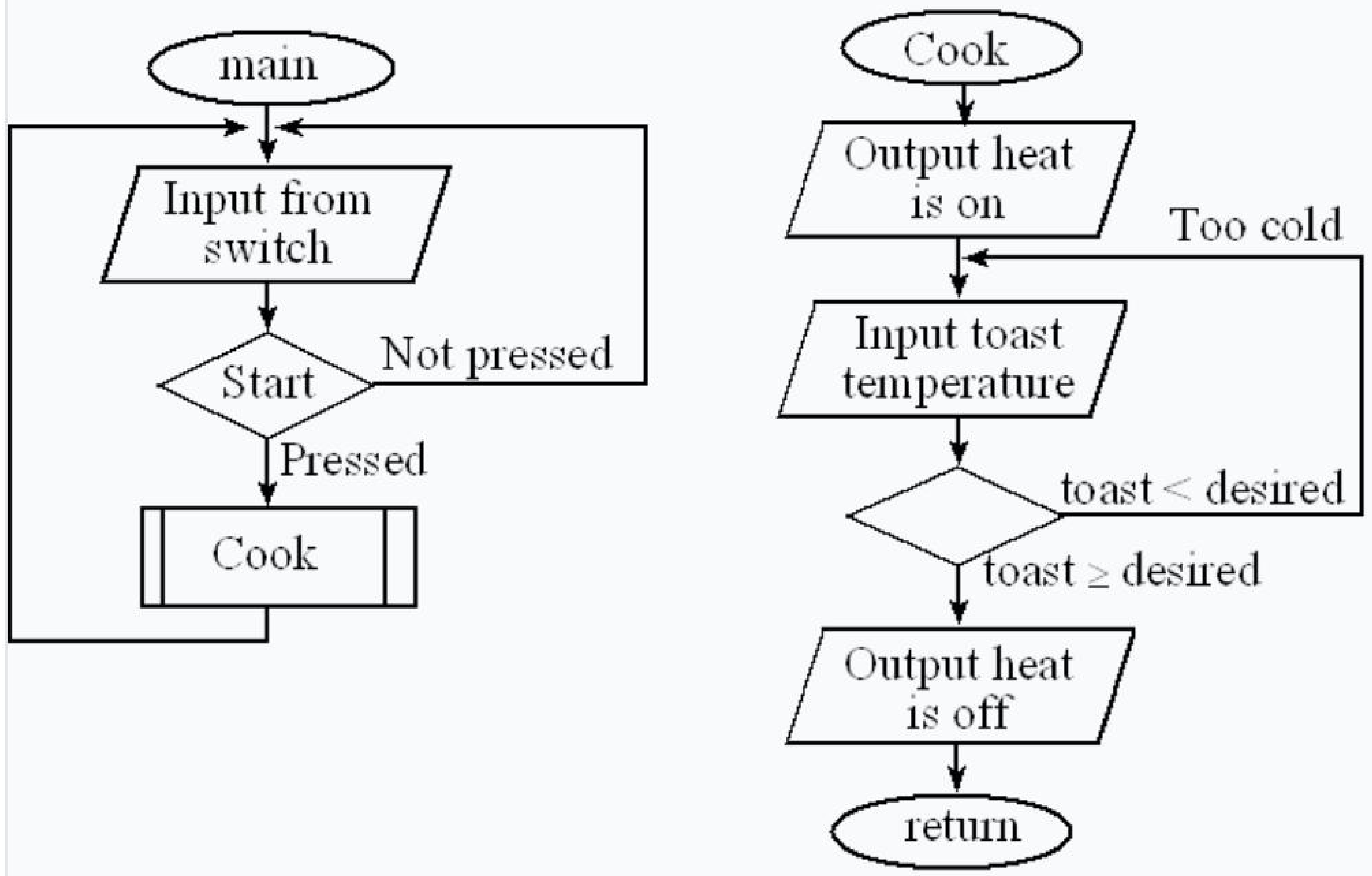
Lecture 1: Concepts and Basics of C++

Key Topics: Programming Paradigms & I/O Streams

Topic 1: Procedural vs. Object-Oriented

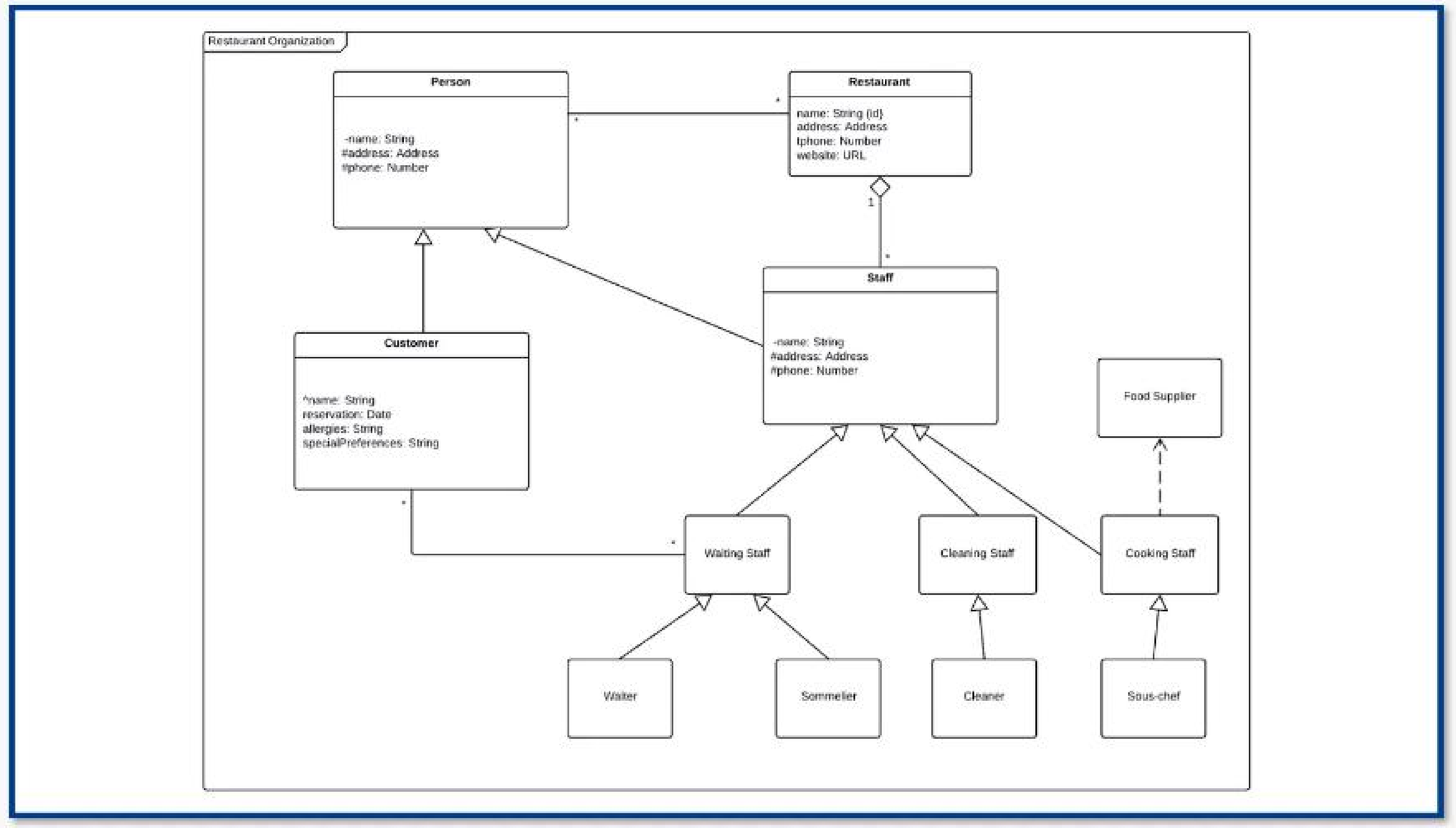
Procedural Programming (POP)

- A language where the program is divided into multiple **functions** (also called procedures or routines).
- It is also known as "function-oriented programming."
- Follows a **Top-Down Approach**, breaking a main task into smaller functions.
- The primary focus is on functions and logic, rather than on data.
- Data is often stored in **global variables**, making it accessible (and changeable) by any function.
- **Examples:** C, Pascal, FORTRAN.



Object-Oriented Programming (OOP)

- A language where the program is divided into multiple **objects** and **classes**.
- Follows a **Bottom-Up Approach**, building complex systems from smaller, self-contained objects.
- The primary focus is on **data**, which is bundled with the functions that operate on it.
- **Key Features:**
  - **Encapsulation:** Bundling data and functions.
  - **Data Hiding:** Restricting direct access to data.
  - **Inheritance:** Reusing code from existing classes.
  - **Polymorphism:** Objects taking many forms.
- **Examples:** C++, Java, Javascript, Python.



IONOS

POP vs. OOP: Key Differences

Feature	Procedural (POP)	Object-Oriented (OOP)
Approach	Top-Down	Bottom-Up
Focus	Functions / Logic	Objects / Data
Program Unit	Functions	Objects & Classes
Data	Exposed (often global)	Encapsulated (hidden & protected)
Examples	C, Pascal, FORTRAN	C++, Java, Python

Topic 2: C++ Input/Output Streams

C++ Input & Output Streams

