

Welcome to Programming Fundamentals with C++

This course is designed to teach you the fundamental concepts of programming using C++. We'll explore problem-solving methodologies and build a solid foundation for your programming journey. Get ready to design, analyze, and decompose complex problems!



Course Overview: Mastering C++ Fundamentals

This course aims to master fundamental programming concepts using C++. We'll cover key topics, grading, and the schedule. There are helpful resources such as textbooks and online platforms like tutorials point and w3schools.

Versatility

Highly versatile language

Performance

Excellent performance

Relevance

Industry relevance

```

    mat fon ane(enuedbly {
        cmarrcts wbiel;; // sticton;
        chlide.;
        beffen is = caf+ cnuch. + clentl);
        cattnrs, instrual {
            caaut clnobor+{
                costrunt ();
                dedfing, = del(lar + is (low, uned))
            }
        facutr()
        ++= ct ++ . alerplutian convection)
        vald.(lt + lx l);
        detplynt ar, folien));
    )
    c+++ l);
    compoutions is, = (lawl){ + blac mact(conservat)

        citini);
        (++ _nait ();
        contract (lon) si resack) |

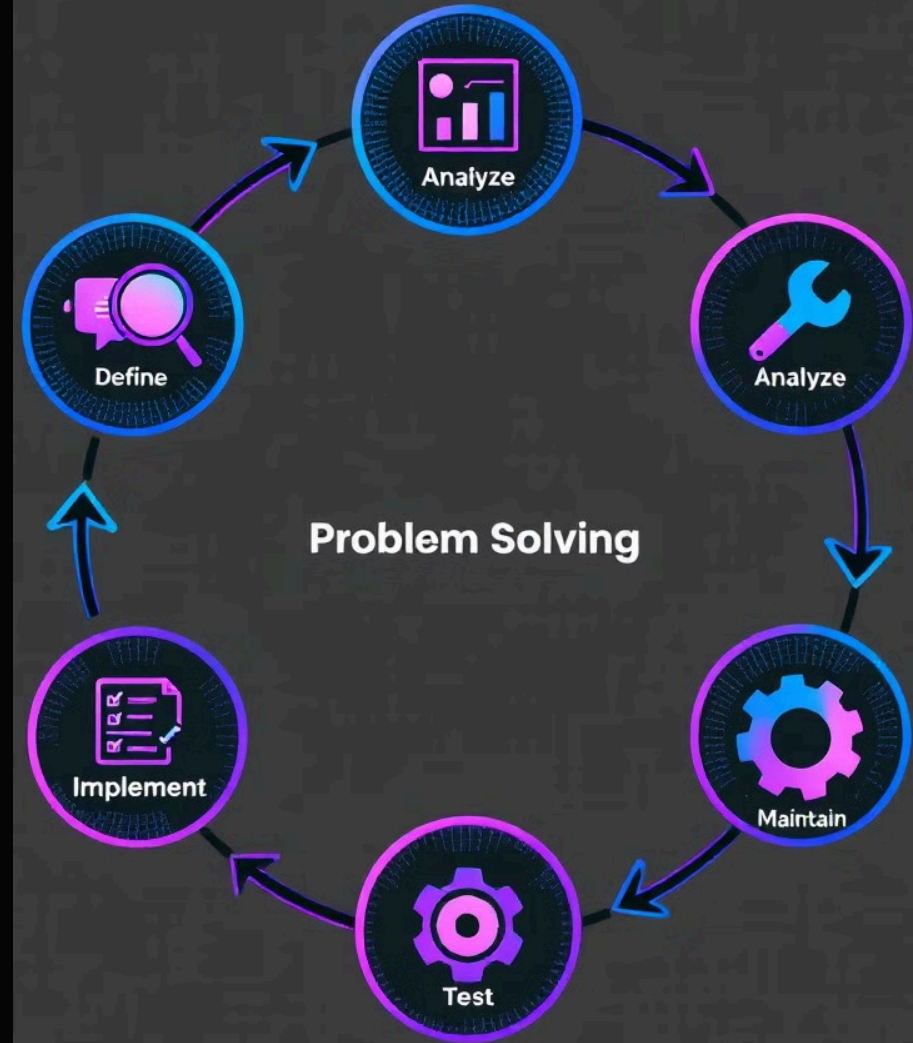
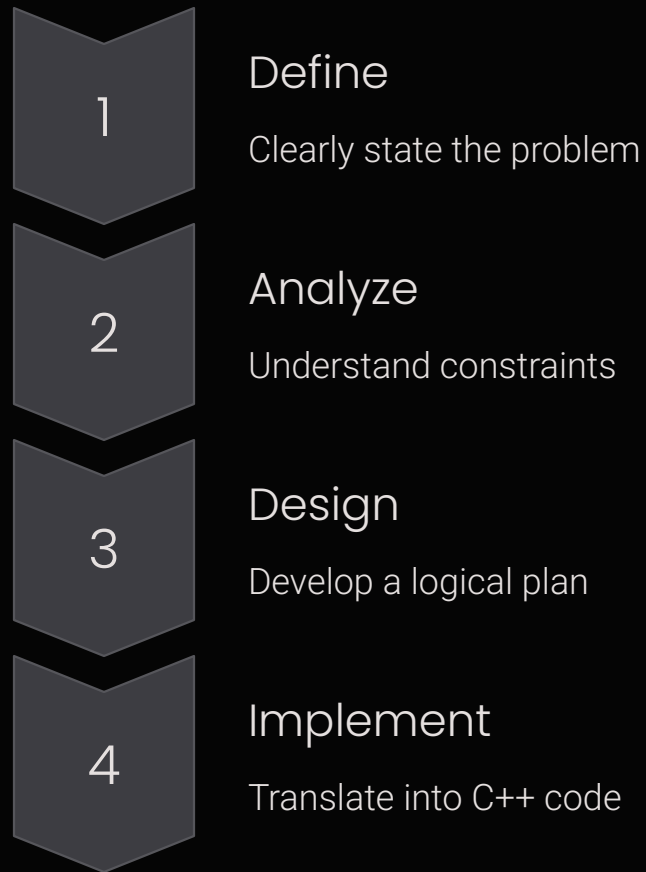
        d))
        callention(( = r connect+ #lype
        at y);

        claualle, neccentsional);
        blup()
        iif, sdat(lar, or constan)) {
            clatent(estion'(low low conservat, vblat)
            ndeterit, (lawt lobari law)
            / statlernpstill, = choalic int zilon + lawm + blac(lype
            )
            gnations((lar))
            incitic consack

