# Introduction to Lab #3: Lab_CubeStats_New

José Nelson Amaral

CMPUT 229

University of Alberta

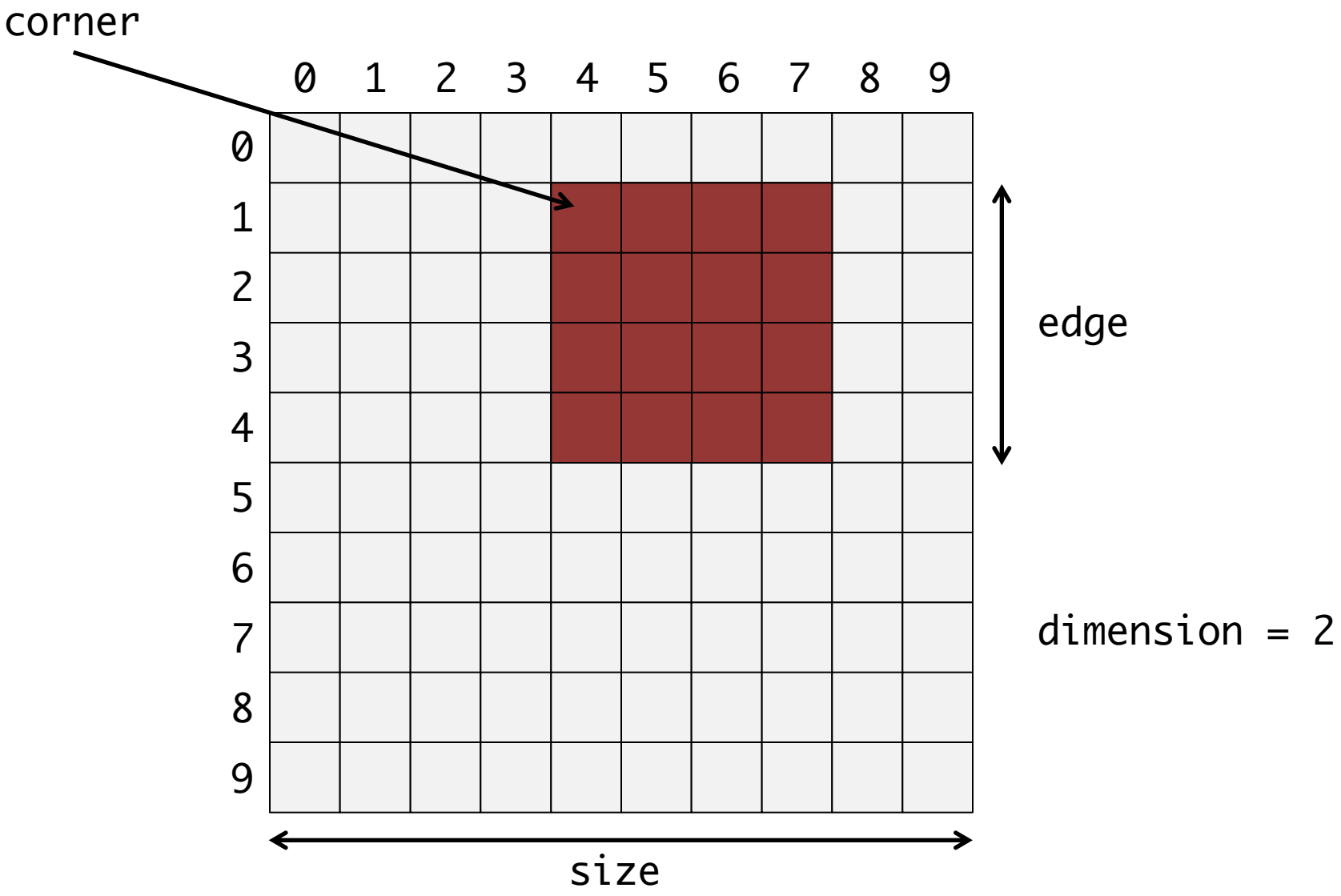
# Requirements

- Follow all subroutine calling conventions
- Must use $fp to access anything that is stored in the stack
  - Only can use $sp in this assignment to change the size of the stack.

# CubeStats

- Receives the following parameters:
  - `corner`: the address of the first element of a cube in an *n*-dimensional array.
  - `edge`: the size of the edge of the cube.
  - `dimensions`: the number of dimensions of the cube (and base array).
  - `size`: the size of the base array
    - Assume that the size of the base array is the same in all dimensions*, i.e.* the base array is itself a cube

# A two-dimensional example



# CubeStats Return Values

- `$v0`: a signed integer representing the floor of the average of all negative elements in the specified cube.

