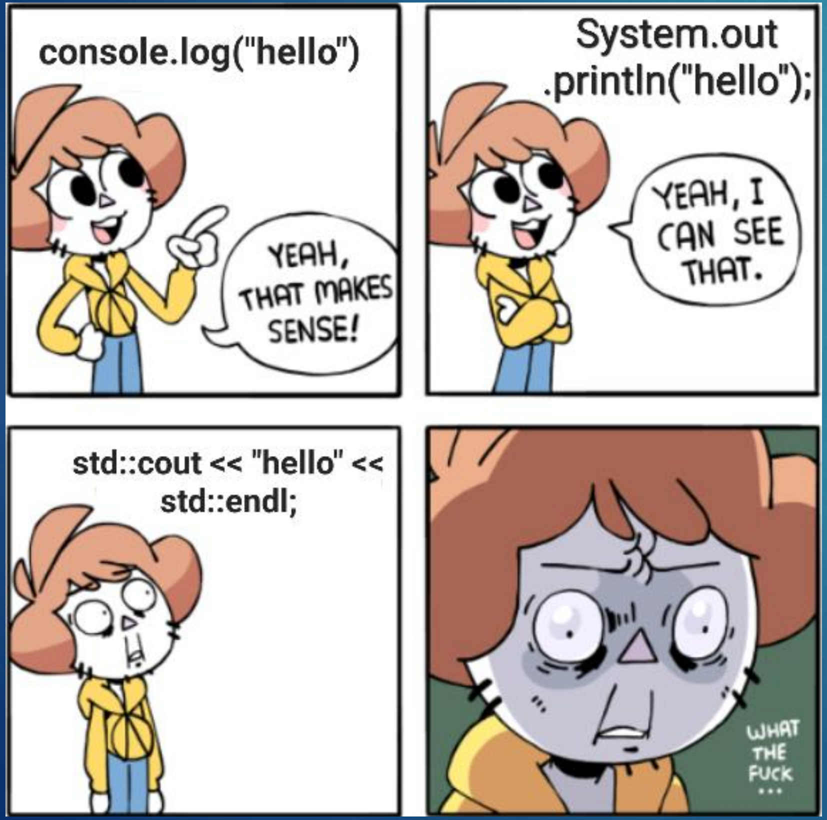
- Suitable for high-performance applications
- Object-oriented programming and scientific modeling go hand in hand
- And so forth (https://www.geeksforgeeks.org/top-10-reasons-to-learn-c-plus-plus/)
- Used by millions of people worldwide



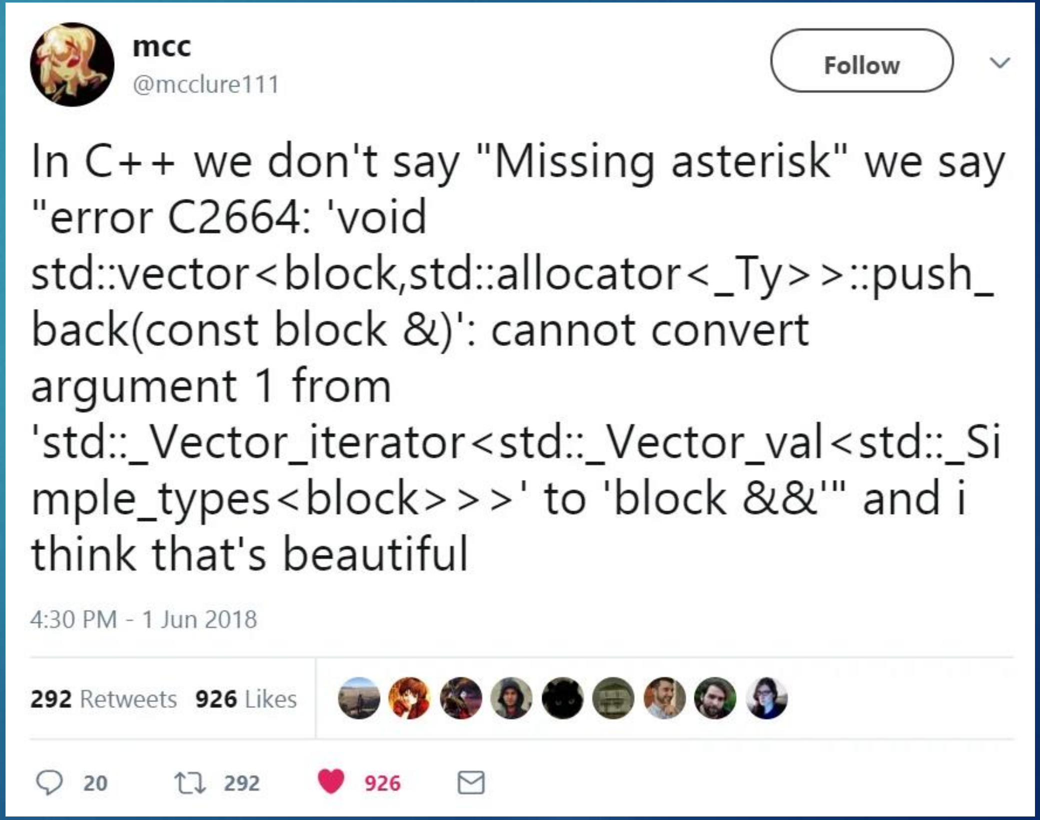
https://madnight.github.io/githut/#/pull_requests/2023/2

