INDEX

|  |  |  |
| --- | --- | --- |
| S.NO | TOPIC | Page No |
| 1 | Abstract | 4 |
| 2 | History | 5 |
| 3 | Introduction to Arrays | 6 |
| 4 | Declaration of Arrays | 7 |
| 5 | Initializing Arrays | 8 |
| 6 | Performing operation on Arrays | 9 |
| 7 | Copy One Array into Another | 13 |
| 8 | Types of Array | 14 |
| 8a | Single dimensional array |  |
| 8b | Multi dimensional array |  |
| 9 |  Application | 19 |
| 10 | Advantages | 21 |
| 11 | Disadvantages | 21 |
| 12 | Conclusion | 23 |
|  13 | References | 24 |

 **ABSTRACT**

In Computer science, an **array type** is a data type which is meant for describing a collection of *elements* such as values or variables and each selected by one or more identifying keys which can be computed at run time by the program. Such a collection is known as an **array variable**, or simply **array.** By analogous to the mathematical concepts of vector and matrix, array types with one and two identifying keys which are often known as **vector type** and **matrix type** respectively.

Language which support an array types may include certain built-in array data types or some syntactic constructions (*array type constructors*) which the programmer may use in order to define such types and declaring array variables and notation for the purpose of indexing an array elements. For example, in the Pascal programming language, we declare the type my data = array [1.4 ,1.2] of integer, defines a new array data type that is called my data. The declaration var A : my data then defines a variable A of same type, which is an average of eight elements, each being an integer variable which is identified by two identifying keys. In the Pascal program, those elements are denoted A[1,1],A[1,2], A[2,1],… A[4,2]. Special array types which are often defined by the language's standard libraries.

Array types are differentiate from record types mainly because they allow the element indices to be computed at run time , as in the Pascal assignment such as A[I,J] := A[N-I,2\*J]. Among other things, whereas feature allows only a single iterative statement to process arbitrarily many elements of an array variable.

In more theoretical contexts, mainly in type theory and in the description of abstract algorithms, the terms "array" and "array type" usually refer to an Abstract data type (ADT) also known *abstract array* or may refer to an associative array a mathematical model with the basic operations and behavior of a typical array type in most languages — basically, a collection of elements that are selected by indices which are computed at run-time.

Depending upon the language array types which may overlap with other data types that describe aggregates of values, such as lists and strings. Array types that are often implemented by array such as data structures, but sometimes by it can other means, as following hash tables, linked list, and search trees.

**HISTORY**

The first digital computers which uses the machine-language programming in order to set up and for data tables, and matrix computations and for many other jobs array structures were used. Von Neumann wrote “first array-sorting program which was about Merge Sort in 1945”, during the time of building of the first stored-program in the computer. Array indexing was originally done by self modifying code and later by using index register and indirect addressing. Some mainframes which were designed during the 1960s, which are Burroughs B5000  and its successors, had special instructions for array indexing that included index-bounds checking.

Assembly languages generally doesn’t have any special support for arrays, other than which the machine itself provide. The earliest high-level programming languages, including FORTRAN, COBOL, and ALGOL 60, had support for multi-dimensional arrays, and so has C. In C++, class templates exist for the purpose of multi-dimensional arrays whose dimension is fixed at runtime as well as for runtime-flexible arrays.

.

|  |
| --- |
|  |

**1.INTRODUCTION**

Arrays can store the same data type and which may be reference through the same variable name.  An array is used for storing a collection of data. We can define an array as a collection of variables of the similar type.

An array is a collective name for a group of similar elements. These similar quantities could be percentage marks of 75 students, number of tables in home, or payment of 450 employees or marks of 50 students. Thus an array is a collection of similar type elements. These similar elements can be all int or all flts or all char etc. Usually, the array of characters is known a “string”, where as an array of int or flts are called an array. All elements of any given array must be of the same type i.e. we can’t have an array of 15 numbers, of which 10 are integer value and 5 are float value.

