header.h : This is the header file with struct definitions and function prototypes

```c
typedef struct Array{
    int *A;
    int size;
    int len;
} Array;

void init(Array *a, int size);
void append(Array *a, int element);
void insertAtIndex(Array *a, int element, int position);
int removeAtIndex(Array *a, int position);
void display(Array a);
int max(Array a);
int min(Array a);
Array merge(Array a1, Array a2);
void sort(Array *a);
void fill(Array *a);
void reverse(Array *a);
```

logic.c : contains all the functions.

```c
#include "./header.h"
#include<stdio.h>
#include<stdlib.h>
#include<limits.h>


//function to initialize the array
void init(Array *a, int size){
    a->A = (int *)malloc(sizeof(int) * size); //allocate memory
    if(!a->A) return; // if memory allocation fails, return

    a->size = size; //set size
    a->len = 0;
}
```

```c
//function to append an element to array
void append(Array *a, int element){
    if(a->len ==  a->size) return; //if array is full return

    a->A[a->len++] = element; // append element
    return;
}
```

```c
// function to insert an element at a particular index
void insertAtIndex(Array *a, int element, int position){
    int length = a->len;
    // if the index is invalid or if the array is full return
    if (length == a->size || position < 0 || position > length)
return;

    // shift elements to the next index
    for(int i = length; i >= position; i--){
        a->A[i] = a->A[i - 1];
    }

    // add element at index
    a->A[position] = element;
    a->len++; //increment the length
    return;
}
```

```c
// function to remove element at a particular index
int removeAtIndex(Array *a, int position){
    int length = a->len;
    // if the index is invalid return
    if (position < 0 || position >= length) return INT_MIN;

    // store the element to remove as we will return it
    int removedElement = a->A[position];

    // shift elements after the index to one position back
    for(int i = position + 1; i < length; i++){
        a->A[i - 1] = a->A[i];
    }
    a->len--; //decrement the length
    return removedElement; //return the removed element
}
```

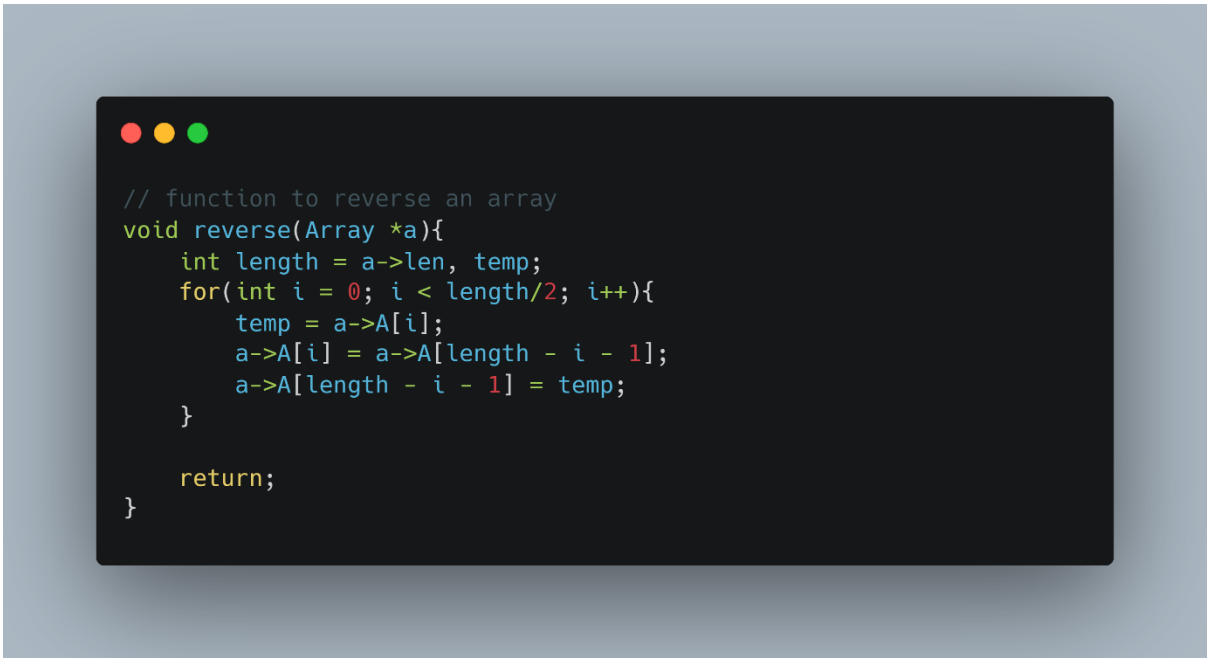
```c
// function to display the array
void display(Array a){
    int length = a.len;
    // if array is empty return
    if(length == 0){
        printf("Array is empty!");
        return;
    }

    printf("Displaying Array: [");

    for (int i = 0; i < length; i++)
    {
        printf("%d, ", a.A[i]);
    }

    printf("\b\b]\n"); //remove the ending comma
    return;
}
```

```
// function to get the maximum element
int max(Array a){
    // if array is empty return
    if(a.len == 0) return INT_MIN;
    int maxElement = a.A[0], length = a.len;

    // traverse through entire array and check for the maximum
element
    for(int i = 1; i < length; i++){
        int currentElement = a.A[i];
        if(currentElement > maxElement) maxElement = currentElement;
    }
    return maxElement;
}

// function to get the minimum element
int min(Array a){
    // if array is empty return
    if(a.len == 0) return INT_MAX;
    int minElement = a.A[0], length = a.len;

    // traverse through entire array and check for the minimum
element
    for(int i = 1; i < length; i++){
        int currentElement = a.A[i];
        if(currentElement < minElement) minElement = currentElement;
    }
    return minElement;
}
```

```
// function to fill array with random elements
void fill(Array *a){
    int length = a->len;

    for (int i = length; i < a->size; i++){
        a->A[i] = rand() % 100; // random number from 0 - 99
    }

    a->len = a->size;
    return;
}
```

```
// function to merge two arrays
Array merge(Array a1, Array a2){
    Array merged; //create a new array
    int length1 = a1.len , length2 = a2.len;
    init(&merged, length1 + length2); // initialze the new array with
length = length1 + length2

    //append elements of first array
    for(int i = 0; i < length1; i++){
        append(&merged, a1.A[i]);
    }

    //append elements of second array
    for(int i = 0; i < length2; i++){
        append(&merged, a2.A[i]);
    }

    return merged;
}
```

```
//simple selection sort algorithm

void swap(int *x, int *y){
    int temp = *x;
    *x = *y;
    *y = temp;
}


void sort(Array *a){
    int length = a->len;
    int i, j, min;
    for(i = 0; i < length - 1; i++){
        min = i;
        for(j = i + 1; j < length; j++){
            if(a->A[j] < a->A[min]) min = j;
        }
        swap(&a->A[i], &a->A[min]);
    }
}
```



```c
// function to reverse an array
void reverse(Array *a){
    int length = a->len, temp;
    for(int i = 0; i < length/2; i++){
        temp = a->A[i];
        a->A[i] = a->A[length - i - 1];
        a->A[length - i - 1] = temp;
    }

    return;
}
```

