CS201

Array

Overview

- What is Array?
- Types of Arrays.
- Array operations.
- Merging of arrays.
- Arrays of pointers.
- Arrays and Polynomials.

ARRAY

- An array is a linear data structure. Which is a finite collection of similar data items stored in successive or consecutive memory locations.
- For example an array may contains all integer or character elements, but not both.

- Each array can be accessed by using array index and it is must be positive integer value enclosed in square braces.
- This is starts from the numerical value 0 and ends at 1 less than of the array index value.
- For example an array[n] containing n number of elements are denoted by array[0],array[1],…..array[n-1]. where ‘0’ is called lower bound and the ‘n-1’ is called higher bound of the array.

Types of Arrays

Array can be categorized into different types. They are

1. One dimensional array
2. Two dimensional array
3. Multi dimensional array

One dimensional array:-

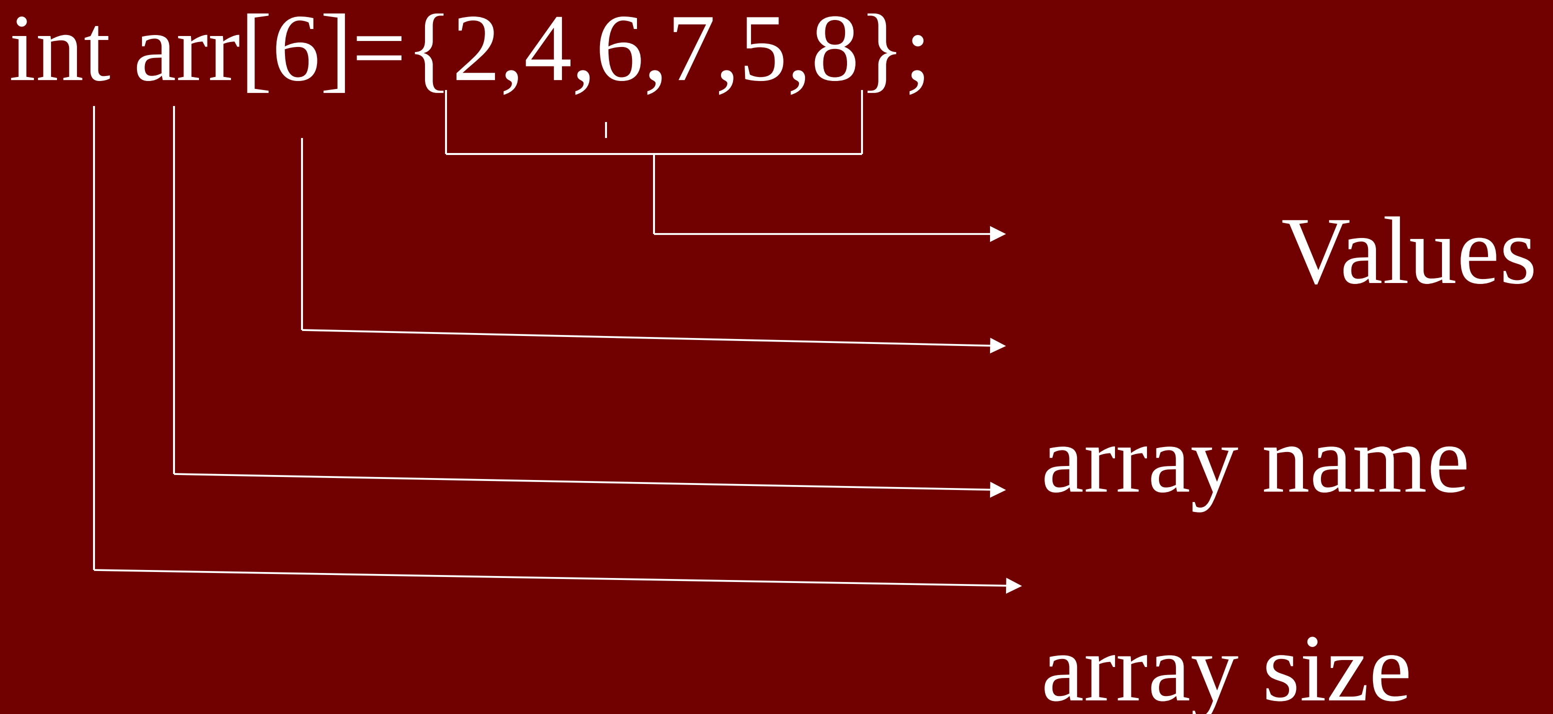
- One dimensional array is also called as linear array. It is also represents 1-D array.
- the one dimensional array stores the data elements in a single row or column.
- The syntax to declare a linear array is as fallows

