ICS 2105
Introduction to Data Structures

Arrays

Acknowledgement

- Notes adapted from tutorialspoint.com

Learning Outcomes

- By the end of this chapter, the learner should be able to:
  - Describe an array.
  - List operations supported by array data structure.
  - Write algorithms for the supported operations.
  - Write C++ programs to implement the operations.

Arrays

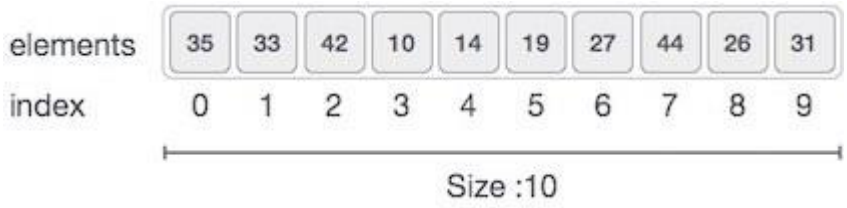
- Array is a container which can hold a fix number of items and these items should be of the same type.
- Most of the data structures make use of arrays to implement their algorithms.

Array Terms

- Element
  - Each item stored in an array is called an element.
- Index
  - Each location of an element in an array has a numerical index, which is used to identify the element.

Array Representation

- Arrays can be declared in various ways in different languages.



- Index starts with 0.
- Array length is 10 which means it can store 10 elements.
- Each element can be accessed via its index. For example, we can fetch an element at index 6 as 9.

Basic Operations Supported by Array

- **Traverse**
  - print all the array elements one by one.
- **Insertion**
  - Adds an element at the given index.
- **Deletion**
  - Deletes an element at the given index.
- **Search**
  - Searches an element using the given index or by the value.
- **Update**
  - Updates an element at the given index.

Insertion

- Insert operation is to insert one or more data elements into an array.
- Based on the requirement, a new element can be added at the beginning, end, or any given index of array.

Insertion Algorithm

- Let **Array** be a linear unordered array of **MAX** elements.
- Let **LA** be a Linear Array (unordered) with **N** elements and **K** is a positive integer such that **K<=N**.

Insertion Algorithm

- Following is the algorithm where ITEM is inserted into the K^{th} position of LA

1. Start
2. Set J = N
3. Set N = N+1
4. Repeat steps 5 and 6 while J >= K
5. Set LA[J+1] = LA[J]
6. Set J = J-1
7. Set LA[K] = ITEM
8. Stop

Insertion Program

```
#include<iostream>
using namespace std;
main() {
  int LA[] = {1,3,5,7,8};
  int item = 10, k = 3, n = 5;
  int i = 0, j = n;
  cout<<"\n\n The original array elements are :\n\n";
  for(i = 0; i<n; i++) {
    cout<<"LA["<<i<<"] = "<<LA[i]<<"\n";
                    }
  n = n + 1;
  while( j >= k) {
    LA[j+1] = LA[j];
    j = j - 1;
  }
  LA[k] = item;
  cout<<"\n \n The array elements after insertion :\n\n";

  for(i = 0; i<n; i++) {
     cout<<"LA["<<i<<"] = "<<LA[i]<<"\n";
  }
}
```

Deletion Operation

- Deletion refers to removing an existing element from the array and re-organizing all elements of an array.
- **Algorithm**
  - Consider **LA** is a linear array with **N** elements and **K** is a positive integer such that **K<=N**.

Deletion Operation

- Following is the algorithm to delete an element available at the K^{th} position of LA.

```
1. Start
2. Set J = K
3. Repeat steps 4 and 5 while J < N
4. Set LA[J] = LA[J + 1]
5. Set J = J+1
6. Set N = N-1
7. Stop
```

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

int main() {
  int LA[] = {1,3,5,7,8};
  int k = 3, n = 5;
  int i, j;

  cout<<"\n\n The original array elements are :\n\n";

  for(i = 0; i<n; i++) {
    cout<<"LA["<<i<<"] = "<<LA[i]<<"\n";
                    }
  j = k;
  while( j < n) {
    LA[j-1] = LA[j];
    j = j + 1;
  }
  n = n -1;
   cout<<"\n \n The array elements after deletion :\n\n";

  for(i = 0; i<n; i++) {
     cout<<"LA["<<i<<"] = "<<LA[i]<<"\n";
  }
  return 0;
}
```

Search Operation

- You can perform a search for an array element based on its value or its index.
- **Algorithm**
  - Consider **LA** is a linear array with **N** elements and **K** is a positive integer such that **K<=N**.

Search Operation

- Below is the algorithm to find an element with a value of ITEM using sequential search.

1. Start
2. Set J = 0
3. Repeat steps 4 and 5 while J < N
4. IF LA[J] is equal ITEM THEN
GOTO STEP 6
5. Set J = J +1
6. PRINT J, ITEM
7. Stop

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

int main() {
  int LA[] = {1,3,5,7,8};
  int item = 5, n = 5;
  int i = 0, j = 0;

  cout<<"\n\n The original array elements are :\n\n";

  for(i = 0; i<n; i++) {
    cout<<"LA["<<i<<"] = "<<LA[i]<<"\n";
                }

  while( j < n){
    if( LA[j] == item ) {
      break;
    }

    j = j + 1;
  }

  cout<<"\n\n Found element  "<<item<<" at position " <<j+1;
  return 0;
}
```

Update Operation

- Update operation refers to updating an existing element from the array at a given index.
- **Algorithm**
  - Consider **LA** is a linear array with **N** elements and **K** is a positive integer such that **K<=N**.

Update Operation

- Below is the algorithm to update an element available at the K^{th} position of LA.

1. Start
2. Set LA[K-1] = ITEM
3. Stop


```
#include <iostream>
using namespace std;
int main() {
  int LA[] = {1,3,5,7,8};
  int k = 3, n = 5, item = 10;
  int i, j;
   cout<<"\n\n The original array elements are :\n\n";

  for(i = 0; i<n; i++) {
    cout<<"LA["<<i<<"] = "<<LA[i]<<"\n";
                }

  LA[k-1] = item;

   cout<<"\n\n The array elements after update Operation are :\n\n";

  for(i = 0; i<n; i++) {
    cout<<"LA["<<i<<"] = "<<LA[i]<<"\n";
                }
  return 0;
}
```

End of lesson