Arrays and Strings

Arash Rafiey

September 12, 2017

Arrays

Array is a collection of variables with the **same data type**.

Arrays

Array is a collection of variables with the **same data type**.

Instead of declaring individual variables, such as number0, number1, ..., and number99, one array variable such as numbers can be used.

And numbers[0], numbers[1], and ..., numbers[99] represent individual variables.

Arrays

Array is a collection of variables with the **same data type**.

Instead of declaring individual variables, such as number0, number1, ..., and number99, one array variable such as numbers can be used.

And numbers[0], numbers[1], and ..., numbers[99] represent individual variables.

All arrays consist of contiguous memory locations.

Declaration

Syntax:

type arrayName [number of elements in array];

Here type can be any data type such as int or char.

Declaration

Syntax:

type arrayName [number of elements in array];

Here type can be any data type such as int or char.

For example:

int arr[20];

Is an integer array of size 20.

Multi-dimensional arrays

C supports multi-dimensional arrays.The simplest form of the multidimensional array is the two-dimensional array.

Multi-dimensional arrays

C supports multi-dimensional arrays.The simplest form of the multidimensional array is the two-dimensional array.

Syntax for two-dimensional array:

type arrayName [x][y];

Here x is the number of rows and y is the number of columns

For example:

int two_dimensional_array[3][2];

Initializing an array

An array can be initialized in many ways.

- **Initializing each element separately:**

```
int arr[10];
int i = 0;
for(i=0;i< sizeof(arr);i++)
{
arr[i] = i; // Initializing each element separately
}
```

Initializing an array

- **Initializing array at the time of declaration.**

  int arr[ ] = {'1','2','3'};

  Since we are initializing at the time of declaration so there is no need to mention any value in the subscripts [ ].

  The size will automatically be calculated from the number of values.

  In this case, the size will be 3.

Initializing an array (string)

- **Initializing array with a string (Method 1):**

  The string in C is actually a one-dimensional array of characters which is terminated by a null character \0.

  Since strings are nothing but a series of characters so the array containing a string will be containing characters.

  char arr[ ] = {'c','o','d','e','\0' };

  The null byte is required as a terminating byte when string is read as a whole.

Initializing an array (string)

- **Initializing array with a string (Method 2):**

  char arr[ ] = "code";

  Here we can specify the whole string using double coats.

```
char test-string[20];
printf( "enter a string \n " );

scanf("%s", &test-string); // here s is for string
int j=0;
while (test-string[j] != '\0' )
{
   printf( "%c",test-string[j]);
   j++;
}
printf( "\n"); // print a new line
```

```
char test-string[20];
printf( "enter a string \n " );

scanf("%s", &test-string); // here s is for string
int j=0;
while (test-string[j] != '\0' )
{
   printf( "%c",test-string[j]);
   j++;
}
printf( "\n"); // print a new line

char chr1='a';
printf( "%d \n", chr1); // it prints out 97. ASCII code for 'z' is
122
```

```
char test-string[20];
printf( "enter a string \n " );

scanf("%s", &test-string); // here s is for string
int j=0;
while (test-string[j] != '\0' )
{
   printf( "%c",test-string[j]);
   j++;
}
printf( "\n"); // print a new line

char chr1='a';
printf( "%d \n", chr1); // it prints out 97. ASCII code for 'z' is
122

chr1='A';

printf( "%d \n", chr1); // it prints out 65. ASCII code for 'Z' is 90
```

Accessing values in an array

The first element of array always has index 0.

```
int arr[10];
int i = 0;
for(i=0;i<sizeof(arr);i++)
{
arr[i] = i; // Initializing each element separately
}
int j = arr[5]; // Accessing the 6th element of integer array arr and assigning its value to integer 'j'.
```

Example 1 Searching for an element in an array

```
# include <stdio.h>
void main() {
int i, find, n;
printf("Enter number of elements in array \n");
scanf("%d",&n);
int arr[n];
printf("Enter the elements of the array \n");
for (i = 0; i<n; i++)
  scanf("%d", &arr[i]);
```

Example 1 Searching for an element in an array

```
   printf("Enter the search element");
   scanf("%d",&find);
   for (i = 0; i<n; i++){
   if(arr[i]==find){
     printf("Element is present at position %d",i+1);
     break; }
}
if(i==n)
   printf("Element not found");
```

Sorting an Array

```
# include <stdio.h>
void main() {
int i, n, smallest, index;
printf("Enter number of elements in array \n");
scanf("%d",&n);
int arr[n];
printf("Enter the elements of the array \n");
for (i = 0; i<n; i++)
  scanf("%d", &arr[i]);
```

Sorting an array in ascending order

```
for (i = 0; i<n; i++){
   smallest=arr[i];
   for (j = i + 1; j<n; j++)
    if (samllest > arr[j]){
     smallest= arr[j];
     index = j;
   }
   arr[index]= arr[i]; arr[i]=smallest;
}
printf("The numbers in ascending order are: \n");
for (i = 0; i<n; i++)
     printf("%d \n", arr[i]);
}
```

Tutorial for Thursday September 14

Question (1)

Write a program that reads n integer numbers from the user and prints them in the reverse order.

Your program should read *n* from the user first. Then ask the user to enter *n* integer numbers and print them in reverse order.

Tutorial for Thursday September 14

Question (1)

Write a program that reads n integer numbers from the user and prints them in the reverse order.

Your program should read n from the user first. Then ask the user to enter n integer numbers and print them in reverse order.

Hint : You need to use array. Declare an array of size n. Read n numbers from input. Place them in the array. Read the array from index $n-1$ to 0 and print the elements of the array.

Question (2)

We say a list of n integer numbers is 100*-good if the difference between the smallest element and the largest element is at most* 100*. Write a program that reads a list of n integer numbers from the user and decides whether it is* 100*-good or not.*

Your program should read *n* from the user first. Then ask the user to enter *n* integer numbers. If it is not 100-good then your program should print out "NO" otherwise it should print the difference between the largest and smallest number in the list.

Question (2)

We say a list of n integer numbers is 100*-good if the difference between the smallest element and the largest element is at most* 100*. Write a program that reads a list of n integer numbers from the user and decides whether it is* 100*-good or not.*

Your program should read n from the user first. Then ask the user to enter n integer numbers. If it is not 100-good then your program should print out "NO" otherwise it should print the difference between the largest and smallest number in the list.

Hint : You need to use array. Declare an array of size n. Read n numbers from input. Place them in the array. Take two dummy variables x, y (x for the smallest, y for the largest element).
As you read the current value of the array compare it with x, y and update x, y accordingly.
At the end check whether $y - x \leq 100$. If yes then print out $y - x$. Otherwise print NO.

Question (3)

Write a program that reads n numbers from the user and finds their median.

Your program should read n from the user first. Then ask the user to enter n integer numbers.

Example : The median of $1, 7, 3, 5, 6, 11, 10$ is 6

Example : The median of $2, 7, 3, 3, 9, 6$ is $(3 + 6)/2$

Question (3)

Write a program that reads n numbers from the user and finds their median.

Your program should read n from the user first. Then ask the user to enter n integer numbers.

Example : The median of $1, 7, 3, 5, 6, 11, 10$ is 6

Example : The median of $2, 7, 3, 3, 9, 6$ is $(3 + 6)/2$

Hint : You need to sort the array and see the definition of the median.

Question (4)

Write a program that converts all the temperature values from 0 F to 100 F (Fahrenheit) to Celsius.