■■ Slide-by-Slide Narration Script

Visual Cue	Narration
Slide 1 Title: Title: “e-Lecture on C Programming”	Welcome! Grab a coffee—today we’re touring C, the language that still runs the world.
Slide 2 Title: Photo: Dennis Ritchie at Bell Labs	Meet Dennis Ritchie. In 1972 he sketched C on a blackboard to build UNIX.
Slide 3 Title: Terminal scrolls of UNIX kernel	That same kernel you’re looking at? Still 80 % C, half a century later.
Slide 4 Title: Bullets: “Robust, fast, portable”	Three magic words: fast like a race car, tough like a tank, travels like a passport.
Slide 5 Title: Side-by-side: Assembly vs C	Left: cryptic MOV AX, BX. Right: friendly printf. Same speed, less pain.
Slide 6 Title: World map dotted with C logos	Write once, compile anywhere—from your phone to a Mars rover.
Slide 7 Title: Puzzle pieces snap into main()	Think Lego: small functions click together to build giant programs.
Slide 8 Title: Layered diagram: low → mid → high	C sits in the comfy middle: close to the metal, readable like English.
Slide 9 Title: Flowchart: problem → modules → functions	Big scary task? Slice it into bite-sized functions—C keeps the pieces tidy.
Slide 10 Title: Stopwatch over compiled code	Hit compile and blink—you’ll miss it. Your program’s ready to run.
Slide 11 Title: Editor shows malloc(), calloc()	Need memory on the fly? Ask malloc for a chunk, calloc for a clean one.
Slide 12 Title: Recursion spiral animation	A function that calls itself—like mirrors facing mirrors—solves factorials in style.
Slide 13 Title: Letters, digits, symbols float in	Before we write, let’s meet the alphabet C understands: A-Z, 0-9, and friends.
Slide 14 Title: Binary rain of 0s and 1s	Machines speak only 0 and 1—powerful, but imagine writing War and Peace in Morse.

Slide 15 Title: Assembly snippet: MOV AX, BX	Assembly is better, but still tied to one chip. C breaks those chains.
Slide 16 Title: High-level code: printf("Hello");	With C you just say “Hello” and the compiler worries about the bits.
Slide 17 Title: Compiler flowchart: source → object	The compiler reads your whole file, checks grammar, and bakes a fast binary.
Slide 18 Title: Interpreter stepping line by line	Interpreters are patient teachers—great for learning, slow for shipping.
Slide 19 Title: Comment: /* Compute area */	Comments are sticky notes for humans; the compiler politely ignores them.
Slide 20 Title: Preprocessor: #include	This line grabs the standard I/O toolbox so we can print and read.
Slide 21 Title: Skeleton C program on screen	Every C program starts with #includes, then main(), then your story.
Slide 22 Title: Zoom into main() highlighted	main() is the front door—execution always rings here first.
Slide 23 Title: Table: int, float, char…	Pick your container: int for whole numbers, float for decimals, char for letters.
Slide 24 Title: Range table: int –32,768 to 32,767	Know your limits—cross them and your numbers wrap around like a faulty odometer.
Slide 25 Title: Keyword collage: if, else, while…	These colored words are reserved—use them as names and the compiler sulks.
Slide 26 Title: Operators chart: +, -, *, /, %	Meet the math crew: plus, minus, times, divide, and modulo for leftovers.
Slide 27 Title: Declaration: int score = 0;	A variable is a labeled box—here, score starts life at zero.
Slide 28 Title: Diamond: condition?	Control statements let your program choose paths—like a GPS recalculating.
Slide 29 Title: Code: if (marks >= 40)	Simple if: if marks are 40 or more, you pass—otherwise, silence.
Slide 30 Title: Code: if…else pass/fail	Add else: low marks print “Fail”, high marks print “Pass”.
Slide 31 Title: Ladder: if…else if…else	Need more choices? Stack else-if steps—only one branch wins.

Slide 32 Title: Example: grade A/B/C/D/F	Five grades, five doors—only one opens based on the score.
Slide 33 Title: Switch-case diagram	Switch is a light switch with many positions—jump straight to the matching bulb.
Slide 34 Title: Loop arrows icon	Loops repeat tasks—like a washing machine cycle—until the socks are clean.
Slide 35 Title: For loop: for(i=0;i<10;i++)	For loop counts for you—start, test, update—all in one tidy breath.
Slide 36 Title: While loop: while(x<5)	While loop checks first, runs second—perfect for cautious programs.
Slide 37 Title: Do-while loop	Do-while runs at least once—great for menus that must appear before asking.
Slide 38 Title: Closing: “Thanks”	That’s our whirlwind tour! Questions? Ping me—I’d love to help.