Instead of declaring individual variables, such as num0, num1... and num99, we declare one array variable such as numbers and use num[0], num[1], and ..., num[99] to represent all variables. A specific element in an array is accessed by an identifying key.

All arrays having contiguous memory locations,in which lowest address corresponds to the first element whereas the highest address to the last element of series as show below:-

 

`Arrays and pointers having a special relationship as arrays use pointers to reference memory locations.

**1.1 Declaration of an Arrays**

Arrays must be declared in the beginning of the program. Array is declared as

**Typeofvariable name [lenofarray];**

Here type indicates to the variable type of the element which is going to be stored in the array such as int ,float, char etc. In C programming language we declare the array of any basic type which can be supported by C language. For eg

**Doubleheight [15];**

**Floatamount [20];**

**Introllno [12];**

**Charnameofperson[25];**

Adjacent to the locations in the memory. C Language treats the name of an array as if it was a pointer to the first element it is important in understanding how to do arithmetic operations with arrays. Any item in the array can be accessed through its identifying key, and it can be used anywhere from within the program. So

**n=height [0];**

Variable n will have the value of first item of array height.

The program below will declare an array of ten integers and print all the 10 elements of the array.

**Int Array [10] = {0,1,2,3,4,5,6,7,8,9};**

**/\* for print all the elements of the an array**

**for (int i=0; i<10; i++)**

**{**

**printf ("%d", Array[i]);**

**}**

## 1.2Initializing Arrays

Initializing of an array is very easy in c programming. For initializing the values we enclosed it within the curly braces along with the declaration and is equalize to the array name. Here is an example in which declaration and initializing an array of six elements having type int..

Below is given an example od initializing and declaring an array:-

**intArray[6] = {1, 2, 3, 4, 5,6};**

 **/\*declaration and initializing of the an array**

**intstudentmarks[5];**

**Studentmarks[0]=53;**

**Studentmarks[1]=86;**

**Studentmarks[2]=96;**

**Studentmarks[3]=88;**

**Studentmarks[4]=95;**

## 1.3Performing Operation On Arrays

Below is a program that will show the simple operations on the array.

**#include <stdio.h>**

**Voidarray1(void);**

**Voidarray2(void);**

**int main(void)**

**{**

**printf("\array1:\n");**

**array1();**

**printf("\array2:\n");**

**array2();**

**}**

**//Array initialized with aggregate**

**Voidarray1(void) {**

**intaar[9] = {1,2,3,4,5,6,7,8,9,};**

**int i;**

**for (i=0; i<9; i++)**

**{**

**printf("i = %2d arr[i] = %2d\n", i, arr[i]);**

 **}**

**}**

**//Array initialized along with the loop**

**Voidarray2(void)**

**{**

**intar[9];**

**int i;**

**for (i=0; i<9; i++)**

**arr[i] = i+1;**

**for (i=0; i<9; i++)**

**printf("i = %2d arr[i] = %2d\n", i, arr[i]);**

**}**

**Output**

**array1:**

**i = 0 arr[i] = 1**

**i = 1 arr[i] = 2**

**i = 2 arr[i] = 3**

**i = 3 arr[i] = 4**

**i = 4 arr[i] = 5**

**i = 5 arr[i] = 6**

**i = 6 arr[i] = 7**

**i = 7 arr[i] = 8**

**i = 8 arr[i] = 9**

**i = 9 arr[i] = 0**

**array2:**

**i = 0 arr[i] = 1**

**i = 1 arr[i] = 2**

**i = 2 arr[i] = 3**

**i = 3 arr[i] = 4**

**i = 4 arr[i] = 5**

**i = 5 arr[i] = 6**

**i = 6 arr[i] = 7**

**i = 7 arr[i] = 8**

**i = 8 arr[i] = 9**

**i = 9 arr[i] = 10**

It is a another program that will show how to read, write and traverse the integer arrays

