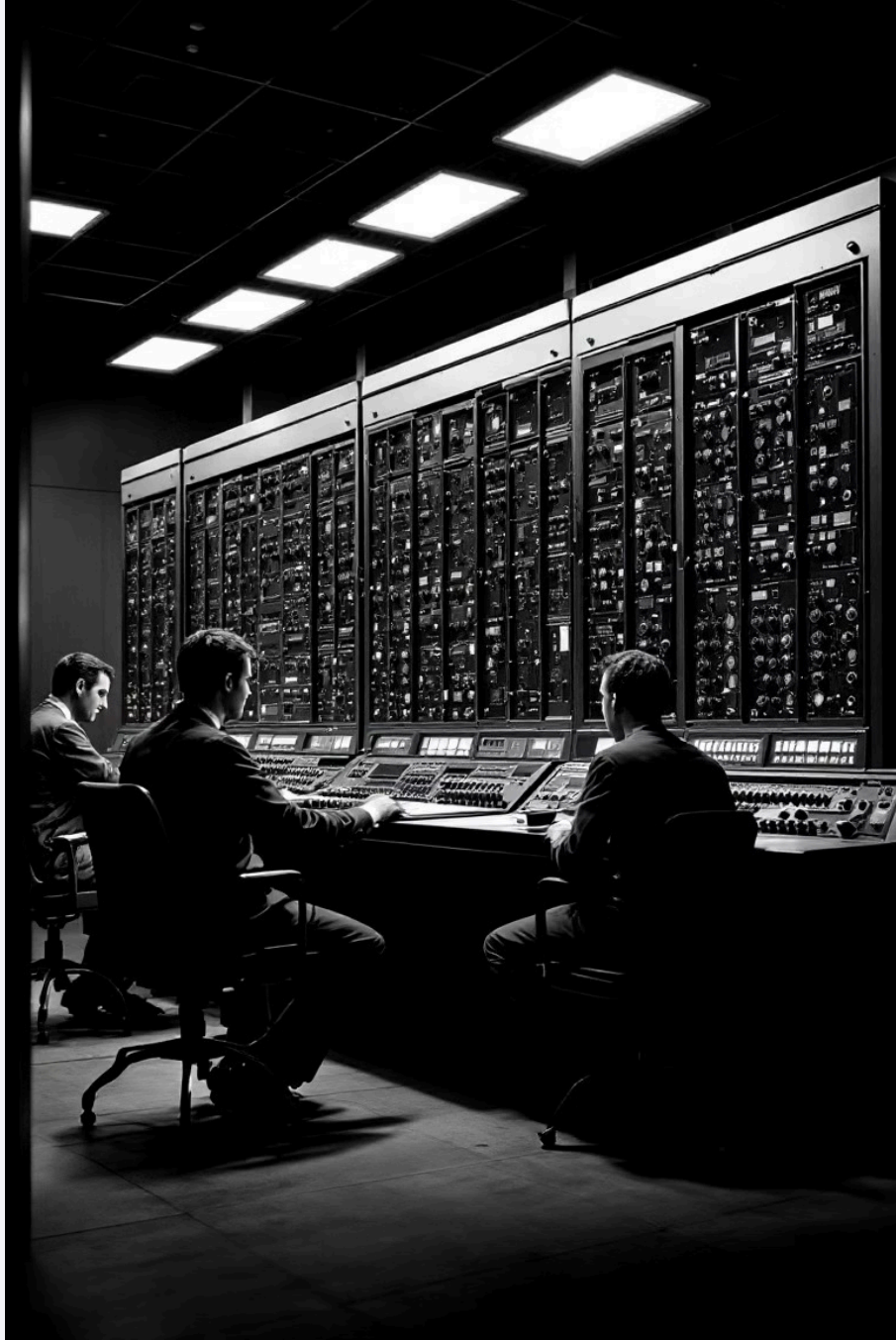
A Journey Through Code: Exploring the History and Features of C++

Embark on a journey through the evolution of programming languages. We'll focus on C++, its origins, features, and core concepts. The presentation will cover everything from early languages to modern C++.



