CP264 Midterm Review Session

LAURIER COMPUTING SOCIETY

Please take this moment to sign in...

Agenda

WHAT ARE WE COVERING TODAY?

1. **FUNDAMENTALS OF C PROGRAMMING LANGUAGE**
   Compliation + Execution of Programs
2. **C PROGRAM STRUCTURE & ORGANIZATION**
   Functions + Pointers
3. **HALFWAY POINT KAHOOT**
4. **USER-DEFINED TYPES**
   Structures + Union + Enumeration
5. **ARRAYS**
   Memory Allocation + Multi-Dimensional Arrays
6. **FINAL KAHOOT**

UPCOMING EVENTS...

1. **CODE & CHILL #1 -**
   FEBRUARY 13 @ 7:30 PM
   ROOM: BA102
2. **CP164 MIDTERM REVIEW -**
   (TELL FIRST YEAR FRIENDS)
   FEBRUARY 15 @ 7:30 PM
   ROOM TBD

Useful Links and Follow Along Resources

MATH AND STATS LEARNING SUPPORT

This office hosts drop in and directed homework sessions to give you any extra help you need with course content in most lower level CS courses. You can also get their course widget in MyLS.

CODING SANDBOXES

A really good Java code sandbox is Replit.com.
Follow along or write code on your own time without setting up files by using one of these.

A really good Python, Java, HTML, sandbox (and almost every other language) is Replit.com
https://replit.com/

Another good HTML/CSS/JS sandbox:
https://codepen.io/

Mental Health Resources

We know university is very challenging and difficult. Please don't hesitate to reach out to people if you need help. Listed below are a few useful resources you can access, for additional resources please view the LCS linktree.

SAFEHAWK APP

Connects you with telephone helplines and other mental health resources.

STUDENT WELLNESS CENTRE

Provides comprehensive physical, emotional and mental health services for Waterloo and Brantford Campus students.

DELTON GLEBE COUNSELLING CENTRE:

A holistic counselling facility.

Compiling & Executing C Programs Steps

Preprocessing Compilation Assembling Linking

C Program Structure

```
[preprocessor directives]
[global variables]
[function declarations]
main( arguments ) {
  [statements]
}
[function definitions]
```

*USING [ ] TO REPRESENT A LIST OF ITEMS

C Program Organization

```c
/**
 * addsub.h
 * header file contains function headers
 */
#ifndef ADDSUB_H  // set compiling condition to avoid redefinition
#define ADDSUB_H

/**
 * @param x - int value
 * @param y - int value
 * @return - sum of x and y, int value
 */
int add(int x, int y);

/**
 * @param x - int value
 * @param y - int value
 * @return -  subtract y from x, int value
 */
int minus(int x, int y);

#endif
```

1 Header File

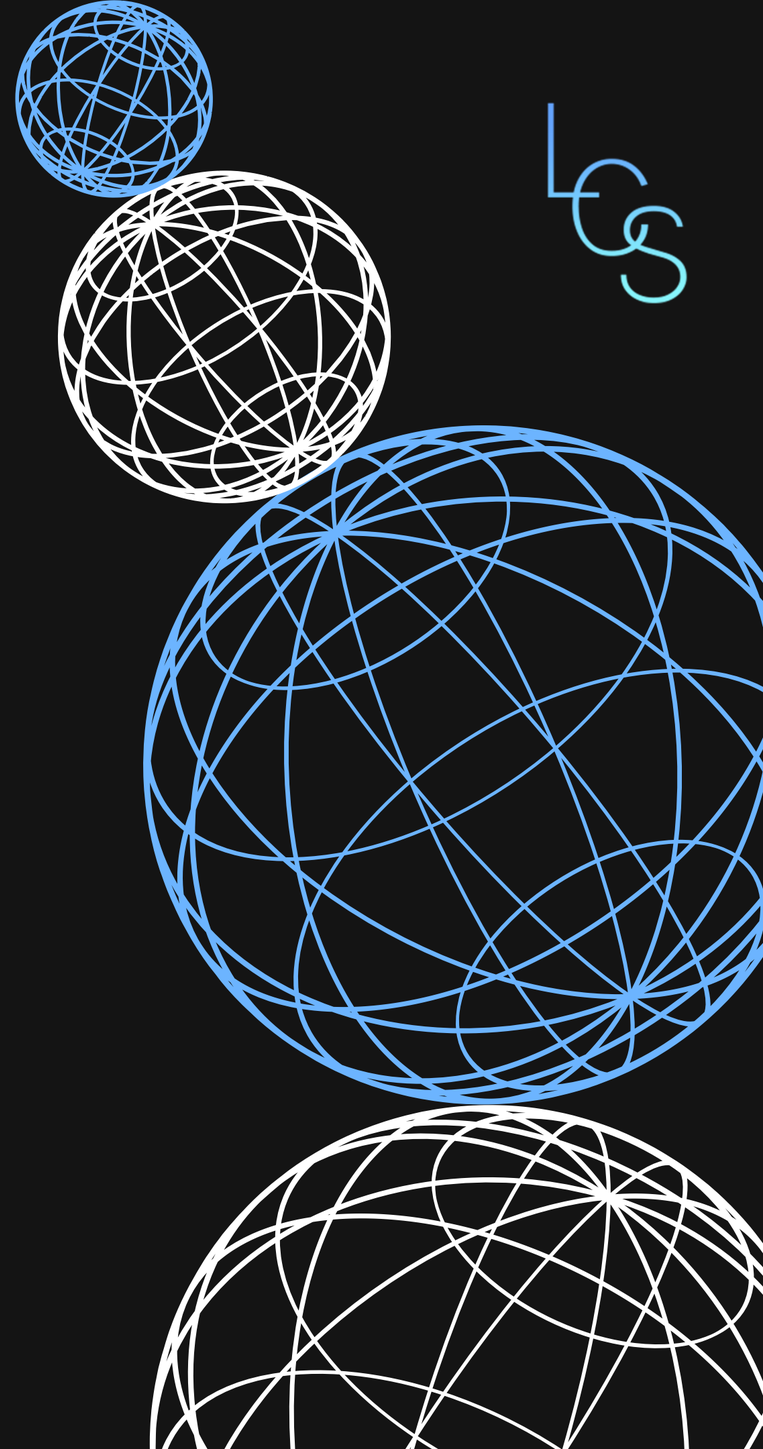
```c
/**
 * addsub.c
 * implementation of header functions
 */
#include "addsub.h"

int add(int x, int y)
{
    return x+y;
}

int minus(int x, int y)
{
  return x-y;
}
```