main.c

```c
#include <stdio.h>
#include "logic.c"

#define MAX_ARRAYS 10

void printMenu();
void handleChoice(int choice, Array* arrays[], int maxArrays);

int main() {
    int choice;
    Array *arrays[MAX_ARRAYS] = {NULL};
    while(1){
        printMenu();
        scanf("%d", &choice);
        handleChoice(choice, arrays, MAX_ARRAYS);
    }

    return 0;
}
```

```c
// function to print a menu for user
void printMenu(){
    printf("\n\n***********Array ADT Menu:***********\n\n");
    printf("Select any of the instructions given below:\n");
    printf("1. Initialize a new Array.\n");
    printf("2. Append an element to an existing Array.\n");
    printf("3. Insert element at a given index.\n");
    printf("4. Remove element at a given index.\n");
    printf("5. Display an Array.\n");
    printf("6. Display the maximum element in an Array.\n");
    printf("7. Display the minimum element in an Array.\n");
    printf("8. Reverse an Array.\n");
    printf("9. Merge two Arrays.\n");
    printf("10. Fill an Array with random elements\n");
    printf("11. Exit\n");
    printf("Enter your choice: ");
}
```

```c
// function to handle user choices
void handleChoice(int choice, Array *arrays[], int maxArrays){
    int arrayNumber;
    if (choice != 1 && choice != 11) {
        printf("\nEnter array number (0-%d): ", maxArrays - 1);
        scanf("%d", &arrayNumber);

        if (arrayNumber < 0 || arrayNumber >= maxArrays) {
            printf("Invalid array number!\n");
            return;
        }

        if (arrays[arrayNumber] == NULL) {
            printf("Array %d is not initialized. Please initialize it
first.\n", arrayNumber);
            return;
        }
    }
```

```
    switch (choice) {
        case 1: {
            int size;
            printf("\nEnter array number (0-%d): ", maxArrays - 1);
            scanf("%d", &arrayNumber);

            if (arrayNumber < 0 || arrayNumber >= maxArrays) {
                printf("Invalid array number!\n");
                return;
            }

            printf("\nEnter the size of the Array: ");
            scanf("%d", &size);
            if (arrays[arrayNumber] == NULL) {
                arrays[arrayNumber] = (Array *)malloc(sizeof(Array));
            }
            init(arrays[arrayNumber], size);

            printf("\nArray %d initialized with size %d.\n",
arrayNumber, size);
            break;
        }
        case 2: {
            int element;
            printf("\nEnter the element to append to the Array: ");
            scanf("%d", &element);
            append(arrays[arrayNumber], element);
            printf("\nElement appended to array %d: %d.\n",
arrayNumber, element);
            break;
        }
```

```
        case 3: {
            int element, index;
            printf("\nEnter the element to add to the Array: ");
            scanf("%d", &element);
            printf("\nEnter the index at which the element is to be
added: ");
            scanf("%d", &index);
            insertAtIndex(arrays[arrayNumber], element, index);
            printf("\nElement %d inserted at index %d in array
%d.\n", element, index, arrayNumber);
            break;
        }
        case 4: {
            int index;
            printf("\nEnter the index at which the element is to be
removed: ");
            scanf("%d", &index);
            int removedElement = removeAtIndex(arrays[arrayNumber],
index);
            printf("\nRemoved Element from array %d: %d\n",
arrayNumber, removedElement);
            break;
        }
```

```
        case 5: {
            display(*arrays[arrayNumber]);
            break;
        }
        case 6: {
            int maxElement = max(*arrays[arrayNumber]);
            printf("\nMaximum element in the Array %d: %d\n",
arrayNumber, maxElement);
            break;
        }
        case 7: {
            int minElement = min(*arrays[arrayNumber]);
            printf("\nMinimum element in the Array %d: %d\n",
arrayNumber, minElement);
            break;
        }
        case 8: {
            reverse(arrays[arrayNumber]);
            printf("\nArray %d reversed.\n", arrayNumber);
            break;
        }
```

```
      case 9: {
            int arrayNumber2;
            printf("\nEnter the second array number (0-%d): ",
maxArrays - 1);
            scanf("%d", &arrayNumber2);

            if (arrayNumber2 < 0 || arrayNumber2 >= maxArrays ||
arrays[arrayNumber2] == NULL) {
                printf("Invalid second array number or array not
initialized!\n");
                return;
            }

            Array mergedArray = merge(*arrays[arrayNumber],
*arrays[arrayNumber2]);
            printf("\nArrays %d and %d merged.\n", arrayNumber,
arrayNumber2);
            display(mergedArray);
            break;
        }
        case 10: {
            fill(arrays[arrayNumber]);
            break;
        }
        case 11:
            printf("\nExiting....\n");
            exit(0);
            break;
        default:
            printf("\nInvalid choice. Please try again.\n");
            break;
    }
}
```

