ARRAYS: REVIEW

MonikaYadav
Student, departnment computer science engineering
Dronacharya college of engineering, gurgoan

ABSTRACT:- **This paper contains a description of the array data type and some discussion as to how we Allocate memory space to arrays.**
The value of defining arrays as an abstract data type is primarily for systems programmers, Who work behind the scenes and bring you all the wonderful software that comes with an Operatingsystem, such as compilers, linkers, files managers, text editors, etc. For most of us Mortal people, we simply use arrays in our programming languages without thought of the more Abstract nature of arrays.The theory of arrays is ubiquitous in the context of software and hardware verification and symbolic analysis. The basic array theory was introduced by McCarthy and allows to symbolically representing array updates.

INTRODUCTION

This paper will give us a small brief review about array that is used in the data structure.
This will also tell us how they are useful.Array is a collection mainly using similar data types that are stored into a common variable, forming (at least conceptually that may even be replicated into the memory hardware) a linear data structure.An array is a particular method of storing **elements** of indexed data. Elements of data are logically stored sequentially in blocks within the array. Each element is referenced by an **index**, or subscripts.Arrays can hold primitives as well as references.

DISSCUSSION

Array is a collection mainly using similar data types that are stored into a common variable, forming (at least conceptually that may even be replicated into the memory hardware) a linear data structure.
An array is a particular method of storing **elements** of indexed data. Elements of data are logically stored sequentially in blocks within the array. Each element is referenced by an **index**, or subscripts.Arrays can hold primitives as well as references.

Array stores similar type of object. Array can be classified into three categories.

1. One dimension array
2. Two dimension array
3. Three dimension array

Example :
Consider a multi-set of discrete domains Di = [li; ui], i 2 f1; 2; : : : ;Ng where each domain Di contains integers between li and ui. An N-dimensional array with M attributes Aj , j 2 f1; 2; : : : ;Mg, can be thought of as a function defined over dimensions and taking values attribute tuples ,i.e.,:

Array: D1 _ D2 _ _ _ _ _ DN 7 ! (A1;A2; : : : ;AM) (1)

where the type of the attributes can be any simple data type encountered in the relational data model. Using the same ideas behind extended data types, or user-defined data types, it is possible to have attributes with composite types, e.g., array, case in which we have nested arrays.

Some mostimportant array features

- copying and cloning
- insertion and deletion
- searching and sorting

An array is not a primitive data type - it has a field (and only one), called *length*. One of the major differences between references and primitives is that you cannot copy arrays by assigning one to another:

```
int[] a = {9, 5, 4};
int[] b = a;
```

Class in arrays
This class is a set of *static* methods that are all useful for working with arrays. The code below demonstrates a proper invocation of equals:

```
int[] a = {1,2,3};
int[] b = {1,2,3};
if(Arrays.equals(a, b) )
System.out.println("arrays with identical contents");
```

Copying arrays
There are four ways to copy arrays

(1) using a loop structure

```
int[] a = {1, 2, 3};
```

```
int[] b = new int[a.length];
for(inti = 0; i ‹ a.length; i++) b[i] = a[i];
```

(2) Arrays.copyOf()

```
int[] a = {1, 2, 3};
int[] b = Arrays.copyOf(a, a.length);
```

(3) The most efficient copying data between arrays is provided by System.arraycopy() method.

```
public static void arraycopy(Object source,
intsrcIndex,
                Object destination,
intdestIndex,
int length)
```

(4) The clone() method is defined in the Object class and its invocation is demonstrated by this code segment

```
int[] a = {1, 2, 3};
int[] b = (int[]) a.clone();
```

Insertion and Deletion

We allocate the array with a different size and copy the contents of the old array to the new array. This code example demonstrates deletion from an array of primitives

```
public char[] delete(char[] data, intpos)
{
        if(pos>= 0 &&pos<data.length)
        {
                char[] tmp = new char[data.length-1];
                System.arraycopy(data, 0, tmp, 0, pos);
                System.arraycopy(data, pos+1, tmp, pos, data.length-pos-1);
                returntmp;
        }
        else
                return data;
}
```

Multi-dimensional arrays

In many practical application there is a need to use two- or multi-dimensional arrays. A two-dimensional array can be thought of as a table of rows and columns. This creates a table of 2 rows and 4 columns:

```
int[][] ar1 = new int[2][4];
```

we can create and initialize an array by using nested curcly braces. For example, this creates a table of 3 rows and 2 columns:

```
int[][] ar2 = {{1,2},{3,4},{5,6}};
```

A two-dimensional array is not exactly a table - each row in such array can have a different length. Consider this code fragment

```
Object[][] obj = {{new Integer(1),new Integer(2)},
        {new Integer(10), "bozo", new Double(1.95)}};
```

obj has two elements obj[0] and obj[1] that are arrays of length 2 and 3 respectively.

Cloning 2D arrays

The procedure is even more confusing and less expected. Consider the following code segment

```
Object[][] obj = {{new Integer(1),new Integer(2)},
        {new Integer(10), "bozo", new Double(1.95)}};

Object[][] twin = (Object[][]) obj.clone();
```

The procedure of cloning 2d arrays is virtually the same as cloning an array of references. Unfortunately, built-in *clone()* method does not actually clone each row, but rather creates references to them.

CONCLUSION

There is also a lack of a unified resourcethat summarizes and analyzes array processing research over its long existence. In this survey, we providethis missing resource as a guide for current and new research in array processing. We present the problem from a database perspective, thus we focus our attention on clarifying the subtle differences between the relational data model and ordered arrays.We will create data structures of immutable objects; therefore implementing the clone method will require copying a structure (a shape) and sharing its internal data.

REFERANCES

[1] http://www.eecs.yorku.ca/course_archive/2008-09/F/2011/slides/06-ArrayADT.pdf
[2] http://iss.ices.utexas.edu/Publications/Papers/TOPLAS1989.pdf
[3] http://www.cs.cmu.edu/~adamchik/15-121/lectures/Arrays/arrays.html
[4] http://www.haskell.org/haskellwiki/Research_papers/Data_structures
[5] http://en.wikipedia.org/wiki/Array_data_structure