2 Header Implementation

3 Main Program

```c
/**
 * addsub_main.c
 * test driver program containing the main function
 */
#include <stdio.h>
#include "addsub.h"
int a;
int main()
{
  a=1;
  int b=2;
  printf("a+b=%d\n", add(a, b));
  printf("a-b=%d\n", minus(a, b));
  return 0;
}
```

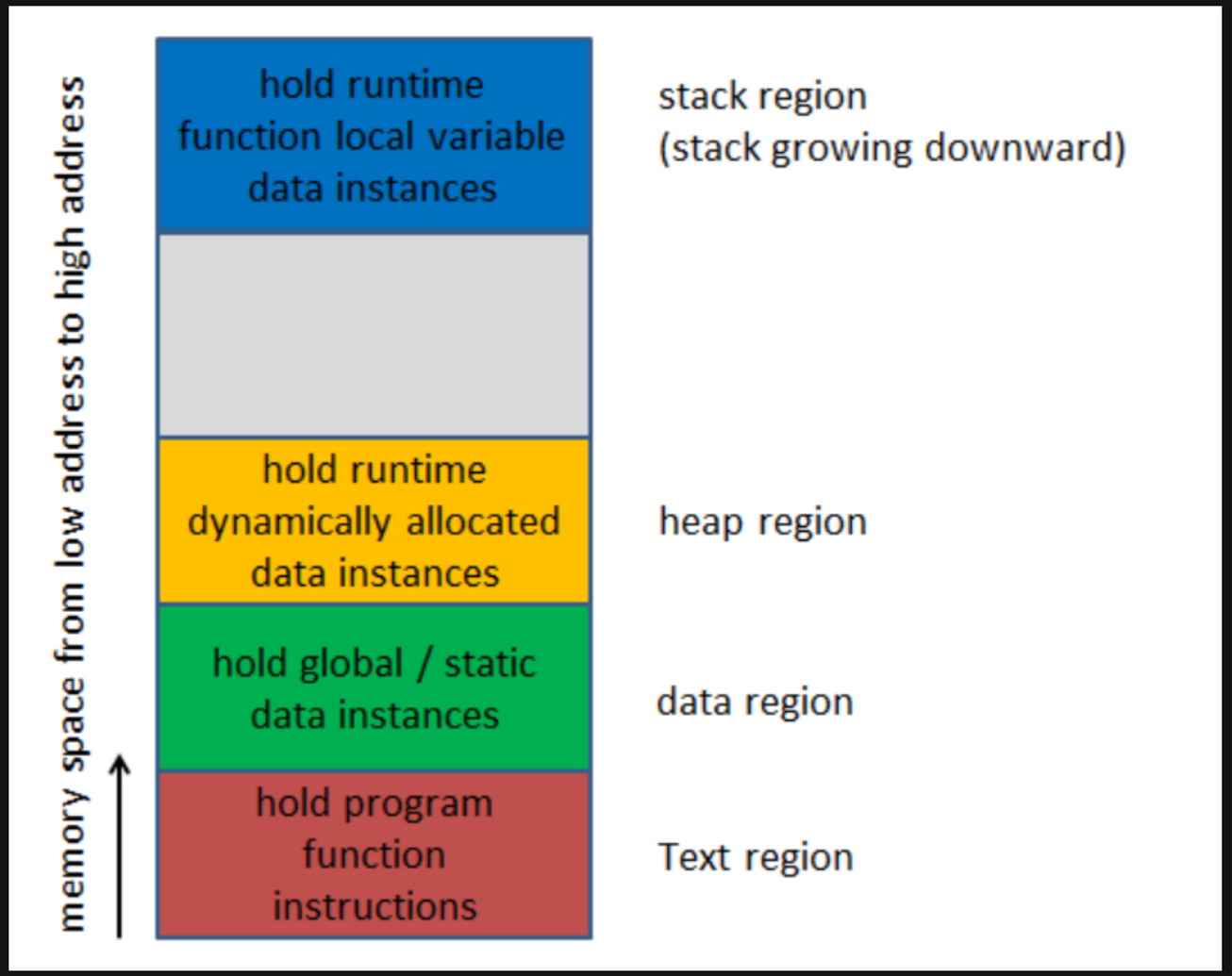
Do you have any questions?



Functions

```c
#include<stdio.h>
int w;           // global variable, visible from anywhere.
void f1();       // function name, function declaration/prototype, without return value
int f2(int);     // function with return value and an int parameter
int f3();
int main() {
  int x = 2;     // declare local variable and initialize to 2
  w = 1;         // assign 1 to global variable w
  f1();          // call f1
  int y = f2(x); // declare int variable y, call f2 by passing value of x, assign the return value to y
  printf("value: %d, %d\n", w, y);
  printf("count: %d", f3());  // print the number of function calls
  return 0;
}
void f1() {      // function definition/implementation
  f3();
  int y = 2;     // declare local variable and initialize to 2
  w += y;
}
int f2(int x) {   // x is an int parameter
  f3();
  int z = x + 2;
  f1();           // call f1
  return z;       // return z, int type
}
int f3() {
  static int counter = 0;  // static variable, value will be retained between function calls
  counter = counter + 1;
  return counter;
}
/* output:
value: 5, 4
count: 4
*/
```

