



UNIVERSITAS  
GADJAH MADA



LOCALLY ROOTED,  
GLOBALLY RESPECTED

Meeting 3

# Demo

*Fundamentals of Programming*  
*TKU211131*

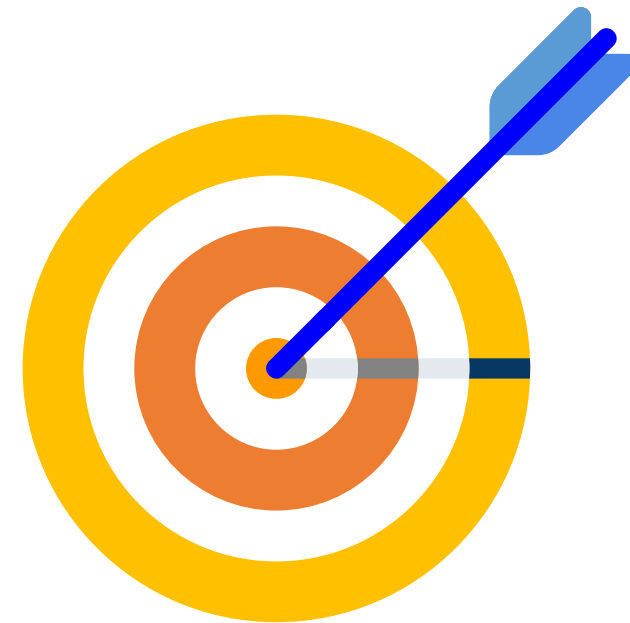
DTETI FT UGM

**Bimo Sunarfri Hantono**  
**[bhe@ugm.ac.id](mailto:bhe@ugm.ac.id)**

ugm.ac.id

# Topics

- **Command Line Interface (CLI)**
- **Demo session: VS Code (editor) + CLI.**
  - **Implement Computational Thinking using:**
    - **Elements of a C++ program**
    - **Data types and operators that C++ can process**
    - **Basic operations in fundamental programming:**
    - **Performing calculations**
    - **Inputting data**
    - **Displaying results**



