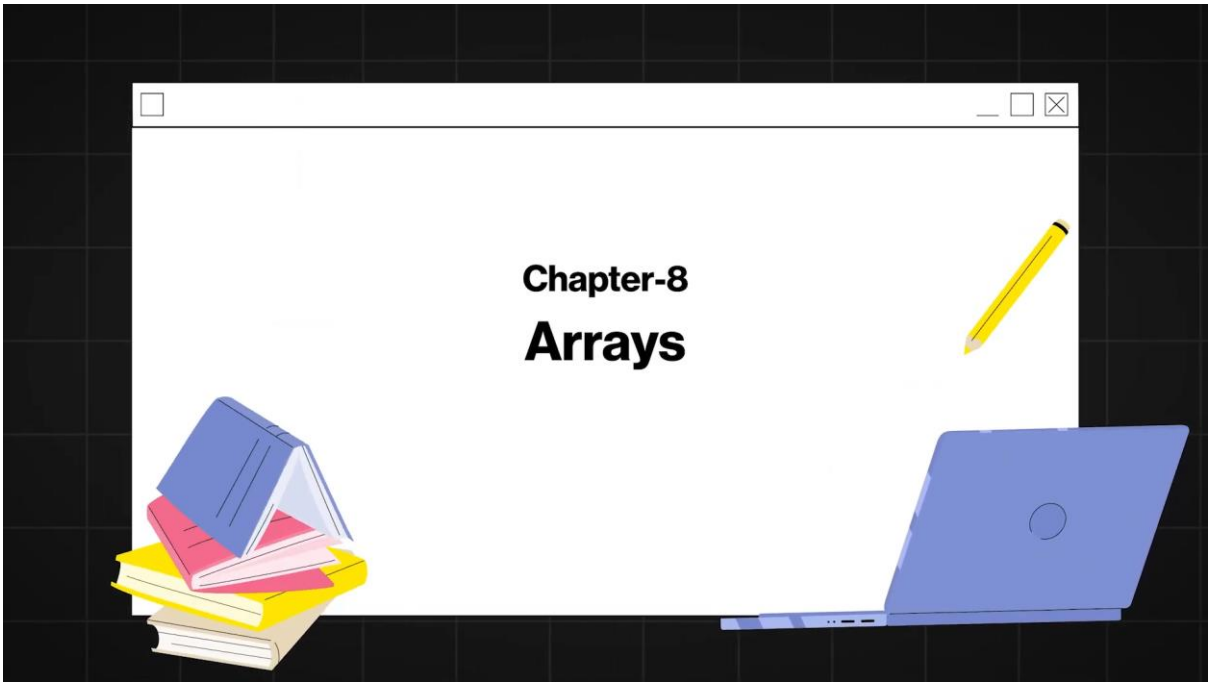
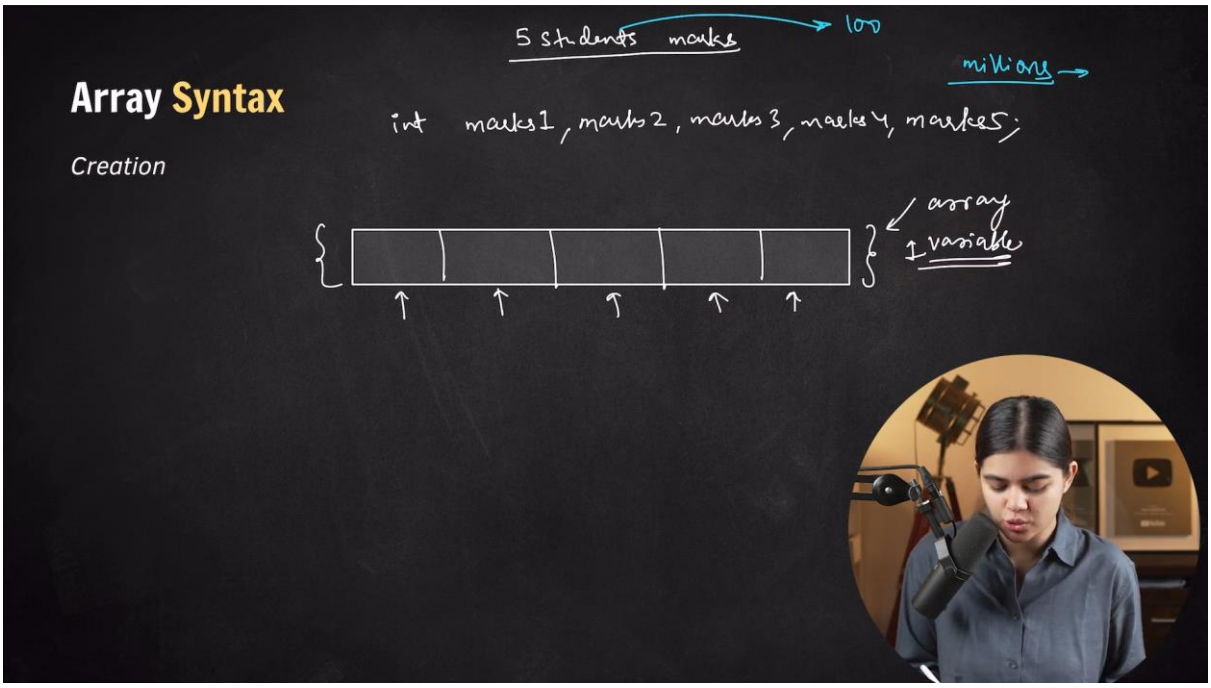
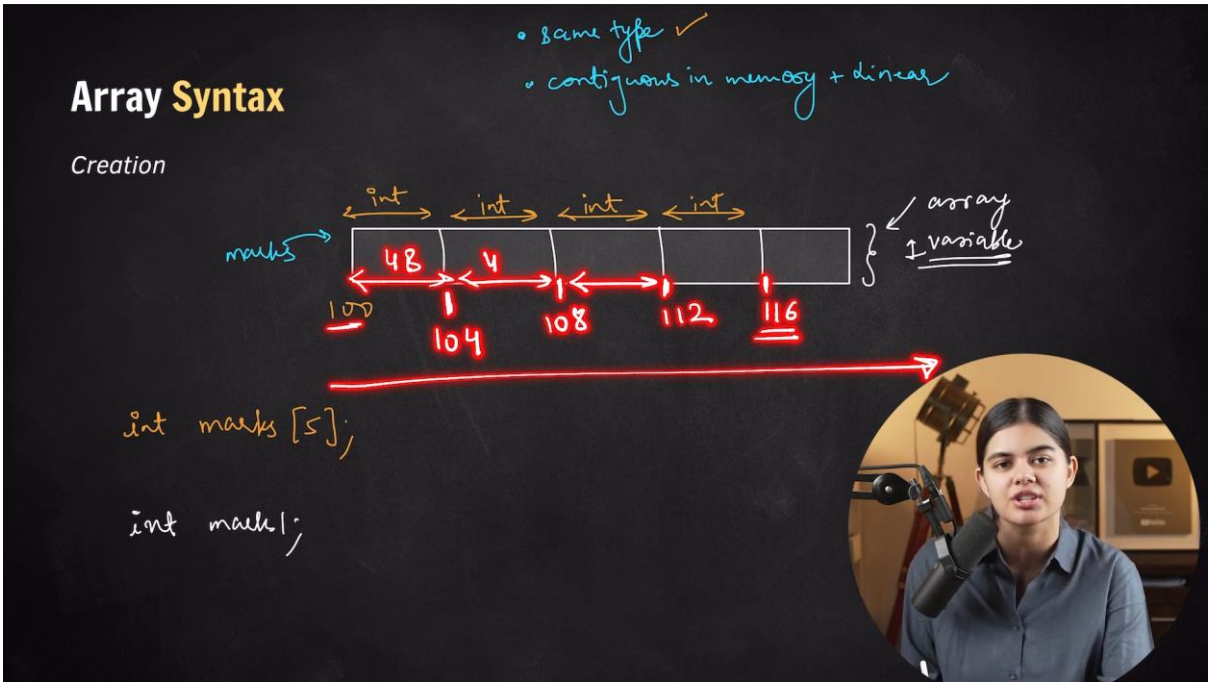