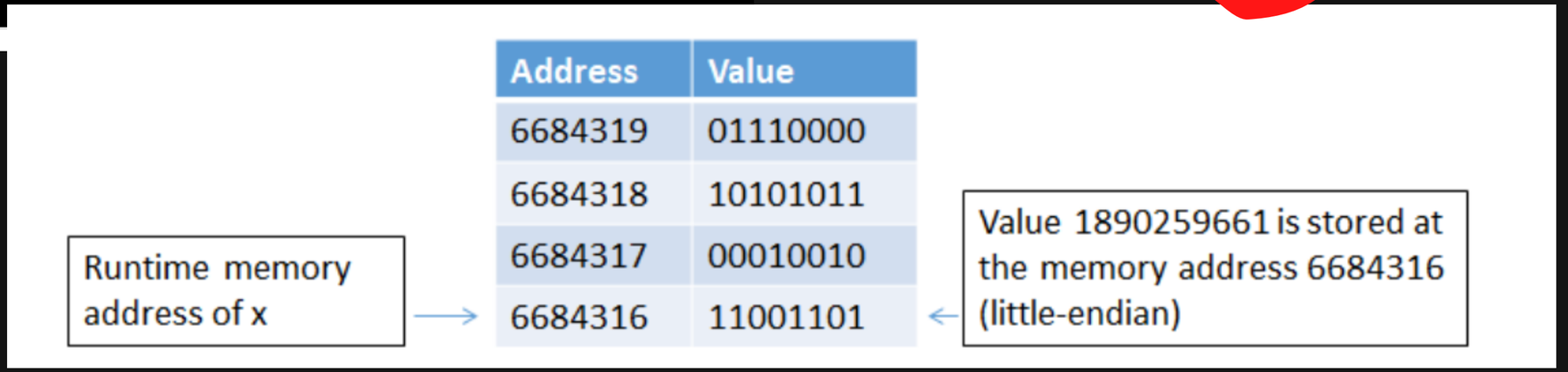
Memory Management of Program Executions



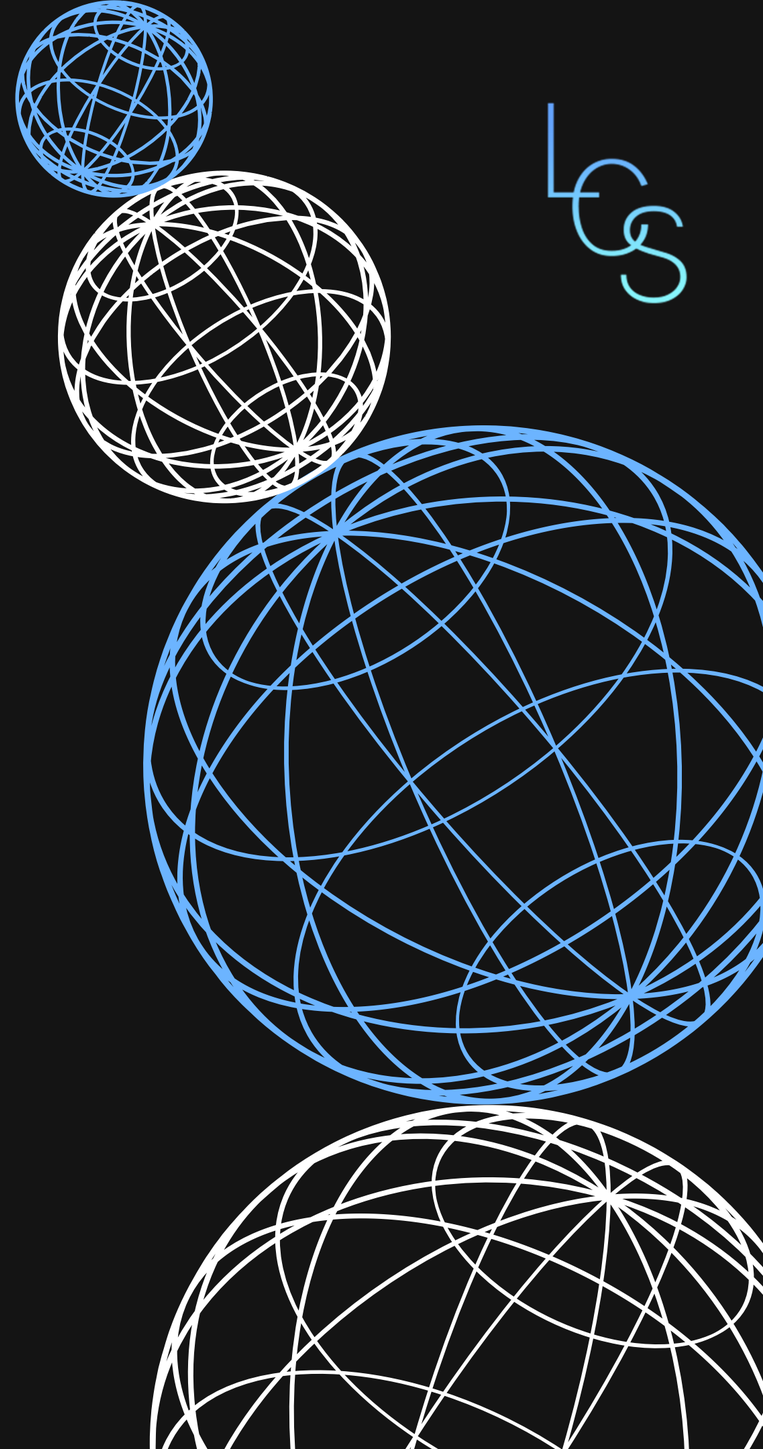
Pointers

```c
#include<stdio.h>
int main()
{
  int x = 1890259661;
  printf("Value of x is %d\n",x);
  printf("Runtime memory address of x in Hex is %p\n", &x);
  printf("Runtime memory address of x in decimal is %lu\n", &x);
  printf("Value stored at address %lu is %d\n", &x, *(&x));
  return 0;
}
```

