**AI-Pattern Pack — C Programs**

**Curated AI-inspired pattern programs in C** — 15+ interview-ready .c files, detailed comments, AIthemed README descriptions, and a GitHub-ready folder structure. Maintained by **Sania Anzum**.

# Suggested folder structure

AI\_Patterns\_C/

├──

README.md

└──

patterns/

├──

01\_neural\_connections.c

├──

02\_multilayer\_nn.c

├──

03\_weight\_grid.c

├──

04\_activation\_heatmap.c

├──

05\_decision\_tree.c

├──

06\_data\_pipeline.c

├──

07\_confusion\_matrix.c

├──

08\_feature\_bars.c

├──

09\_gradient\_steps.c

├──

10\_backprop\_flow.c

├──

11\_ai\_brain\_ascii.c

├──

12\_circuit\_chip.c

├──

13\_probability\_dist.c

├──

14\_matrix\_visual.c

└──

15\_flowchart\_layers.c

## patterns/01\_neural\_connections.c

/\*

Neural Connections Pattern

Visualizes a small grid of neurons with horizontal connections.

Purpose: show structured ASCII art resembling neuron layers.

\*/

#include

<

stdio.h

>

int

main()

{

int

layers

=

3

;

// number of nodes per row

for

(

int

r

=

0

;

r

<

layers;

r++)

{

// print nodes with horizontal links

for

(

int

c

=

0

;

c

<

layers;

c++)

{

printf(

"O"

)

;

if

(

c

!=

layers

-

1)

printf(

"---"

)

;

}

printf(

"\n"

)

;

// print crossing slashes to indicate connections between rows

if

(

r

!=

layers

-

1)

{

for

(

int

c

=

0

;

c

<

layers;

c++)

{

if

(

c

%

2

==

0)

printf(

" \\ /"

)

;

else

printf(

" / \");

}

printf(

"\n"

)

;

}

}

return

0

;

}

## patterns/02\_multilayer\_nn.c

/\*

Multi-layer Neural Network Visualization

Prints staggered layers of circles to represent neurons.

Parameters: width controls spacing.

\*/

#include

<

stdio.h

>

int

main()

{

int

layers

=

4

;

// number of rows

for

(

int

i

=

0

;

i

<

layers;

i++)

{

// indent to show layer depth

for

(

int

s

=

0

;

s

<

(

layers

-

i

-

1)

\*

2

;

s++)

printf(

" "

)

;

for

(

int

j

=

0

;

j

<=

i;

j++)

{

printf(

"(O)"

)

;

if

(

j

!=

i)

printf(

" "

)

;

}

printf(

"\n"

)

;

}

return

0

;

}

## patterns/03\_weight\_grid.c

/\*

Weight Grid Visualization

Shows a 5x5 grid of weights with simple float formatting to mimic weight

magnitudes.

\*/

#include

<

stdio.h

>

int

main()

{

int

rows

=

5

,

cols

=

5

;

float

w

=

0.1

f;

for

(

int

i

=

0

;

i

<

rows;

i++)

{

for

(

int

j

=

0

;

j

<

cols;

j++)

{

printf(

"%4.1f "

,

w);

w

+=

0.1

f;

if

(

w

>

1.0f)

w

=

0.1

f;

// wrap

}

printf(

"\n"

)

;

}

return

0

;

}

## patterns/04\_activation\_heatmap.c

/\*

Activation Heatmap (ASCII)

Uses characters to approximate different activation strengths.

\*/

#include

<

stdio.h

>

int

main()

{

int

n

=

9

;

// 9 cells to show a 3x3 heatmap

int

strength[]

=

{1

,2,3,2,4,3,1,2,3};

for

(

int

i

=

0

;

i

<

9

;

i++)

{

int

v

=

strength[i];

char

ch

=

(

v

>=

4)

?

'@'

:

(

v

==

3)

?

'O'

:

(

v

==

2)

?

'+'

:

'.'

;

printf(

" %c "

,

ch);

if

((i+1)

%

3

==

0)

printf(

"\n"

)

;

}

return

0

;

}

Binary Decision Tree ASCII Prints a small tree with '?' as decision nodes and slashes as branches.

\*/

#include <stdio.h> int main() { int depth = 3; for (int i = 0; i < depth; i++) { for (int s = 0; s < depth - i - 1; s++) printf(" ");

**patterns/05\_decision\_tree.c**

/\*

}

for (int j = 0; j < (1 << i); j++) printf("? ");

printf("\n");

if (i != depth - 1) { for (int s = 0; s < depth - i - 1; s++) printf(" ");

for (int j = 0; j < (1 << i); j++) printf("/ \\ ");

printf("\n");

}

return

0

;

}

## patterns/06\_data\_pipeline.c

/\*

Data Pipeline Flow

ASCII art representing Input => Process => Output pipeline.