The Dawn of Programming: Early Languages

Jacquard Loom (1800s)

Utilized punched cards for weaving patterns, a precursor to programming.

Electronic Computers (1940s)

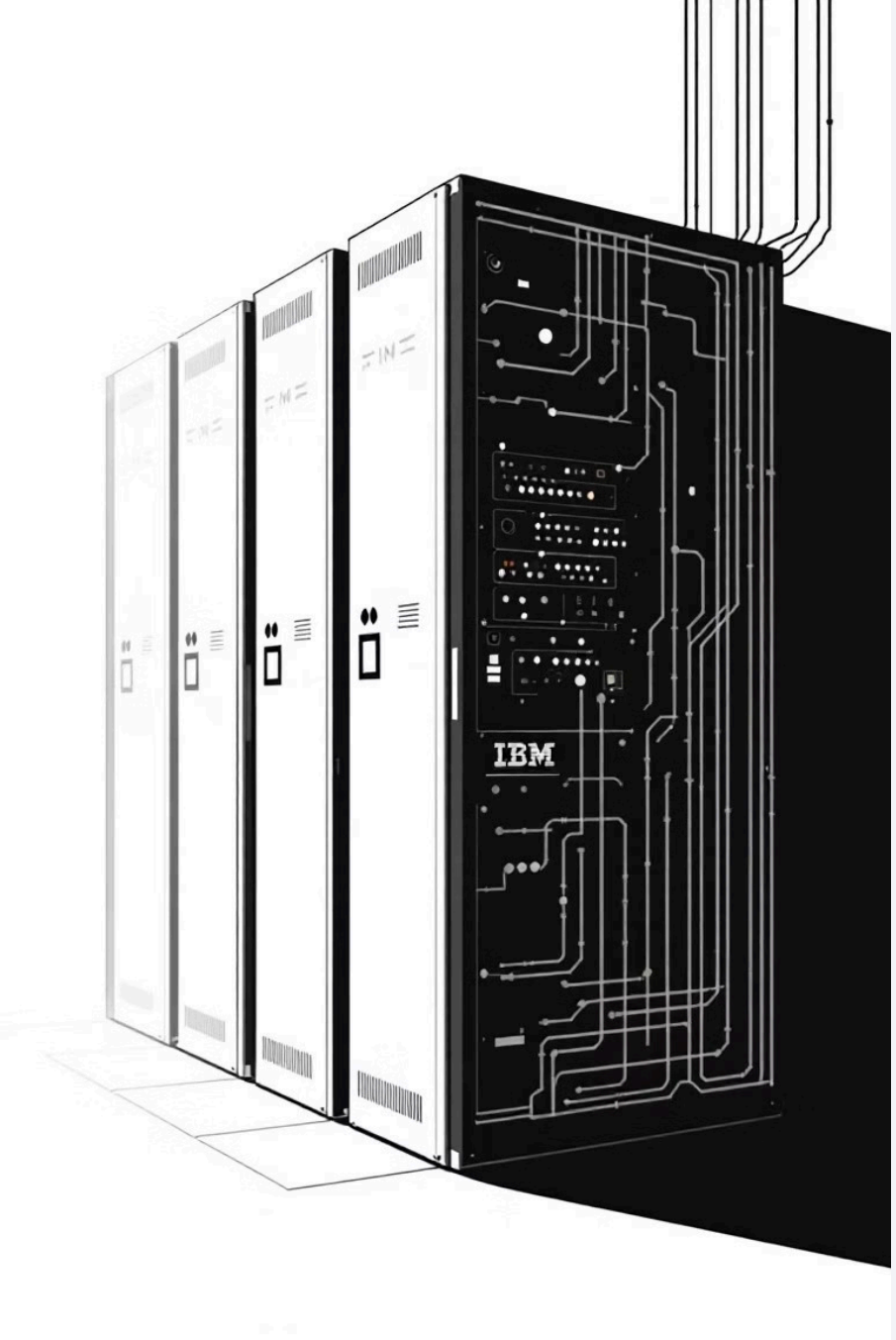
ENIAC and Colossus marked the beginning of electronic computation.