**#include <stdio.h>**

**voidintSwap(int \*a, int \*b);**

**intgotIntArray(int a[], int m max, int sntinel);**

**voidprintIntArray(int a[], int m);**

**voidreverseIntArray(int a[], int m);**

**int main(void) {**

**int x[10];**

**int mny;**

**mny = gotIntArray(x, 10, 0);**

**printf("The elements of array : \n");**

**printIntArray(x,mny);**

**reverseIntArray(x,mny);**

**printf("after swaping it is:\n");**

**printIntArray(x,mny);**

**}**

**Void intSwap(int \*a, int \*b)**

**/\* It swaps the content of a and b \*/**

**{**

**int temp = \*a;**

 **\*a = \*b;**

 **\*b = temp;**

**}**

**void printIntArray(int a[], int m)**

**{**

**inti;**

**for (i=0; i<m; )**

**{**

 **printf("\t%d ", a[i++]);**

 **if (i%5==0)**

 **printf ("\n");**

 **}**

**printf("\n");**

**}**

**int gotIntArray(int a[], intmmax, int sntinel)**

**{**

**int m = 0;**

**int temp;**

**do {**

**printf("Enter integer [%d for terminating] ", sntinel);**

**scanf("%d", &temp);**

**if (temp==sentinel) break;**

**if (m==mmax)**

**printf("array is full\n");**

**else**

**a[m++] = temp;**

 **}**

**while (1);**

**return m;**

**}**

**void swapIntArray(int a[], int m)**

**{**

**inti;**

**for(i=0;i<m/2;i++)**

**{**

**intSwap (&a[i],&a[m-i-1]);**

 **}**

} ,

|  |
| --- |
| Output**a=2****b=4****after swaping****a=4****b=2** |

## 1.4COPY ONE ARRAY INTO ANOTHER

There is no any statement in C language which can be used for directly copy an array into another array. For that we will copy each item separately from one into second array.

**#include <stdio.h>**

**int main()**

**{**

**intMarks[5];**

**shortenterMarks[5];**

**Marks[0]=78;**

**Marks[1]=64;**

**Marks[2]=66;**

**Marks[3]=74;**

**Marks[4]=86;**

**for(i=0; i<5; i++)**

**enterMarks[i]=Marks[i];**

**For(j=0; j<5; j++)**

**printf("%d\n", enterMarks[j]);**

**return 0;**

}

In order to summarize, arrays are provides a simple procedure where more than one elements having the same type can be used. We can store, alter and maintain multiple elements of same type in single array variable and ause them tby index.

**2.Types of Arrays**

Types of Arrays

1: Single dimensional 2: Multi dimensional

2.1 Single Dimensional Arrays

1. Single or one dimensional array is used to represent and store data in linear form.

2. Array having only one subscript variable is called One-dimensional array.

3.It is also called linear array.

**Syntax:**

<data-type> <array name> [size];

**Example of One Dimensional Array**

Int iaar[3]={2,3,8}

Char charr[20]=”c4learn”;

# 2.2Multi Dimensional arrays

1. Array having more than one subscript variable is called Multi Dimensional array.
2. Multi Dimensional array also called **Matrix**.

**Syntax:**

**<data-type> <array name> [row-subscript] [column-subscript];**

**Example**

In C Language one can have an arrays of any dimensions. For understanding the concept of multidimensional array, let consider the array of order 4\*5

|  |  |
| --- | --- |
| **Row number (i)** | **Column numbers (j)** |
| **0** | 11 | 3 | 5 | -9 | -6 |
| **1** | 5 | 6 | -8 | 7 | 24 |
| **2** | -8 | 9 | 2 | 12 | 45 |
| **3** | 10 | 13 | -10 | 4 | 5 |

Let us assume the name of matrix is A

For accessing a particular element of the array we have to use two subscripts one for row number and second for column number the notation is of the form A [i] [j] where i stands for row subscript and j stands for column subscript. Thus A [0] [0] refers to 10, A [2] [1] refers to 16 and so on In short multi dimensional arrays are defined more or less in the same manner as single dimensional arrays, except that for subscripts you require two squire two square brackets. We will restrict our decision to two dimensional arrays.