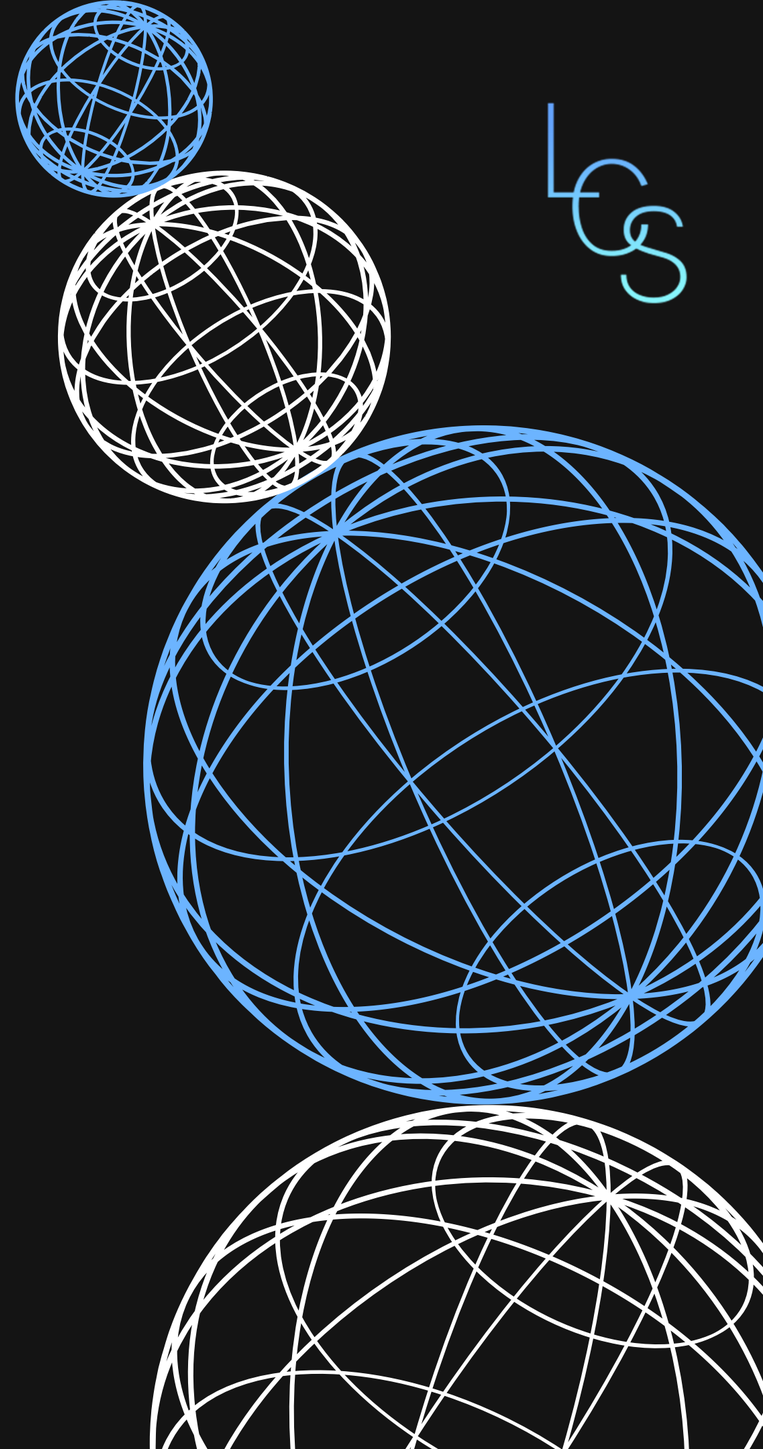
```
Value of x is 1890259661
Runtime memory address of x in Hex is 0065FE9C
Runtime memory address of x in decimal is 6684316
Value stored at address 6684316 is 1890259661
```



Do you have any questions?



Live coding time: Pointers!



HALFWAY POINT KAHOOT TIME!!!!

USE YOUR LAURIER LOG IN FOR THE KAHOOT!!!

User Defined Types: Justification

User Defined Types: Structs

- EXAMPLE:

```c
#include <stdio.h>

// Define the struct for a pizza
typedef struct {
    char name[50];
    float diameter; // in inches
    int slices;
    float price; // in dollars
} PIZZA;  // Creating an alias "PIZZA" for the struct

int main() {
    // Declare and initialize a pizza using the typedef
    PIZZA pepperoniPizza = {"Pepperoni", 12.0, 8, 10.99};

    // Access and print information about the pizza
    printf("Pizza Details:\n");
    printf("Name: %s\n", pepperoniPizza.name);
    printf("Diameter: %.2f inches\n", pepperoniPizza.diameter);
    printf("Number of Slices: %d\n", pepperoniPizza.slices);
    printf("Price: $%.2f\n", pepperoniPizza.price);

    return 0;
}
```

Member Access: Arrow Notation

- EXAMPLE:

```
11  // Typedef for the car struct
12  typedef struct Car {
13      char name[50];
14      int year;
15  } CAR;
16
17
18  // Typedef for the tow truck struct
19  typedef struct TowTruck {
20      char name[50];
21      CAR *towedCar;      // Pointer to the car being towed
22  } TOWTRUCK;
23
24
25  // Function to simulate towing
26  void towCar(TOWTRUCK *towTruck, CAR *car) {
27      towTruck->towedCar = car;
28  }
```

User Defined Types: Unions

- EXAMPLE:

