

Course synthesis and review

Putting computing together
From information to working systems

Prof. Dr. Mehmet Göktürk · Gebze Technical University

A final review, not a new unit

Explain an idea using a diagram, an example and its limitations.

Trace a small program before running it; justify the result.

Use today's questions to identify which earlier material needs another look.

One system, many computing ideas



Revisit the room sensor: a measurement becomes data, computation, communication and a decision.

The course as a connected whole

Question	Concepts that help answer it
How is information represented?	Bits, encodings, numbers and sampled signals
How does work execute?	Instructions, processors, memory and the operating system
How is a solution designed?	Algorithms, data structures, functions and testing
How do systems cooperate safely?	Networks, files, databases, AI, security and privacy

Different layers answer different questions about the same system.

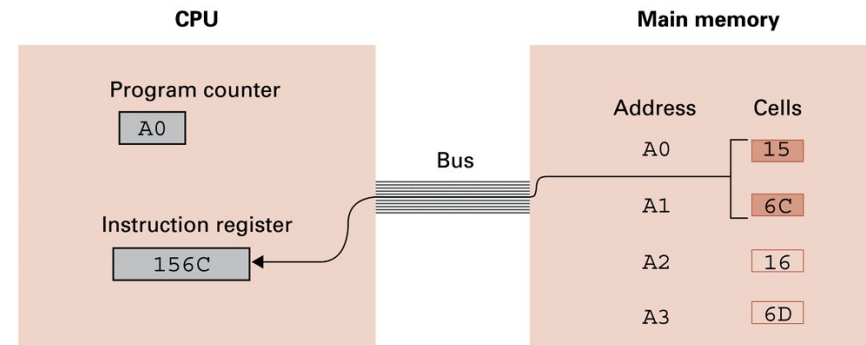
A number needs an interpretation

The bit pattern 01000001 can represent the unsigned number 65 or the character A in ASCII.

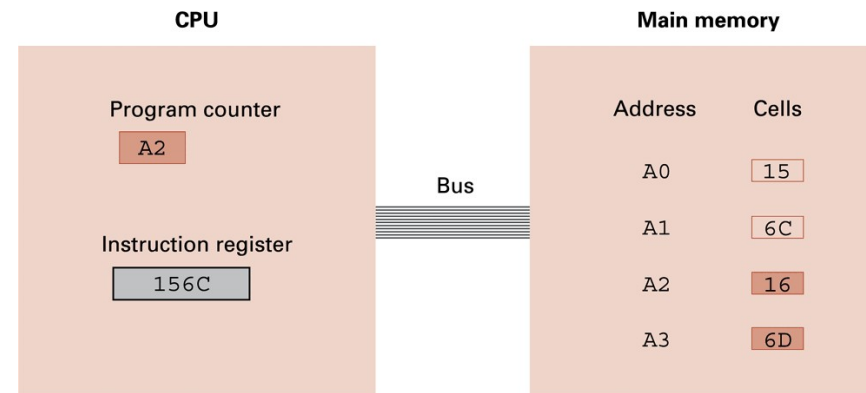
A raw sensor value is not automatically a temperature: a conversion and units give it meaning.

Ask what the representation permits, what it loses and which interpretation the program assumes.

From an instruction to an action



a. At the beginning of the fetch step the instruction starting at address A0 is retrieved from memory and placed in the instruction register.



b. Then the program counter is incremented so that it points to the next instruction.

Follow the original fetch diagram. What is an instruction address, and what is the instruction stored there?

What the processor does—and does not know

The processor executes instructions and updates machine state.

Variable names and a programmer's intention are not a substitute for the operations actually executed.

A program can run successfully and still compute the wrong answer to the original problem.

Revisit the mechanic shop



Several unfinished jobs can share one mechanic.

Two mechanics can work at the same time.

A job waiting for a part is not ready for CPU time.

One mechanic represents a core; each car's repair job represents a process.

Explain these distinctions

Pair	What separates them?
Program / process	Stored instructions / an executing instance with state
Ready / blocked	Waiting for CPU time / waiting for an event
Concurrency / parallelism	Overlapping progress / simultaneous execution
Function / process	A callable part of a program / an execution context

An analogy is useful when you can map it back to the technical terms.