Below given are some typical two-dimensional array definitions

**intflowchart[50] [50];**

**charline[25] [25];**

The first example defines flowchart as a int point array having 50 rows and 50 columns. the number of elements will be 2500 (50 X50).

The second declaration example establishes an array line of type character with 25 rows and 25 columns. The number of elements will be (25 X 25) 625 consider the following two dimensional array definition int values [3] [4] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10. 11, 12 };

**Values [0] [0] = 1 Values [0] [1] = 2 Values [0] [2] = 3 Values [0] [3] = 4**

**Values [1] [0] = 5 Values [1] [1] = 6 Values [1] [2] = 7 Values [1] [3] = 8**

**Values [2] [0] = 9 Values [2] [1] = 10 Values [2] [2] = 11 Values [2] [3] = 12**

Here the first subscript stands for the row number and second for the column number. First subscript ranges from 0 to 2 and there are altogether 3 rows second one ranges from 0 to 3 and there are altogether 4 columns.

Otherwise we can also define the above definition and initialize it as

**int values [3] [4] = {**

 **{**

 **1, 2, 3, 4**

 **}**

 **{**

 **5, 6, 7, 8**

 **}**

 **{**

 **9, 10, 11, 12**

 **}**

**};**

In this the values in first pair of braces are initializing to the elements of first row, the values of 2nd pair of inner braces are assigned to second row and so on. Note that outer pair of curly braces is nessary. If there are two few values within a pair of braces the remaining elements will be assigned as zeros.

It is a program that stores name and roll no obtained by a student side by side in matrix

**main ( )**

**{**

 **int student [4] [2];**

 **int i, j;**

 **for(i =0; i< =3; i ++)**

 **{**

 **printf ("\n Enter name and roll no");**

 **scanf ("%d%d", &student [i] [0], &student [i] [1] );**

 **}**

 **for (i = 0; i< = 3; i ++)**

 **printf ("\n %d %d", student [i] [0], student [i] [1]);**

**}**

The given example will show how a two dimensional array can be read and how the values stored in the array can be showed on screen.

Another Example of multi-dimensional array:-

#include <stdio.h>

void main()

{

 float x[3][3], y[3][3], z[3][3];

 int i,j;

 printf(" Elements of matrix1\n");

/\*

 for(i=0;i<3;++i)

 for(j=0;j<3;++j){

 printf("Enter a %d%d: ",i+1,j+1);

 scanf("%f",&a[i][j]);

 }

 printf(" Elements of matrix2\n");

 for(i=0;i<3;++i)

 for(j=0;j<3;++j){

 printf("Enter b%d%d: ",i+1,j+1);

 scanf("%f",&b[i][j]);

 }

 for(i=0;i<3;++i)

 for(j=0;j<3;++j){

 c[i][j]=a[i][j]+b[i][j];

 }

 printf("\nSum :");

 for(i=0;i<3;++i)

 for(j=0;j<3;++j){

 printf("%.1f\t",c[i][j]);

 if(j==1)

 printf("\n");

 }

return 0;

}

**Ouput**

 Elements of matrix1

Enter a11: 0;

Enter a12: 0.5;

Entera13:0.9;

Enter a21: 0;

Enter a22: 0;

Enter a23:0;

Enter a31:0;

Enter a32:0;

Enter a33:0;

 Elements of matrix2

Enter b11: 0;

Enter b12: 0;

Enter b13;0;

Enter b21: 0;

Enter b22: 0;

Enter b31:0;

Enter b32:0;

Enter b33:0;

Sum :

0 0.5 0.9

0 0 0

0 0 0

**3.APPLICATION**

Arrays are used to implement mathematical vectors and matrices as well as other kinds of rectangular tables. Many database, small and large, consist of (or include) one-dimensional arrays whose elements are records

Arrays are used to implement other data structures, such as heaps, hash tables, deques,queues,stacks,string, and VLists.