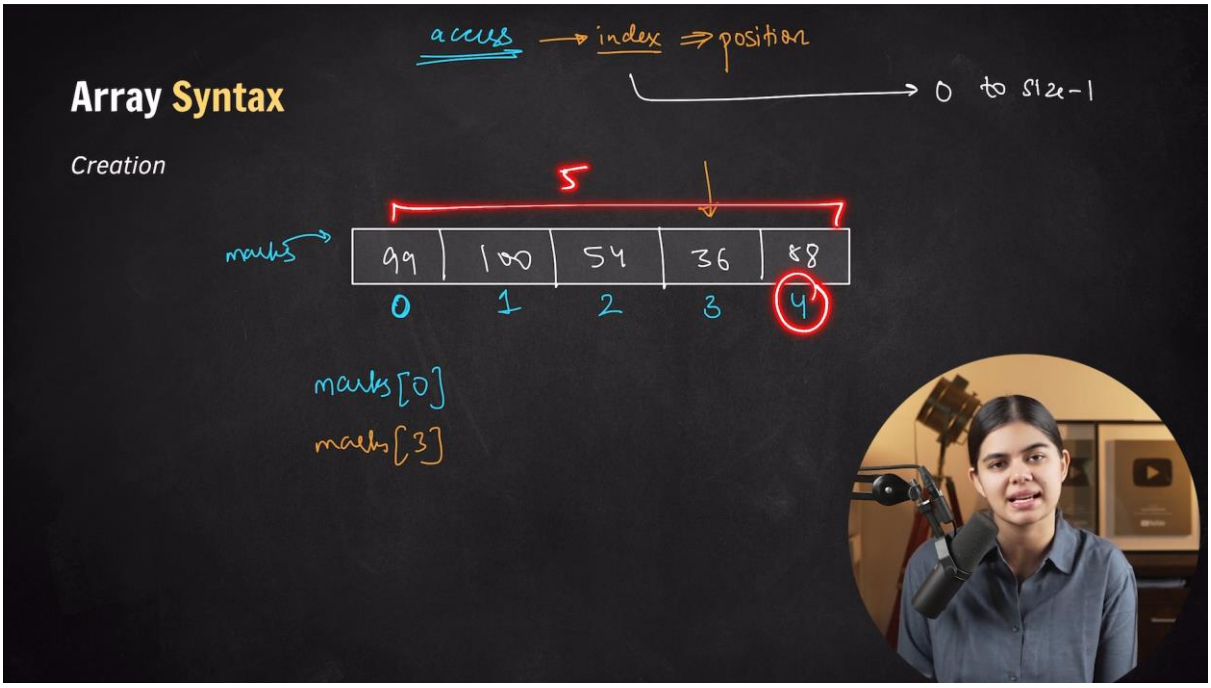
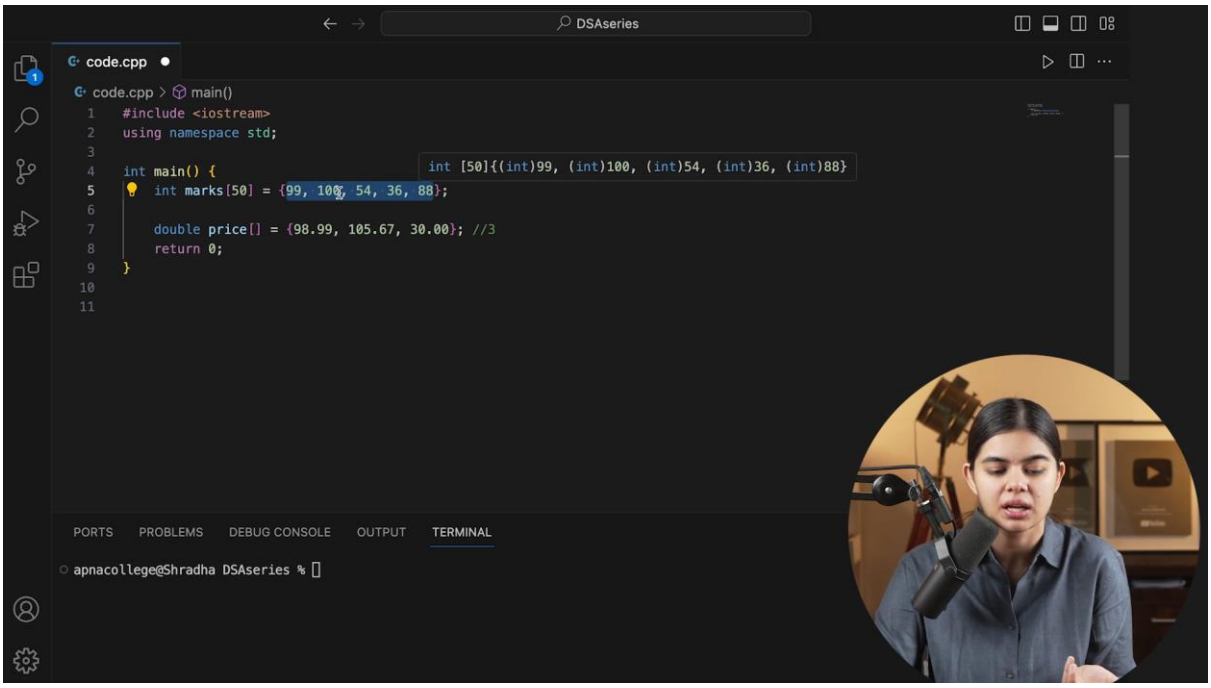