Source code is not one execution method

Stage or mechanism	What happens?
Compilation	Translate one representation into another before its execution
Interpretation	An interpreter executes the operations of a representation
Bytecode	An intermediate instruction representation; may be interpreted or JIT compiled
Ordinary CPython	Python source becomes bytecode, which the Python runtime executes

A language can have multiple implementations. “Compiled” and “interpreted” are not exclusive language labels.

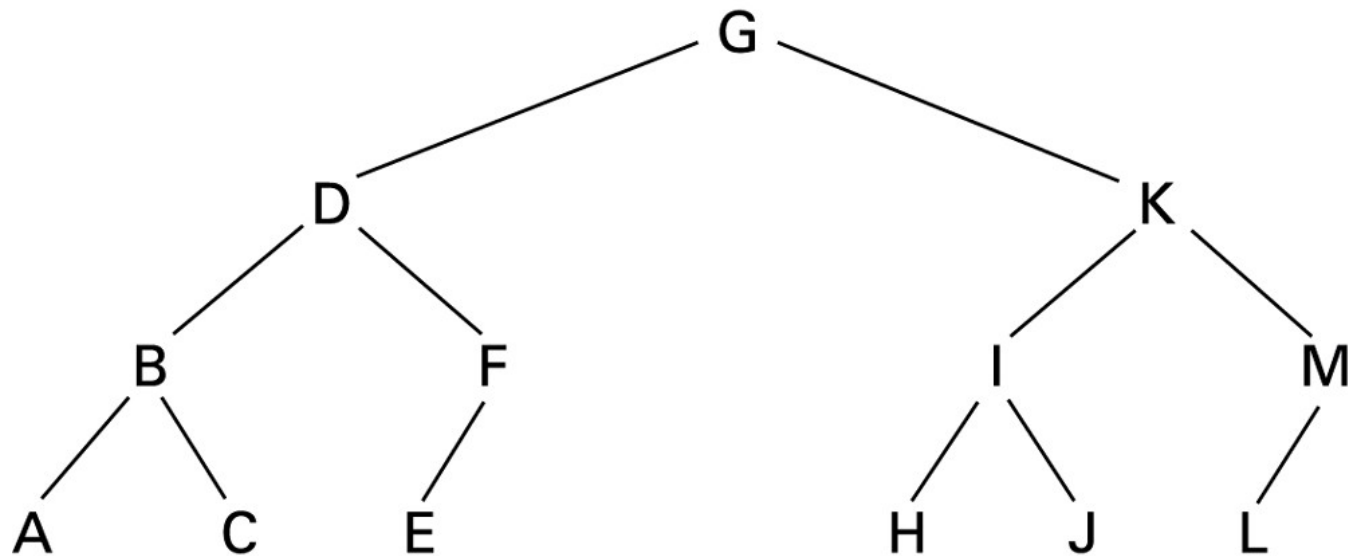
An algorithm needs a reason to work

State the inputs, the desired result and the conditions under which the method applies.

Identify sequence, selection and repetition; a flowchart makes paths and stopping conditions visible.

Then argue correctness and termination before comparing costs.

Why can a search discard half?



Compare with checking A through J in order.

The tree illustrates comparisons; an array uses indices.

A badly shaped search tree can still be slow.

Use the ordered tree to find J: $G \rightarrow K \rightarrow I \rightarrow J$. Ordering justifies each eliminated region.