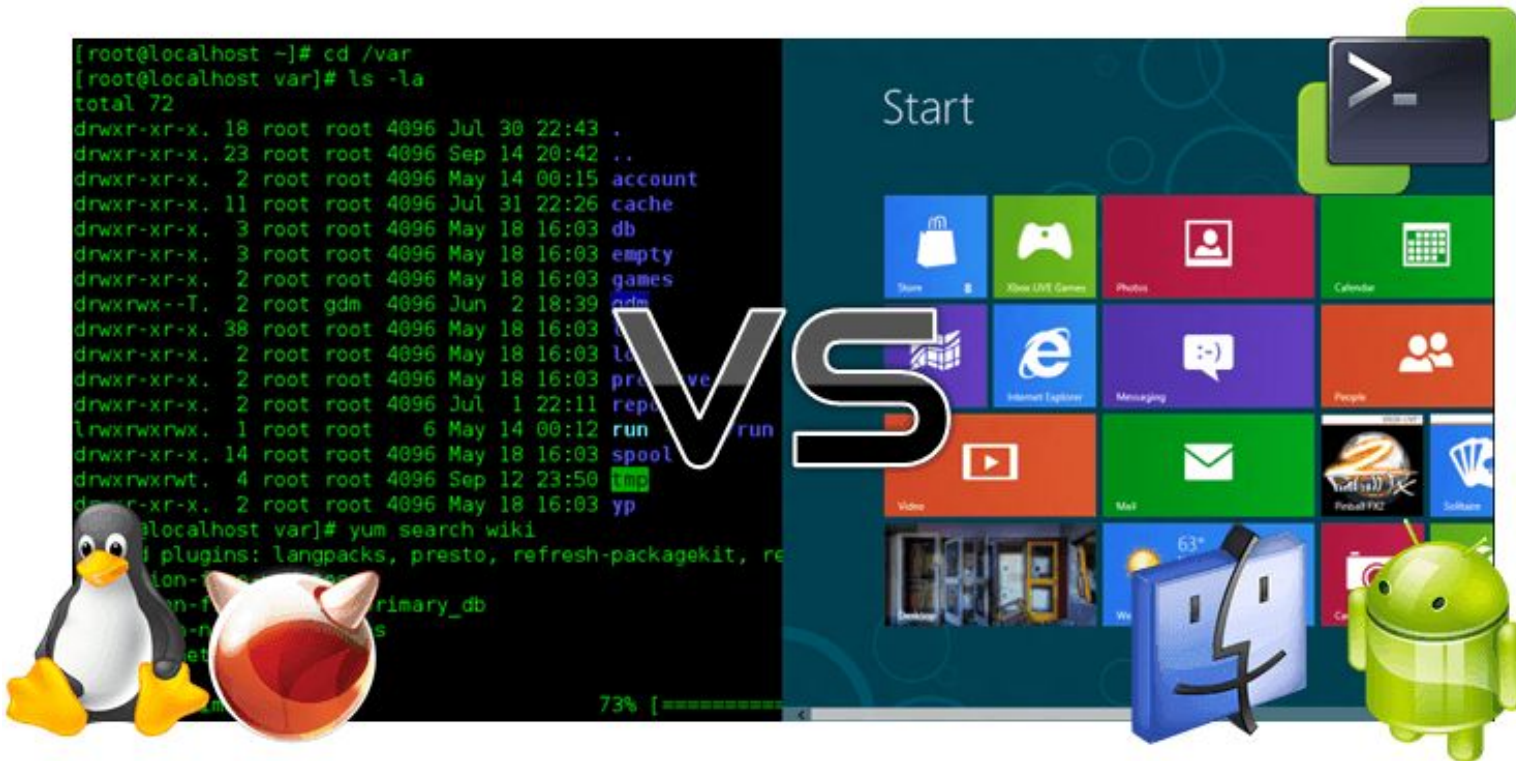


UNIVERSITAS  
GADJAH MADA

# Command Line Interface

# What is CLI?

A text-based way to interact with the computer





# Example of CLI in Different OS

```
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Users\Ted> gci

Directory: C:\Users\Ted

Mode                LastWriteTime         Length Name
----                -
d----- 4/1/2016 9:12 AM             .oracle_jre_usage
d----- 3/11/2016 5:19 PM             .ssh
d-r----- 3/3/2016 3:47 AM             Contacts
d-r----- 4/1/2016 10:19 PM            Desktop
d-r----- 3/13/2016 11:15 AM            Documents
d-r----- 4/3/2016 11:00 PM            Downloads
d-r----- 3/3/2016 3:47 AM             Favorites
d-r----- 3/30/2016 10:35 AM           ForReal
d-r----- 4/1/2016 3:16 PM            funWithPowerShell
d-r----- 3/3/2016 3:47 AM             Links
d-r----- 3/3/2016 3:47 AM             Music
dar----- 3/10/2016 5:51 PM            OneDrive
d-r----- 4/4/2016 3:03 PM             Pictures
d-r----- 3/3/2016 3:47 AM             Saved Games
d-r----- 3/3/2016 3:47 AM             Searches
d----- 2/20/2016 12:15 PM            temp
d----- 8/30/2015 7:48 PM             Tracing
d-r----- 3/3/2016 3:47 AM             Videos
d----- 4/3/2016 7:34 AM             wget-activehistory
-a----- 4/1/2016 9:42 PM             1505 .bash_history
-a----- 4/3/2016 1:13 PM             155 .gitconfig
-a----- 2/10/2016 11:29 AM           69 .node_repl_history
-a----- 2/25/2016 10:45 AM           599 .viminfo

PS C:\Users\Ted>
```

```
Terminal — top — Basic — 80x24

Processes: 421 total, 3 running, 418 sleeping, 1383 threads          16:01:01
Load Avg: 1.49, 1.20, 1.14 CPU usage: 0.48% user, 0.84% sys, 98.67% idle
SharedLibs
MemRegions
PhysMem: 1
VM: 2461G
Networks:
Disks: 246

PID COMM
147 Wind
3724 top
3725 top
3723 top
2947 Term
0 kern
260 mtr
1323 com
81 powe
329 Touc
2734 Safa
2881 app
238 nsur
110 cont
```

```
Terminal — top — Red Sands — 80x24

Processes: 421 total, 3 running, 418 sleeping, 1383 threads          16:01:01
Load Avg: 1.49, 1.20, 1.14 CPU usage: 0.48% user, 0.84% sys, 98.67% idle
SharedLibs
MemRegions
PhysMem: 1
VM: 2461G
Networks:
Disks: 246

PID COMM
147 Wind
3724 top
3725 top
3723 top
2947 Term
0 kern
260 mtr
1323 com
81 powe
329 Touc
2734 Safa
2881 app
238 nsur
110 cont
```

```
Terminal — top — Pro — 94x24

Processes: 421 total, 2 running, 419 sleeping, 1383 threads          16:01:01