Why not C++?

- The syntax is cumbersome
- Making very simple programs can be easier on other languages like Python
  - But in scientific context, you usually don't make very simple programs



https://www.reddit.com/r/ProgrammerHumor/comments/x8ri80/seriously_wtf_c/



mcc
@mcclure111

In C++ we don't say "Missing asterisk" we say "error C2664: 'void std::vector<block,std::allocator<_Ty>>::push_back(const block &)': cannot convert argument 1 from 'std::_Vector_iterator<std::_Vector_val<std::_Simple_types<block>>>' to 'block &&'" and i think that's beautiful

4:30 PM - 1 Jun 2018

292 Retweets **926** Likes

20 292 926

https://www.reddit.com/r/ProgrammerHumor/comments/8nwyma/c_is_a_beautiful_language/

When to use C++ in research?

- I need to…
  - create plots of data I already have
    - MATLAB/Python
  - conduct statistical analysis for my research paper
    - MATLAB/Python
  - train shallow neural networks for time series prediction
    - MATLAB/Python (or C++ if I have too much spare time)
  - extend the API of existing software libraries so I can create my own analysis pipeline
    - MATLAB/C++/Python
  - make a program that reads data from measurement devices, processes it immediately and uses the results to control the measurement or provide real-time feedback
    - C++
- The more complex or performance-heavy the task is, the more likely I will use C++ for it
- For quick independent tasks that require little customization, I use something else

Some characteristics of C++

- Object-oriented
  - Programming is done by instancing objects from classes, which model systems with different properties, unique behavior, and inheritance
- High-/middle-level language
  - Both written English and more complex statements in syntax
- Supports multi-threading
  - Useful in high-performance applications
- Memory management is dynamic but not fully automated
  - Memory leaks are possible
- The programmer has a lot of control over fine details
  - And a lot of responsibility

Hello world compared with MATLAB and Python

MATLAB

```matlab
% This is a comment
disp("Hello world")
```

C++

```cpp
#include <iostream>

// This is a comment
int main()
{
    std::cout << "Hello world";
}
```