- `$v1`: a signed integer representing the floor of the average of all positive elements in the specified cube.

# CubeStats Return Values --- more formally

# CubeStats Return Values --- more formally

$$C = \text{Set of elements that are in the specified cube}$$

$$N = \{x_i | x_i \in C \wedge x_i < 0\}$$

$$P = \{x_i | x_i \in C \wedge x_i > 0\}$$

$$\$\text{v0} = \left\lfloor \frac{\sum_{x_i \in N} x_i}{|N|} \right\rfloor$$

$$\$\text{v1} = \left\lfloor \frac{\sum_{x_i \in P} x_i}{|P|} \right\rfloor$$

# CubeStats (cont.)

- Assume that the parameters are correct:
  - Parameters are positive
  - The Cube is contained within the base array

# 1-d Array Storage

What is the address of element -1 (i=2) ?

$$A + \quad \times 4$$


One-dimensional matrix A.

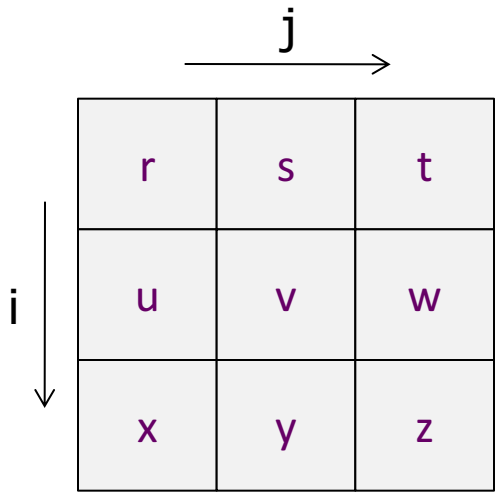
| Address | Value |
|---|---|
| 0x10001024 | |
| 0x10001020 | |
| 0x1000101C | . . . |
| 0x10001018 | 15 |
| 0x10001014 | -5 |
| 0x10001010 | 4 |
| 0x1000100C | 1 |
| 0x10001008 | -1 |
| 0x10001004 | 3 |
| 0x10001000 | 7 |
| 0x10000FFC | . . . |

Organization of B in memory in row-major style.

# 2-d Array Storage

What is the address of element w (i=1, j =2) ?

$$B + (\qquad\qquad) \times 4$$




B[i][j]

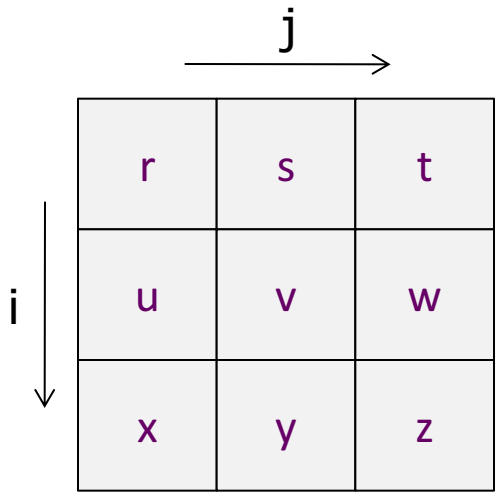
Two-dimensional 3 × 3 matrix B.

| Address | Value |
|---|---|
| 0x10001024 | |
| 0x10001020 | z |
| 0x1000101C | y |
| 0x10001018 | x |
| 0x10001014 | w |
| 0x10001010 | v |
| 0x1000100C | u |
| 0x10001008 | t |
| 0x10001004 | s |
| 0x10001000 | r |
| 0x10000FFC | |

Organization of B in memory in row-major style.

# 2-d Array Storage

Which elements belong to a Cube at position (1,1) with an edge = 2?