Chapter-8
Arrays


Arrays → 1st DS
Data Structures
linear
Hierarchial
store data
Algorithms
O/S


5 students marks → 100
millions →
Array Syntax
Creation
int marks1, marks2, marks3, marks4, marks5;
array
1 variable


Array Syntax
Creation
• same type
• contiguous in memory + linear
int
int
int
int
marks
48
4
100
104
108
112
116
array
1 variable
int marks[5];
int marks1;


access → index ⇒ position
0 to size-1
Array Syntax
Creation
5
marks
99
100
54
36
88
0
1
2
3
4
marks[0]
marks[3]


DSAseries
code.cpp
code.cpp > main()
#include <iostream>
using namespace std;
int main() {
int marks[50] = {99, 100, 54, 36, 88};
double price[] = {98.99, 105.67, 30.00}; //3
return 0;
}
int [50]{(int)99, (int)100, (int)54, (int)36, (int)88}
PORTS
PROBLEMS
DEBUG CONSOLE
OUTPUT
TERMINAL
apnacollege@Shradha DSAseries %

DSAseries

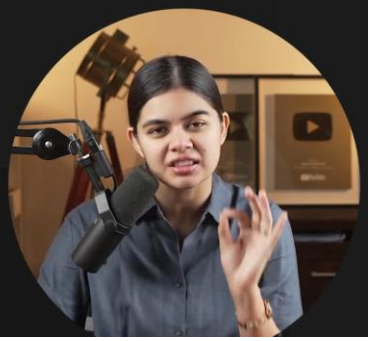
code.cpp

code.cpp > main()

```cpp
#include <iostream>
using namespace std;

int main() {
    int marks[5] = {99, 100, 54, 36, 88};
    marks[0] = 101;
    cout << marks[0] << endl;
    cout << marks[1] << endl;
    cout << marks[2] << endl;
    cout << marks[3] << endl;
    cout << marks[4] << endl;

    //0 to size-1
    cout << marks[-1] << endl;
    return 0;
}
```

PORTS PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL

```
apnacollege@Shradha DSAseries % g++ code.cpp && ./a.out
code.cpp:14:13: warning: array index -1 is before the beginning of the array [-Warray-bounds]
    cout << marks[-1] << endl;
            ^    ~~
code.cpp:5:5: note: array 'marks' declared here
    int marks[5] = {99, 100, 54, 36, 88};
    ^
1 warning generated.
```



DSAseries

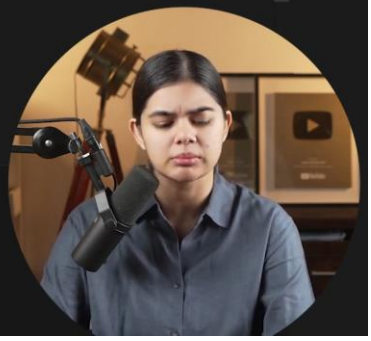
code.cpp

code.cpp > main()

```cpp
#include <iostream>
using namespace std;

int main() {
    int marks[5] = {99, 100, 54, 36, 88};
    int size = 5;

    //int sz = sizeof(marks) /
    cout << sizeof(marks) / sizeof(int) << endl;
    //loops : 0 to size-1

    return 0;
}
```

PORTS PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL

```
apnacollege@Shradha DSAseries % g++ code.cpp && ./a.out
20
apnacollege@Shradha DSAseries % g++ code.cpp && ./a.out
5
apnacollege@Shradha DSAseries %
```



DSAseries

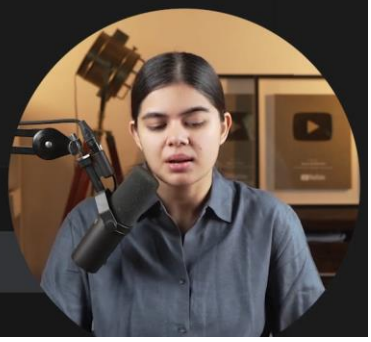
code.cpp

code.cpp > main()

```cpp
#include <iostream>
using namespace std;

int main() {
    int marks[5] = {99, 100, 54, 36, 88};
    int size = 5;

    //loops : 0 to size-1
    for(int i=0; i<size; i++) {
        cout << marks[i] << endl;
    }

    return 0;
}
```

PORTS PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL

```
apnacollege@Shradha DSAseries % g++ code.cpp && ./a.out
99
100
54
36
88
apnacollege@Shradha DSAseries %
```



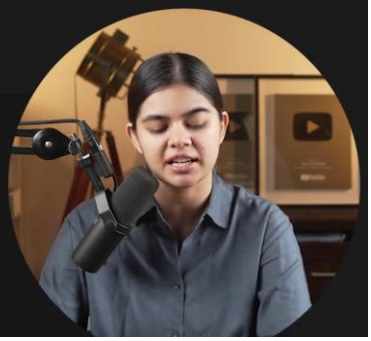
DSAseries

code.cpp

```cpp
#include <iostream>
using namespace std;

int main() {
    int size = 5;
    int marks[size];

    for(int i=0; i<size; i++) {
        cin >> marks[i];
    }

    //loops : 0 to size-1
    for(int i=0; i<size; i++) {
        cout << marks[i] << endl;
    }

    return 0;
```

PORTS PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL

```
apnacollege@Shradha DSAseries % g++ code.cpp && ./a.out
12 13 14 15 16

apnacollege@Shradha DSAseries %
```



