



Programming for Problem Solving (PPS)

Chapter-4

Arrays and Strings





ARRAYS



ARRAY

- Many applications require multiple data items that have common characteristics.
 - In mathematics, we often express such groups of data items in indexed form:
 - $x_1, x_2, x_3, \dots, x_n$
- Array is a data structure which can represent a collection of data items which have the same data type (float/int/char)



USING ARRAYS

- All the data items constituting the group share the same name.

```
int x[10];
```

- Individual elements are accessed by specifying the index.



X is a 10-element one dimensional array



DECLARING ARRAYS

Like variables, the arrays that are used in a program must be declared before they are used.

General syntax:

`type array-name [size];`

- type specifies the type of element that will be contained in the array (int, float, char, etc.)
- size is an integer constant which indicates the maximum number of elements that can be stored inside the array.
- marks is an array containing a maximum of 5 integers.



EXAMPLES

- Examples:

```
int x[10];
```

```
char line[80];
```

```
float points[150];
```

```
char name[35];
```

- If we are not sure of the exact size of the array, we can define an array of a large size.

```
int marks[50];
```

though in a particular run we may only be using, say, 10 elements.



ACCESSING ARRAY ELEMENTS

- A particular element of the array can be accessed by specifying two things:
 - Name of the array.
 - Index (relative position) of the element in the array.
- In C, the index of an array starts from zero.
- Example:
 - An array is defined as `int x[10];`
 - The first element of the array x can be accessed as `x[0]`, fourth element as `x[3]`, tenth element as `x[9]`, etc.



INITIALIZATION OF ARRAYS

- General form:

```
type array_name[size] = { list of values };
```

- Examples:

```
int marks[5] = {72, 83, 65, 80, 76};
```

```
char name[4] = {'A', 'm', 'i', 't'};
```

- Some special cases:

- If the number of values in the list is less than the number of elements, the remaining elements are automatically set to zero.

```
float total[5] = {24.2, -12.5, 35.1};
```

➔ total[0]=24.2, total[1]=-12.5, total[2]=35.1, total[3]=0,
total[4]=0



INITIALIZATION OF ARRAYS

- The size may be omitted. In such cases the compiler automatically allocates enough space for all initialized elements.

```
int flag[] = {1, 1, 1, 0};  
char name[] = {'A', 'm', 'i', 't'};
```



CHARACTER ARRAYS AND STRINGS

```
char C[8] = { 'a', 'b', 'h', 'i', 'j', 'i', 't', '\0' };
```

- C[0] gets the value 'a', C[1] the value 'b', and so on. The last (7th) location receives the null character '\0'.
- Null-terminated character arrays are also called strings.
- Strings can be initialized in an alternative way. The last declaration is equivalent to:

```
char C[8] = "abhijit";
```

- The trailing null character is missing here. C automatically puts it at the end.
- Note also that for individual characters, C uses single quotes, whereas for strings, it uses double quotes.



Example 1: Find the minimum of a set of 10 numbers

```
#include <stdio.h>
int main()
{
    int a[10], i, min;

    for (i=0; i<10; i++)
        scanf ("%d", &a[i]);

    min = 99999;
    for (i=0; i<10; i++)
    {
        if (a[i] < min)
            min = a[i];
    }
    printf ("\n Minimum is %d", min);
}
```



How to read the elements of an array?

By reading them one element at a time

- `for (j=0; j<25; j++)`
- `scanf ("%f", &a[j]);`

The ampersand (&) is necessary.

The elements can be entered all in one line or in different lines.



How to print the elements of an array?

- By printing them one element at a time.

```
for (j=0; j<25; j++)  
    printf ("\n %f", a[j]);
```

- The elements are printed one per line.

```
printf ("\n");  
for (j=0; j<25; j++)  
    printf (" %f", a[j]);
```

- The elements are printed all in one line
(starting with a new line).



Two Dimensional Arrays

- We have seen that an array variable can store a list of values.
- Many applications require us to store a **table** of values.

	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5
Student 1	75	82	90	65	76
Student 2	68	75	80	70	72
Student 3	88	74	85	76	80
Student 4	50	65	68	40	70



Two Dimensional Arrays

- The table contains a total of 20 values, five in each line.
 - The table can be regarded as a **matrix** consisting of **four rows** and **five columns**.
- C allows us to define such tables of items by using **two-dimensional** arrays.