B[i][j]

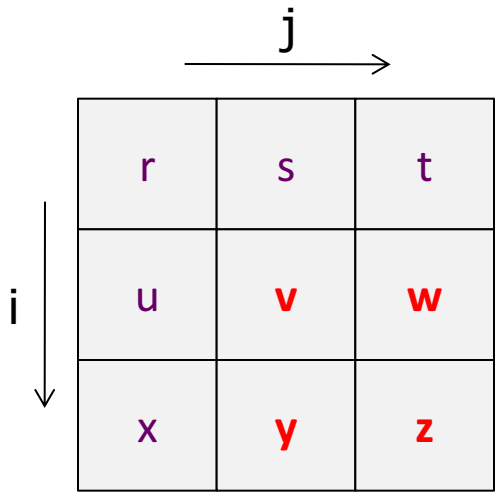
Two-dimensional 3 × 3 matrix B.

| Address | Value |
|---|---|
| 0x10001024 | |
| 0x10001020 | z |
| 0x1000101C | y |
| 0x10001018 | x |
| 0x10001014 | w |
| 0x10001010 | v |
| 0x1000100C | u |
| 0x10001008 | t |
| 0x10001004 | s |
| 0x10001000 | r |
| 0x10000FFC | |

Organization of B in memory in row-major style.

# 2-d Array Storage

Which elements belong to a Cube at position (1,1) with an edge = 2?




B[i][j]

Two-dimensional 3 × 3 matrix B.

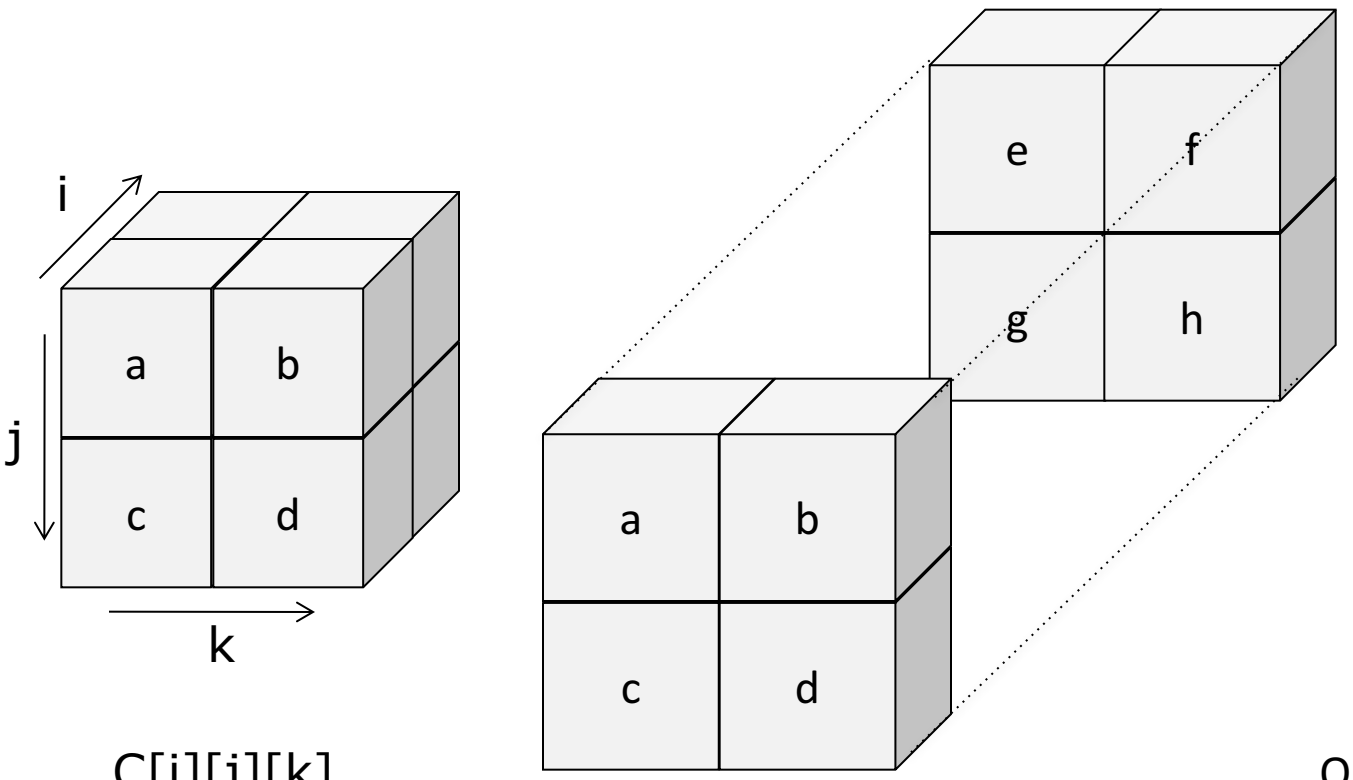
| Address | Value |
|---|---|
| 0x10001024 | |
| 0x10001020 | z |
| 0x1000101C | y |
| 0x10001018 | x |
| 0x10001014 | w |
| 0x10001010 | v |
| 0x1000100C | u |
| 0x10001008 | t |
| 0x10001004 | s |
| 0x10001000 | r |
| 0x10000FFC | |

Organization of B in memory in row-major style.