Syntax: <data type> <array name> [size];

- Syntax for the initialization of the linear array is as fallows
- Syntax:

  <data type><array name>[size]={values};

- Example:



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

Values

array name

array size

Memory representation of the one dimensional array:-

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
2	4	6	7	5	8
100	102	104	106	108	110

- The memory blocks a[0],a[1],a[2],a[3],a[4] , a[5] with base addresses 100,102,104,106,108, 110 store the values 2,4,6,7,5,8 respectively.

- Here need not to keep the track of the address of the data elements of an array to perform any operation on data element.
- We can track the memory location of any element of the linear array by using the base address of the array.
- To calculate the memory location of an element in an array by using formulae.

  Loc (a[k])=base address +w(k-lower bound)

- Here k specifies the element whose location to find.
- w means word length.
- Ex: We can find the location of the element 5, present at a[3],base address is 100, then

$$loc(a[3])=100+2(3-0)$$

$$=100+6$$

$$=106.$$

Two dimensional array

- A two dimensional array is a collection of elements placed in rows and columns.
- The syntax used to declare two dimensional array includes two subscripts, of which one specifies the number of rows and the other specifies the number of columns.
- These two subscripts are used to reference an element in an array.

- Syntax to declare the two dimensional array is as follows
- Syntax:

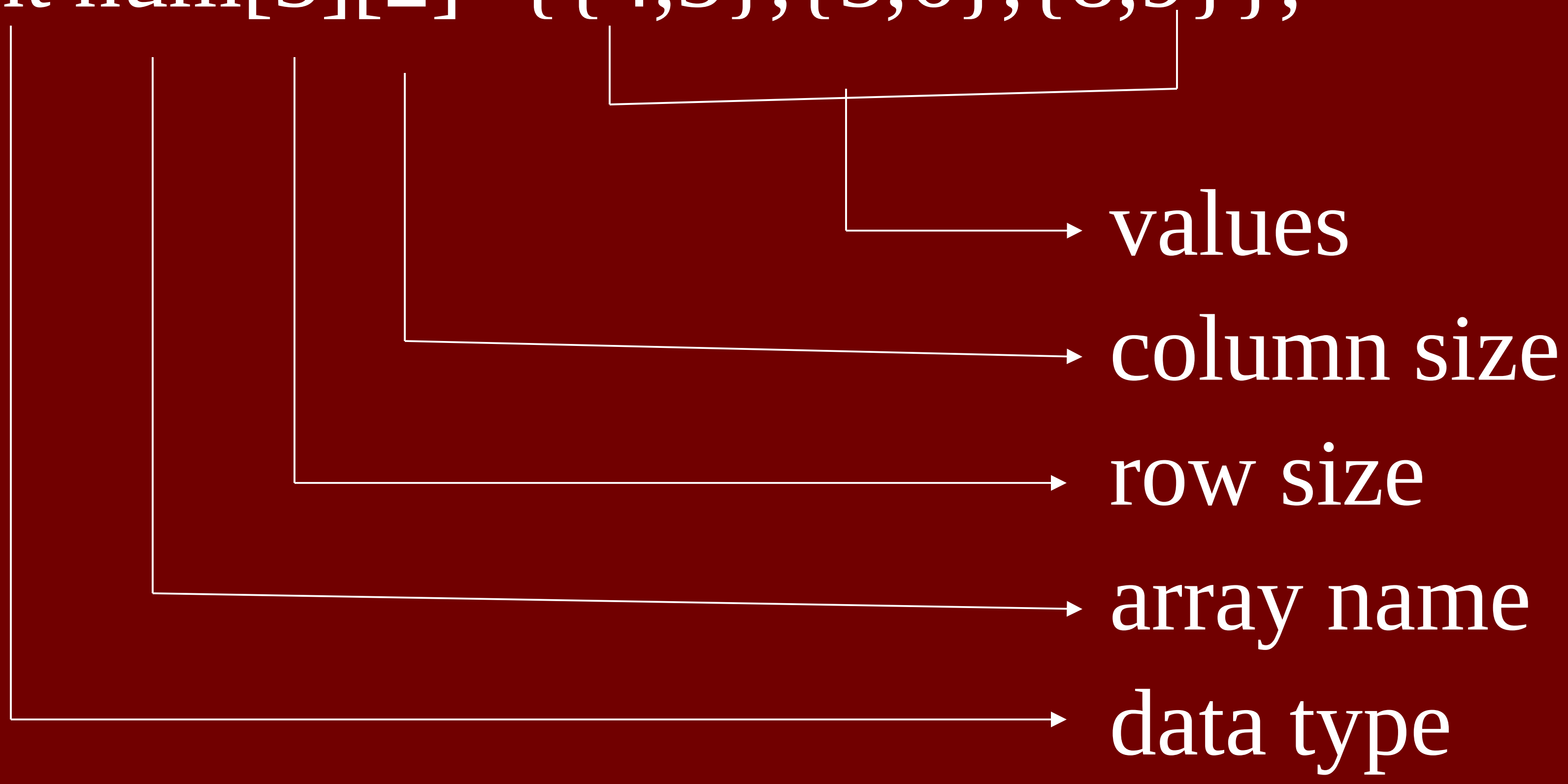
  <data type> <array name> [row size] [column size];

