# Print All Distinct Elements of a given integer

# array

* Simple way as below.
* Sorted first and then find unique values using the way below, or use std::sort, std::unique.
* Use set (similar to the hash way next).

Given an integer array, print all distinct elements in array. The given array may contain duplicates and the output should print every element only once. The given array is not sorted.

Examples:

Input: arr[] = {12, 10, 9, 45, 2, 10, 10, 45}

Output: 12, 10, 9, 45, 2

Input: arr[] = {1, 2, 3, 4, 5}

Output: 1, 2, 3, 4, 5

Input: arr[] = {1, 1, 1, 1, 1}

Output: 1

A **Simple Solution**is to use twp nested loops. The outer loop picks an element one by one starting from the leftmost element. The inner loop checks if the element is present on left side of it. If present, then ignores the element, else prints the element. Following is C++ implementation of the simple algorithm.

|  |
| --- |
| // C++ program to print all distinct elements in a given array  #include <iostream>  #include <algorithm>  using namespace std;    void printDistinct(int arr[], int n)  {      // Pick all elements one by one      for (int i=0; i<n; i++)      {          // Check if the picked element is already printed          int j;          for (j=0; j<i; j++)             if (arr[i] == arr[j])                 break;            // If not printed earlier, then print it          if (i == j)            cout << arr[i] << " ";      }  }    // Driver program to test above function  int main()  {      int arr[] = {6, 10, 5, 4, 9, 120, 4, 6, 10};      int n = sizeof(arr)/sizeof(arr[0]);      printDistinct(arr, n);      return 0;  } |

Time Complexity of above solution is O(n2). We can **Use Sorting** to solve the problem in O(nLogn) time. The idea is simple, first sort the array so that all occurrences of every element become consecutive. Once the occurrences become consecutive, we can traverse the sorted array and print distinct elements in O(n) time. Following is C++ implementation of the idea.

|  |
| --- |
| // C++ program to print all distinct elements in a given array  #include <iostream>  #include <algorithm>  using namespace std;    void printDistinct(int arr[], int n)  {      // First sort the array so that all occurrences become consecutive      sort(arr, arr + n);        // Traverse the sorted array      for (int i=0; i<n; i++)      {         // Move the index ahead while there are duplicates         while (i < n-1 && arr[i] == arr[i+1])            i++;           // print last occurrence of the current element         cout << arr[i] << " ";      }  }    // Driver program to test above function  int main()  {      int arr[] = {6, 10, 5, 4, 9, 120, 4, 6, 10};      int n = sizeof(arr)/sizeof(arr[0]);      printDistinct(arr, n);      return 0;  } |

4 5 6 9 10 120

We can **Use**[**Hashing**](http://quiz.geeksforgeeks.org/hashing-set-1-introduction/) to solve this in O(n) time on average. The idea is to traverse the given array from left to right and keep track of visited elements in a hash table. Following is Java implementation of the idea.

|  |
| --- |
| /\* Java program to print all distinct elements of a given array \*/  import java.util.\*;    class Main  {      // This function prints all distinct elements      static void printDistinct(int arr[])      {          // Creates an empty hashset          HashSet<Integer> set = new HashSet<>();            // Traverse the input array          for (int i=0; i<arr.length; i++)          {              // If not present, then put it in hashtable and print it              if (!set.contains(arr[i]))              {                  set.add(arr[i]);                  System.out.print(arr[i] + " ");              }          }      }        // Driver method to test above method      public static void main (String[] args)      {          int arr[] = {10, 5, 3, 4, 3, 5, 6};          printDistinct(arr);      }  } |

10 5 3 4 6