Load Avg: 1.49, 1.20, 1.14 CPU usage: 0.54% user, 0.96% sys, 98.49% idle
SharedLibs: 515M resident, 86M data, 305M linkedit.
MemRegions: 98214 total, 2353M resident, 242M private, 1558M shared.
PhysMem: 14G used (2687M wired), 18G unused.
VM: 2461G vsz, 2277M framework vsz, 0(0) swapins, 0(0) swapouts.
Networks: packets: 168275/66M in, 191369/56M out.
Disks: 246661/3854M read, 146809/3327M written.

PID COMMAND %CPU TIME #TH #WQ #PORT MEM PURG CMPR P
147 WindowServer 6.7 15:40.52 14 5 1731 746M+ 31M- 0B 1:
3725 top 3.2 00:17.37 1 0 25 3864K 0B 0B 3:
3724 top 3.1 00:17.41 1 0 23 3824K 0B 0B 3:
3723 top 3.0 00:17.39 1/1 0 35 4808K 0B 0B 3:
2947 Terminal 1.7 00:28.08 6 1 402 106M+ 6916K 0B 2:
0 kernel_task 1.7 03:48.16 270/16 0 0 55M- 0B 0B 0
2734 Safa 260 mtr 1.2 01:46.76 2 1 67 5728K 0B 0B 2:
2881 app 3123 com.apple.Ap 0.9 00:37.87 3 2 77 1224K 0B 0B 3:
238 nsur 81 powe 0.1 00:07.37 3 2 137 2080K 0B 0B 8:
329 TouchBarServ 0.0 01:11.82 4 1 325 23M 3200K 0B 3:
2734 SafariBookma 0.0 00:05.08 5 3 72 4788K 12K 0B 2:
2881 appstoreagen 0.0 00:01.49 4 2 123 6236K 208K 0B 2:
141 AirPlayXPCh 0.0 00:03.40 6 2 172 2696K 0B 0B 1:
238 nsur 0.0 00:03.44 6 3 101+ 3000K+ 0B 0B 2:

chinmay29hub@UbuntuMint: ~$ neofetch

chinmay29hub@UbuntuMint.com
OS: Ubuntu 23.04 x86_64
Host: HP Pavilion Gaming Laptop 15-ec2xxx
Kernel: 6.2.0-24-generic
Uptime: 7 hours, 9 mins
Packages: 2664 (dpkg), 26 (snap)
Shell: bash 5.2.15
Resolution: 1920x1080
Terminal: /dev/pts/1
CPU: AMD Ryzen 5 5600H with Radeon Graphics (12) @ 3.300GHz
GPU: AMD ATI Radeon Vega Series / Radeon Vega Mobile Series
GPU: NVIDIA 01:00.0 NVIDIA Corporation TU117M
Memory: 4511MiB / 7267MiB
```

Aspect	CLI (Command Line Interface)	GUI (Graphical User Interface)
Interaction	Text commands typed by user	Icons, menus, and buttons
Ease of Use	Steeper learning curve (need to memorize commands)	Intuitive, easy for beginners
Speed	Very fast for experienced users	Slower due to multiple clicks
Resource Usage	Low (lightweight)	High (needs more CPU & memory)
Automation	Easy to script & automate	Limited automation
Flexibility	High (fine-grained control)	Moderate (depends on GUI features)
Examples	Terminal, PowerShell, Bash	Windows Explorer, macOS Finder

# Example of CLI Command

- `cd` → change directory
- `ls / dir` → list files
- `g++ file.cpp -o file.exe` → compile C++ program
- `./file.exe` → run program