- Syntax to initialize the two dimensional array is as follows
- Syntax:

  <data type> <array name> [row size] [column size]={values};

- Example:

```
int num[3][2]={4,3,5,6,8,9};
or
int num[3][2]={{4,3},{5,6},{8,9}};
```



values

column size

row size

array name

data type

Representation of the 2-D array:-

Row	columns	
	0th column	1st column
0th row	a[0][0]	a[0][1]
1st row	a[1][0]	a[1][1]
2nd row	a[2][0]	a[2][1]

Memory representation of 2-D array:-

- Memory representation of a 2-D array is different from the linear array.
- in 2-D array possible two types of memory arrangements. They are

Row major arrangement

Column major arrangement

- Row major arrangement:

0th row		1st row		2nd row	
4	3	5	6	8	9
502	504	506	508	510	512

- Column major arrangement:

0th column			1st column		
4	5	8	3	6	9
502	504	506	508	510	512

- We can access any element of the array once we know the base address of the array and number of row and columns present in the array.
- In general for an array a[m][n] the address of element a[i][j] would be,
  - In row major arrangement

    Base address+2(i*n+j)
  - In column major arrangement

    Base adress+2(j*m+i)

- Ex:

  we can find the location of the element 8 then an array a[3][2] , the address of element would be a[2][0] would be

- In row major arrangement

  loc(a[2][0])=502+2(2*2+0)

  =502+8

  =510

- In column major arrangement

  loc(a[2][0])=502+2(0*3+2)

  =502+4

  =506

Multi dimensional arrays:-

- An array haves 2 or more subscripts, that type of array is called multi dimensional array.
- The 3-D array is called as multidimensional array this can be thought of as an array of two dimensional arrays.
- Each element of a 3-D array is accessed using subscripts, one for each dimension.

- Syntax for the declaration and initialization as follows Syntax
- <data type><array name>[s1][s2][s3]
  ={values};

- Ex:

```
int a[2][3][2]={
                    {  {2,1},{3,6},{5,3}  },
                     { {0,9},{2,3},{5,8}  }
                      };
```

Memory representation of 3-D array:-

- In multi dimensional arrays permits only a row major arrangement.

0th 2-D array						1st 2-D array					
2	1	3	6	5	3	0	9	2	3	5	8
10	12	14	16	18	20	22	24	26	28	30	32

- For any 3-D array a [x][y][z], the element a[i][j][k] can be accessed as

  Base address+2(i*y*z +j*z+ k)

- Array a can be defined as int a [2][3][2] , element 9 is present at a[1][0][1]
- Hence address of 9 can be obtained as

  =10+2(1*3*2+0*2+1)

  =10+14

  =24

ARRAY OPERATIONS

- There are several operations that can be performed on an array. They are
  - ♣Insertion
  - ♣Deletion
  - ♣Traversal
  - ♣Reversing
  - ♣Sorting
  - ♣Searching

Insertion:

- Insertion is nothing but adding a new element to an array.
- Here through a loop, we have shifted the numbers, from the specified position, one place to the right of their existing position.
- Then we have placed the new number at the vacant place.