Output:

Menu: prints the menu

```
Assignment 1>gcc .\main.c -Wall -o main
Assignment 1>.\main.exe


**********Array ADT Menu:**********

Select any of the instructions given below:
1. Initialize a new Array.
2. Append an element to an existing Array.
3. Insert element at a given index.
4. Remove element at a given index.
5. Display an Array.
6. Display the maximum element in an Array.
7. Display the minimum element in an Array.
8. Reverse an Array.
9. Merge two Arrays.
10. Fill an Array with random elements
11. Exit
Enter your choice:
```

Initialize: Initialize an array

```
Enter your choice: 1

Enter array number (0-9): 0

Enter the size of the Array: 10

Array 0 initialized with size 10.
```

Append: append an element to the Array

```
Enter your choice: 2

Enter array number (0-9): 0

Enter the element to append to the Array: 12

Element appended to array 0: 12.
```

Insert at index: Insert an element at a given index.

```
Enter your choice: 3

Enter array number (0-9): 0

Enter the element to add to the Array: 23

Enter the index at which the element is to be added: 1

Element 23 inserted at index 1 in array 0.
```

remove element at index : remove element at a given index.

```
Enter your choice: 4

Enter array number (0-9): 0

Enter the index at which the element is to be removed: 0

Removed Element from array 0: 12
```

Display array: Display an array

```
Enter your choice: 5

Enter array number (0-9): 0
Displaying Array: [23]
```

Fill : fills array with random elements

```
 Enter your choice: 10

 Enter array number (0-9): 0
```

```
Enter your choice: 5

Enter array number (0-9): 0
Displaying Array: [23, 41, 67, 34, 0, 69, 24, 78, 58, 62]
```

Display Maximum: display the maximum element in the array.

```
Enter your choice: 6

Enter array number (0-9): 0

Maximum element in the Array 0: 78
```

Display Minimum: display the minimum element in the array

```
Enter your choice: 7

Enter array number (0-9): 0

Minimum element in the Array 0: 0
```

Reverse array: reverse an array

```
Enter your choice: 8

Enter array number (0-9): 0

Array 0 reversed.
```

```
Enter your choice: 5

Enter array number (0-9): 0
Displaying Array: [62, 58, 78, 24, 69, 0, 34, 67, 41, 23]
```

Merge: merge two arrays

```
Enter your choice: 5

Enter array number (0-9): 1
Displaying Array: [64, 5, 45, 81, 27, 61, 91, 95, 42, 27]
```

```
Enter your choice: 9

Enter array number (0-9): 0

Enter the second array number (0-9): 1

Arrays 0 and 1 merged.
Displaying Array: [62, 58, 78, 24, 69, 0, 34, 67, 41, 23, 64, 5, 45, 81, 27, 61, 91, 95, 42, 27]
```