Loops on Arrays

Find smallest/ largest in Array

int smallest = +∞ = INT_MAX

5

1

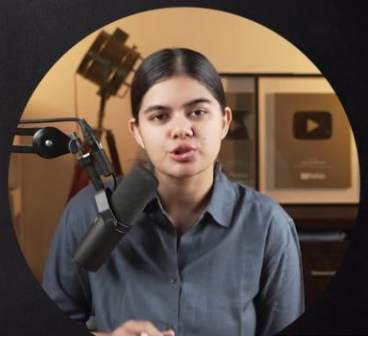
−15

nums

5	15	22	1	-15	24

if(nums[i] < smallest)

smallest = nums[i]



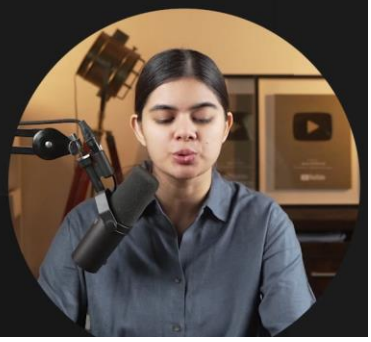
DSAseries

code.cpp

```cpp
#include <iostream>
using namespace std;

int main() {
    int nums[] = {5, 15, 22, 1, -15, -24};
    int size = 6;

    int smallest = INT_MAX;

    for(int i=0; i<size; i++) {//min, max
        if(nums[i] < smallest) {
            smallest = nums[i];
        }
    }

    cout << "smallest = " << smallest << endl;
    return 0;
}
```

PORTS PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL

```
apnacollege@Shradha DSAseries % g++ code.cpp && ./a.out
smallest = -24
apnacollege@Shradha DSAseries %
```



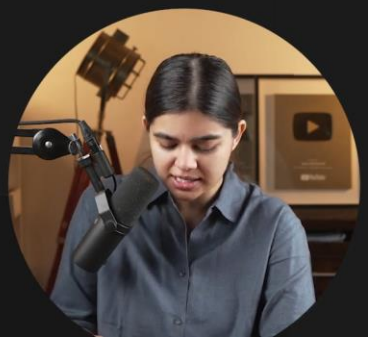
DSAseries

code.cpp

```cpp
#include <iostream>
using namespace std;

int main() {
    int nums[] = {5, 15, 22, 1, -15, -24};
    int size = 6;

    int smallest = INT_MAX;
    int largest = INT_MIN;

    for(int i=0; i<size; i++) {//min, max
        smallest = min(nums[i], smallest);
        largest = max(nums[i], largest);
    }

    cout << "smallest = " << smallest << endl;
    cout << "largest = " << largest << endl;
    return 0;
}
```

PORTS PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL

```
apnacollege@Shradha DSAseries % g++ code.cpp && ./a.out
smallest = -24
largest = 22
apnacollege@Shradha DSAseries %
```



Loops on Arrays

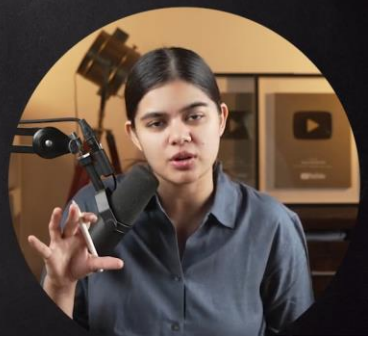
smallest idx , largest idx

Find smallest/ largest in Array

nums

5	15	22	1	-15	24

if(nums[i] < smallest)
smallest = nums[i]

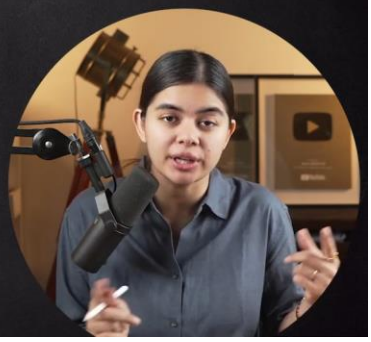


Pass by reference

address

Pass by Value

In C++ a reference is an alternative name for an object or function, and its address is the address of the object or function it refers to. For sake of understanding we are calling it address here.