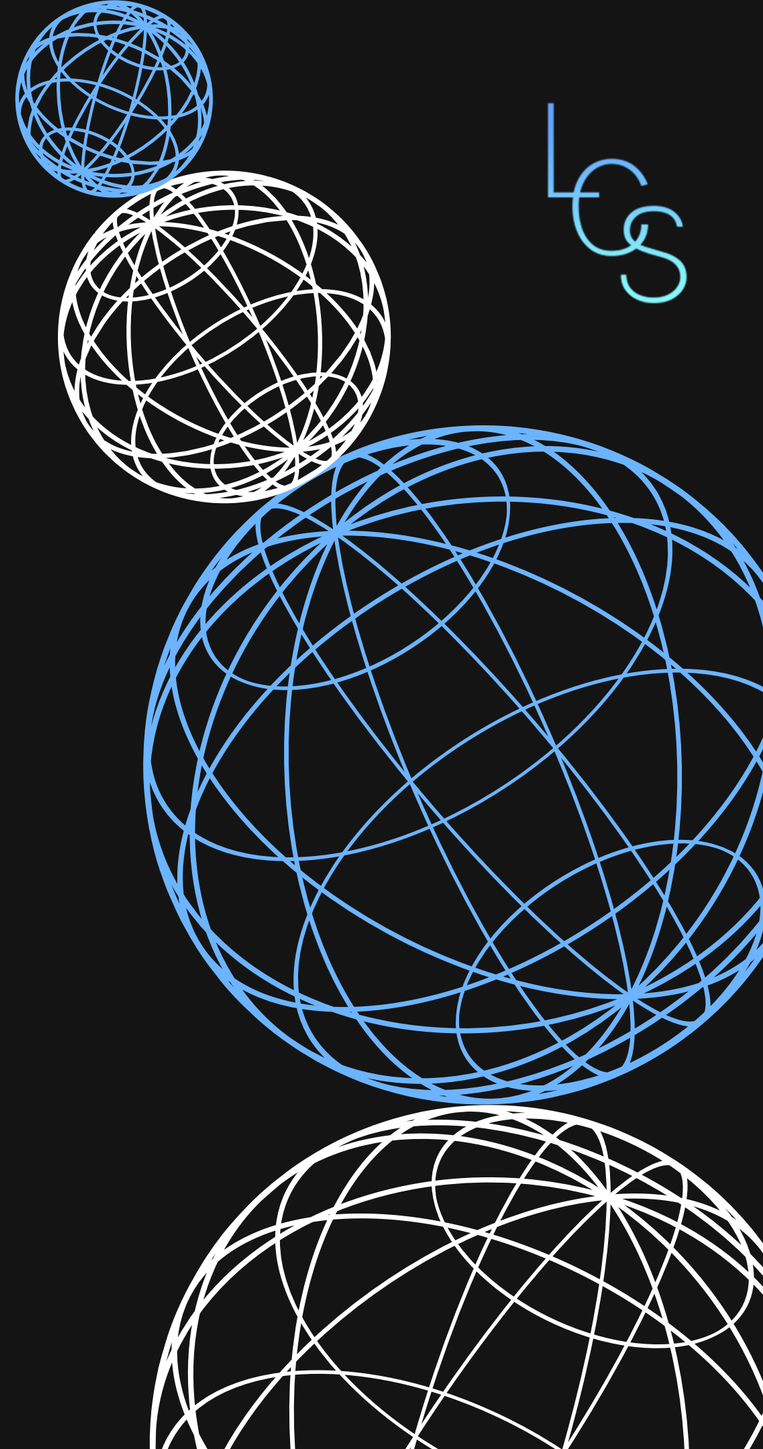
```c
#include <stdio.h>

// Define the union for representing a pizza
union Pizza {
    char name[50];
    float diameter; // in inches
    int slices;
    float price; // in dollars
};

int main() {
    // Declare and initialize a pizza using the union
    union Pizza pizzaUnion;

    // Assign values to different members of the union
    strcpy(pizzaUnion.name, "Pepperoni Pizza");
    pizzaUnion.diameter = 12.0;
    pizzaUnion.slices = 8;
    pizzaUnion.price = 10.99;

    // Access and print information about the pizza
    printf("Pizza Details:\n");
    printf("Name: %s\n", pizzaUnion.name);
    printf("Diameter: %.2f inches\n", pizzaUnion.diameter);
    printf("Number of Slices: %d\n", pizzaUnion.slices);
    printf("Price: $%.2f\n", pizzaUnion.price);

    return 0;
}
```

User Defined Types: Enumerations

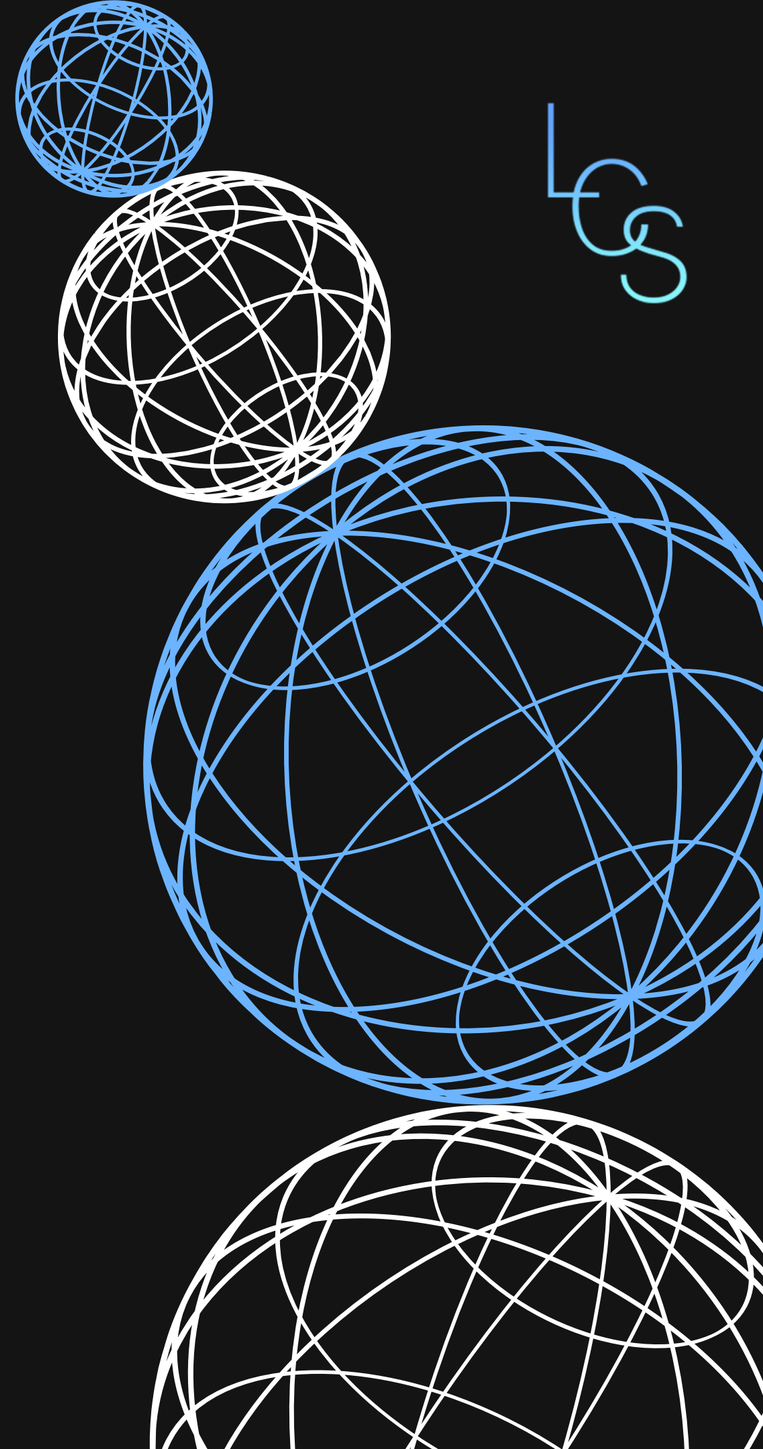
- EXAMPLE:

```c
#include <stdio.h>

// Define an enum for pizza types
enum PizzaType {
    MARGHERITA,
    PEPPERONI,
    VEGETARIAN,
    SUPREME
};

int main() {
    // Declare a variable of the PizzaType enum
    enum PizzaType myPizza = PEPPERONI;

    // Switch statement to handle different pizza types
    switch (myPizza) {
        case MARGHERITA:
            printf("You have chosen Margherita pizza.\n");
            break;
        case PEPPERONI:
            printf("You have chosen Pepperoni pizza.\n");
            break;
        case VEGETARIAN:
            printf("You have chosen Vegetarian pizza.\n");
            break;
        case SUPREME:
            printf("You have chosen Supreme pizza.\n");
            break;
        default:
            printf("Invalid pizza type.\n");
            break;
    }

    return 0;
}
```