Declaring 2-D Arrays

- General form:

```
type array_name [row_size][column_size];
```

- Examples:

```
int marks[4][5];
```

```
float sales[12][25];
```

```
double matrix[100][100];
```

Accessing Elements of a 2-D Array

- Similar to that for 1-D array, but use two indices.
 - First indicates row, second indicates column.
 - Both the indices should be expressions which evaluate to integer values.
- Examples:

$x[m][n] = 0;$

$c[i][k] += a[i][j] * b[j][k];$

$a = \sqrt{a[j^*3][k]};$



How to read the elements of a 2-D array?

- By reading them one element at a time
 - for (i=0; i<nrow; i++)
 - for (j=0; j<ncol; j++)
 - scanf ("%f", &a[i][j]);
- The ampersand (&) is necessary.
- The elements can be entered all in one line or in different lines.



How to print the elements of a 2-D array?

- By printing them one element at a time.

```
for (i=0; i<nrow; i++)  
    for (j=0; j<ncol; j++)  
        printf ("\n %f", a[i][j]);
```

- The elements are printed one per line.

```
for (i=0; i<nrow; i++)  
    for (j=0; j<ncol; j++)  
        printf ("%f", a[i][j]);
```

- The elements are all printed on the same line.



How to print the elements of a 2-D array?

```
for (i=0; i<nrow; i++)  
{  
    printf ("\n");  
    for (j=0; j<ncol; j++)  
        printf ("%f ", a[i][j]);  
}
```

- The elements are printed nicely in matrix form.



Multi-Dimensional Arrays

- Syntax: type array_name[s1][s2].....[sm];
- Eg: int survey[3][5][12][11];
- Survey is a four dimensional array
- ANSI C does not specify any limit for array dimension. However, most of the compilers permit seven to ten dimensions. Some allow even more.



STRING

- A group of characters
- A string constant is a one-dimensional array of characters terminated by a null ('\0').

declaration of Character:

```
char mychar;
```

declaration of String:

```
char myString[10];
```

The characters after the null character are ignored.



STRING INITIALIZATION

Initialization Syntax:

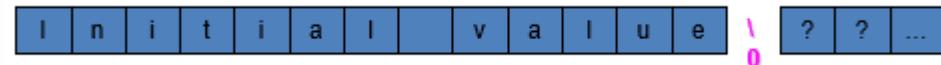
```
char myString [] = { 'H', 'A', 'E', 'S', 'L', 'E', 'R',  
    '\0' } ;
```

```
char myString[13] = "Initial value"
```

```
char myString[] = "Initial value";
```

\0 = null character

Compilation time
initialization



Note: that '\0' and '0' are not same.

- When declaring a string don't forget to leave a space for the null character which is also known as the string terminator character

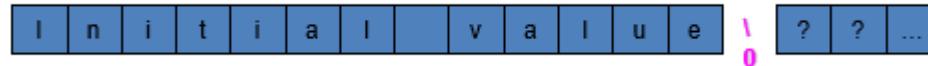


STRING INITIALIZATION

Why Null
char ?

only way the functions that work with a string
can know where the string ends.

```
char myString[100] = "Initial value"
```





STRING INITIALIZATION: scanf() with %c

Character array :

```
#include <stdio.h>
int main() {
    int i;
    char mystring[21]; // one extra for '\0'
    for(i = 0; i < 20; i++) {
        scanf("%c", &mystring[i]);
    }
    mystring[10] = '\0'; // null terminate. Try mystring[20] also
    printf("%s", mystring);
    return 0;
}
```

Run time
Initializatio
n

Scanf("%c",);

- The **scanf** function in C is used to read formatted input from the standard input stream (typically the keyboard).
- When using **scanf** to read a single character, the **%c** format specifier is employed.

Using Input/ output function :

Scanf() **gets()** **getchar()**



Input Function: `scanf()` with `%s`

- The `scanf()` Function
 - header file `stdio.h`
 - Syntax:

```
char mystring[100];
scanf("%s", mystring);
```
 - The name of a string is a pointer constant to the first character in the character array.
- Problem:
terminates its input on the first white space it finds.
white space includes blanks, tabs, carriage returns(CR), form feeds & new line.



Input Function: `scanf()` with `%s`

Example

```
#include<stdio.h>
int main(){
char mystring [100];
printf("Enter String:\n");
scanf("%s", mystring);
printf("\n\n%s",mystring);
}
```

`scanf("%s", ...);`

- Reads characters until it encounters any whitespace (space, tab, newline) or End-Of-File (EOF).
- It only captures the first word of a multi-word input string.