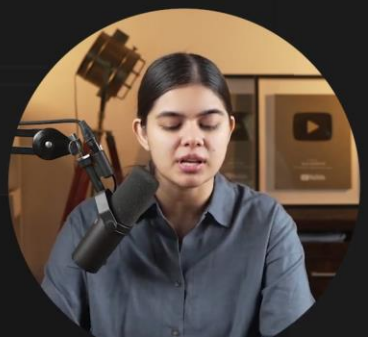
DSAseries

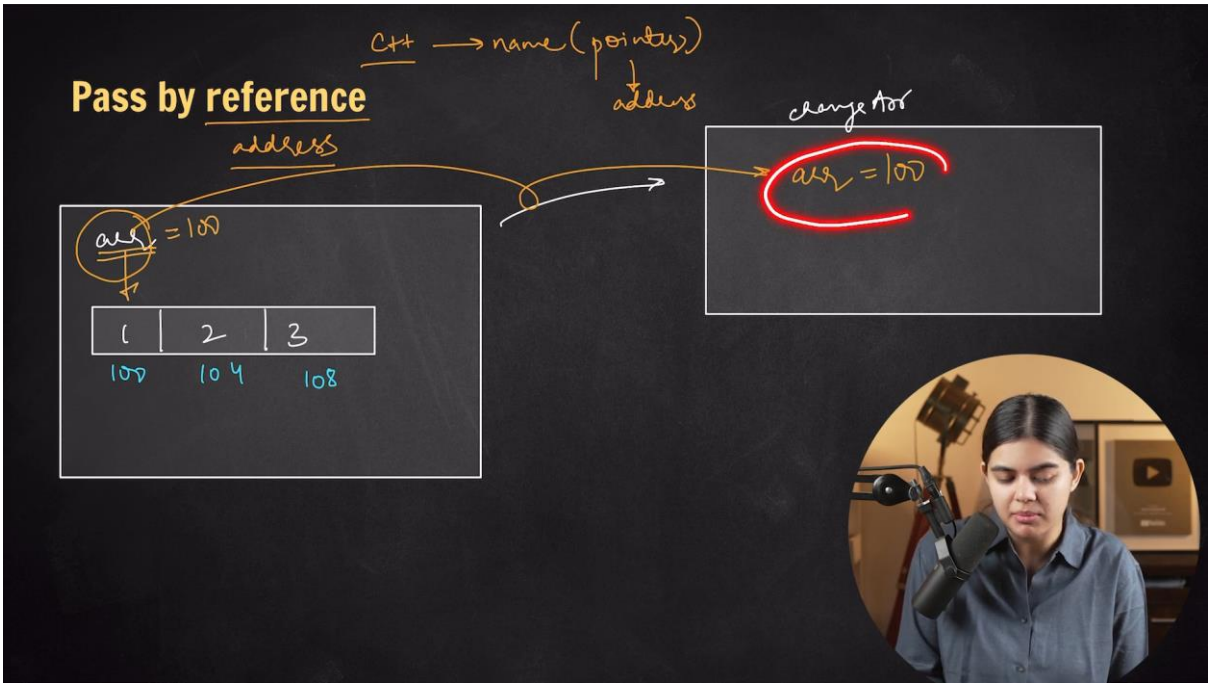
code.cpp

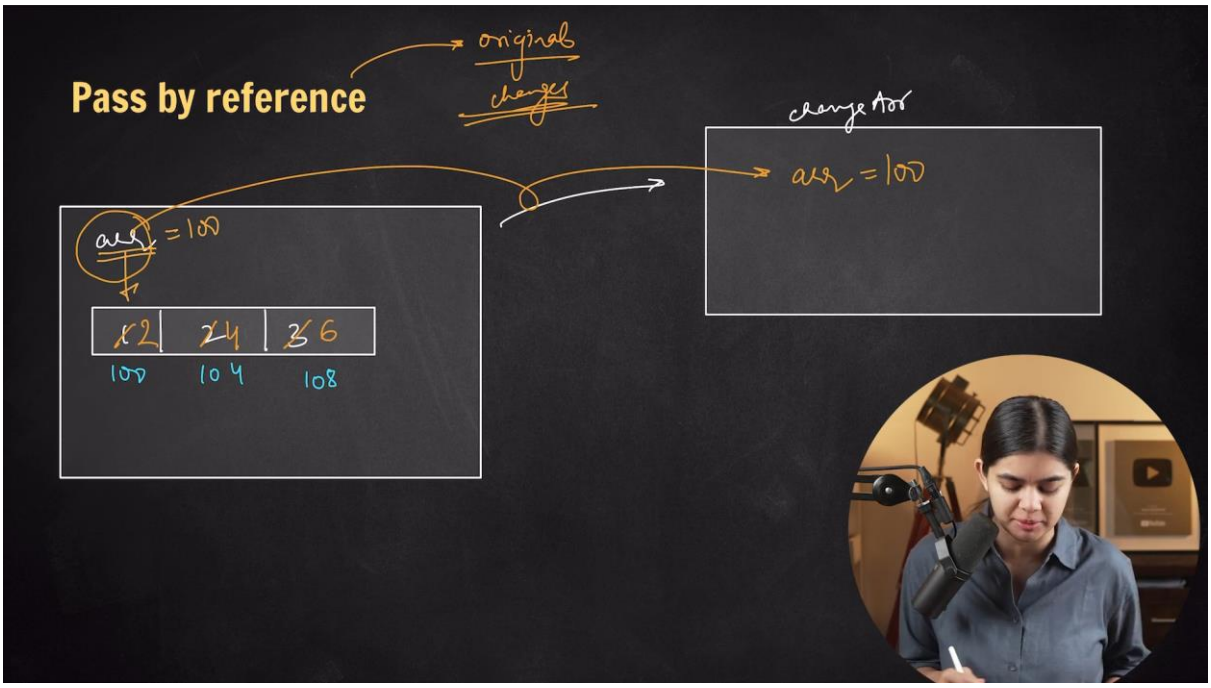
```cpp
#include <iostream>
using namespace std;

void changeArr(int arr[], int size) {
    cout << "in function\n";
    for(int i=0; i<size; i++) {
        arr[i] = 2* arr[i];
    }
}

int main() {
    int arr[] = {1, 2, 3};

    changeArr(arr, 3);

    cout << "in main\n";
    for(int i=0; i<3; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

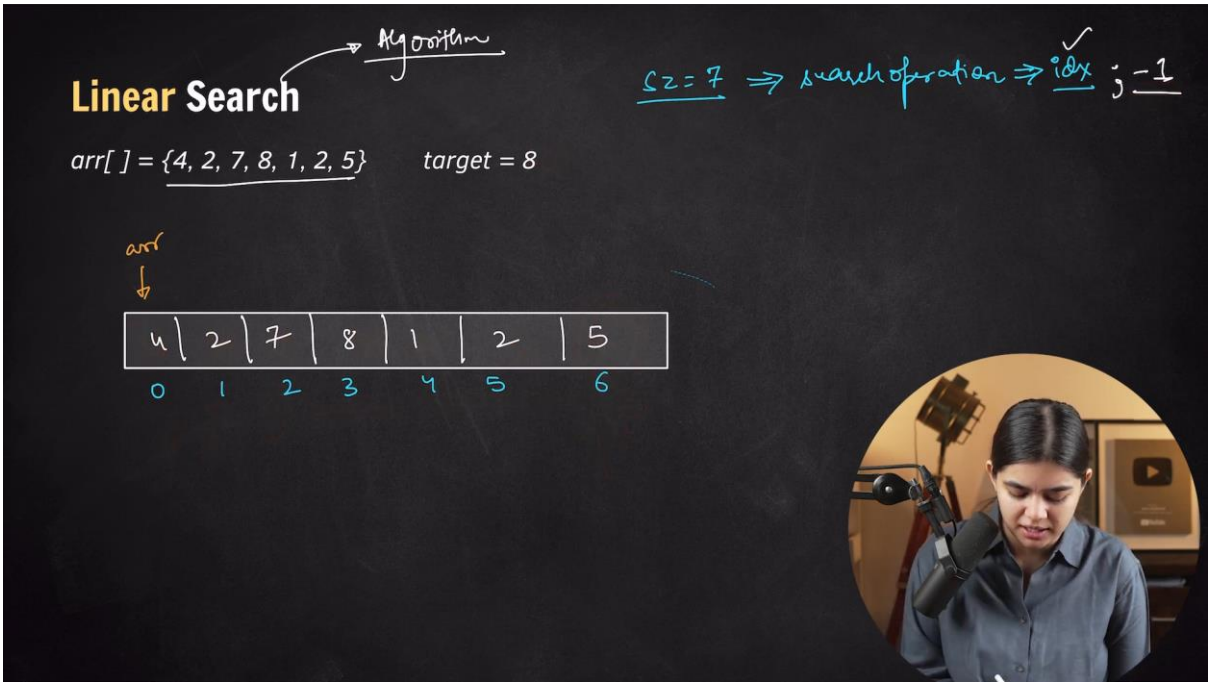
PORTS PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL

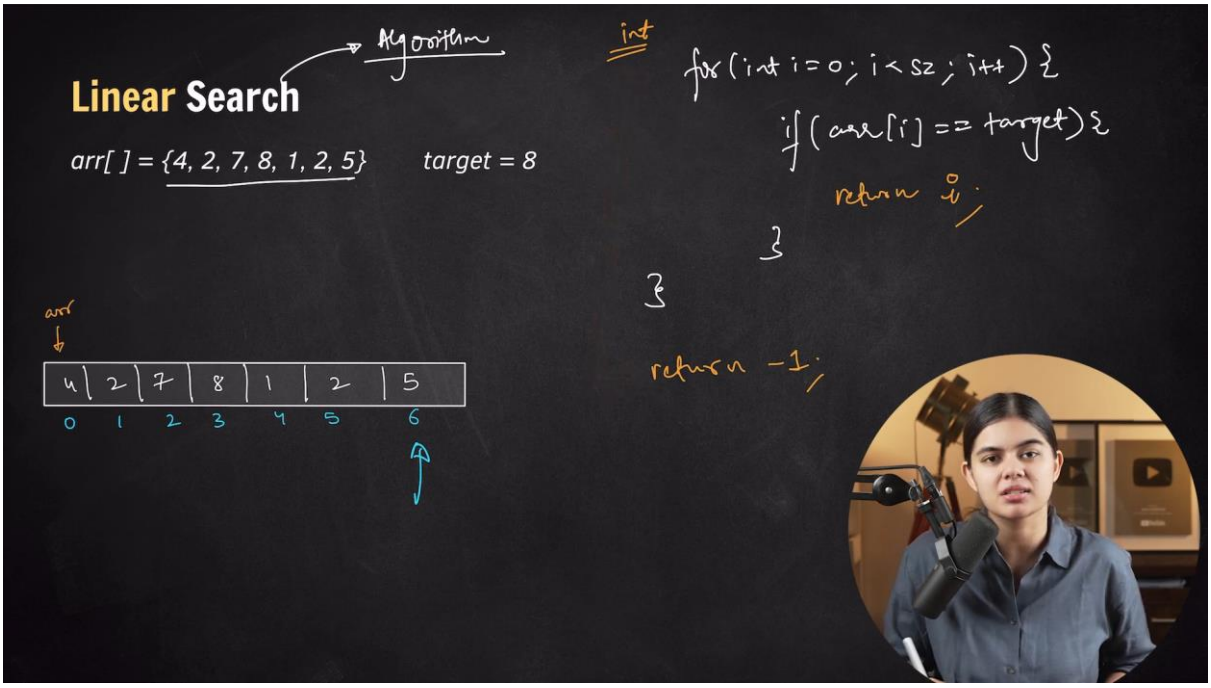
```
apnacollege@Shradha DSAseries % g++ code.cpp && ./a.out
in function
in main
2 4 6
apnacollege@Shradha DSAseries %
```

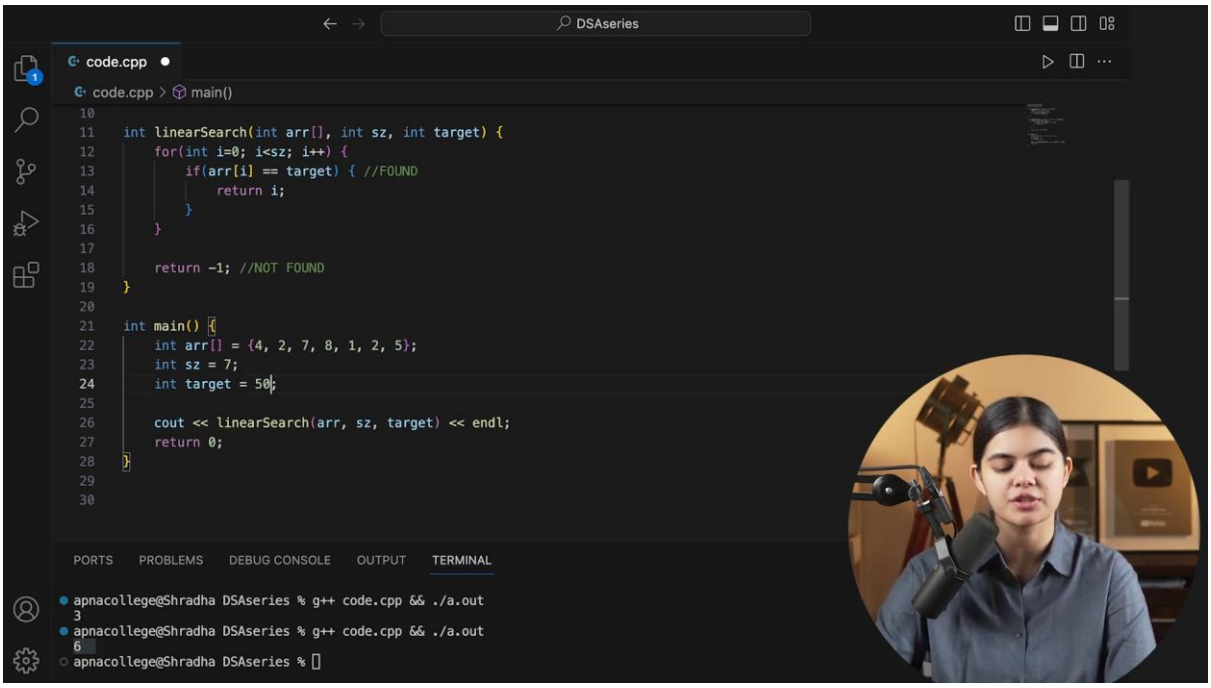







Linear Search
Algorithm
sz = 7 ⇒ search operation ⇒ idx ; -1
arr[ ] = {4, 2, 7, 8, 1, 2, 5}
target = 8
arr
4 2 7 8 1 2 5
0 1 2 3 4 5 6


Linear Search
Algorithm
arr[ ] = {4, 2, 7, 8, 1, 2, 5}
target = 8
int
for(int i=0; i<sz; i++) {
if(arr[i] == target) {
return i;
}
}
return -1;
arr
4 2 7 8 1 2 5
0 1 2 3 4 5 6


DSAseries
code.cpp
code.cpp > main()
int linearSearch(int arr[], int sz, int target) {
for(int i=0; i<sz; i++) {
if(arr[i] == target) { //FOUND