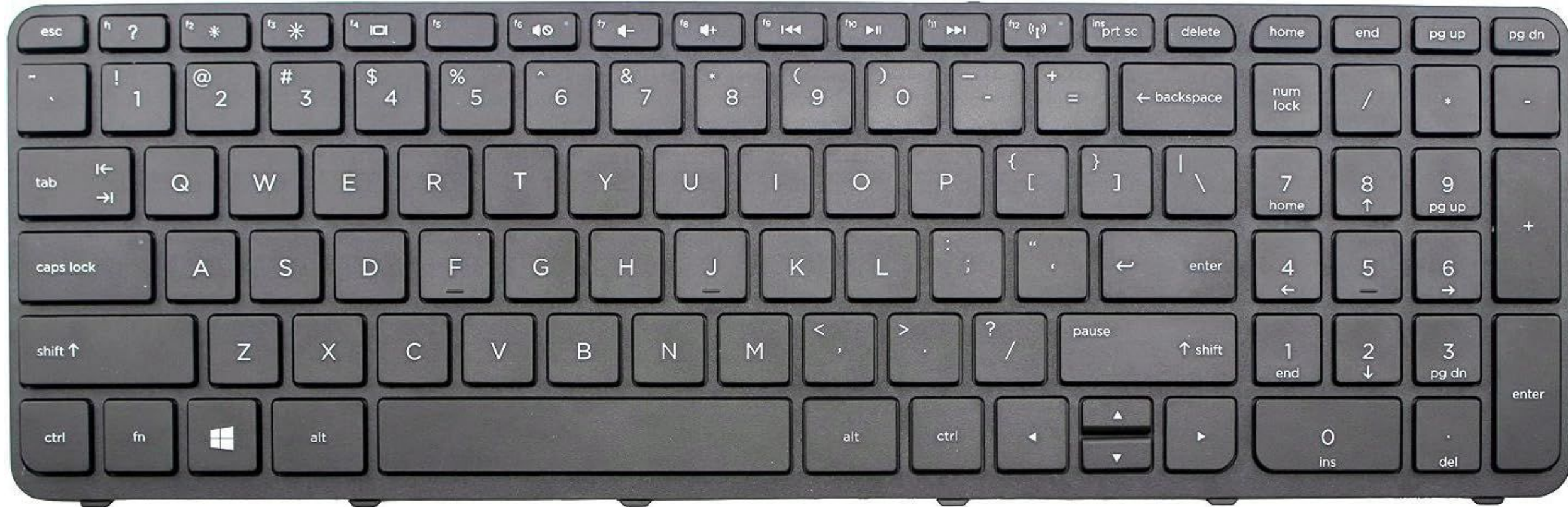
## Main tools in **Visual Studio Code**:

- Explorer → see project files
- Editor → write code
- Terminal → compile & run code
- Extensions → add support for C++

# Useful VS Code Shortcuts

- **Ctrl + N** → new file
- **Ctrl + S** → save file
- **Ctrl + /** → comment/uncomment line
- **Ctrl + `** → open terminal
- **Ctrl + Shift + B** → build program
- **F5** → run & debug program

# Keyboard



# VS Code Demo





UNIVERSITAS  
GADJAH MADA

# Demo Session

# Case: Car Breakdown with Low Fuel

## Problem Definition

- **Problem:** Car stops in the middle of the road; fuel is almost empty.
- **Goal:** Get the car running again or select a safe alternative.
- **Constraints:** Time, roadside safety, limited tools.