Python

```python
# This is a comment
print('Hello world')
```

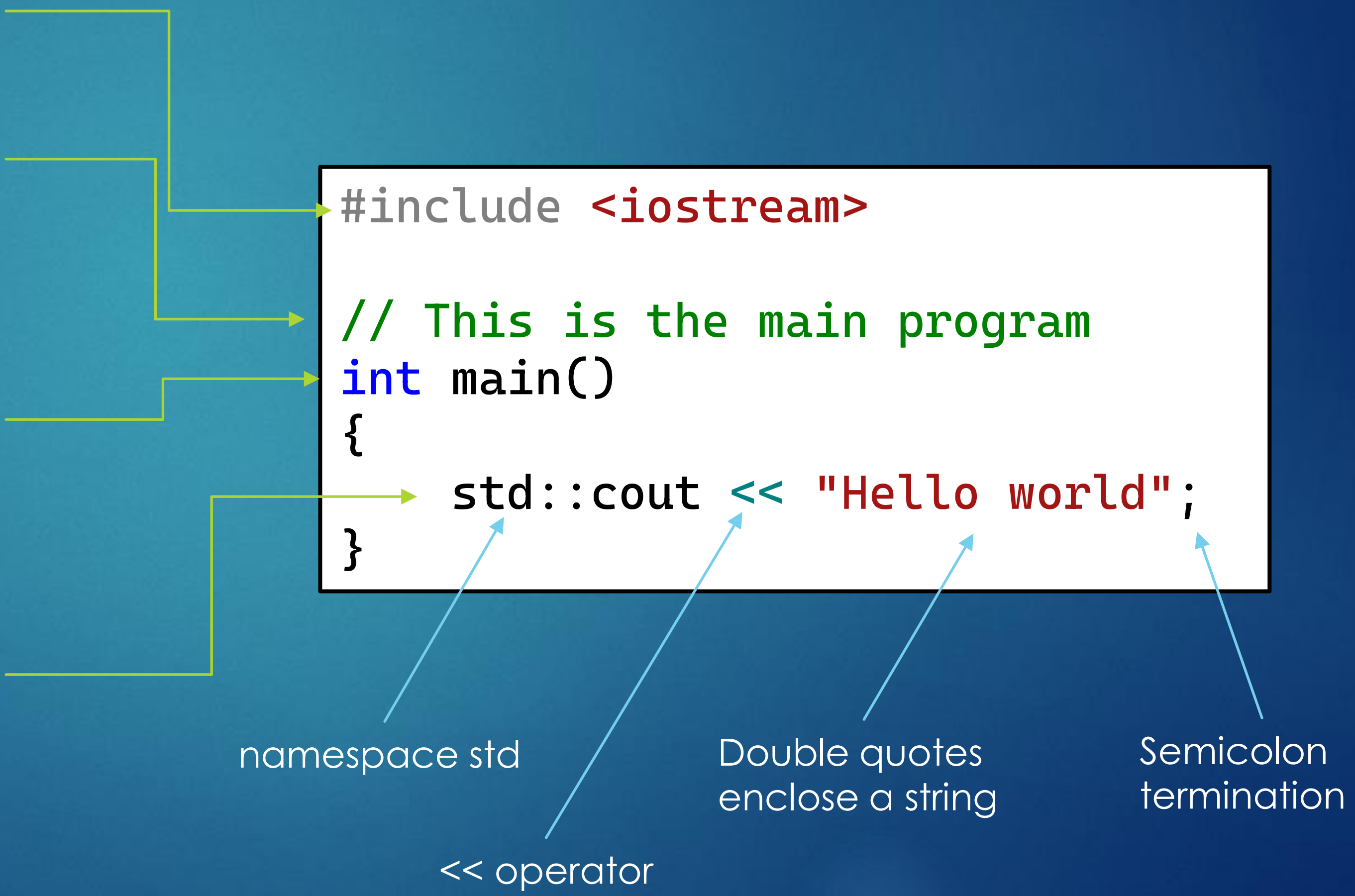
Line-by-line breakdown of “Hello world”

A preprocessor directive to include objects from the iostream header

A comment for the human reader, not compiled in the program.

A function begins with its type (int) and name (main). The parentheses enclose its input arguments (in this case none). The “main” function is where all functionality of the program is located (functions that aren’t called directly or indirectly from main are never called) inside the curly brackets.

cout is an object from the iostream library that can be used to print text.



```
#include <iostream>

// This is the main program
int main()
{
    std::cout << "Hello world";
}
```