```

The Problem-Solving Methodology

A step-by-step guide to solve any problem.



A close-up, low-angle shot of a desk at night. On the left, a dark brown mug sits on a saucer. Next to it is a small potted plant. In the foreground, two papers are spread out. The top paper is titled 'Algorithms' and features a flowchart with boxes labeled 'Input', 'Process', and 'Output', along with some handwritten notes. The bottom paper is titled 'DESIGNING' and has some diagrams. The background is softly blurred, showing more plants and warm, ambient lighting from a lamp or string lights.

Design: Crafting Solutions with Algorithms

An algorithm is a step-by-step procedure for solving a problem. A good algorithm needs clarity, efficiency, and correctness. Here's the algorithm for calculating the area of a triangle given its base and height.

1

Base Input

Get the base of a triangle

2

Height Input

Get the height of a triangle

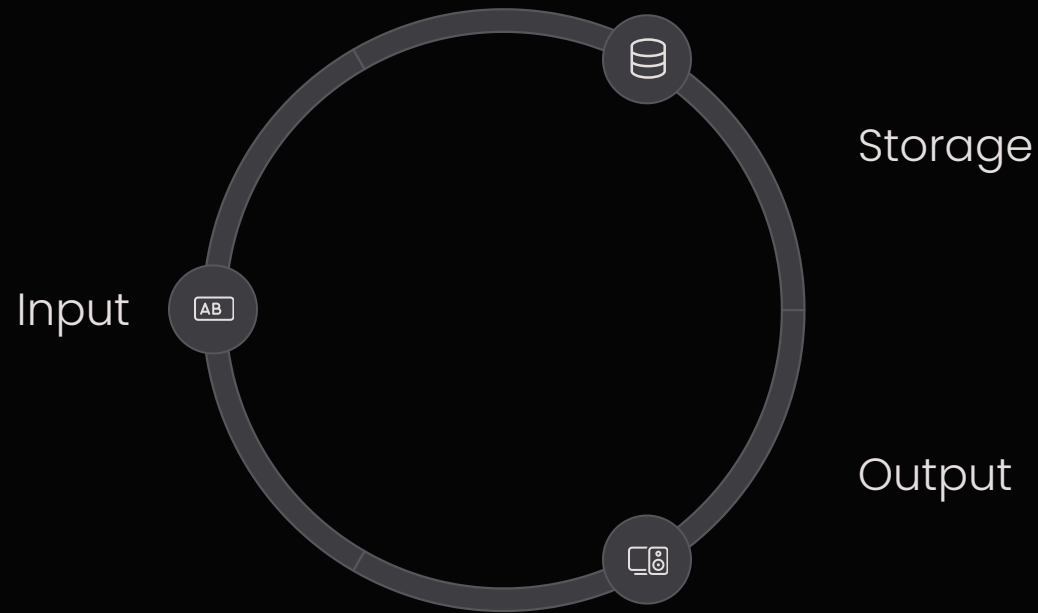
3

Calculate

Multiply base by height then divide by 2

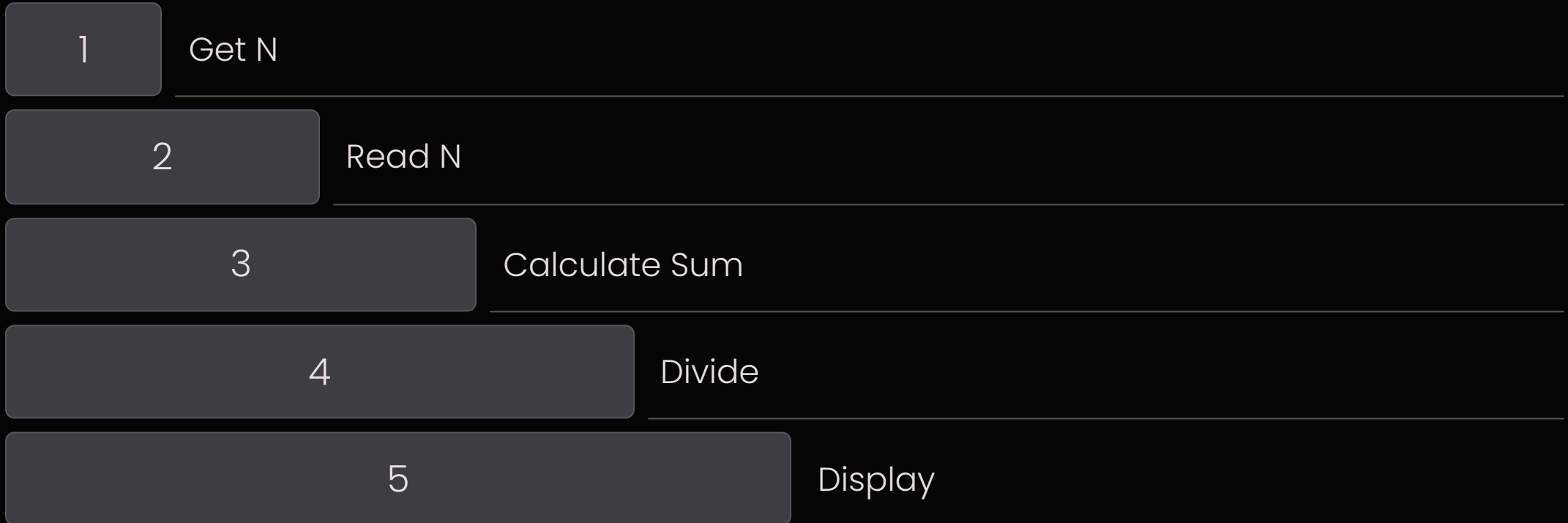
Analyze: Breaking Down Complex Problems

Decompose complex problems into smaller, manageable sub-problems, such as building a calculator app with components like input, storage and output. Simplifies development, enhances maintainability, and facilitates teamwork.



Decompose: Calculating the Average

Write a program to calculate the average of N numbers.



Algorithms: The Heart of Programming Logic

A finite sequence of well-defined instructions to solve a specific problem. They must be unambiguous, feasible, and produce the correct output.