Output:

```
Enter String:
Hello students !  
  
Hello
```



Input Function: gets() (similar to scanf() with %s, but till string ends)

- The `gets()` Function
 - Header file stdio.h
 - takes a string from standard input and assigns it to a character array.
 - It replaces the \n with \0.
 - Syntax:

```
char mystring[100] ;  
gets(myString) ;
```
 - `fgets()` it keeps the \n and includes it as part of the string.



Input Function: gets() (similar to scanf() with %s, but till string ends)

Example:

```
#include<stdio.h>
int main(){
char mystring [100];
printf("Enter String:\n");
gets(mystring);
printf("\n\n%s",mystring);
}
```

Output:

```
Enter String:
Hello students !
Hello students !
```



How to scan string using scanf() with %s?

Example:

```
#include<stdio.h>
int main(){
char mystring [100];
printf("Enter String:\n");
scanf("%[^\\n]s", &mystring); //scans till new line
printf("\n\n%s",mystring);
}
```

Output:

```
Enter String:
Hello students !
Hello students !
```



Input Function: `getchar()` (similar to `scanf()` with `%c`)

- The `getchar()` Function
 - Takes single character at a time.
 - Syntax:
`char mychar;`
`mychar=getchar();`
 - It can used to read each character of a string from keyboard

```
#include<stdio.h>
int main(){
int i;
char mystring[100];
printf("Enter String:\n");
for(i=0;i<10;i++)
mystring[i]=getchar();
mystring[9]='\0';
printf("\n\n%s",mystring);
}
```

```
Enter String:
Hello boys
Hello boy
```



Output function: putchar() (similar to printf() with %c)

- The **putchar()** Function
 - output single character at a time.
 - Syntax:
char mychar;
putchar(mychar);

Example:

```
#include<stdio.h>
int main(){
int i;
char mystring[100];
printf("Enter String:\n");
gets(mystring);
for(i=0;i<20;i++)
putchar(mystring[i]);
}
```

Output:

Enter String:
Hello Friends!
Hello Friends!



Output function: puts() (similar to printf() with %s)

- The **puts()** Function
 - Outputs string.
 - Automatically appends newline character (\n) at the end
 - Syntax:

**char mychar;
puts (mychar) ;**

Example:

```
#include<stdio.h>
int main(){
    char mystring[100];
    printf("Enter String:\n");
    gets(mystring);
    puts(mystring);
}
```

**Output:
Enter String:
Hello Friends!
Hello Friends!**



String operation – string.h

- **Four main library function which is define in `string.h` header file**

`strcpy()` - copy one string into another

`strcat()` - append one string onto the right side of the other

`strcmp()` – compare alphabetic order of two strings

`strlen()` – return the length of a string



strcpy()

- **strcpy(destinationstring, sourcestring)**
- Copies sourcestring into destinationstring

Example

```
#include<stdio.h>
#include<string.h>
void main(){
    char str1[20], str2[20];
    printf("Enter String:\n");
    gets(str1);
    strcpy(str2, str1);
    puts(str2);
}
```

Output

```
Enter String:
Hello students
Hello students
```



strcat()

- **strcat()** function to combine two strings into a new string.
- **strcat(destinationstring, sourcestring)**
- appends sourcestring to right hand side of destinationstring
- We need to be certain that the array to which we assign the resulting string is large enough to hold all the characters from the two contributing strings.
- **Syntax:**

strcat(str1, str2);



Example: **strcat()**

```
#include<stdio.h>
#include<string.h>
void main(){
    char str1[20]="Hello ";
    char str2[20]="Students!";
    strcat(str1, str2);
    puts(str1);
}
```

Output

```
Hello students!
```



strcmp()

- Compares str1 and str2 alphabetically

`strcmp(str1, str2)`

- If the two strings are equal, `strcmp()` returns 0.
- If `str1` is greater than `str2`, `strcmp()` returns a positive number.
- If `str1` is less than `str2`, `strcmp()` returns a negative number.



Example: strcmp()

```
#include<stdio.h>
#include<string.h>
void main(){
    int d;
    char str1[20]="Hello ";
    char str2[20]="Students!";
    d = strcmp(str1, str2);
    if(d==0)
        printf("Both Str are same");
    else if(d>0)
        printf("Str1 is greater");
    else
        printf("Str2 is greater");
}
```

Output

```
str2 is greater
```



strlen()

- **strlen()** function to determine the length of a string.
- It counts the number of characters in a string, excluding the null character.
- Syntax:
strcount = strlen(myString);



Example: **strlen()**

```
#include<stdio.h>
#include<string.h>
int main(){
    int d;
    char str1[20]="Hello Students!";
    d=strlen(str1);
    printf("Length of String is: %d", d);
}
```

Output:

Length of string is : 15



String Functions

- **strlwr()** : converts a string to lowercase
- **strupr()** : converts a string to uppercase
- **Strncat()** : Appends first n characters of a string at the end of another
- **Strncmp()** : Compares first n characters of two strings
- **Strcmpi()** : Compares two strings without regard to case ("i" denotes that this function ignores case)
- **Strrev()** : Reverses string
- **Strncpy()** : copy first n character of one string to another.



