Agenda

- simple and dynamic Array(1D and 2D)
- enum
- ~~multiple files~~
- ~~Hierarchy and its type.~~
- ~~Association~~

Array

- Array is a data structure that is used to store the elements of same type in contagious memory locations.
- the elements stored in the array can be accessed using their index number;
- Types of array
    1. Single Dimension Array
    2. Multi Dimension Array
- we can create array for fundamental data types as well as derived data types

Single Dimension Array

```
// single dimension array
int main()
{
    // int arr[5] = {10, 20, 30, 40, 50};
    int arr[] = {10, 20, 30, 40, 50};
    // int arr[5];
    //  arr[0] = 10;
    //  ...

    for (int i = 0; i < 5; i++)
        cout << arr[i] << ",";
    cout << endl;
    return 0;
}

// single dimension array of ptrs (Dynamic memory allocation)
int main()
{
    int *arr[5];
    for (int i = 0; i < 5; i++)
        arr[i] = new int(10 * (i + 1));

    for (int i = 0; i < 5; i++)
        cout << *arr[i] << ",";
    cout << endl;

    for (int i = 0; i < 5; i++)
        delete arr[i];
    return 0;
```

```
}

// single dimension array with Dynamic memory allocation
int main()
{
    int *arr = new int[5]{10, 20, 30, 40, 50};

    for (int i = 0; i < 5; i++)
        cout << arr[i] << ",";
    cout << endl;
    delete[] arr;
    return 0;
}
```

Multi Dimension Array

```
// multi dimension array
int main()
{
    int arr[][3] = {10, 20, 30, 40, 50, 60};
    // int arr[2][3] = {10, 20, 30, 40, 50, 60};
    for (int i = 0; i < 2; i++)
        for (int j = 0; j < 3; j++)
            cout << arr[i][j] << ",";
    cout << endl;
    return 0;
}

// multi dimension array of ptrs (Dynamic memory allocation)
int main()
{
    int *arr[2][3];
    for (int i = 0; i < 2; i++)
        for (int j = 0; j < 3; j++)
            arr[i][j] = new int(i + j);

    for (int i = 0; i < 2; i++)
        for (int j = 0; j < 3; j++)
            cout << arr[i][j] << endl;

    for (int i = 0; i < 2; i++)
        for (int j = 0; j < 3; j++)
            delete arr[i][j];
    return 0;
}


// multi dimension array with Dynamic memory allocation
int main2()
```

```
{

    int **arr = new int *[2];
    arr[0] = new int[3]{10, 20, 30};
    arr[1] = new int[3]{40, 50, 60};

    for (int i = 0; i < 2; i++)
        for (int j = 0; j < 3; j++)
            cout << arr[i][j] << ",";
    cout << endl;
    delete[] arr[0];
    delete[] arr[1];
    delete[] arr;
    return 0;
}
```

enum

- Enumeration (Enumerated type) is a user-defined data type that can be assigned some limited values. These values are defined by the programmer at the time of declaring the enumerated type.
- Enums provide a way to define symbolic names for sets of integers, making the code more readable and maintainable.

```
#include <iostream>

// Define an enum named Color
enum Color {
    RED,    // 0
    GREEN,  // 1
    BLUE    // 2
};

int main() {
    // Declare a variable of type Color
    Color myColor = GREEN;

    // Check the value of myColor
    if (myColor == GREEN) {
        cout << "The color is green." << endl;
    } else {
        cout << "The color is not green." << std::endl;
    }
    return 0;
}
```