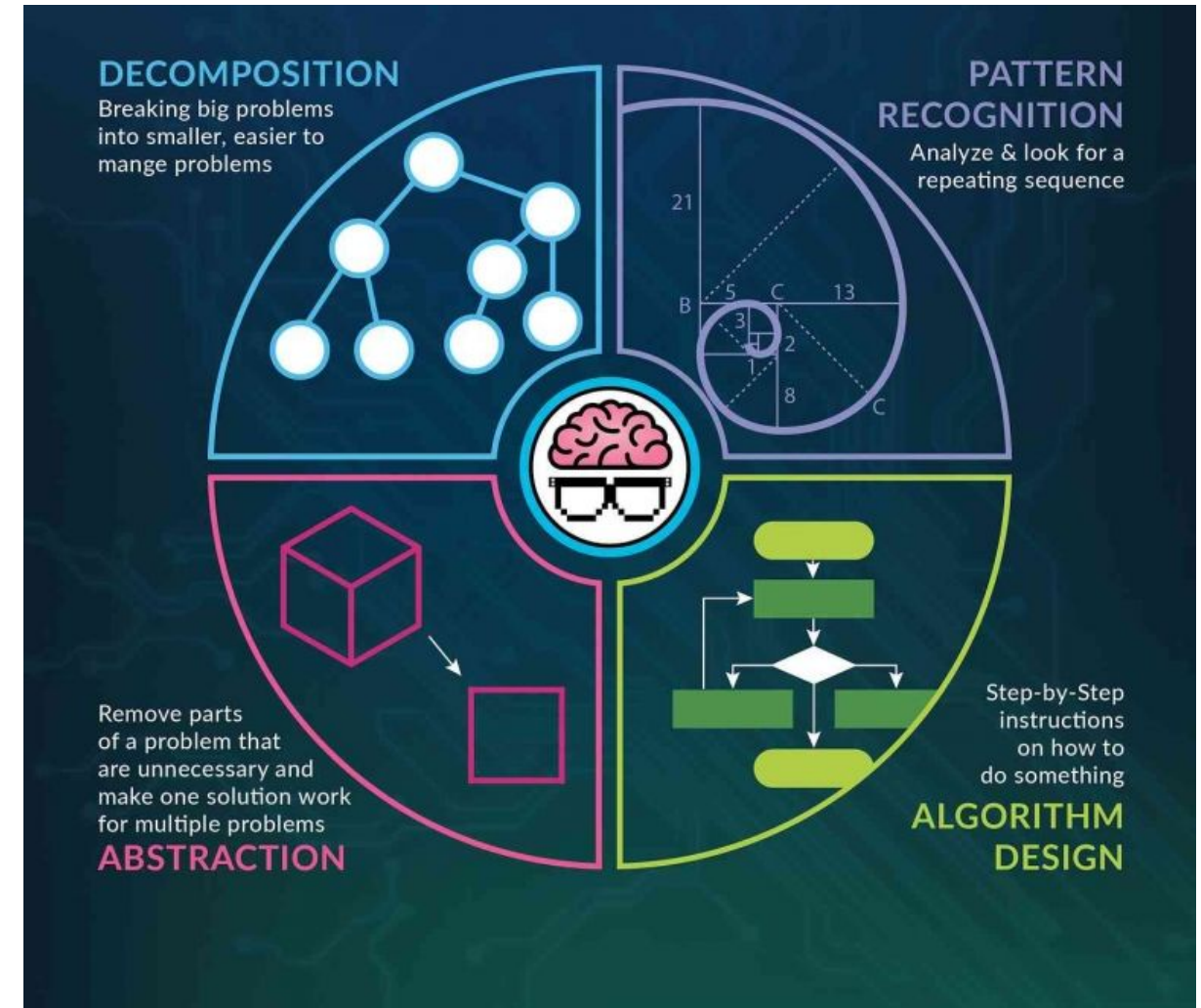




# Computational Thinking Steps

## CT pillars:

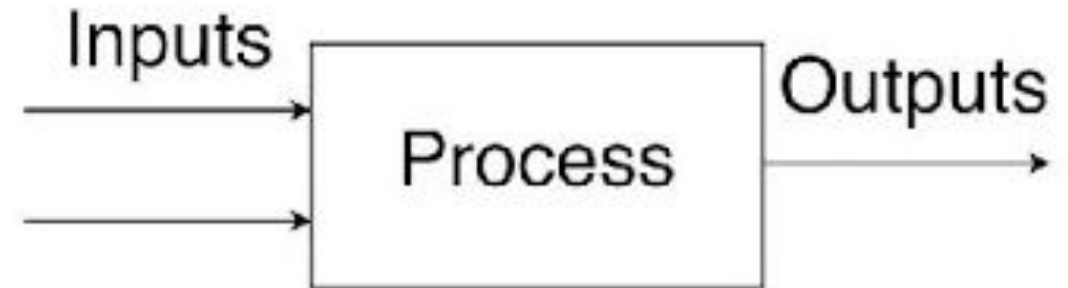
1. **Decomposition** – break into checks & actions
2. **Pattern Recognition** – common car-stop causes
3. **Abstraction** – focus on fuel/battery essentials
4. **Algorithm Design** – clear, testable steps



# IPO Diagram (Input – Process – Output)

## Input

- **fuel\_level** (0 = empty, 1 = low, 2 = ok)
- **battery\_status** (0 = weak, 1 = ok)
- **has\_spare\_battery** (true/false)



## Process

1. Ensure safety (hazard lights)
2. If fuel > 0 and (battery weak or spare available) → replace battery
3. Try starting engine; if fail, recheck or call assistance
4. If fuel == empty → refuel/assistance