namespace std

<< operator

Double quotes enclose a string

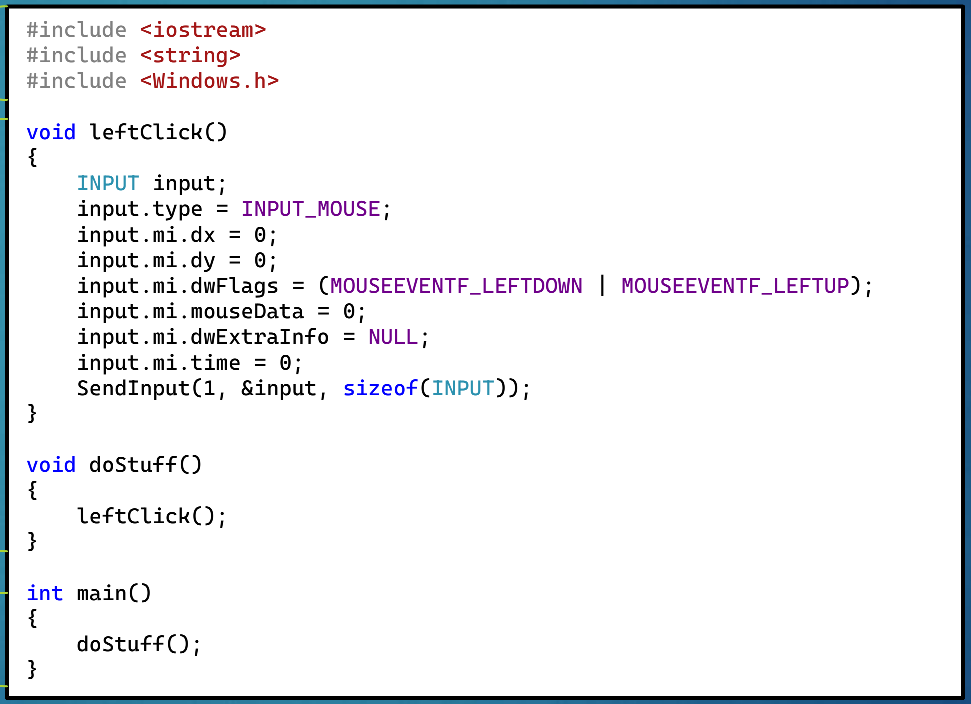
Semicolon termination

Anatomy of C++ code

headers to include

functions in order of dependency

main() function



```
#include <iostream>
#include <string>
#include <Windows.h>

void leftClick()
{
    INPUT input;
    input.type = INPUT_MOUSE;
    input.mi.dx = 0;
    input.mi.dy = 0;
    input.mi.dwFlags = (MOUSEEVENTF_LEFTDOWN | MOUSEEVENTF_LEFTUP);
    input.mi.mouseData = 0;
    input.mi.dwExtraInfo = NULL;
    input.mi.time = 0;
    SendInput(1, &input, sizeof(INPUT));
}

void doStuff()
{
    leftClick();
}

int main()
{
    doStuff();
}
```

C++ basics

- camelCase
- Operators
- Indexing from 0
- Loops
- If statements and conditions
- Blocks
- Namespaces
- Comments

C++ basics: camelCase

- Variables and functions are named using lowerCamelCase
  - All separate words in the name except the first one start with an uppercase letter
  - Examples: coordinatePoint, calculateTotalArea, userIdentifier
- Classes are named using UpperCamelCase
  - All words in the name start with an uppercase letter
  - Examples: Vehicle, KinematicsSolver, DataValueManager
- This is not enforced in the language itself, but is a common convention and helps understanding your and others' code
  - Other styles exist, for example snake_case
  - **Use a consistent style in your projects**, whatever the style is

C++ basics: operators

- Characters (usually symbols) that have some defined behaviour for variables
- Some (but not all) operators:
  - Assignment (=)
    - Assigns a value (right) to a variable (left)
  - Arithmetic (+, -, *, /, %)
    - Basic maths
  - Compound assignments (+=, -=, *=, /=, and so forth)
    - Modifies the value of a variable (left)
  - Increment and decrement (++, --)
    - Increase or decrease the value of a variable by 1
  - Relational and comparison (==, !=, >, <, >=, <=)
  - Logical (!, &&, | |)

```cpp
int a = 1;
```

```cpp
int b = 2 * a;
```

```cpp
a += 2; // same as a = a+2;
```

```cpp
++a; // same as a = a+1;
```

```cpp
bool c = (a == b);
```

```cpp
bool d = !c;
```

C++ basics: indexing

- The first element of a container or array is at index 0
- Elements are usually accessed with square brackets []
- Example:

```
std::string exampleString = "programming";
```

  - exampleString has 11 elements
  - 1st element is ‘p’ and accessed with exampleString[0]
  - 2nd element is ‘r’ and accessed with exampleString[1]
  - Last element is ‘g’ and accessed with exampleString[10]
    - Even though exampleString has 11 elements!

C++ basics: loops

- Three types: for loop, while loop, and do-while loop
- For loop is an incrementing loop
- While loop repeats as long as a defined condition is met
  - Ideally you will change this condition at some point in the loop to avoid endless looping
- Do-while loops repeats as long as a defined condition is met (like while loops) but the loop is executed at least once even if the condition is never met
- Loops can be (and often are) nested
  - Loops within loops
  - The “break” keyword can be used to exit the innermost loop

C++ basics: for loop

Define and initialize the looping variable “i” to be an unsigned integer starting from 0

Set the condition for the loop to be: “variable i is smaller than 10”

Increment i by one after each loop

```
for (unsigned int i = 0; i < 10; ++i) {
        std::cout << i << std::endl;
    }
```