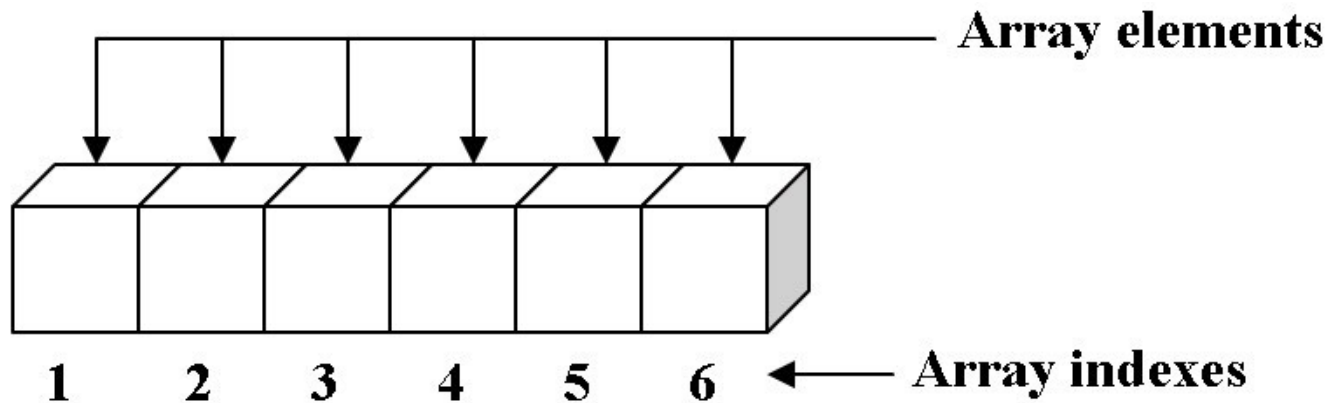
Efficiency is a question about growth

A linear scan may inspect every element. Binary search on a sorted array repeatedly halves the candidates.

A faster search can require earlier work: sorting, building an index or maintaining an ordered structure.

Compare total work for the actual use: one lookup and a million lookups can justify different choices.

Storage structure and access rules



One-dimensional array with six elements

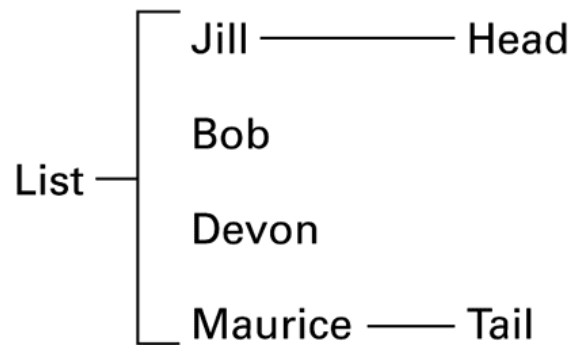
An index identifies a position.

The element stores a value.

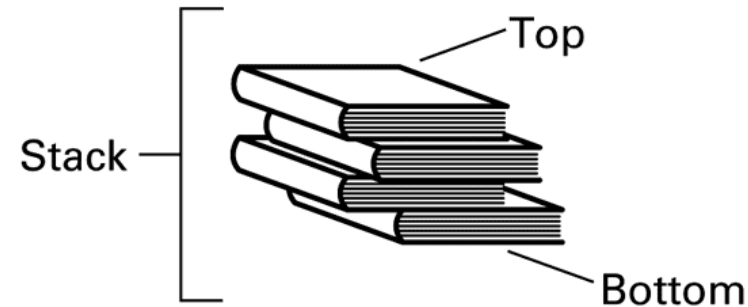
The array length sets the valid range.

The source drawing labels positions 1–6. In C, these six array elements have indices 0–5.