Sorting

- Bubble sort
- Insertion sort

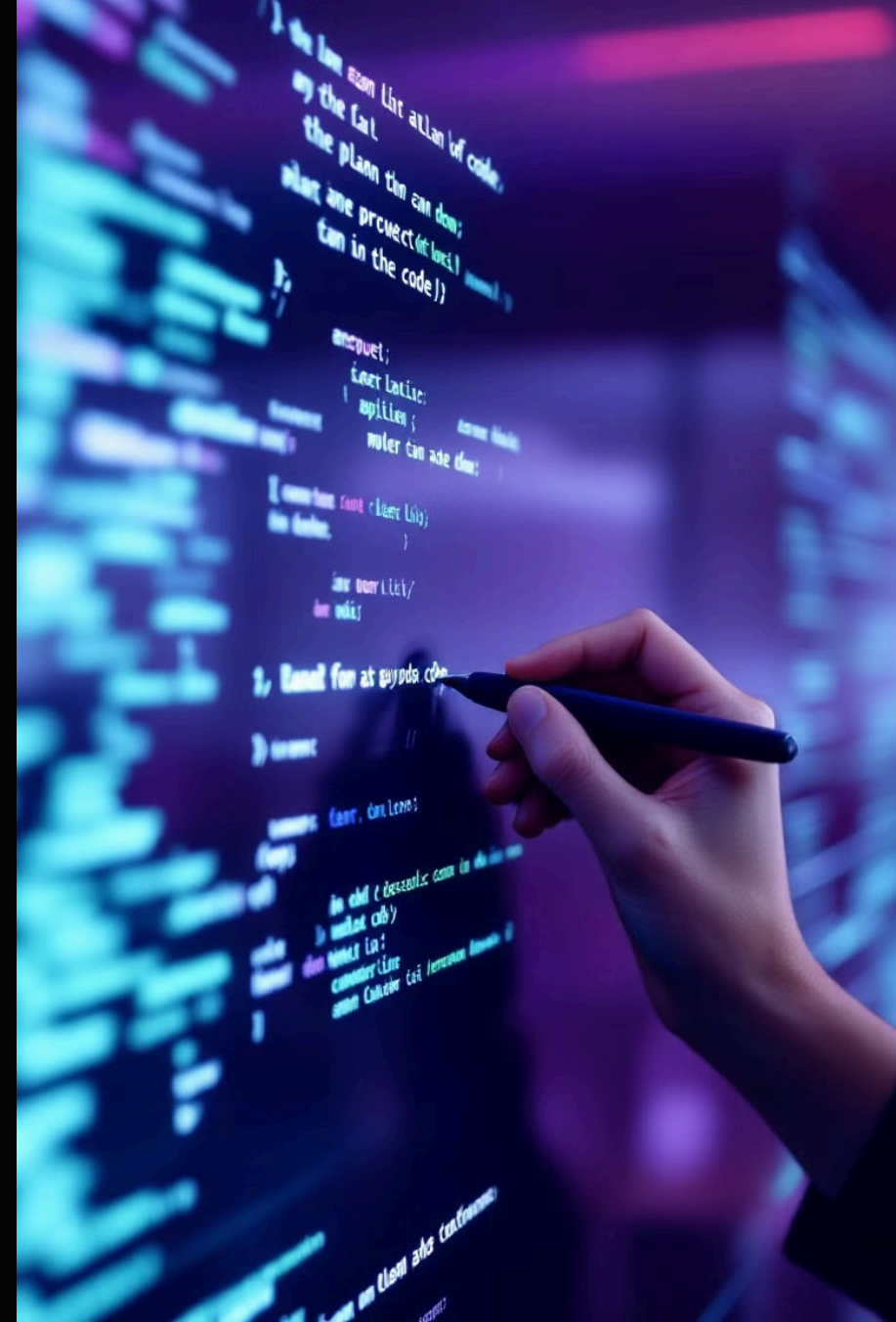
Searching

- Linear search
- Binary search

Pseudocode: Bridging the Gap to Code

An informal, high-level description of an algorithm. It outlines the logic of a program before writing actual code, making it easier to understand, facilitate communication, and helps in debugging.

```
Input: array A of numbers
max = A[0]
for each element in A:
    if element > max:
        max = element
Output: max
```



Week 1 Recap & Looking Ahead

Key takeaways: Problem-solving methodology, algorithms, pseudocode, and flow charts. Next up are the C++ basics: Variables, data types, and operators. Implement simple algorithms in C++ to reinforce concepts.