\*/

#include

<

stdio.h

>

int

main()

{

printf(

"[INPUT] "

)

;

for

(

int

i

=

0

;

i

<

4

;

i++)

printf(

"=>"

)

;

printf(

" [PROCESS] "

)

;

for

(

int

i

=

0

;

i

<

4

;

i++)

printf(

"=>"

)

;

printf(

" [OUTPUT]\n"

)

;

return

0

;

}

## patterns/07\_confusion\_matrix.c

/\*

Confusion Matrix Layout

Prints the TP/FP/FN/TN matrix styled for small datasets.

\*/

#include

<

stdio.h

>

int

main()

{

printf(

"\tPred\_Pos\tPred\_Neg\n"

)

;

printf(

"Actual\_Pos\tTP=50\tFP=5\n"

)

;

printf(

"Actual\_Neg\tFN=3\tTN=42\n"

)

;

return

0

;

}

## patterns/08\_feature\_bars.c

/\*

Feature Importance Bars

Draws horizontal bars using '#' to indicate importance scores.

\*/

#include

<

stdio.h

>

int

main()

{

int

scores[]

=

{5

,

3

,

8

,

6}

;

int

features

=

4

;

for

(

int

i

=

0

;

i

<

features;

i++)

{

printf(

"Feature\_%d: "

,

i+1);

for

(

int

j

=

0

;

j

<

scores[i];

j++)

printf(

"#"

)

;

printf(

" (%d)\n"

,

scores[i]);

}

return

0

;

}

## patterns/09\_gradient\_steps.c

/\*

Gradient Descent Step Indicators

Shows a simple progression of arrows to indicate iterative steps.

\*/

#include

<

stdio.h

>

int

main()

{

int

steps

=

6

;

for

(

int

i

=

0

;

i

<

steps;

i++)

{

for

(

int

s

=

0

;

s

<

i;

s++)

printf(

" "

)

;

printf(

"--> step %d\n"

,

i+1);

}

return

0

;

}

## patterns/10\_backprop\_flow.c

/\*

Backpropagation Flow (simplified)

Visualizes forward then backward arrows for a tiny network.

\*/

#include

<

stdio.h

>

int

main()

{

printf(

"Input -> Hidden -> Output\n"

)

;

printf(

" ^\n"

)

;

printf(

" |\n"

)

;

printf(

" Gradients flow back\n"

)

;

return

0

;

}

## patterns/11\_ai\_brain\_ascii.c

/\*

AI Brain ASCII Art

Decorative ASCII representing a stylized brain.

\*/

#include

<

stdio.h

>

int

main()

{

printf(

" .-""-.\n"

)

;

printf(

" / \\\\n"

)

;

printf(

" | AI |\n"

)

;

printf(

" \\ /\n"

)

;

printf(

" `-..-'\n"

)

;

return

0

;

}

## patterns/12\_circuit\_chip.c

/\*

Circuit Chip Pattern

Draws a simple chip/circuit ASCII block.

\*/

#include

<

stdio.h

>

int

main()

{

printf(

"+------------+\n"

)

;

printf(

"| [ AI ] |\n"

)

;

printf(

"| CHIP V1 |\n"

)

;

printf(

"+------------+\n"

)

;

printf(

" | | | |\n"

)

;

printf(

"--+--+--+--+--\n"

)

;

return

0

;

}

## patterns/13\_probability\_dist.c

/\*

Probability Distribution Bars

Simple ASCII histogram for discrete probabilities.

\*/

#include

<

stdio.h

>

int

main()

{

float

probs[]

=

{0.1

f,

0.4

f,

0.25

f,

0.25f}

;

int

n

=

4

;

for

(

int

i

=

0

;

i

<

n;

i++)

{

int

len

=

(

int

)(

probs[i

]

\*

20)

;

// scale for display

printf(

"P%d: "

,

i+1);

for

(

int

j

=

0

;

j

<

len;

j++)

printf(

"\*"

)

;

printf(

" (%.2f)\n"

,

probs[i]);

}

return

0

;

}

## patterns/14\_matrix\_visual.c

/\*

Matrix Visualization

Prints a simple square matrix with indices to mimic tensors.

\*/

#include

<

stdio.h

>

int

main()

{

int

n

=

4

;

for

(

int

i

=

0

;

i

<

n;

i++)

{

for

(

int

j

=

0

;

j

<

n;

j++)

{

printf(

"[%d,%d] "

,

i,

j);

}

printf(

"\n"

)

;

}

return

0

;

}

## patterns/15\_flowchart\_layers.c

/\*

Layered Flowchart

Shows Input -> Hidden -> Hidden -> Output with brackets for clarity.

\*/

#include

<

stdio.h

>

int

main()

{

printf(

"[Input]\n"

)

;

printf(

" |\n"

)

;

printf(

"[Hidden Layer 1]\n"

)

;

printf(

" |\n"

)

;

printf(

"[Hidden Layer 2]\n"

)

;

printf(

" |\n"

)

;

printf(

"[Output]\n"

)

;

return

0

;

}