return i;
}
}
return -1; //NOT FOUND
}
int main() {
int arr[] = {4, 2, 7, 8, 1, 2, 5};
int sz = 7;
int target = 50;
cout << linearSearch(arr, sz, target) << endl;
return 0;
}
PORTS PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL
apnacollege@Shradha DSAseries % g++ code.cpp && ./a.out
3
apnacollege@Shradha DSAseries % g++ code.cpp && ./a.out
6
apnacollege@Shradha DSAseries %

Linear Search

Algorithm

Time Complexity ⇒ O(n)

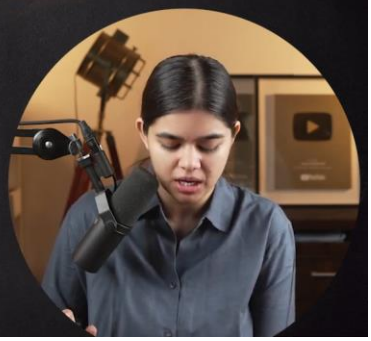
linear

Binary Search O(log n)

arr[ ] = {4, 2, 7, 8, 1, 2, 5} target = 8

arr

4	2	7	8	1	2	5
0	1	2	3	4	5	6



Reverse an Array

2nd

arr[ ] = {4, 2, 7, 8, 1, 2, 5} [5, 2, 1, 8, 7, 2, 4]

4	2	7	8	1	2	5

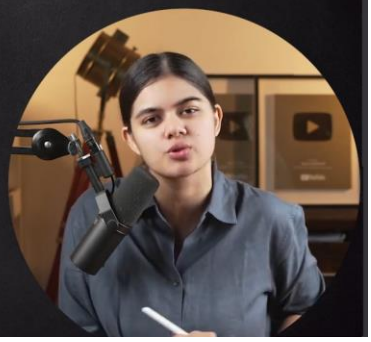
Reverse an Array

strings BS

* 2 pointer approach

arr[ ] = {4, 2, 7, 8, 1, 2, 5}

4	2	7	8	1	2	5



Reverse an Array

arr[ ] = {4, 2, 7, 8, 1, 2, 5}

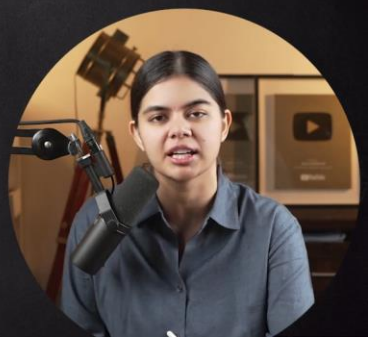
start = 0 1 2 end = 6 5 4

4	2	7	8	1	2	5
5	2	1	8	7	2	4

* 2 pointer approach

start = 0 ; end = size - 1

↓ ++ ↓ --



Reverse an Array

arr[ ] = {4, 2, 7, 8, 1, 2, 5}

start = 0 1 2 end = 6 5 4

4	2	7	8	1	2	5
5	2	1	8	7	2	4

even st > end

end st

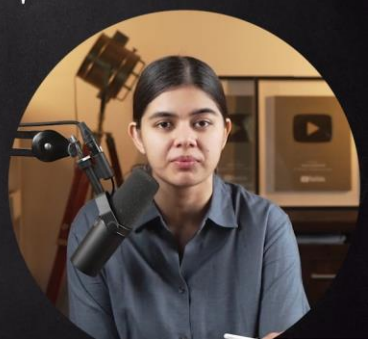
odd start = end

* 2 pointer approach

start = 0 ; end = size - 1

↓ ++ ↓ --