C++ Basics

Variables, data types

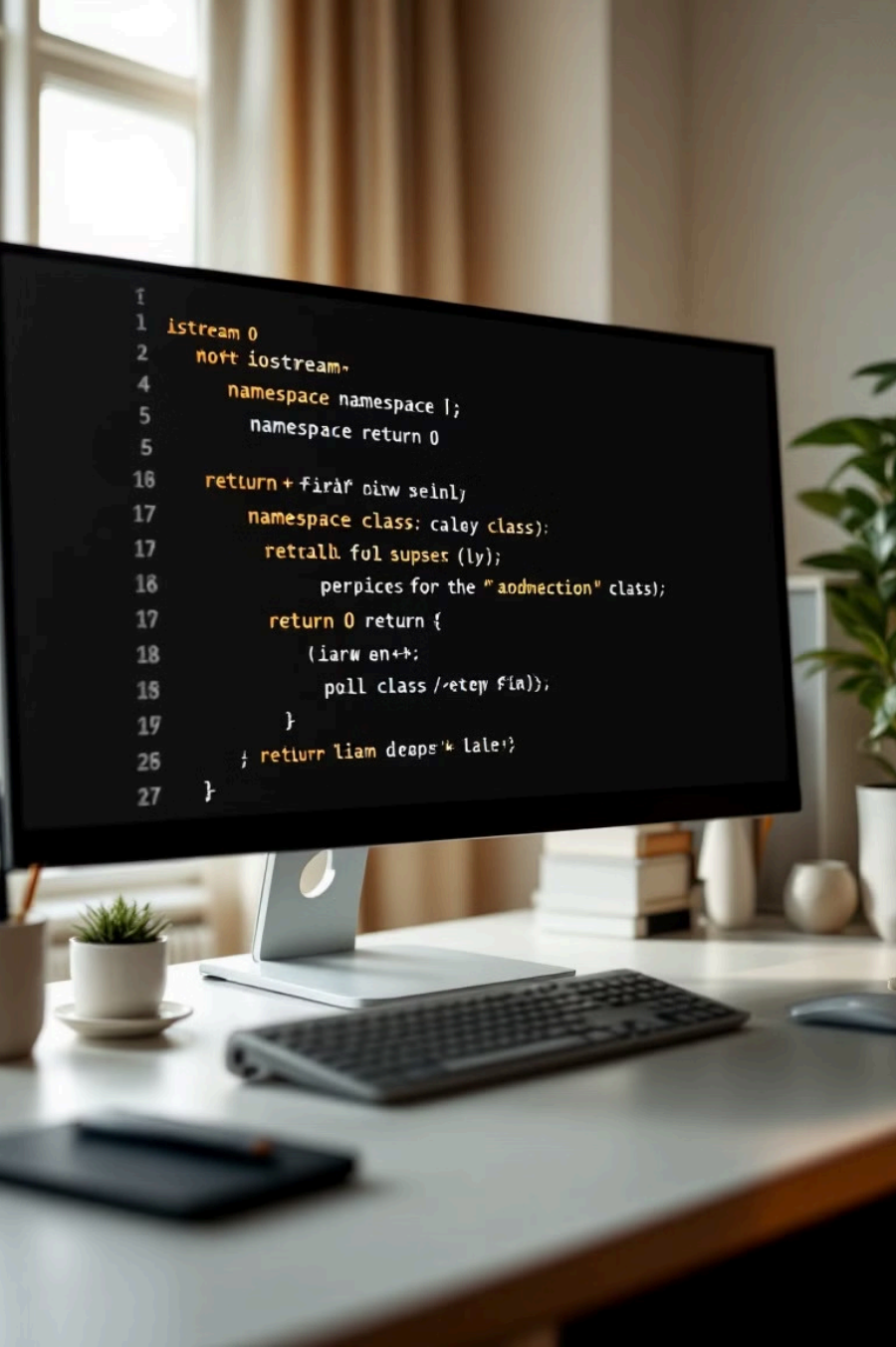
Practice

Implement algorithms in C++

Next Week

Diving into C++ syntax





```
1 #include <iostream>
2 using namespace std;
3
4 namespace namespace1;
5 namespace namespace2 {
6
7     return + first only;
8     namespace class: caley class;
9     retrah. ful supse (ly);
10    perpires for the "addrection" class;
11    return 0 return {
12        (iarw en++);
13        poll class /-etey fia);
14    }
15 }
16
17 ; return liam deaps "ale";
18 }
```

Programming Fundamentals: C++

Welcome to Programming Fundamentals! This presentation covers the history of C++, basic program structure, and essential elements like directives and comments. We'll also explore output methods and manipulators. Let's begin our journey into the world of C++.

The History of C++

C++ evolved from "C with Classes" in 1983. Bjarne Stroustrup sought efficiency and flexibility. The language added object-oriented features. It had its first ISO standard in 1998. C++ continues to evolve with new standards.

1

1979

C with Classes

2

1983

Becomes C++

3

1998

C++98 Standard

4

2011 Onward

Modern Updates



Compilers vs. Interpreters

C++ employs compilers to translate code into machine code before execution. This leads to faster execution speeds. Interpreters translate and execute code line by line. C++ is typically a compiled language.

Compilers

- Translates the entire code at once
- Faster execution
- Examples: g++, Clang

Interpreters

- Translates line by line
- Easier debugging

Basic C++ Program Structure

Let's explore the structure of a simple C++ program. This structure includes directives, comments, output using "cout", escape sequences, setw, endl, and manipulators. Understanding this structure is crucial for writing C++ code.

```
#include iostream
<int main() >
    int main(>;>
std-coutHello-World! endl:
    strings
return 0:
```

```
#include <iostream>
```

```
int main() {
    // This is a comment
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```