C++ basics: while loop

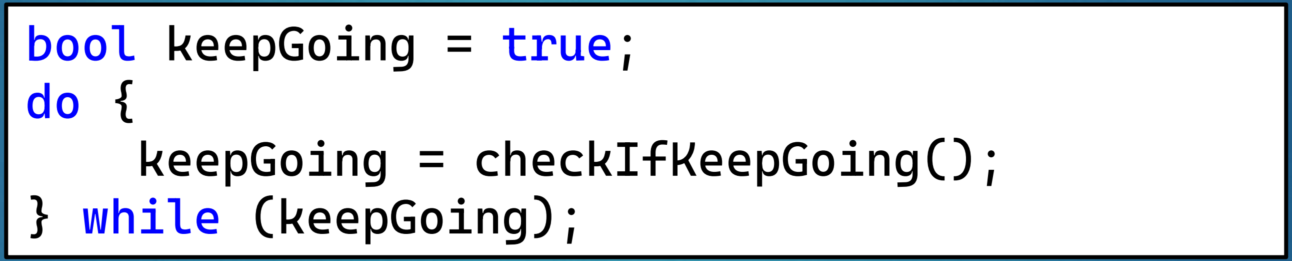
Define and initialize a boolean variable as true

```cpp
bool keepGoing = true;
while (keepGoing) {
    keepGoing = checkIfKeepGoing();
}
```

Create a while loop with the condition “while keepGoing is true”

Update the value of keepGoing from another function

C++ basics: do-while loop



```cpp
bool keepGoing = true;
do {
    keepGoing = checkIfKeepGoing();
} while (keepGoing);
```

C++ basics: if statements and conditions

```cpp
if (condition) {
    // implement what to do here
    doThings();
}
else if (anotherCondition) {
    doStuff();
}
else {
    doSomething();
}
```

- Condition: a statement that returns a Boolean value

a<b	a<=b	a>=b	a>b	a==b	a!=b

- Note! == is equality operator, = is assignment operator
- Note! If a and b are different types, comparison may not work properly

C++ basics: blocks

- Curly brackets are used to organize C++ code into separate sections called blocks (or compound statements)
  - Both mandatory (functions, multiline loops and if statements) and voluntary
- Can be nested
  - First block is “level 1”, its inner block is “level 2“ etc
- Defines the scope of variables
  - A variable definition inside a block is not valid in outer blocks, or lower-level blocks
- Can be used for a variety of handy tricks like variable shadowing

C++ basics: namespaces

- Provides scope and grouping to identifiers such as functions, classes, variables etc
- Prevents ambiguity in identifier names
  - What if you use two different libraries that both have a class “Matrix”?

```
Matrix someMatrix;
```

    - If the library namespaces are “abc” and “xyz”

```
abc::Matrix oneMatrix;
xyz::Matrix anotherMatrix;
```

- Accessed with
  - The fully qualified name (the safer way)

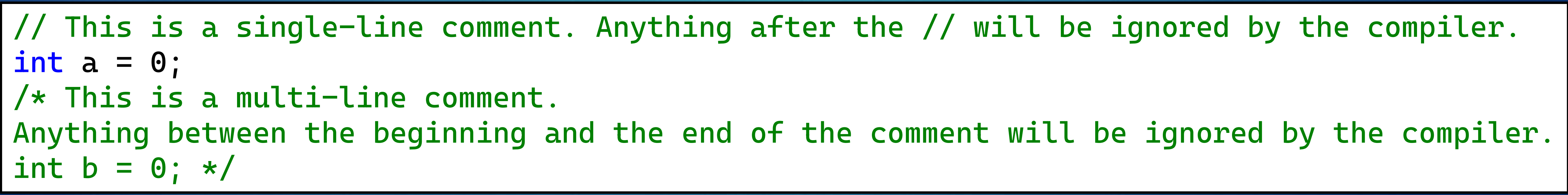
```
std::cout << "Hello world";
```

  - The “using” declaration (the easier way)

```
using namespace std;
cout << "Hello world";
```

C++ basics: comments

- Use comments to explain your code
  - To others
  - To yourself
- Temporarily disable parts of the code
- Single-line comments start with //
- Multi-line comments start with /* and end with */



```
// This is a single-line comment. Anything after the // will be ignored by the compiler.
int a = 0;
/* This is a multi-line comment.
Anything between the beginning and the end of the comment will be ignored by the compiler.
int b = 0; */
```