- **Header:** C++ uses the header file for all standard input and output operations.
- **Library:** This header defines the standard `iostream` library that handles I/O.
- **Stream:** A stream is an abstraction that represents a flow of characters, either from an input device (like a keyboard) or to an output device (like a screen).
- **Standard Objects:** The library defines key global objects to manage this flow.

Standard I/O Stream Objects

std::cin

Standard input object (type `istream`). It reads data from the console/keyboard. It uses the **extraction operator** (>>).

std::cout

Standard output object (type `ostream`). It writes data to the console/screen. It uses the **insertion operator** (<<).

std::cerr & std::clog

Standard error (`cerr`) and log (`clog`) objects. Used for outputting error messages. `cerr` is unbuffered, `clog` is buffered.

Example Program Analysis

The Code

```
#include

int main() {
  int n1, n2;

  std::cout << "Enter two numbers: ";
  std::cin >> n1 >> n2;

  int sum = n1 + n2;

  std::cout << "The sum of " << n1
            << " and " << n2
            << " is " << sum
            << std::endl;

  return 0;
}
```

The Breakdown

- `#include` : Includes the I/O library.
- `int n1, n2;`: Declares integer variables to store input.
- `std::cout`: The standard output object (prints to screen).
- `std::cin`: The standard input object (reads from keyboard).
- << **(Insertion)**: Operator used to "insert" data into the `cout` stream.
- >> **(Extraction)**: Operator used to "extract" data from the `cin` stream into a variable.
- `std::endl`: Inserts a newline character and flushes the output buffer.
- `return 0;`: Indicates the program finished successfully.