Below are the some application of Arrays:-

**3.1 Stores Element Of Same Data type**

Arrays is used to store the number of elements belonging to same data type.

|  |
| --- |
| Int arr[30]; |

Above arrays is used to store integer number in an array.

|  |
| --- |
| arr[0] = 50;arr[1] = 60;arr[2] = 70;arr[3] = 80;arr[4] = 90; |

Similarly if we declare the character array then it can hold only character. So in short character array can store character variables while floating array stores only floating numbers.

**3.2 Array Used for maintain Multiple variable names using single name**

Suppose we need to store 5 roll number of students then without declaration of array we need to declare following:-

|  |
| --- |
| Int roll1,roll2,roll3,roll4,roll5; |

1. Now in order to get roll number of first student we need to access roll1.
2. Guess if we need to store roll number of 100 student then what will be the procedure.
3. Maintaing all the variables and remembering all these things is very difficult.

**Consider the Array**

|  |
| --- |
| Int roll[5]; |

So we are using array which can store multiple values and we have to remember just single variable name

**3.3 Array can be used for sorting elements`**

We can store elements to be sorted in an array and then by using different sorting technique we can sort the elements.

**Different Sorting Techniques are** :-

1. Bubble sort
2. Insertion Sort
3. Selection Sort
4. Bucket Sort

**3.3.1 Bubble sort :**

Bubble sort, is a simple sorting algorithm that works by repeatedly stepping through the list to be sorted, comparing each pair of adjacent items and then swapping them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted.

**3.3.2 Insertion sort:**

Insertion sort is a simple Sorting Algorithm that builds the final sorted array (or list) one item at a time. It is much less efficient on large lists than more advanced algorithms such as quick sort heap sort or Merge sort.

**3.3.3 Selection Sort:**

In computer science, **selection sort** is a sorting algorithm, specifically an in-place comparison sort .It has O(*n*2) time complexity, making it inefficient on large lists, and generally performs worse than the similar insertion sort. Selection sort is noted for its simplicity, and it has performance advantages over more complicated algorithms in certain situations, particularly where auxiliary memory is limited.

**3.3.4 Bucket Sort:**

**Bucket sort**, or **bin sort**, is a sorting algorithm that works by partitioning an array into a number of buckets. Each bucket is then sorted individually, either using a different sorting algorithm, or by recursively applying the bucket sorting algorithm. It is a distribution sort, and is a cousin of radix sort in the most to least significant digit flavor. Bucket sort is a generalization of pigeonhole sort. Since bucket sort is not a comparison sort, the Ω(*n* log *n*) lower bound is inapplicable

**3.4 Array can perform Matrix Operation**

Matrix operations can be performed using the array. we can use 2-D array to store the matrix

For example:-

#include<stdio.h>

int main()

{

int a[5][5],b[5][5],c[][5],i,j,k,sum=0,m,n,o,p;

 printf("\nEnter the row and column of first matrix");

 scanf("%d %d",&m,&n);

 printf("\nEnter the row and column of second matrix");

 scanf("%d %d",&o,&p);

 if(n!=o){ printf("Matrix mutiplication is not possible");

 printf("\nColumn of first matrix must be same as row of second matrix");

}

 Else

{

printf("\nEnter the First matrix->");

for(i=0;i<m;i++)

 for(j=0;j<n;j++)

scanf("%d",&a[i][j]);

 printf("\nEnter the Second matrix->");

for(i=0;i<o;i++)

 for(j=0;j<p;j++)

 scanf("%d",&b[i][j]);

 printf("\nThe First matrix is\n");

 for(i=0;i<m;i++){ printf("\n");

 for(j=0;j<n;j++)

{

printf("%d\t",a[i][j]);