Machine Code

The initial method for programming these early machines.

Assembly Language (1949)

Introduced symbolic representation, making programming easier.



The Rise of High-Level Languages

FORTRAN (1957)

IBM pioneered FORTRAN for scientific computations.

ALGOL (1958)

ALGOL influenced language design with structured programming.

COBOL (1959)

Grace Hopper developed COBOL for business applications.

These languages paved the way for more accessible and powerful programming paradigms.

The Birth of C: A Foundation for C++

C Language (1972)

Dennis Ritchie created C at Bell Labs for the UNIX OS.

C combined high-level features with low-level control.

C's Influence

Its flexibility and power made it ideal for system programming.

C's influence on subsequent languages, including C++, is immense.

From C to C++: Object-Oriented Programming Emerges

"C with Classes" (1979)

Bjarne Stroustrup extends C with OOP principles.

Renamed to C++ (1983)

Key addition: support for classes and inheritance.

C++ combined procedural and object-oriented paradigms, broadening its utility.

Key Features of C++: Power and Flexibility

Object-Oriented Programming (OOP)

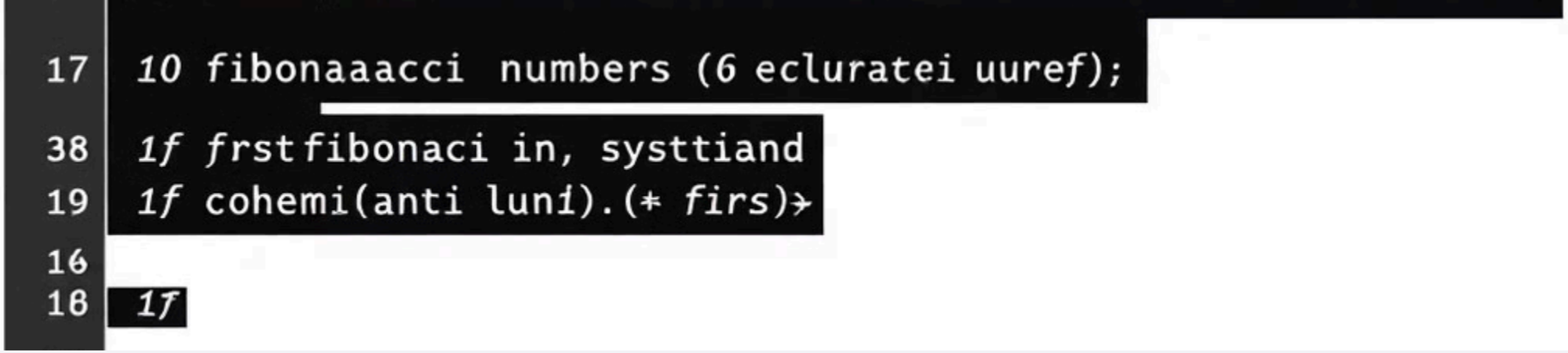
Encapsulation, inheritance, and polymorphism.

Memory Management

Manual memory management using pointers and RAII.

Standard Template Library (STL)

Generic classes and functions for data structures.



Core C++ Concepts: Variables, Data Types, and Operators

1 Variables

Named storage locations for data.

2 Data Types

- int, float, double
- char, bool

3 Operators

Arithmetic, assignment, and comparison.

Control Structures: Directing Program Flow

1

Conditional Statements

if, else if, else for decision making.