# 3-d Array Storage

What is the address of element h (i=1, j =1, k=1) ?

$$C + (\qquad\qquad\qquad\qquad\qquad\qquad) \times 4$$




C[i][j][k]

Three-dimensional 2 × 2 × 2 matrix C.

| Address | Value |
|---|---|
| 0x10001024 | |
| 0x10001020 | |
| 0x1000101C | h |
| 0x10001018 | g |
| 0x10001014 | f |
| 0x10001010 | e |
| 0x1000100C | d |
| 0x10001008 | c |
| 0x10001004 | b |
| 0x10001000 | a |
| 0x10000FFC | |

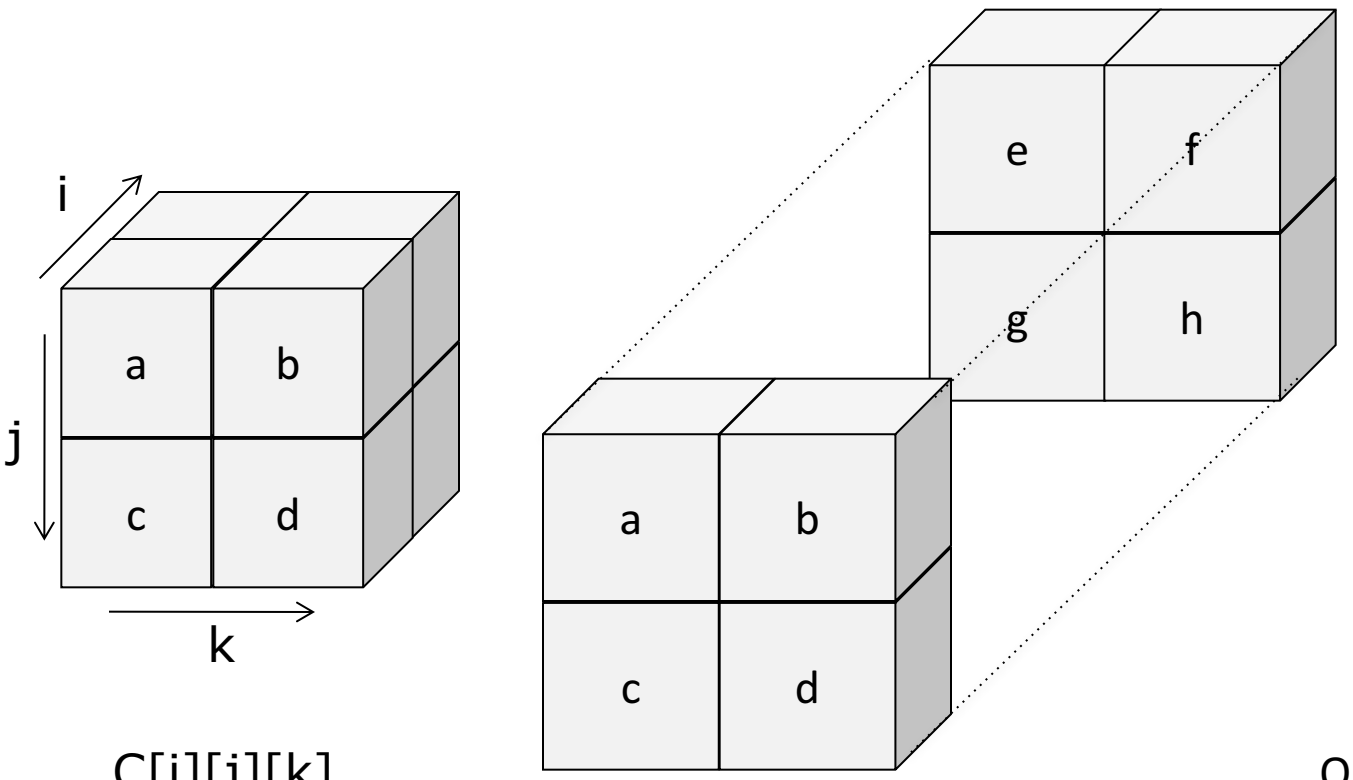
Organization of C in memory in row-major style.