```
while ( st < end) {
swap(arr[start], arr[end])
}
```



Reverse an Array

arr[ ] = {4, 2, 7, 8, 1, 2, 5}

end = 1 st = 2

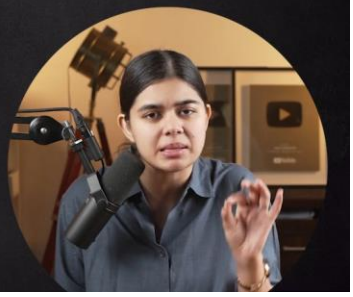
4	3	2	1

* 2 pointer approach

start = 0 ; end = size - 1

↓ ++ ↓ --

```
while ( st < end) {
swap(arr[start], arr[end])
start++; end--;
}
```



DSAseries

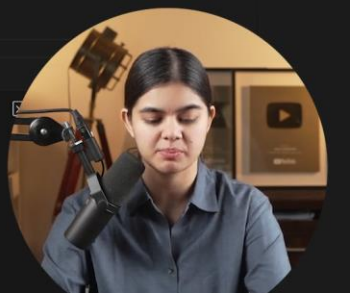
code.cpp

code.cpp > main()

```cpp
void reverseArray(int arr[], int sz) {
    int start = 0, end = sz-1;

    while(start < end) {
        swap(arr[start], arr[end]);
        start++;
        end--;
    }
}

int main() {
    int arr[] = {4, 2, 7, 8, 1, 2, 5};
    int sz = 7;

    reverseArray(arr, sz);

    for(int i=0; i<sz; i++) {
        cout << arr[i] << " ";
    }
    return 0;
```

PORTS PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL

```
apnacollege@Shradha DSAseries %
```



DSAseries

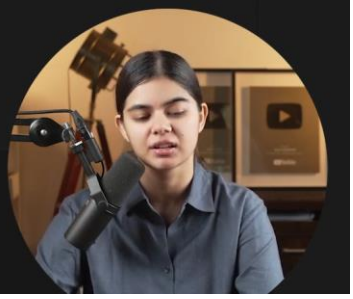
code.cpp

code.cpp > main()

```cpp
void reverseArray(int arr[], int sz) {
    int start = 0, end = sz-1;

    while(start < end) {
        swap(arr[start], arr[end]);
        start++;
        end--;
    }
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int sz = 5;

    reverseArray(arr, sz);

    for(int i=0; i<sz; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
    return 0;
```

PORTS PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL

