perplexity

Arrays - Single dimensional arrays, Defining an array, Array initialization,

Accessing array elements; Enumerated data type; Type Definition; Twodimensional arrays – Defining a two-dimensional array; Programs for
matrix processing; Programs for sequential search; Bubble sort;
teach me all this with appropriate points, differences to note, syntax and an example

Here's a structured guide to **arrays, enums, typedef, 2D arrays, and key algorithms** in C, covering syntax, examples, and key differences:

1. Single-Dimensional Arrays

Definition & Initialization

```
// Declaration without initialization
int arr[5]; // Contains garbage values

// Declaration with initialization
int arr[5] = {1, 2, 3, 4, 5};
int arr[] = {1, 2, 3}; // Size inferred as 3
```

Accessing Elements

```
arr[0] = 10; // First element (index 0)
printf("%d", arr[2]); // Third element (index 2)
```

- **Zero-indexed**: Indexes range from `0` to `n-1`.

2. Enumerated Data Type (`enum`)

Syntax & Example

```
enum Weekday {MON, TUE, WED}; // MON=0, TUE=1, WED=2
enum Weekday day = MON;

// Explicit values
enum Status {OK=200, ERROR=404};
```

- **Default values**: Start at `0` and increment by 1.

- **Use Case**: Improve readability for fixed sets of constants.

3. Type Definition (`typedef`)

Syntax & Example

```
typedef int Age; // Alias for int
Age userAge = 25;

typedef struct { // Alias for struct
    int x, y;
} Point;
Point p1 = {10, 20};
```

- **Difference from `#define`**: `typedef` handles types, not text substitution.

4. Two-Dimensional Arrays

Definition & Initialization

```
int matrix[2][3] = { {1, 2, 3},
                     {4, 5, 6} };
```

Accessing Elements

```
matrix[0][1] = 10; // Row 0, Column 1
printf("%d", matrix[1][2]); // 6
```

5. Matrix Processing (Addition)

```
#include <stdio.h>
#define ROWS 2
#define COLS 3

void addMatrices(int a[ROWS][COLS], int b[ROWS][COLS], int result[ROWS][COLS]) {
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            result[i][j] = a[i][j] + b[i][j];
        }
    }
}

int main() {
    int a[ROWS][COLS] = {{1,2,3}, {4,5,6}};
    int b[ROWS][COLS] = {{6,5,4}, {3,2,1}};
```

```
    int result[ROWS][COLS];
    addMatrices(a, b, result);
    // Print result matrix...
    return 0;
}
```

6. Sequential Search

```
int sequentialSearch(int arr[], int size, int key) {
    for (int i = 0; i < size; i++) {
        if (arr[i] == key)
            return i; // Return index if found
    }
    return -1; // Not found
}
```

- **Time Complexity**: O(n).

7. Bubble Sort

```
void bubbleSort(int arr[], int size) {
    for (int i = 0; i < size-1; i++) {
        for (int j = 0; j < size-i-1; j++) {
            if (arr[j] > arr[j+1]) { // Swap if greater
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}
```

- **Time Complexity**: $O(n^2)$.

Key Differences & Notes

Concept	Key Points
1D vs 2D Arrays	1D: Single row; 2D: Multiple rows and columns.
`enum` **vs** `#define`	`enum` groups related constants; `#define` is for macros.
`typedef` **vs** `#define`	`typedef` creates type aliases; `#define` replaces text.
Sequential vs Binary Search	Sequential works on unsorted arrays; binary requires sorted data.
Bubble Sort	Simple but inefficient for large datasets.

Summary Table: Array Types

Array Type	Declaration	Use Case
1D Array	`int arr;`	Lists, vectors
2D Array	`int matrix;`	Matrices, grids
Dynamic Array	`int *arr = malloc(n);`	Size determined at runtime

This guide covers syntax, examples, and key differences for arrays, enums, typedef, and essential algorithms. Use the code snippets as templates for your programs.