C++ basics takeaways

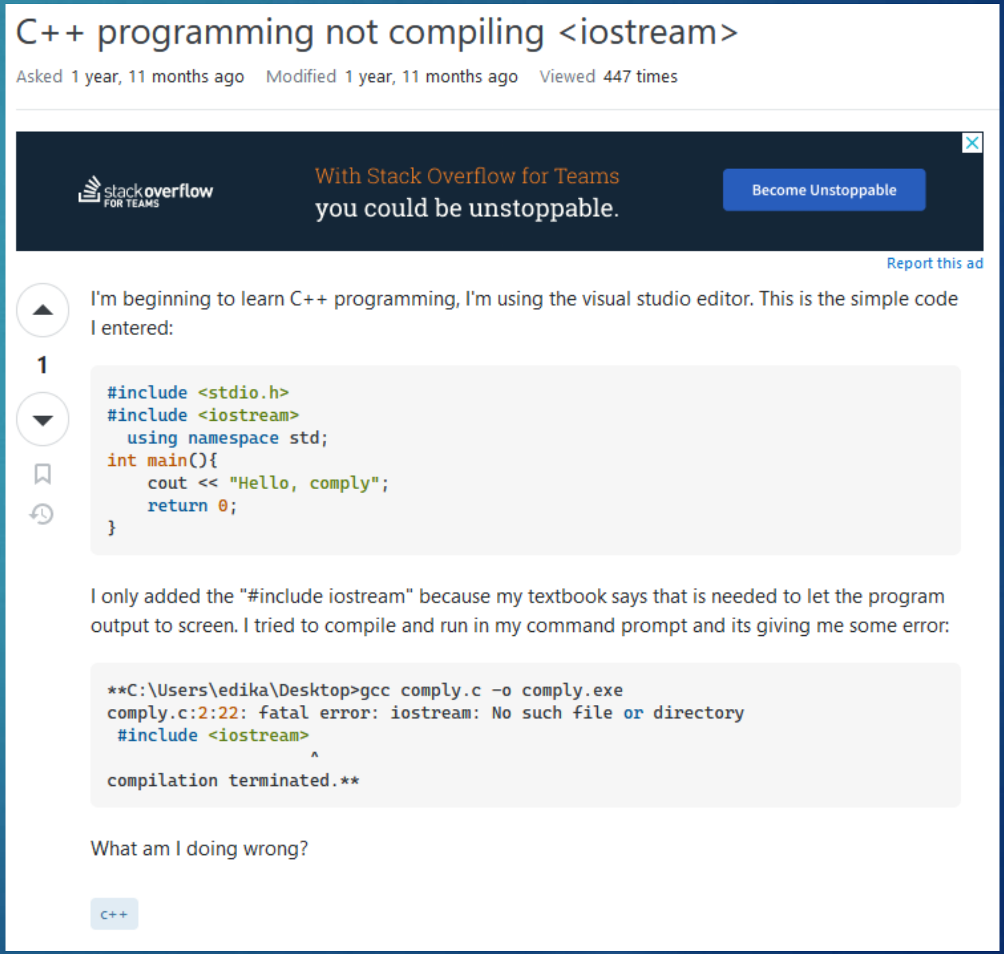
- camelCase
  - Use a consistent style
- Blocks
  - Understand variable scope
- Namespaces
  - Avoid mixing up similarly named objects or functions
  - Group identifiers by libraries
    - Example: C++ Standard Library (std)
- Comments
  - Explain what your code should do to others and yourself

Surviving the course

- Software tools
  - Visual Studio, VSC, NetBeans, Eclipse, …
  - Matplot++ or another data visualization library recommended
  - Online C++ compilers such as https://www.onlinegdb.com/online_c++_compiler
- Useful websites
  - Stack Overflow
  - W3Schools
  - Tutorialspoint
  - GeeksforGeeks
  - ChatGPT (chat.openai.com)
    - University has ruled that you must mention when turning in your assignment if you used AI tools

Stack Overflow

- Any issues you encounter while writing code have already been asked (and usually answered) by someone else



C++ programming not compiling <iostream>

Asked 1 year, 11 months ago Modified 1 year, 11 months ago Viewed 447 times

stackoverflow FOR TEAMS
With Stack Overflow for Teams you could be unstoppable.
Become Unstoppable
Report this ad

I'm beginning to learn C++ programming, I'm using the visual studio editor. This is the simple code I entered:

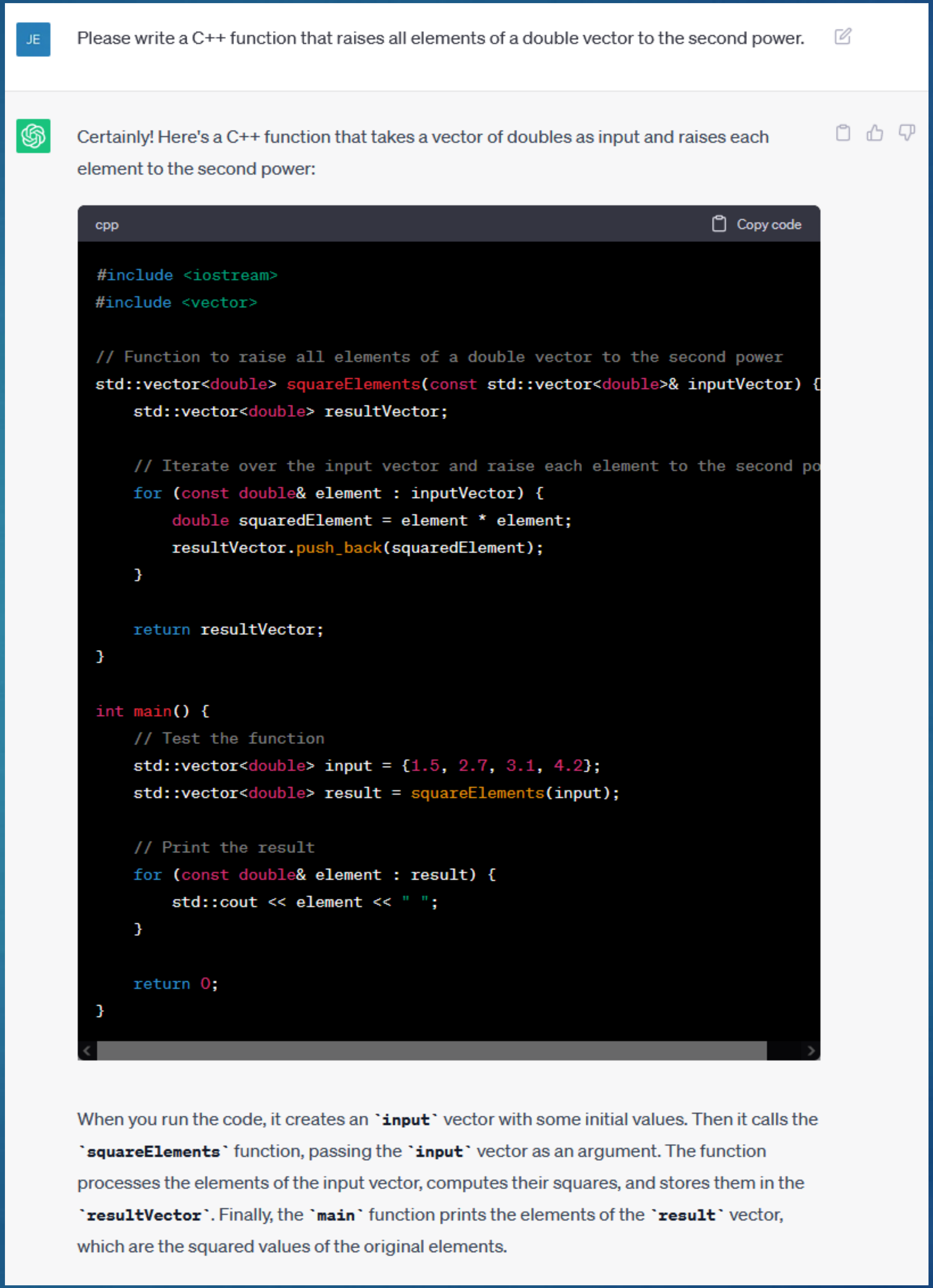
```
#include <stdio.h>
#include <iostream>
  using namespace std;
int main(){
    cout << "Hello, comply";
    return 0;
}
```

I only added the "#include iostream" because my textbook says that is needed to let the program output to screen. I tried to compile and run in my command prompt and its giving me some error:

```
**C:\Users\edika\Desktop>gcc comply.c -o comply.exe
comply.c:2:22: fatal error: iostream: No such file or directory
 #include <iostream>
                    ^
compilation terminated.**
```

What am I doing wrong?

ChatGPT



Please write a C++ function that raises all elements of a double vector to the second power.