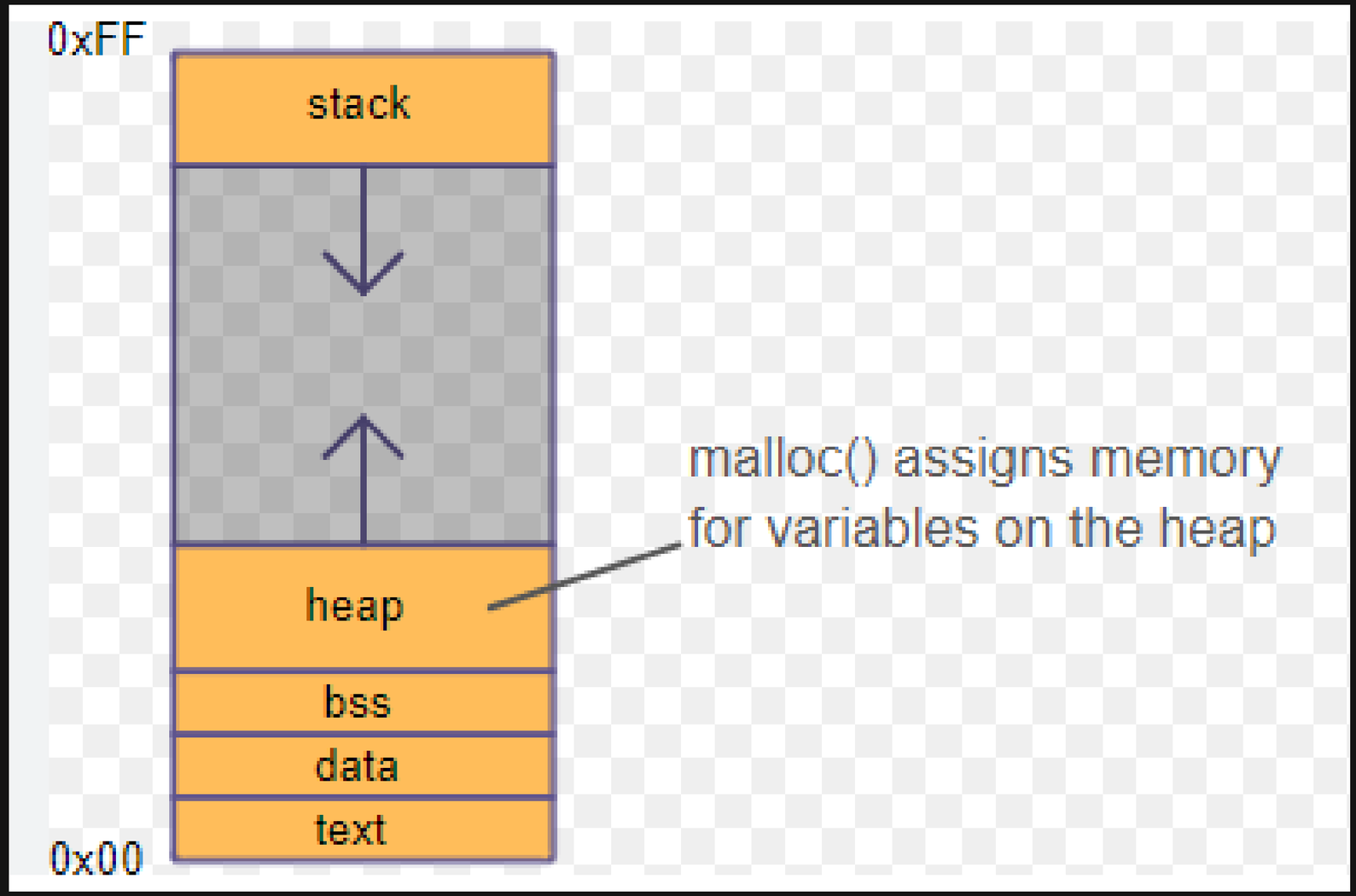
Do you have any questions?



Live coding time: Struct!



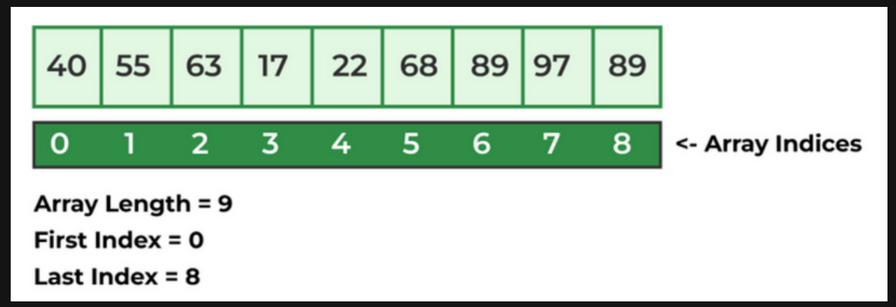
What is Memory Allocation?



