Comp 2401

Arrays and Pointers

Class Notes

2013

Introduction To Arrays:

In C programming, one of the frequently problem is to handle similar types of data. For example: if the user wants to store marks of 500 students, this can be done by creating 500 variables individually but, this is rather tedious and impracticable. These types of problem can be handled in C programming using arrays.

An array in C Programing can be defined as number of memory locations, each of which can store the same data type and which can be references through the same variable name. It is a collective name given to a group of similar quantities. These similar quantities could be marks of 500 students, number of chairs in university, salaries of 300 employees or ages of 250 students. Thus we can say array is a sequence of data item of homogeneous values (same type). These values could be all integers, floats or characters etc.

We have two types of arrays:

1. One-dimensional arrays.
2. Multidimensional arrays.

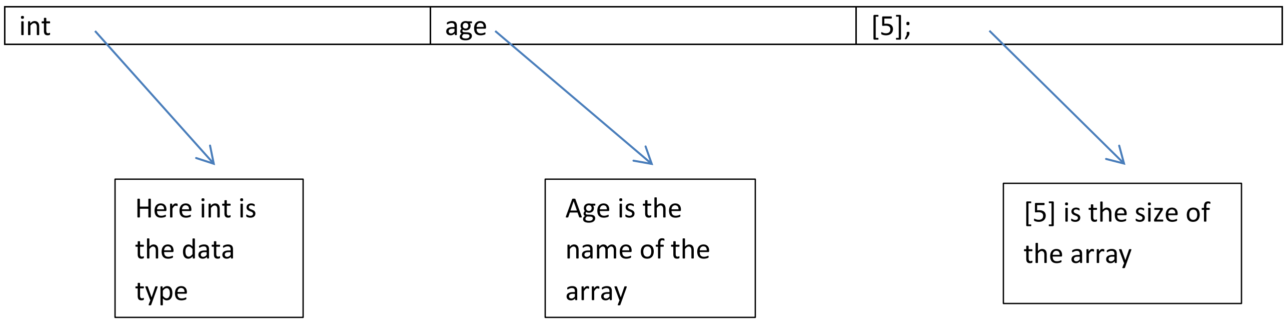
One Dimensional Arrays:

A **one-dimensional array** is a structured collection of components (often called *array elements*) that can be accessed individually by specifying the position of a component with a single *index* value. Arrays must be declared before they can be used in the program. Here is the declaration syntax of one dimensional array:

```
data_type array_name[array_size];
```

Here “data_type” is the type of the array we want to define, “array_name” is the name given to the array and “array_size” is the size of the array that we want to assign to the array. The array size is always mentioned inside the “[]”.

For example:
Int age[5];


int
age
[5];
Here int is the data type
Age is the name of the array
[5] is the size of the array

The following will be the result of the above declarations:

age[0]	age[1]	age[2]	age[3]	age[4]

Initializing Arrays

Initializing of array is very simple in c programming. The initializing values are enclosed within the curly braces in the declaration and placed following an equal sign after the array name. Here is an example which declares and initializes an array of five elements of type int. Array can also be initialized after declaration. Look at the following code, which demonstrate the declaration and initialization of an array.

int age[5]={2,3,4,5,6};

It is not necessary to define the size of arrays during initialization e.g.

int age[]={2,3,4,5,6};

In this case, the compiler determines the size of array by calculating the number of elements of an array.

age[0]	age[1]	age[2]	age[3]	age[4]
2	3	4	5	6

Accessing array elements

In C programming, arrays can be accessed and treated like variables in C.

For example:

- scanf("%d",&age[2]);

//statement to insert value in the third element of array age[]

- printf("%d",age[2]);

//statement to print third element of an array.

Arrays can be accessed and updated using its index.An array of n elements, has indices ranging from 0 to n-1. An element can be updated simply by assigning

A[i] = x;
A great care must be taken in dealing with arrays. Unlike in Java, where array index out of bounds exception is thrown when indices go out of the 0..n-1 range, C arrays may not display any warnings if out of bounds indices are accessed. Instead,compiler may access the elements out of bounds, thus leading to critical run time errors.

Example of array in C programming

```
/* C program to find the sum marks of n students using arrays */

#include <stdio.h>

int main(){

        int i,n;
        int marks[n];
        int sum=0;

        printf("Enter number of students: ");
        scanf("%d",&n);

        for(i=0;i<n;i++){
                printf("Enter marks of student%d: ",i+1);
                scanf("%d",&marks[i]); //saving the marks in array
                sum+=marks[i];
        }

        printf("Sum of marks = %d",sum);

return 0;

}
```

Output :

```
Enter number of students: (input by user)3
Enter marks of student1:  (input by user) 10
Enter marks of student2:  (input by user) 29
Enter marks of student3:  (input by user) 11

Sum of marks = 50
```

Important thing to remember in C arrays

Suppose, you declared the array of 10 students. For example: students[10]. You can use array members from student[0] to student[9]. But, what if you want to use element student[10], student[100] etc. In this case the compiler may not show error using these elements but, may cause fatal error during program execution.

Multidimensional Arrays:

C programming language allows the user to create arrays of arrays known as multidimensional arrays. To access a particular element from the array we have to use two subscripts one for row number and other for column number. The notation is of the form array [i] [j] where i stands for row subscripts and j stands for column subscripts. The array holds i*j elements. Suppose there is a multidimensional array array[i][j][k][m]. Then this array can hold i*j*k*m numbers of data. In the same way, the array of any dimension can be initialized in C programming. For example :

```
int smarks[3][4];
```

Here, smarks is an array of two dimension, which is an example of multidimensional array. This array has 3 rows and 4columns. For better understanding of multidimensional arrays, array elements of above example can be as below:

	Col 1	**Col 2**	**Col 3**	**Col 4**
Row 1	smakrs[0][0]	smarks[0][1]	smarks[0][2]	smarks[0][3]
Row 2	smarks[1][0]	smarks[1][1]	smarks[1][2]	smarks[1][3]
Row 3	smarks[2][0]	smarks[2][1]	smarks[2][2]	smarks[2][3]

Make sure that you remember to put each subscript in its own, correct pair of brackets. All three examples below are wrong.

```
int a2[5, 7];          /* XXX WRONG */

a2[i, j] = 0;          /* XXX WRONG */

a2[j][i] = 0;          /* XXX WRONG */
```

would do anything remotely like what you wanted

Initialization of Multidimensional Arrays

In C, multidimensional arrays can be initialized in different number of ways.

```
int smarks[2][3]={{1,2,3}, {-1,-2,-3}};
          OR
int smarks[][3]={{1,2,3}, {-1,-2,-3}};
          OR
int smarks[2][3]={1,2,3,-1,-2,-3};
```

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE

fastcase® Smarter legal research.