- Ex:

```
for (i=4;i>= 2;i++)
  {
      a[i]=a[i-1];
      }
```

Code for shifting

a[i]=num;(Here i refers to required position)

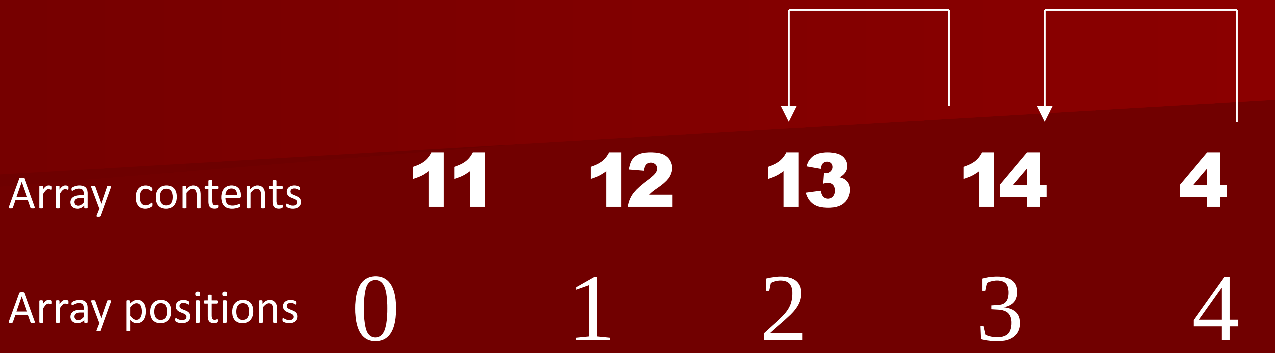
Deletion

- Deletion is nothing but process of remove an element from the array.
- Here we have shifted the numbers of placed after the position from where the number is to be deleted, one place to the left of their existing positions.
- The place that is vacant after deletion of an element is filled with ‘0’.

- Ex:

```
for (i=3;i<5;i++)
{
    a[i-1]=a[i];
      }
   a[i-1]=0; (The
   place that’s
   vacant after
   deletion )
```

- Before deletion:



Array contents	11	12	13	14	4
Array positions	0	1	2	3	4

- After deletion:

Array contents	11	13	14	4	0
Array positions	0	1	2	3	4

- Eig: shifting the elements to the left while deleting 3rd element in an array.

Traversal:

- Traversal is nothing but display the elements in the array.
- Ex:

```
        for (i=0;i<5;i++)
        {
           Printf (“%d\t”, a[i]);
           }
```

Array contents   11   12   14   4   0

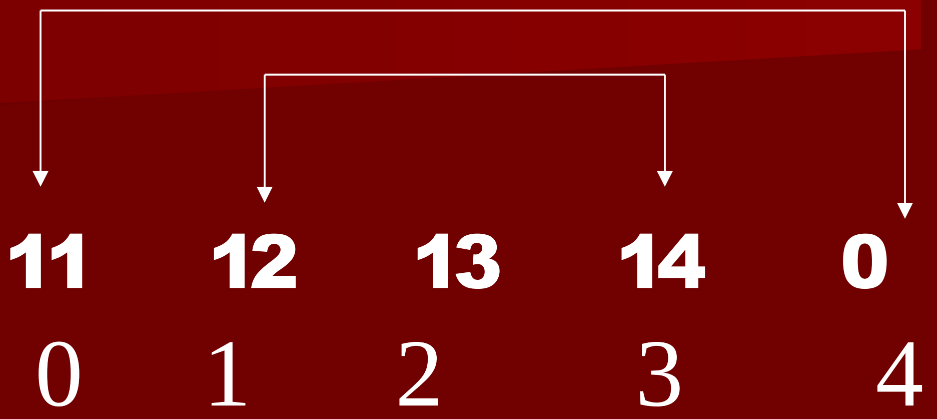
Reversing:

- This is the process of reversing the elements in the array by swapping the elements.
- Here swapping should be done only half times of the array size.
- Ex:

```
 for (i=0;i<5/2;i++)
     {
      int temp=a[i];
      a[i]=a[5-1-i];
       a[5-1-i]=temp;
         }
```

Code for Swapping between ith element and (5-1-i)th element.