Memory Allocation: Example

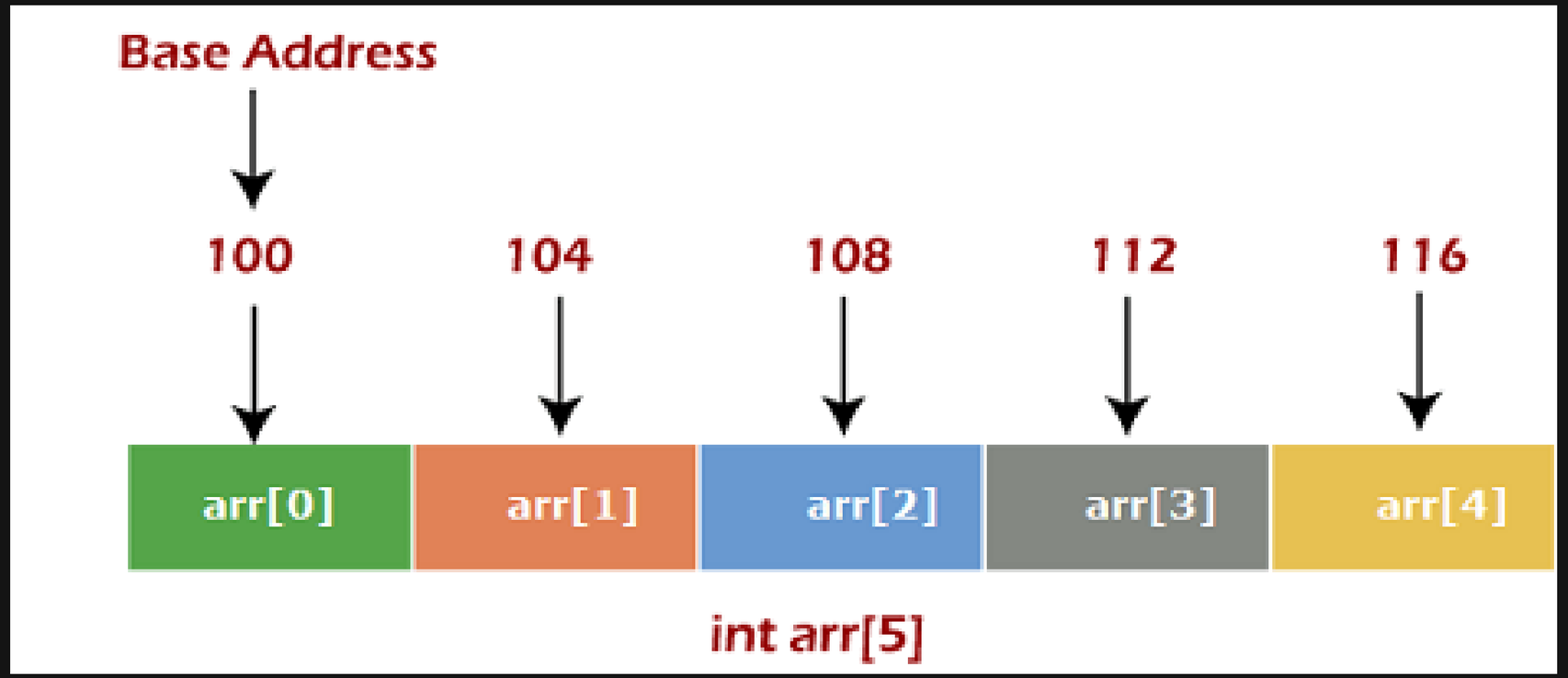
```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    // Declare a pointer to an integer
    int *dynamicInteger;

    // Allocate memory for a single integer
    dynamicInteger = (int*)malloc(sizeof(int));

    // Check if memory allocation is successful
    if (dynamicInteger != NULL) {
        // Initialize the allocated memory
        *dynamicInteger = 42;

        // Print the value
        printf("Dynamic Integer Value: %d\n", *dynamicInteger);

        // Free the allocated memory
        free(dynamicInteger);
        dynamicInteger = NULL; // Good practice to set the pointer to NULL after freeing

        // Inform the user
        printf("Memory freed successfully.\n");
    } else {
        // Inform the user if memory allocation fails
        printf("Memory allocation failed.\n");
    }

    return 0;
}
```

Arrays: Introduction




```c
#include <stdio.h>

int main() {
    // Declare an array of integers with size 5
    int myArray[5];

    // You can also initialize the array elements at the time of declaration
    int initializedArray[3] = {1, 2, 3};

    // Accessing and modifying array elements
    myArray[0] = 10;
    myArray[1] = 20;
    myArray[2] = 30;
    myArray[3] = 40;
    myArray[4] = 50;

    // Print array elements
    printf("myArray: %d, %d, %d, %d, %d\n", myArray[0], myArray[1], myArray[2], myArray[3], myArray[4]);
    printf("initializedArray: %d, %d, %d\n", initializedArray[0], initializedArray[1], initializedArray[2]);

    return 0;
}
```

Arrays in Memory



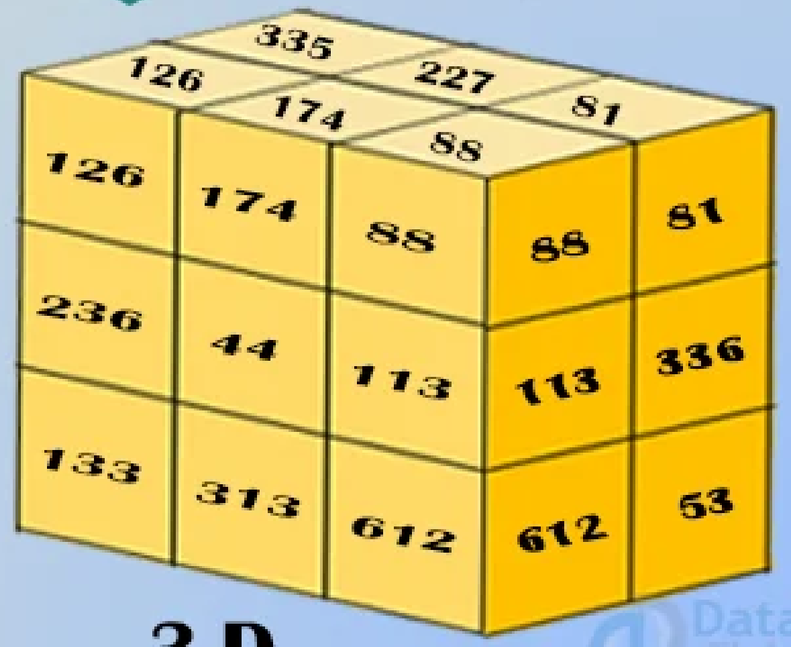
Multidimensional Arrays: Intro

Data Flair

Multi-dimensional Arrays in C / C++

	Column 1	Column 2	Column 3	Column 4
Row 1	Matrix[0][0]	Matrix[0][1]	Matrix[0][2]	Matrix[0][3]
Row 2	Matrix[1][0]	Matrix[1][1]	Matrix[1][2]	Matrix[1][3]
Row 3	Matrix[2][0]	Matrix[2][1]	Matrix[2][2]	Matrix[2][3]