```
apnacollege@Shradha DSAseries % g++ code.cpp && ./a.out
5 4 3 2 1
apnacollege@Shradha DSAseries %
```



array — create
array → index

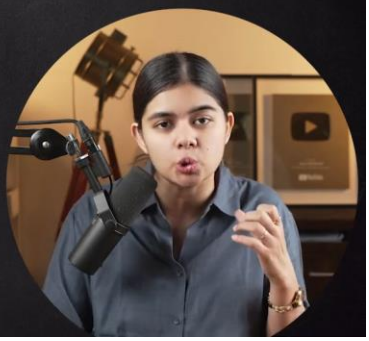
inp/out

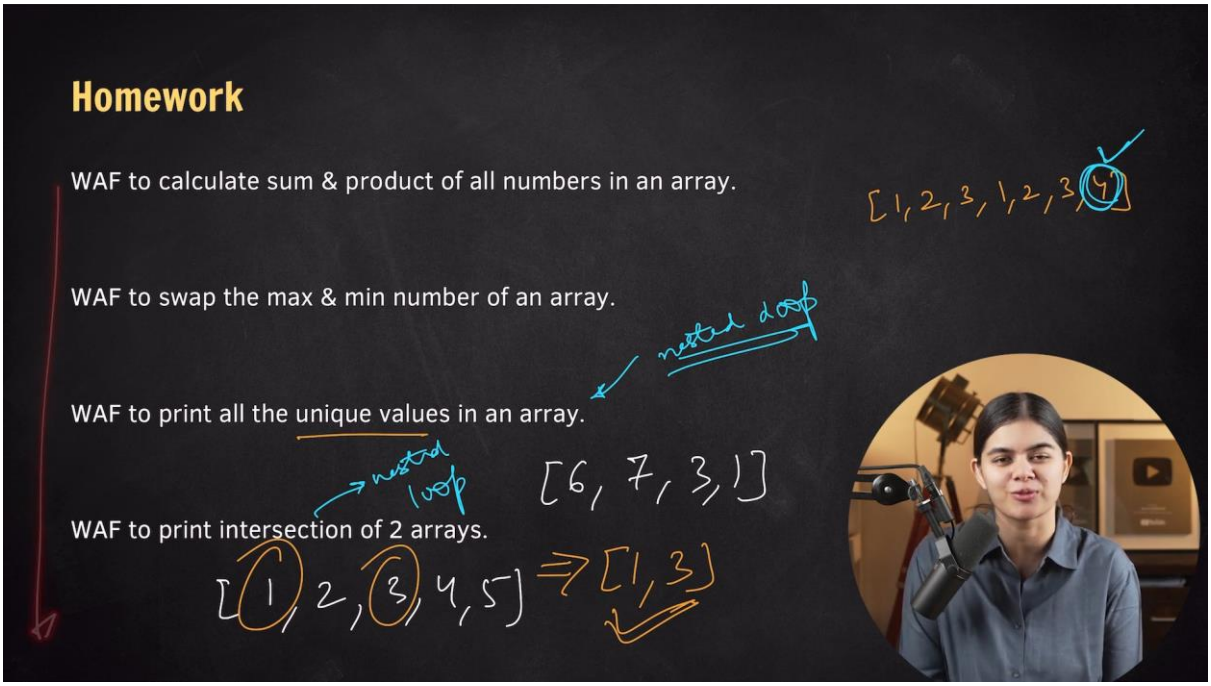
loops — min
— max

LinearSearch

Reverse

array → vector





Homework Solutions:

```
// WAF to calculate sum & product of all numbers in an array.


#include <iostream>

using namespace std;


int sumProduct(int arr[], int size) {

   int sum = 0, product = 1;


   for(int i = 0; i < size; i++) {

      sum += arr[i];

      product *= arr[i];

   }


   cout << "Sum = " << sum << endl;

   cout << "Product = " << product << endl;


   return 0;

}


int main() {

   int arr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9};

   int size = 9;
```

```
    sumProduct(arr, size);

    return 0;
}


// WAF to swap the max & min number of an array.

#include <iostream>
using namespace std;

int main() {
    int nums[] = {5, 23, 45, 28, 91, -19, -92};
    int size = sizeof(nums) / sizeof(int);

    int mini = INT_MAX;
    int maxi = INT_MIN;

    for(int i = 0; i < size; i++) {
        if(nums[i] < mini) {
            mini = nums[i];
        }

        // min = max(nums[i], min);
        maxi = max(nums[i], maxi);
    }

    cout << "min: " << mini << endl;
    cout << "max: " << maxi << endl;

    return 0;
}
```

```
// WAF to print all the unique values in an array.

#include <iostream>
using namespace std;

void printUnique(int arr[], int size) {
   cout << "Unique elements in the array are: ";

   for (int i = 0; i < size; i++) {
      bool isUnique = true;

      // Check if the element has appeared before in the array
      for (int j = 0; j < i; j++) {
         if (arr[i] == arr[j]) {
            isUnique = false;
            break;
         }
      }
      // If the element is unique, print it
      if (isUnique) {
         cout << arr[i] << " ";
      }
   }

   cout << endl;
}

int main() {
   int arr[] = {1, 2, 3, 4, 5, 3, 2, 1, 6, 7};
   int size = sizeof(arr) / sizeof(arr[0]);

   printUnique(arr, size);

   return 0;
}
```

```
// WAF to print intersection of 2 arrays.

#include <iostream>
using namespace std;

void commonElement(int arr1[], int arr2[], int size1, int size2) {

   for(int i = 0; i < size1; i++) {
      for(int j = 0; j < size2; j++) {
         if(arr1[i] == arr2[j]) {
            cout << arr1[i] << " ";
            break;
         }
      }
   }

   cout << endl;
   return;
}

int main() {
   int arr1[] = {1, 2, 3, 4, 5, 6, 7};
   int arr2[] = {4, 5, 3, 2, 8, 9};

   int size1 = 7;
   int size2 = 6;

   commonElement(arr1, arr2, size1, size2);
   return 0;
}

// #include <iostream>
// using namespace std;

// void commonElement(int arr1[], int arr2[], int size1, int size2) {
```

```
//    for(int i = 0; i < size1; i++) {
//        for(int j = 0; j < size2; j++) {
//            if(arr1[i] == arr2[j]) {
//                cout << arr1[i] << " ";
//                break; // Avoid printing duplicates
//            }
//        }
//    }
//    cout << endl;
// }

// int main() {
//    int arr1[] = {1, 2, 3, 4, 5, 6, 7};
//    int arr2[] = {4, 5, 3, 2, 8, 9};

//    int size1 = sizeof(arr1)/sizeof(arr1[0]);
//    int size2 = sizeof(arr2)/sizeof(arr2[0]);

//    commonElement(arr1, arr2, size1, size2);
//    return 0;
// }
```