2

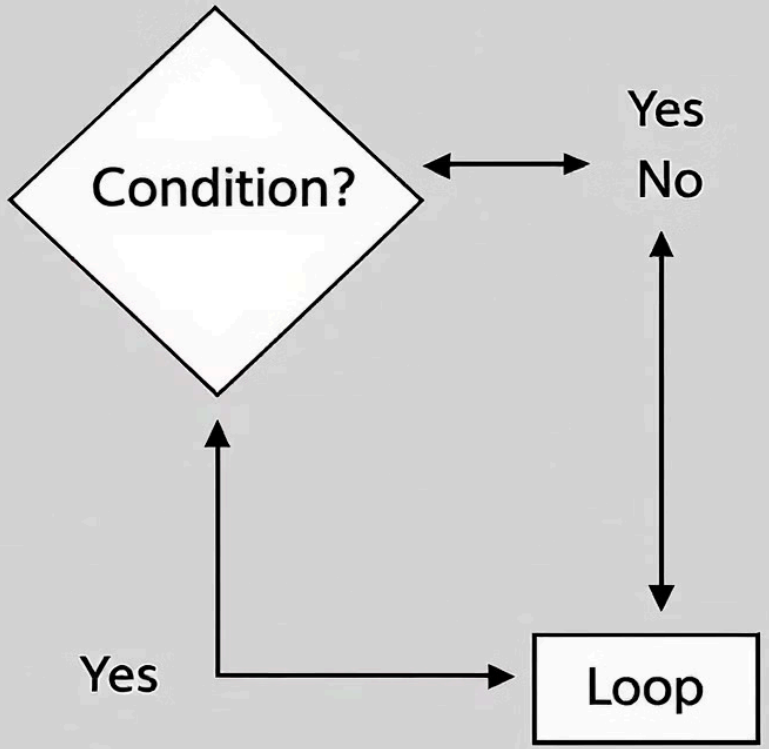
For Loops

Repeat code a specific number of times.

3

While Loops

Repeat code while a condition is true.



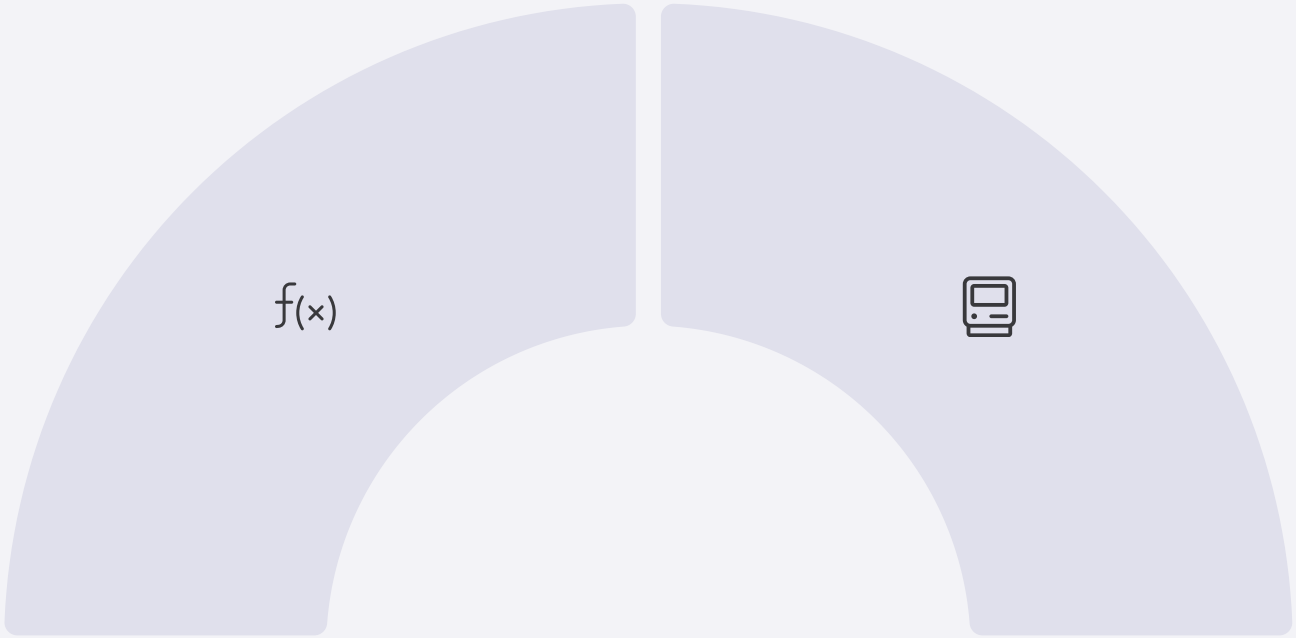
Functions and Classes: Building Blocks of C++

Functions

Reusable blocks of code.

