5. Common Array Operations

5.1 Array Traversal

5.1.1 Forward Traversal

```
// Python equivalent: for item in list:
for (int i = 0; i < size; i++) {
    printf("%d ", arr[i]);
}
```

5.1.2 Reverse Traversal

```
// Python equivalent: for item in reversed(list):
for (int i = size - 1; i >= 0; i--) {
    printf("%d ", arr[i]);
}
```

5.1.3 Traversal with Condition

```
// Print only even numbers
for (int i = 0; i < size; i++) {
    if (arr[i] % 2 == 0) {
        printf("%d ", arr[i]);
    }
}
```

5.2 Finding Maximum and Minimum

Python vs C Comparison:

Python	C
`max(list)`	Manual implementation
`min(list)`	Manual implementation

5.2.1 Finding Maximum

```
int find_max(int arr[], int size) {
    if (size <= 0) {
        printf("Error: Empty array\n");
        return INT_MIN;  // Return minimum integer value
    }

    int max = arr[0];  // Assume first element is maximum

    for (int i = 1; i < size; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }

    return max;
}
```

5.2.2 Finding Minimum

```
int find_min(int arr[], int size) {
    if (size <= 0) {
        printf("Error: Empty array\n");
        return INT_MAX;  // Return maximum integer value
    }

    int min = arr[0];

    for (int i = 1; i < size; i++) {
        if (arr[i] < min) {
            min = arr[i];
        }
    }

    return min;
}
```

5.2.3 Finding Both Max and Min with Position

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

```
typedef struct {
    int max_value;
    int max_index;
    int min_value;
    int min_index;
} MinMaxResult;

MinMaxResult find_min_max(int arr[], int size) {
    MinMaxResult result = {arr[0], 0, arr[0], 0};

    for (int i = 1; i < size; i++) {
        if (arr[i] > result.max_value) {
            result.max_value = arr[i];
            result.max_index = i;
        }
        if (arr[i] < result.min_value) {
            result.min_value = arr[i];
            result.min_index = i;
        }
    }

    return result;
}
```

5.3 Searching in Arrays

5.3.1 Linear Search

```
int linear_search(int arr[], int size, int target) {
    for (int i = 0; i < size; i++) {
        if (arr[i] == target) {
            return i;  // Return index of found element
        }
    }
    return -1;  // Element not found
}

// Usage example
int main() {
```

```
    int numbers[] = {10, 25, 8, 42, 15};
    int size = sizeof(numbers) / sizeof(numbers[0]);
    int target = 42;

    int index = linear_search(numbers, size, target);

    if (index != -1) {
        printf("Element %d found at index %d\n", target, index);
    } else {
        printf("Element %d not found\n", target);
    }

    return 0;
}
```

5.3.2 Count Occurrences

```
int count_occurrences(int arr[], int size, int target) {
    int count = 0;

    for (int i = 0; i < size; i++) {
        if (arr[i] == target) {
            count++;
        }
    }

    return count;
}
```

5.3.3 Find All Occurrences

```
#include <stdio.h>
#define MAX_INDICES 100

int find_all_occurrences(int arr[], int size, int target, int indices[]) {
    int count = 0;

    for (int i = 0; i < size && count < MAX_INDICES; i++) {
        if (arr[i] == target) {
            indices[count] = i;
```

```
            count++;
        }
    }

    return count;  // Number of occurrences found
}

// Usage example
int main() {
    int numbers[] = {1, 3, 7, 3, 9, 3, 2};
    int size = sizeof(numbers) / sizeof(numbers[0]);
    int target = 3;
    int indices[MAX_INDICES];

    int count = find_all_occurrences(numbers, size, target, indices);

    printf("Element %d found %d times at indices: ", target, count);
    for (int i = 0; i < count; i++) {
        printf("%d ", indices[i]);
    }
    printf("\n");

    return 0;
}
```

5.4 Array Modification Operations

5.4.1 Filling Arrays

```
void fill_array(int arr[], int size, int value) {
    for (int i = 0; i < size; i++) {
        arr[i] = value;
    }
}

// Fill with sequential numbers
void fill_sequence(int arr[], int size, int start) {
    for (int i = 0; i < size; i++) {
        arr[i] = start + i;
```

```
    }
}
```

5.4.2 Copying Arrays

```
void copy_array(int source[], int destination[], int size) {
    for (int i = 0; i < size; i++) {
        destination[i] = source[i];
    }
}

// Usage
int main() {
    int original[] = {1, 2, 3, 4, 5};
    int copy[5];

    copy_array(original, copy, 5);

    return 0;
}
```

5.4.3 Reversing Arrays

```
void reverse_array(int arr[], int size) {
    for (int i = 0; i < size / 2; i++) {
        // Swap elements
        int temp = arr[i];
        arr[i] = arr[size - 1 - i];
        arr[size - 1 - i] = temp;
    }
}
```

Revision #1
Created 2025-09-15 00:55:52 UTC by DS
Updated 2025-09-15 00:56:22 UTC by DS