- Before swapping:



11	**12**	**13**	**14**	**0**
0	1	2	3	4

- After swapping:

0	**14**	**13**	**12**	**11**
0	1	2	3	4

Fig: swapping of elements while reversing an array.

Sorting:

- Sorting means arranging a set of data in some order like ascending or descending order.

- Ex:

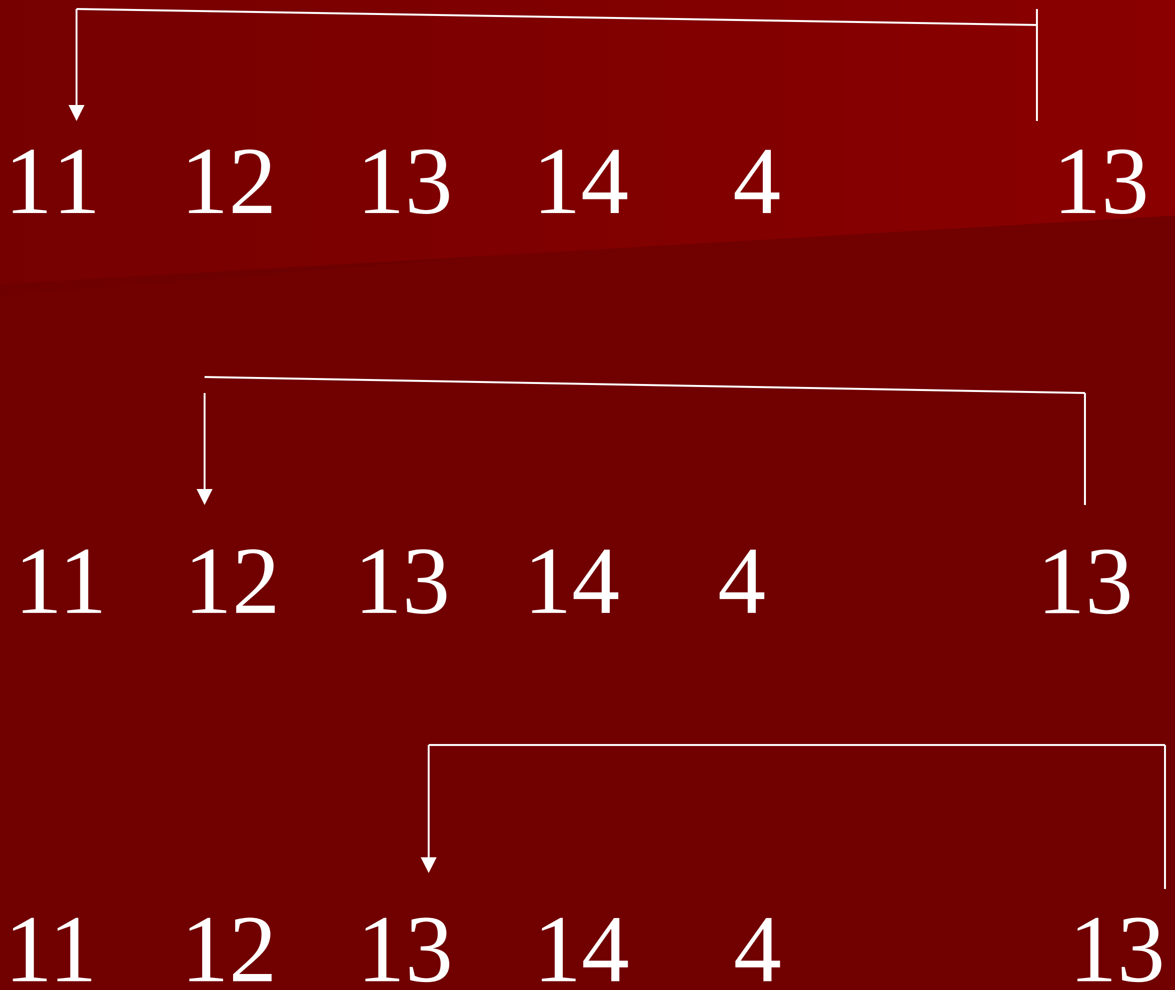
```
for (i=0;i<5;i++)
{
 for (j=i+1;j<5;j++)
 {
   if (a[i]>a[j])
   {
     temp=a[i];
     a[i]=a[j];
     a[j]=temp;
       } } }
```