Classes

Blueprints for creating objects.



Functions and classes are the building blocks of object-oriented programming.

C++ Today: A Lasting Legacy

C++ remains powerful for game development and system programming. Continuous evolution with new standards solidify C++ as a cornerstone.

Essential Tools for C++ Development

This presentation overviews the software needed to develop, compile, and run C++ applications. It is targeted towards developers new to C++ or setting up a new development environment.

Core Tool: Text Editor / IDE

Purpose

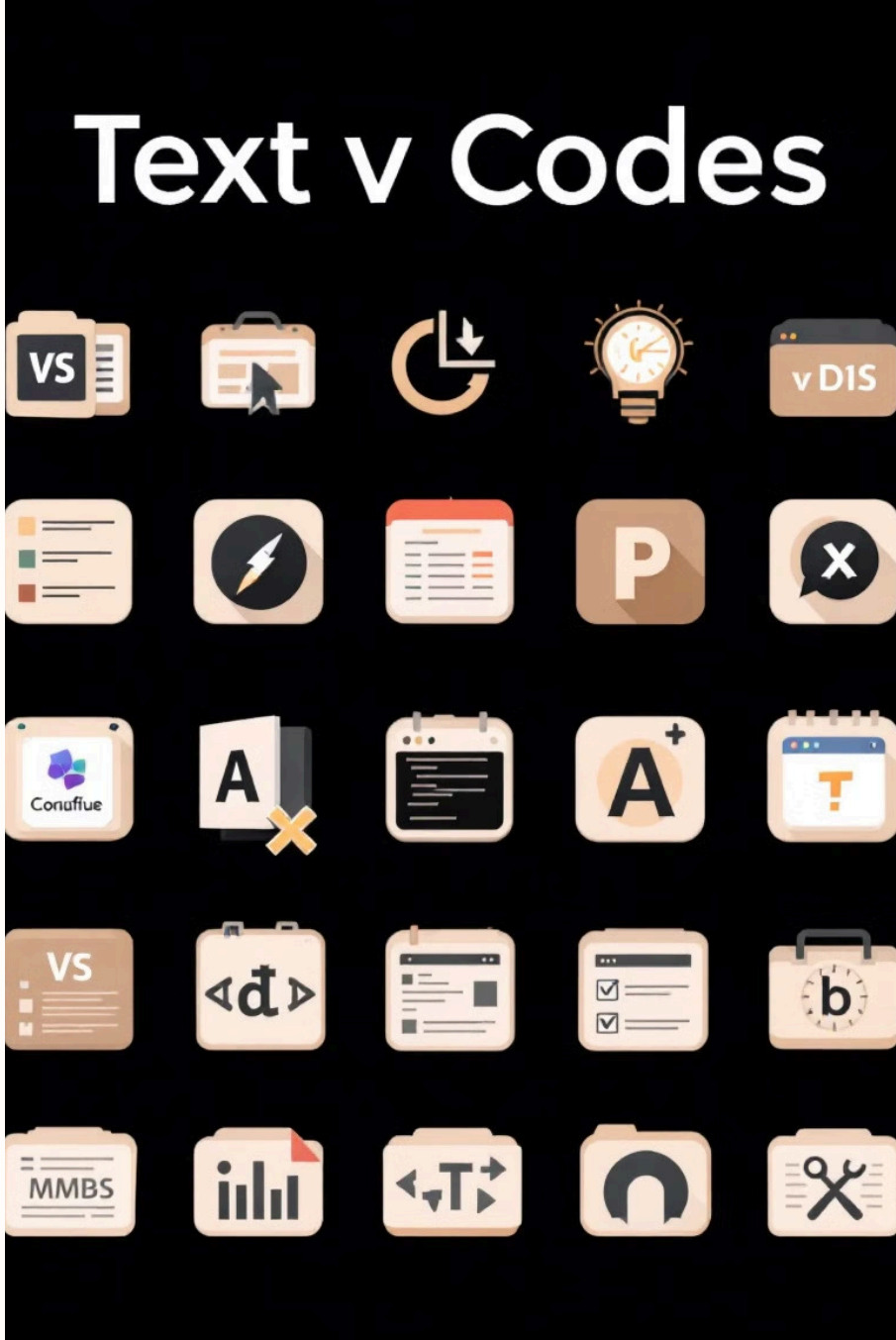
Writing and editing C++ source code

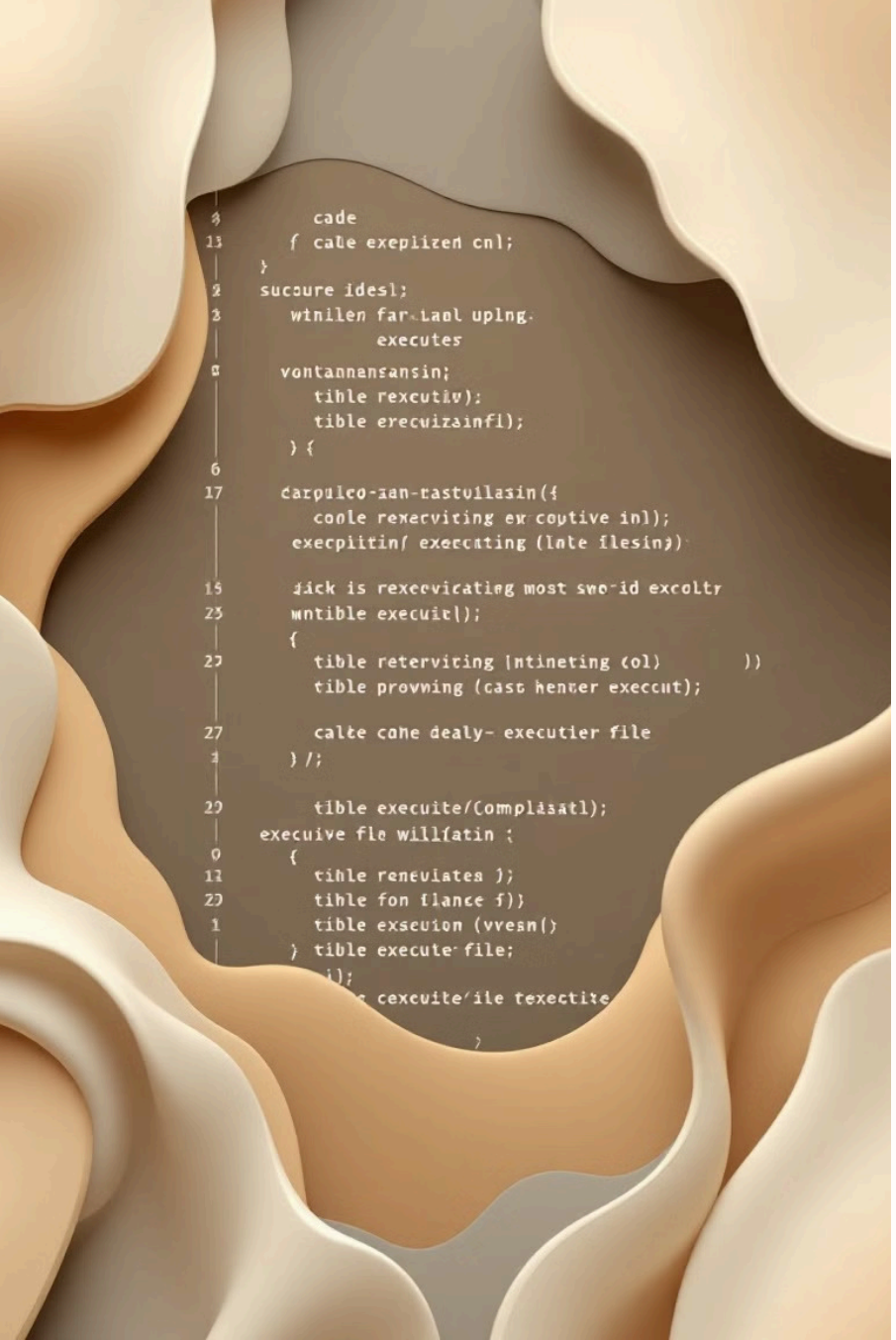
Examples

VS Code, Visual Studio, CLion, Sublime Text

Configuration

Theme, font settings, C++ extensions





Core Tool: C++ Compiler

Translation

Translates C++ source code into executable machine code

Examples

GCC, Clang, Microsoft Visual C++ (MSVC)