# 3-d Array Storage

What is the address of element h (i=1, j =1, k=1) ?

$$C + (((i \times 2) + j) \times 2 + k) \times 4$$

$$C + (i \times 2 \times 2 + j \times 2 + k) \times 4$$




C[i][j][k]

Three-dimensional 2 × 2 × 2 matrix C.

| Address | Value |
|---|---|
| 0x10001024 | |
| 0x10001020 | |
| 0x1000101C | h |
| 0x10001018 | g |
| 0x10001014 | f |
| 0x10001010 | e |
| 0x1000100C | d |
| 0x10001008 | c |
| 0x10001004 | b |
| 0x10001000 | a |
| 0x10000FFC | |

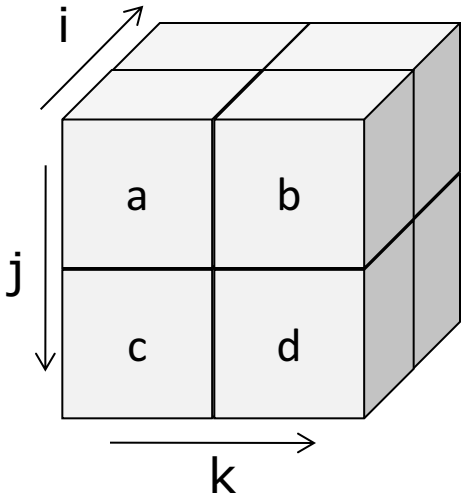
Organization of C in memory in row-major style.

# main program

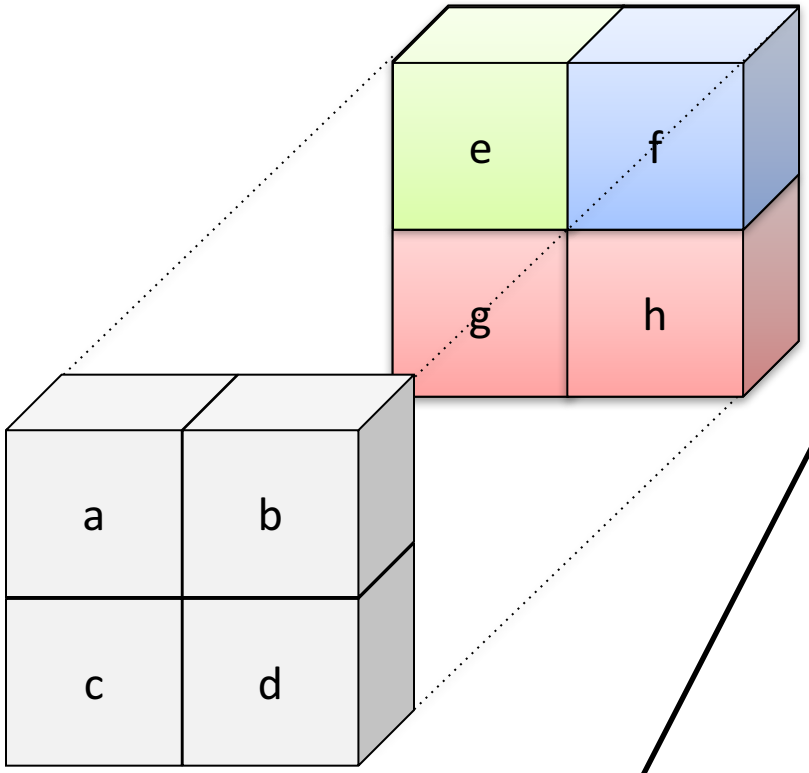
- reads a k-dimensional Cube from a file
- places the values in the memory in row-major format
- for each specification of a cube in the file:
  - initializes four global variables to zero:
    - `totalNeg`, `totalPos`, `countNeg` and `countPos`
  - calls your `CubeStats` subroutine
  - prints the value returned by `CubeStats`

# File format

C[1][0][1] ↔ f




C[i][j][k]




size of array

dimension of array

```
3 2
a b
c d
e f
g h
1 0 1 1
0 0 0 2
-1
```

Cube a position (0,0,0) with edge 2

Cube at position (1,0,1) with edge 1

# main

- Reading and understanding the main routine is part of the assignment.

# Test Generator

- A test generator, written in Python, is provided to you as a convenience.
  - Have fun modifying/playing with it.
- Caution:
  - Large test cases overflow the arena provided
  - Increasing the arena is ok but will eventually run into the static space limit of SPIM.

# What to hand in

- A single file named `CubeStats.s` containing your subroutine `CubeStats` written in MIPS assembly.
- Your subroutine must return to the caller using the instruction:

```
jr $ra
```

- Your file must not contain a main function.