Searching:

- Searching is the process of finding the location of an element with a given element in a list..
- Here searching is starts from 0th element and continue the process until the given specified number is found or end of list is reached.

- Ex:

```
     for (i=0;i<5;i++)
     {
       if (a[i]==num)
       {
         Printf("\n element %d is present at %dth
    position",num,i+1);
          return;
  }}if (i==5)
  Printf ("the element %d is not present in the
    array ",num);
```



11 12 13 14 4 13

11 12 13 14 4 13

11 12 13 14 4 13

Merging of arrays

- Merging means combining two sorted list into one sorted list.
- Merging of arrays involves two steps:

They are

1. sorting the arrays that are to be merged.
2. Adding the sorted elements of both the arrays a to a new array in sorted order.

- Ex:
- Before merging:

1st array 2nd array

1 3 13 2 8 11

- After merging:

1 2 3 8 11 13

Arrays of pointers

- A pointer variable always contains an address.
- An array of pointer would be nothing but a collection of addresses.
- The address present in an array of pointer can be address of isolated variables or even the address of other variables.

- An array of pointers widely used for stoning several strings in the array.
- The rules that apply to an ordinary array also apply to an array of pointer as well.
- The elements of an array of pointer are stored in the memory just like the elements of any other kind of array.
- Memory representation of the array of integers and an array of pointers respectively.

int a[3]

```
char *name[] = {“Aditya”, “NITGoa”}
```

```
char *name[0] = “Aditya”
char *name[1] = “NITGoa”
```

- Fig1:Memory representation of an array of integers and integer variables i and j.

a[0]	a[1]	a[2]	a[3]	i	j
3	4	5	6	1	9
100	102	104	106	200	312

- Fig2:Memory representation of an array of pointers.

b[0]	b[1]	b[2]	b[3]	b[4]	b[5]
100	102	104	106	200	312
8112	8114	8116	8118	8120	8122

Arrays and polynomials

- Polynomials like $5x^4+2x^3+7x^2+10x-8$ can be maintained using an array.
- To achieve each element of the array should have two values coefficient and exponent.

- While maintaining the polynomial it is assumes that the exponent of each successive term is less than that of the previous term.
- Once we build an array to represent polynomial we can use such an array to perform common polynomial operations like addition and multiplication.

***Addition of two polynomials*:**

- Here if the exponents of the 2 terms compared are equal then their coefficients are added and the result is stored in 3rdpolynomial.
- If the exponents of the 2 terms are not equal then the term with the bigger exponent is added to the 3 rd polynomial.

- If the term with an exponent is present in only 1 of the 2 polynomials then that term is added as it is to the 3rd polynomial.
- <u>Ex:</u>
- 1st polynomial is $2x^6+3x^5+5x^2$
- 2nd polynomial is $1x^6+5x^2+1x+2$
- Resultant polynomial is

$$3x^6+3x^5+10x^2+1x+2$$

Multiplication of 2 polynomials:

- Here each term of the coefficient of the 2nd polynomial is multiplied with each term of the coefficient of the 1st polynomial.
- Each term exponent of the 2nd polynomial is added to the each tem of the 1st polynomial.
- Adding the all terms and this equations placed to the resultant polynomial.

- Ex:
- 1st polynomial is

$$1x^4+2x^3+2x^2+2x$$

- 2nd polynomial is

$$2x^3+3x^2+4x$$

- Resultant polynomial is

$$2x^7+7x^6+14x^5+18x^4+14x^3+8x^2$$

References

1. The C Programming Language, Brain W. Kernighan, Dennis M. Ritche

2. Dr. C. Saritha, “ARRAYS IN DATASTRUCTURES USING ‘C’” SSBN Degree & PG College ANANTAPUR

Thank you