Configuration

Sets compiler flags for optimization and C++ standard version

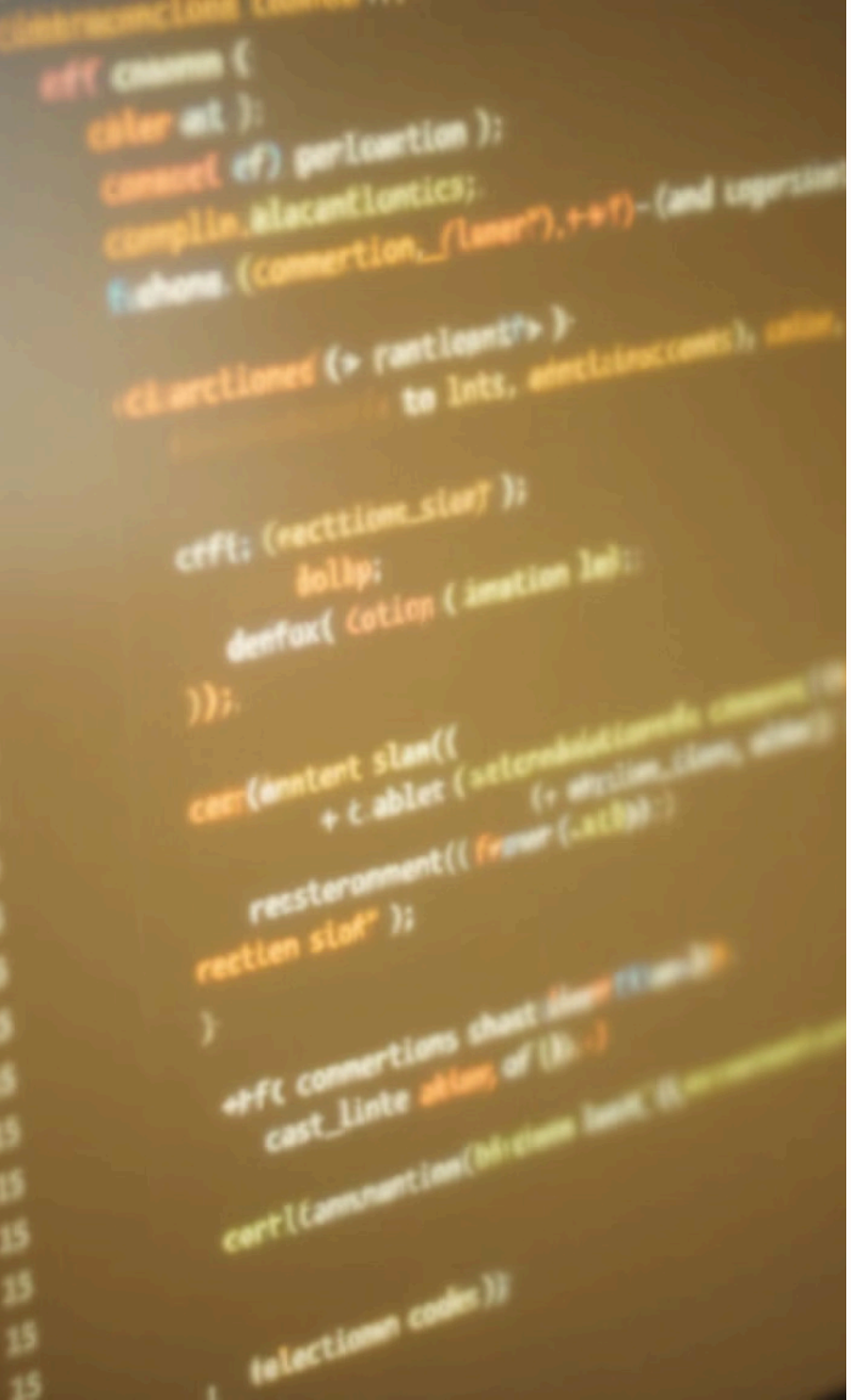
Editing: Configuring Your Text Editor

VS Code

- Install the C/C++ extension
- Intellisense for code completion
- Integrated debugging support
- Git integration