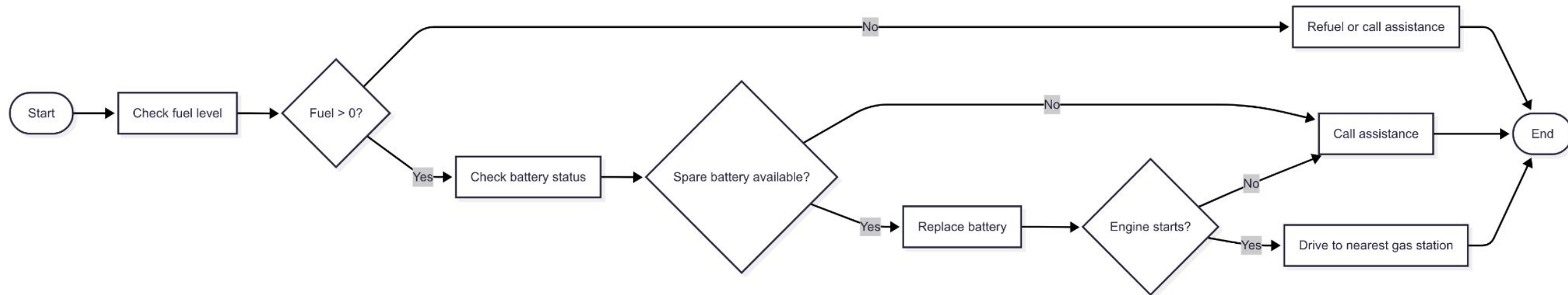
## Output

- Decision & action: **drive to gas station OR call assistance**

**Definition:** A simplified, plain-language description of the steps in an algorithm, not tied to any programming language.

```
IF fuel > 0 AND spare battery available
    Replace battery
    IF engine starts
        Drive to gas station
    ELSE
        Call assistance
ELSE
    Refuel or call assistance
```

# Algorithm Design: Flowchart



## Elements of a C++ Program

- Preprocessor: `#include <iostream>`
- Namespace: `using namespace std;`
- Entry point: `int main() { ... }`

```
#include <iostream>
using namespace std;

int main() {
    // program starts here
    return 0;
}
```

## Data Types and Operators that C++ Can Process

- Data types used: `int`, `bool`
- Operators shown: assignment `=`, comparison `==`, logical `||` (OR)

cpp

```
int fuel_level;      // 0=empty, 1=low, 2=ok
int battery_status;  // 0=weak, 1=ok
bool has_spare_battery; // true/false
```



## Basic Operations in Fundamental Programming

- Read → Process → Decide
- Illustrate control flow with `if` / `else`

```
// basic decision structure
if (fuel_level == 0) {
    // handle empty fuel
} else {
    // handle non-empty fuel
    if (battery_status == 0 || has_spare_battery) {
        // handle weak battery or replace with spare
    }
    // attempt to start engine...
}
```

## Performing Calculations

- Compute/derive a state for decision
- Example: simple start condition

```
// naive demo calculation: assume engine can start if fuel is at least low  
bool engine_starts = (fuel_level >= 1); // derived state for decision
```

## Inputting Data

- Use `cin` to read from keyboard
- Prompt user clearly

```
cout << "Fuel level (0=empty, 1=low, 2=ok): ";  
cin  >> fuel_level;  
  
cout << "Battery status (0=weak, 1=ok): ";  
cin  >> battery_status;  
  
cout << "Has spare battery? (0=no, 1=yes): ";  
cin  >> has_spare_battery;
```

## Displaying Results

- Use `cout` to display actions/outcomes
- Print clear guidance

```
cout << "\n-- Actions --\n";
cout << "Turn on hazard lights and place safety triangle.\n";

if (fuel_level == 0) {
    cout << "Fuel empty. Refuel or call assistance.\n";
} else {
    if (battery_status == 0 || has_spare_battery) {
        cout << "Replace or secure battery connections.\n";
    }
    cout << "Attempting to start engine...\n";
    bool engine_starts = (fuel_level >= 1); // from previous slide
    if (engine_starts) {
        cout << "Engine started. Drive carefully to the nearest gas station.\n";
    } else {
        cout << "Engine failed to start. Call assistance.\n";
    }
}
}
```

- Windows
  - `g++ demo_car.cpp -o demo_car.exe`
  - `demo_car.exe`
- MacOS / Linux
  - `g++ demo_car.cpp -o demo_car`
  - `./demo_car`

- Missing ; → check compile errors
- Wrong types → verify variable declarations
- Input confusion → echo inputs back to user
- Logic branches → print checkpoints (e.g., *entered fuel==0 branch*)





UNIVERSITAS  
GADJAH MADA

# Any Question?



UNIVERSITAS  
GADJAH MADA

**Thank You!**  
**See you, next week,**  
**stay safe!**