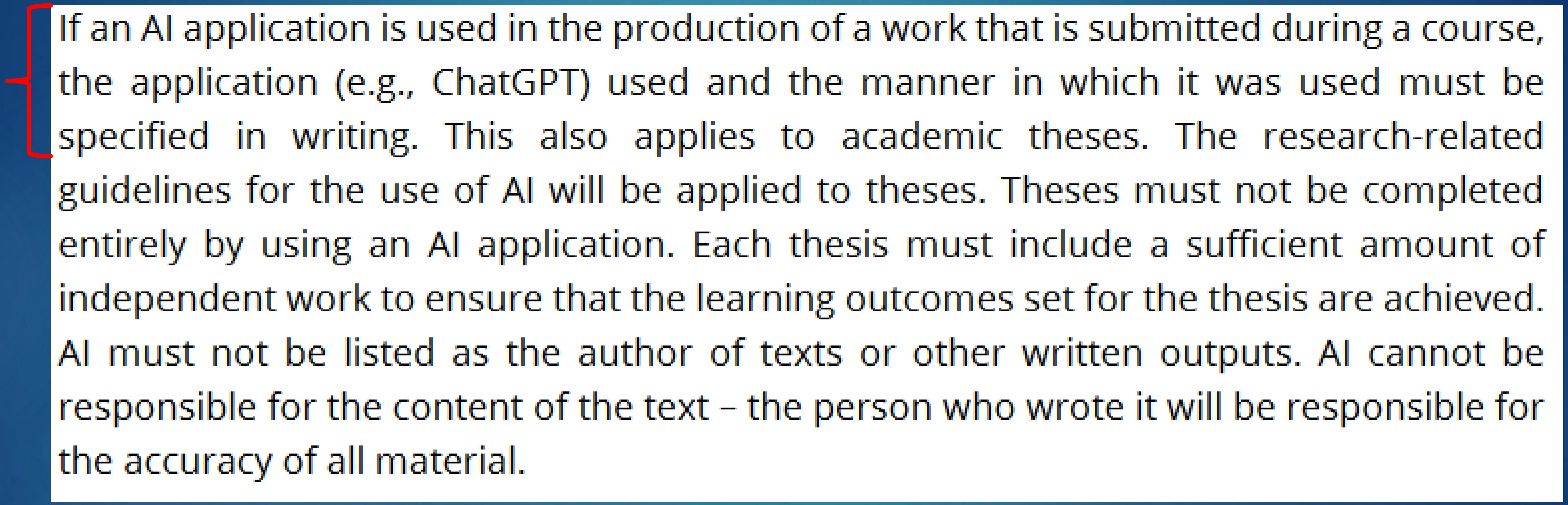
Certainly! Here's a C++ function that takes a vector of doubles as input and raises each element to the second power:

```cpp
#include <iostream>
#include <vector>

// Function to raise all elements of a double vector to the second power
std::vector<double> squareElements(const std::vector<double>& inputVector) {
    std::vector<double> resultVector;

    // Iterate over the input vector and raise each element to the second po
    for (const double& element : inputVector) {
        double squaredElement = element * element;
        resultVector.push_back(squaredElement);
    }

    return resultVector;
}

int main() {
    // Test the function
    std::vector<double> input = {1.5, 2.7, 3.1, 4.2};
    std::vector<double> result = squareElements(input);

    // Print the result
    for (const double& element : result) {
        std::cout << element << " ";
    }

    return 0;
}
```

When you run the code, it creates an `input` vector with some initial values. Then it calls the `squareElements` function, passing the `input` vector as an argument. The function processes the elements of the input vector, computes their squares, and stores them in the `resultVector`. Finally, the `main` function prints the elements of the `result` vector, which are the squared values of the original elements.

ChatGPT: disclaimer

- Guidelines for the use of AI in teaching and research at the University of Eastern Finland (14/6/2023):



If an AI application is used in the production of a work that is submitted during a course, the application (e.g., ChatGPT) used and the manner in which it was used must be specified in writing. This also applies to academic theses. The research-related guidelines for the use of AI will be applied to theses. Theses must not be completed entirely by using an AI application. Each thesis must include a sufficient amount of independent work to ensure that the learning outcomes set for the thesis are achieved. AI must not be listed as the author of texts or other written outputs. AI cannot be responsible for the content of the text – the person who wrote it will be responsible for the accuracy of all material.

- Mention how you used ChatGPT when turning in your assignment

Surviving the course

- Some prior programming experience is assumed but not strictly required
  - If you don’t know basic concepts like loops and functions, you may need some extra work to catch up
- Sources of information organized from best to worst:
  1. Stack Overflow
  2. Google & programming websites
  3. Lecturer
  4. ChatGPT
- Slides are not meant to be exhaustive
  - You are expected to read more elsewhere
- **Reserve plenty of time for the programming assignments**

Introduction survey

- Moodle
    - Previous programming experience
    - Expectations and wishes for the course
- Lots of answers already
    - Python is a common background
    - Some requested topics will be in the course
        - We may be able to explore some others in the last sessions
    - Some asked for challenging and interesting assignments
        - Others hoped they can pass the course

Setting up your C++ environment

- Text editor
  - Preferably with syntax highlighting
  - Notepad++, Visual Studio Code, …
- C++ compiler
  - Builds computer programs from source code
- Integrated development environment (IDE)
  - Text editor, debugger, compiler
  - Visual Studio Community