 }

}

printf("\nThe Second matrix is\n");

 for(i=0;i<o;i++)

{

 printf("\n");

 for(j=0;j<p;j++)

{

 printf("%d\t",b[i][j]);

}

}

for(i=0;i<m;i++)

 for(j=0;j<p;j++) c[i][j]=0;

 for(i=0;i<m;i++)

{ //row of first matrix for(j=0;j<p;j++)

{ //column of second matrix sum=0;

 for(k=0;k<n;k++) sum=sum+a[i][k]\*b[k][j];

c[i][j]=sum;

}

}

}

 printf("\nThe multiplication of two matrix is\n");

for(i=0;i<m;i++)

{ printf("\n");

 for(j=0;j<p;j++)

{

printf("%d\t",c[i][j]);

}

}

 return 0;

}



**3.5 Array can be used in CPU Scheduling**

CPU Scheduling is generally managed by queue. Queue can be managed and implemented using the array. Array can be Dynamically allocated i.e. at run time.

**3.6 Array can be used in Recursive Function**

When the function calls another function or the same function again then the current values are stores onto the stack and those values will be retrieving when control comes back. This is similar to Stack.

 **4. ADVANTAGES/DISADVANTAGE**

**4.1Advantage**

1. You can use one name for similar objects and save then with the same name but different indexes.

2. Arrays are very useful when you are working with sequences of the same kind of data (similar to the first point but has a different meaning).

3. Arrays use reference type and so.

4. We can easily access each element of array. Not necessity to declare too many variables

5.Not necessity to declare too many variables.

6. Array elements are stored in continuous memory location.

7. It is used to represent multiple data items of same type by using only single name.

8. It can be used to implement other data structures like stacks, trees, graphs queues, linked list etc

9. Multidimensional arrays or 2D are used to represent matrices.

**4.2DISADVANTAGES**

1. Sometimes it’s difficult to operate with many identifying arrays.

2. C system doesn’t have checking mechanism for array sizes.

3. An array uses reference mechanism to work with memory which can cause unstable behavior of operating system (unless special methods were created to prevent it) and often even “blus screen” and so on.

4. Wastage of memory space. We doesn’t able to change size of array once it is declared at the run time.

5. It can store only similar type of data.

6. Sometimes it is not easy to operate

7. We must know in advance that how many elements are to be stored in array.

8. Array is static structure. It means that array is of fixed size. The memory which is allocated to array can not be increased or reduced.

9. Since array is of fixed size, if we allocate more memory than requirement then the memory space will be wasted. And if we allocate less memory than requirement, then it will create problem.

10. The elements of an array are stored in consecutive memory locations. So insertions and deletions are very difficult and time consuming

**Conclusion**

An array is a group of variables that have the same name and data type and are related in some way

* One-dimensional
* Visualize as a column of variables
* Two-dimensional
* Visualize as a table
* You must declare an array before you can use it
* Each elements in a one-dimensional array is assigned a subscript
* First element has subscript 0
* Parallel arrays are two or more arrays whose elements are related by their subscript in the arrays
* To create a two-dimensional array specify the number of rows and columns
* Each element is identified by two subscripts
* First subscript represents the element’s row location
* Second subscript represents its column location
* Use two loops to access every element in a two-dimensional array

**References**

1. Mailloux, R. J., "Phased array theory and technology," *Proc. IEEE*, Vol. 70, No. 3, Mar. 2010.
2. Balaguruswami, programming in ANSI C,4th edition,2012
3. By:Mingqiang Li; Jiwu Shu
4. Published in: RC25218 in 2011
5. The C Programming Language by Brian W. Kernighan, Dennis M. Ritchie,2012

 6. C Programming: A Modern Approach by K.N. King, 2011

 7. C Programming in 12 Easy Lessons by Greg Perry, 2012

 8. C for Dummies Vol. II by Dan Gookin, 2012

 9. cprogramming.com