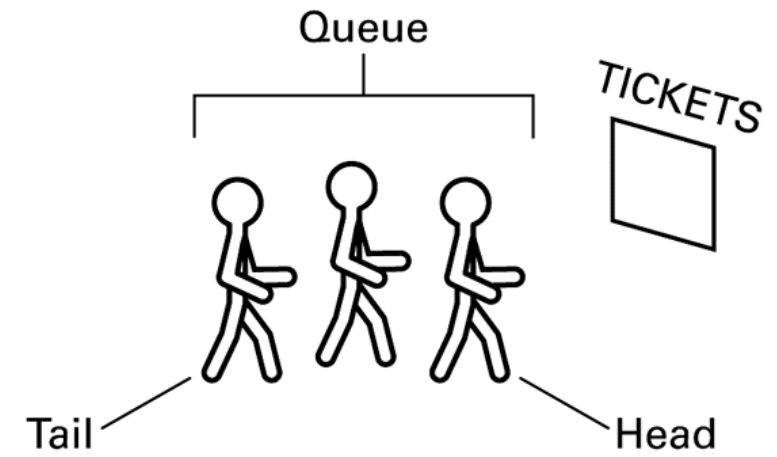
Data structures express useful rules



a. A list of names



b. A stack of books



c. A queue of people

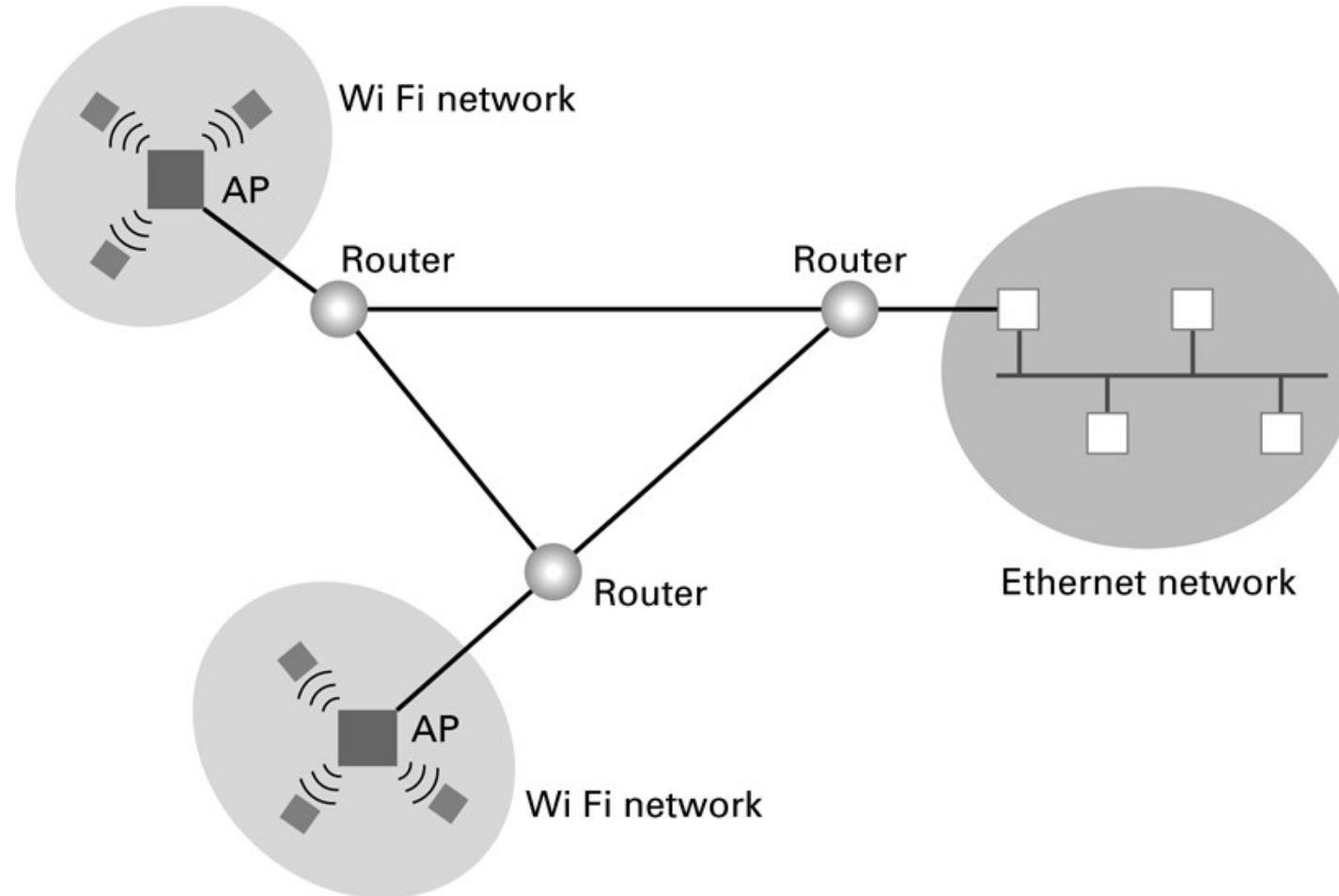
Compare stack and queue access rules. A general list does not impose either rule.

Choose a structure for a reason

Need	A reasonable starting point
Visit fixed samples by position	Array: direct indexing and contiguous elements
Undo the most recent action	Stack: last in, first out
Serve requests in arrival order	Queue: first in, first out
Relink after a known node	Linked list: update links; finding the node still costs work

A structure does not make every operation fast. Match its operations to the problem.

Communication crosses boundaries



A router forwards traffic between networks. Explain the difference between an endpoint and an intermediate router.

The Internet, Web and storage

Concept	Role
Internet	Interconnected networks carrying IP traffic
Web	Resources accessed using Web protocols such as HTTP
Files and databases	Files store data; a DBMS adds queries, constraints and transactions
Protocol	Agreed message formats and interaction rules

Sending bytes does not establish that their meaning, permissions or content are correct.