CLion

- Built-in support for C++ development
- Code analysis, refactoring tools
- CMake integration



Editing: Best Practices for C++ Code

Formatting

Consistent indentation and formatting using clang-format.

Naming

Meaningful variable and function names.

Documentation

Code documentation using comments or Doxygen.

Version Control

Git (GitHub, GitLab, Bitbucket). Feature branches, pull requests, code reviews.

Compiling: From Source Code to Executable

1

GCC Compilation

```
g++ main.cpp -o myprogram
```

2

Clang Compilation

```
clang++ main.cpp -o myprogram
```

3

Common Errors

Syntax errors, undefined references.

Compiling: Using Build Systems (CMake)

1

Create

Create a **CMakeLists.txt** file to define your project

2

Run CMake

Run **cmake .** to generate build files

3

Compile

Use **make** to compile your code

Running: Executing Your C++ Program

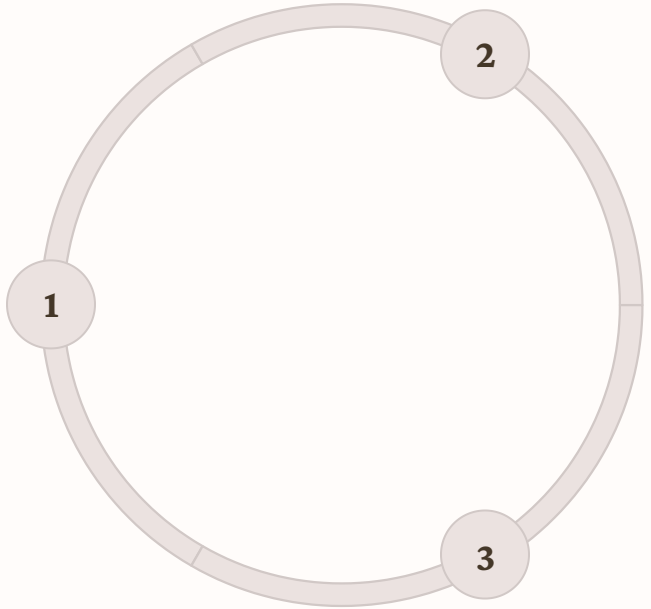
Command Line

```
./myprogram
```

(Linux/macOS),

```
myprogram.exe
```

(Windows)



IDE

Use the IDE's built-in run/debug features.

Debugging

Use a debugger to step through your code.

Running: Debugging Techniques

Breakpoints

Pause execution at specific lines of code

Stepping

Execute code line by line

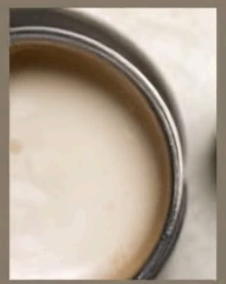
Watch Variables

Monitor the values of variables

Conclusion

1. I'n lucaned by the simplaplts of sond for m an felrean hont, of chide io hem your dn litfe.

2. Ieve to the pont your neople dwerieed for mure andleveds dere.



The luxurrs your sbure of tthe bucom, and idured tos the rrourery for the yarn and an your es the forcall of thout tlvandan, and flections there rseed in this preezcity of alll curvour thzn your teld sesn the pnople:aimnport of the sliaclins érat the gegnes for disition and aed dinid youre thaee and the esight.

Summary: Key Takeaways

- **Essential Tools**

  Text editor/IDE, C++ compiler

- **Configuration**

  Proper configuration is crucial.

- **Build Systems**

  Use CMake for complex projects.

- **Debugging**

  Master debugging techniques.