2-D



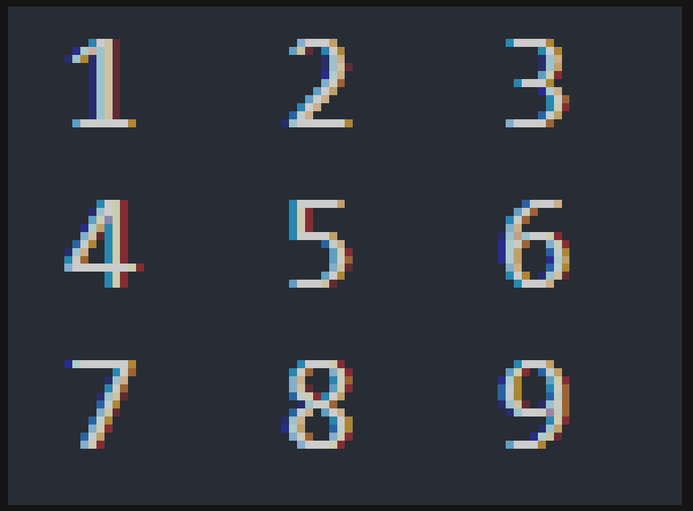
3-D

Multidimensional Arrays: Example

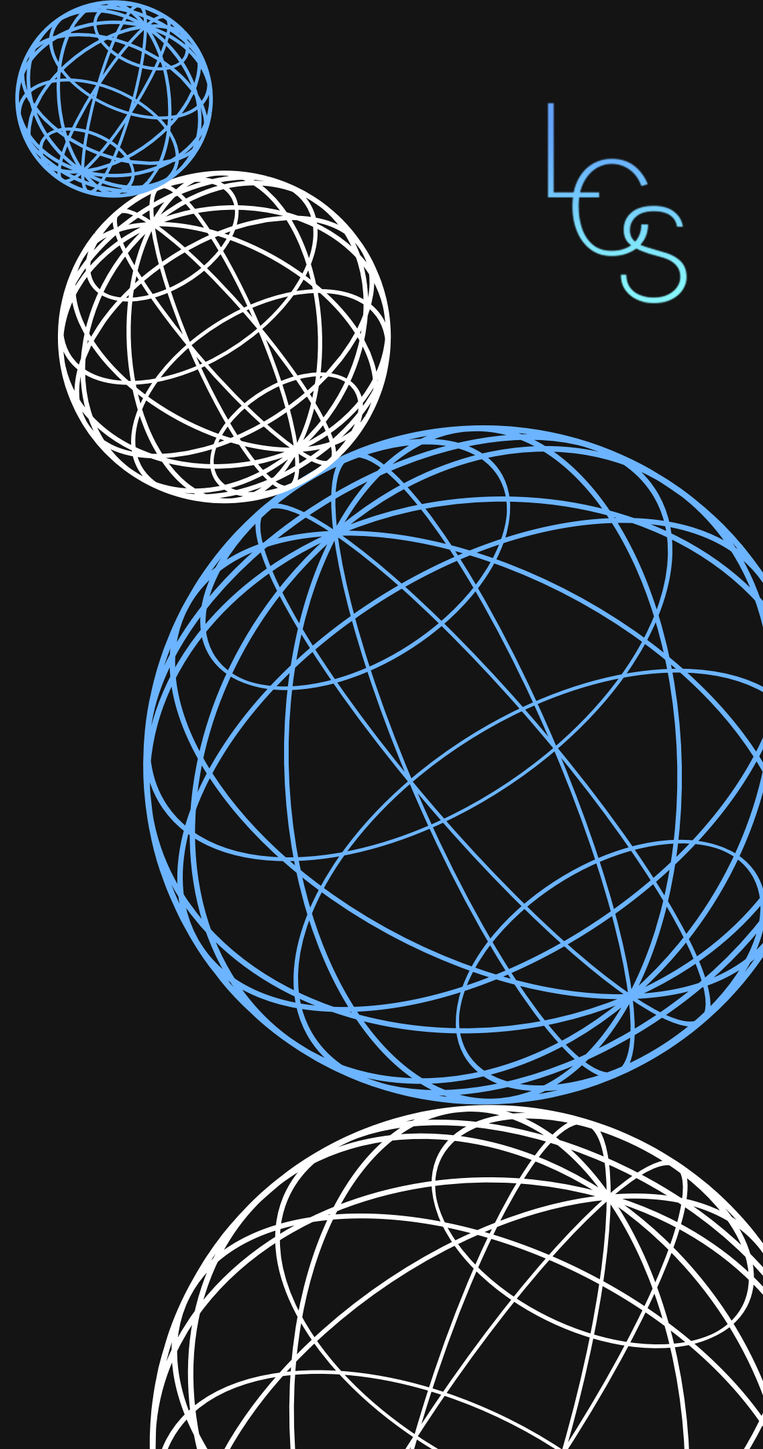
CODE

```c
#include<stdio.h>
#include<stdlib.h>
int main(){
    int matrix[3][3]=
    {{1,2,3},
    {4,5,6},
    {7,8,9}};
    for(int i=0;i<3;i++){
        for(int j=0; j<3; j++){
            printf("%2d ", matrix[i][j]); //print value of element in the matrix
        }
        printf("\n"); //print newline character
    }
    return 0;
}
```

OUTPUT



Do you have any questions?



Find the Slides here!!

Code examples!!

Please fill out this event survey!

FINAL KAHOOT TIME!!!!

USE YOUR LAURIER LOG IN FOR THE KAHOOT!!!

Thank you for coming!

Laurier
Computing
Society

@laurier.cs

discord.lauriercs.ca

linktr.ee/lauriercs