AI: prediction needs evaluation

Training adjusts a model using examples; inference uses the resulting model on an input.

A good training score alone does not show how the system performs on new cases.

Ask which errors matter, whose data is represented and how an output will be checked.

Security is part of system design

Question	Relevant objective or mechanism
Who can read the history?	Confidentiality and access control
Can someone alter a reading?	Integrity checks and authenticated communication
What if the service stops?	Availability, recovery and handling failures
Should we collect this at all?	Privacy: purpose, access and retention

Encryption helps protect data; it does not validate a sensor reading or make every user authorized.

A working demo is only a beginning

A requirement states what should happen, including important failure cases.

Tests compare observed behavior with an independently chosen expected result.

Good design also considers readability, boundaries, maintainability and the people affected.

Predict, explain, then check

Small C examples revisit earlier ideas.

The goal is reasoning, not a new coding assignment.

Trace a loop and a decision

```
int a[4] = {3, 8, 2, 7};  
int total = 0;  
for (int i = 0; i < 4; i++) {  
    if (a[i] >= 5)  
        total += a[i];  
}  
printf("%d\n", total);
```

Which indices are visited?

When does total change?

Predict the output before the next slide.

Trace: preserve what has been counted

i	a[i]	total after iteration
0	3	0
1	8	8
2	2	8
3	7	15

Output: 15. Before each iteration, total is the sum of qualifying values already visited.

Find the boundary error

```
int a[4] = {3, 8, 2, 7};
```

```
/* Incorrect: diagnose, do not run. */  
for (int i = 0; i <= 4; i++)  
    printf("%d\n", a[i]);
```

Valid indices are 0–3.

At $i = 4$, $a[i]$ is outside the array.

Fix: $i < 4$. There is no defined output for the faulty program.

What is copied into a function?

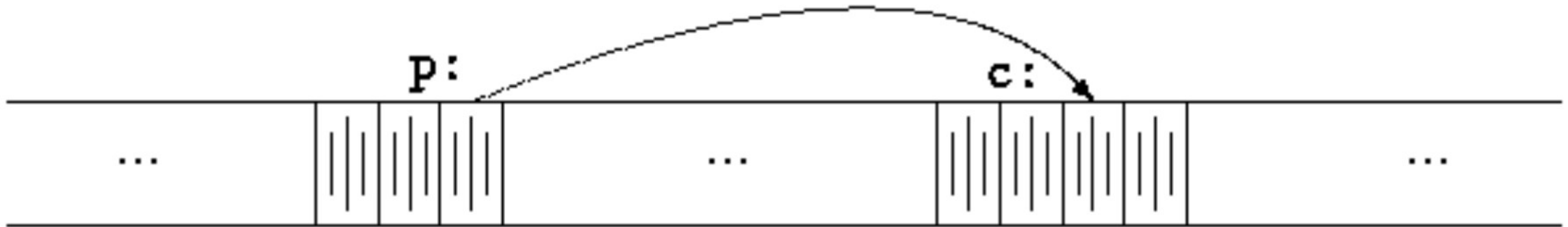
```
void bump(int n) {  
    n++;  
}  
void bump_at(int *p) {  
    (*p)++;  
}  
/* Inside main: */  
int x = 5;  
bump(x);  
bump_at(&x);
```

After bump(x), x is still 5.

bump_at receives a copy of the address.

Writing through that address changes x to 6.

Draw the address before following it



`p = &c;`

Identify the pointer object, the address it contains and the object reached by dereferencing it.

Integer division changes the result

```
int sum = 7;  
int count = 2;
```

```
double a = sum / count;  
double b = (double)sum / count;
```

a is 3.0: integer division happens first.

b is 3.5: conversion occurs before division.

A general averaging function must also handle count == 0.

Two dimensions, two indices

```
int a[2][3] = {{1, 2, 3},  
              {4, 5, 6}};  
  
int total = 0;  
for (int r = 0; r < 2; r++)  
    for (int c = 0; c < 3; c++)  
        total += a[r][c];
```

Six elements are visited.

The inner loop restarts for each row.

Output if printed: 21. What changes if c starts at 1?

Reason about the changed traversal

With c starting at 1, column 0 is skipped in each row.

The values visited are 2, 3, 5 and 6; the total is 16.