Example:

```
#include<stdio.h>
#include<string.h>

int main() {
    int d;
    char str1[20] = "Hello ";
    char str2[20] = "Hello Students!";
    char str3[20] = "HELLO HE";
    char str4[20];

    d = strcmp(str1, str2, 5);
    if (d == 0)
        printf("Both Strings are same\n");

    puts(strlwr(str1));
    puts(strupr(str2));
}
```

```
strncat(str1, str2, 2);
puts(str1);
strcpy(str4, str2, 5);
puts(str4);

d = strcmp(str1, str3);
if (d == 0)
    printf("Both Strings are same\n");

puts(strrev(str2));
```

Output:

```
Both Strings are same
hello
HELLO STUDENTS!
hello HE
HELLO
Both Strings are same
!STNEDUTS OLLEH
```



Problem: Alphabetical Order

Write a program to input a string and rearrange the string in alphabetical order. For example, the word NEPAL should be written as AELNP



Solution: Alphabetical Order

```
#include <stdio.h>
#include <string.h>

int main() {
    char str[100], temp;
    int i, j, len;

    printf("Enter a string: ");
    gets(str);
    len = strlen(str);
```

```
// Sorting characters in ascending (alphabetical) order
for (i = 0; i < len - 1; i++) {
    for (j = i + 1; j < len; j++) {
        if (str[i] > str[j]) {
            temp = str[i];
            str[i] = str[j];
            str[j] = temp;
        }
    }
}
printf("String in alphabetical order: %s\n", str);
return 0;
```

Output

Enter a string: NEPAL
String in alphabetical order: AELNP



What is the key difference between ‘A’ and “A” ?

- The representation of a char (e.g., ‘A’) and a string (e.g., “A”) is essentially different.
 - A string is an array of characters ended with the null character.

A

Character ‘Q’

A \0

String “Q”



Two dimensional- string array

- An array of strings is a two-dimensional array of characters in which each row is one string.

```
char names [std_number] [Name_Lth] ;
```

```
char month[5][10] = {"January", "February",
"March", "April", "May"};
```

//An array to store 5 Strings, each with a maximum length of 9 characters (Plus the null character)

e.g.

```
char names[3][10] =
{"Apple", "Banana",
"Cherry"};
```

Memory Representation

Column->	0	1	2	3	4	5	6	7	8	9
name[0]	A	p	p	I	e	\0				
name[1]	B	a	n	a	n	a	\0			
Name[2]	C	h	e	r	r	y	\0			



Example: 1

```
#include <stdio.h>

int main() {
    // Declare and initialize 2D array of strings
    char month[5][10] = {"January", "February", "March",
    "April", "May"};

    int i;

    printf("First 5 Months of the Year:\n");
    printf("-----\n");

    for (i = 0; i < 5; i++) {
        printf("%d: %s\n", i + 1, month[i]);
    }

    return 0;
}
```

Output

First 5 Months of the Year:

1: January
2: February
3: March
4: April
5: May



Example: 2

```
#include<stdio.h>
int main(){
char std_database[50][10]; // student std_database
int i;
printf("Enter Name of the Students : \n\n");
for(i=0;i<4;i++){
    printf("%d: ", i+1);
    gets(std_database[i]); // read name of the students
}
printf(" \n\nName of the Students :\n");
printf("_____\n");
for(i=0;i<4;i++){
    printf("%d: ", i+1);
    puts(std_database[i]); // display name of the students
}
}
```

Output

Enter Name of the Students :

- 1: Anil
- 2: Patel
- 3: Shail
- 4: Amin

Name of the Students :

-
- 1: Anil
 - 2: Patel
 - 3: Shail
 - 4: Amin

- It means a 2D character array, i.e., an array of strings.
- You can think of it as a table of characters with:
- 50 rows → maximum 50 strings (students)
- 10 columns → each string can have up to 9 characters + 1 for '\0'

- x **DIGITAL LEARNING CONTENT**



Parul® University