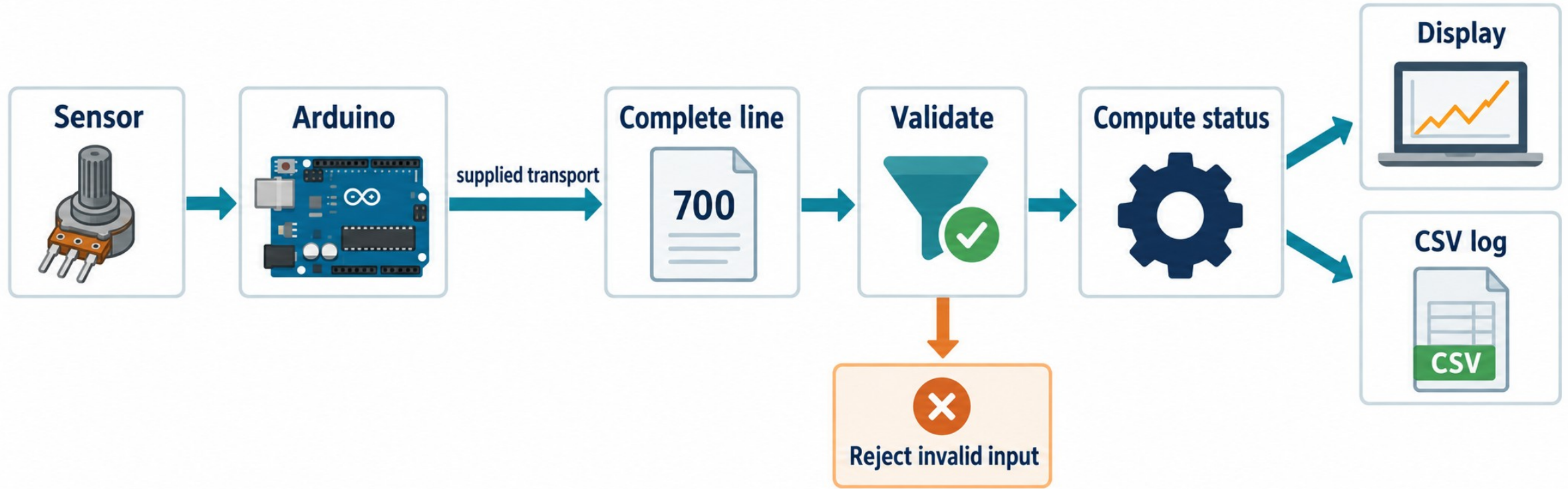
Changing a loop bound changes which data belongs to the computation. Check it against the requirement.

Test a threshold at its boundaries

Input	Rule: high when raw ≥ 600	Expected
599	Just below threshold	Not high
600	At the threshold	High
601	Just above threshold	High
Invalid text	Reject before applying the rule	No accepted sample

Threshold behavior and input validity are separate decisions.

Review the complete data path



Reused Week 13 conceptual diagram: follow one valid reading and one rejected line.

Localize a failure before changing code

The display is wrong. First compare the raw reading, transmitted line, parsed number and computed result.

If the saved file is empty, check whether the program opened the intended path and whether writing and closing succeeded.

Change one assumption at a time and repeat a small test with a known expected result.

Discussion: the system works in the lab

The sensor works on one desk but fails in another room. Which physical assumptions and boundary cases would you inspect?

The network is encrypted but a shared account can view every room. Which access and privacy questions remain?

A classifier is accurate overall but misses the rare dangerous cases. Which results should be examined separately?

Discussion: possible lines of reasoning

Physical: wiring, power, calibration, range and environmental conditions; compare known inputs before blaming the algorithm.

Access: who needs which records, what permissions are granted and how long the data should remain available.

Learning: inspect false negatives and false positives for the relevant cases, then evaluate representative new data.

How to revisit an uncertain topic

Return to its diagram and explain every arrow, state, index or message.

Work one small example by hand; include an empty, boundary or failure case when relevant.

Use the earlier lecture and reading to resolve the uncertainty, then explain the answer in your own words.

Connect the ideas

What can you explain now?

Which assumption would you check first?

Conceptual review: current CSE101 weekly slides and the published readings for Weeks 1–4.

Brookshear and Brylow, Computer Science: An Overview, 12th Global Edition: revisit the chapters cited in each weekly deck.

C review: earlier worked examples and the GNU C Language Introduction and Reference Manual.