Chapter 8 : Array

- Array is fixed sized sequence collection of elements of same data type.
- It is simply a grouping of like-type data.
- It is simplest form, an array can be used to represent a list of numbers, or a list of names.
- Array index starts from 0.

Uses of array

- List of temperatures recorded every hour in a day or a month or a year.
- List of employees in organization.
- List of products and their cost sold by a store.
- Test score of a class of students.
- List of customers and their telephone numbers.
- Table of daily rainfall data.

Types of array

- One-dimensional array
- Two-dimensional array
- Multi-dimensional array

One-dimensional array

- A list of items can be given one variable name using only one subscript and such a variable is called a single-subscripted variable a one-dimensional array.
- If we want to calculate average of n values of x we can do easily with array.
- It can be expressed as x[1],x[2],x[3],….x[n].

- If we want to represent a set of five numbers(11,40,2,45,12) by an array variable number, then we may declare the variable number as int number[5];

Declaration of array

- Array must be declare before they are used so that computer can allocate space for them in memory.
- General form of declaration type variable_name[size];
- The type specifies the type of elements that will be contained in array, such int, float, or char.
- For example float height[50];
- Any reference to the arrays outside the declared limits would not necessarily cause an error. Rather, it might result in unpredictable program results.
- The size should be either a numeric constant or symbolic constant
- The c language treats characters string simply as arrays of

characters . The size in a character string represents the

maximum number of characters . For instance char name[10];

- Declares the name as a character array variable that can hold a maximum of characters. We read string constant into variable name. “WELL DONE”.

‘W’	‘E’	‘L’	‘L’		‘D’	‘O’	‘N’	‘E’	‘\0’

- Each character of string is treated as an element of array name is stored in the memory as follows.

Initialization of one-dimensional array

- After array is declared, its elements must be initialized otherwise, they will contain “garbage”.
- An array can be initialized at either of the following stages:
  - At compile time
  - At run time

Compile time initialization

- We can initialize the the elements of arrays in same way as the ordinary variables when they are declared.
- The general form of initialization of array is

  type array_name[size]= { list of values };
- For example

  int number[3] = { 0, 0, 0 };
- The size may be omitted. In such cases, the compiler allocates space for all initialized elements.

  int counter[] = { 1, 1, 1, 1 };
- Character arrays may be initialized in a similar manner.

  char name[ ] = { ‘j’, ‘o’, ‘h’, ‘n’};
- Declares the name to be an array of five characters, initialized with the string “john” ending with null character.

  char name [ ] = “john”;
- The number of initializers may be less than the declared size. In such cases, the remaining elements are initialized to zero, If the array type is numeric and null if type is char.

  int number [5] = {10, 20 };
- Initialize the first two elements to 10 and 20 respectively, and the remaining elements to 0.

char city [5] = { ‘B’ };

Run time initialization

- An array can be explicitly initialized at run time. This approach is usually applied for initializing large arrays.
- For example consider following segment of c program.

```
  for(i=0;i<100;i++)
{
    if (i<50)
        sum[i] = 0.0;
    else
        sum[i] = 1.0;
 }
```

- First 50 element of array sum are initialized to zero while remaining 50 are 1 at run.

Two-dimensional array

- Array variables can store a list of values. There could be situations where a table of values will have to be stored .
- Consider a table shows marks of 3 subjects of 5 students.

	Subject1	Subject2	subject3
Student1	75	67	66
student2	56	58	62
Student3	88	83	72

This table contains total 9 values, three in each line. Each row represents marks of student, each column represents marks of particular subject.

This table discussed above can be defined in c as a[3][3];

- Two-dimensional array are declared as follow

  type array_name[row_size][column_size];

- Each dimension arrays is indexed from zero to its maximum size minus one, the first index selects the row and second index selects column within that row.

Initialization of two-dimensional array

- Initializes the elements of the first row to zero and second row to one. The initialization is done row by row.

  int table[2][3] = { {0, 0, 0}, {1, 1, 1} };

- We can also initialize two-dimensional array in form of matrix.

```
int table[2][3] = {
                    {0, 0, 0},
                    {1, 1, 1}
                    };
```

- the values are missing in an initializer, they are automatically set to zero.

  int table[2][3] = { {1,1}, {2} };

Multi-dimensional array

- C allows arrays of three or more dimensions. The exact limit is determined by the computers.
- The general form of array is type array_name[s1][s2][s3]...[sm];

- Examples of array : int survey [3][5][12];

survey is a four dimensional array declared to contain 180 integer

type.

- The array survey may represent a survey data of rain fall during the last three years from January to December in five cities.

Dynamic arrays

- An array created at compile time by specifying size in source code has a fixed size and cannot be modified at run time.
- The process of allocating memory at compile time is known as static memory allocation and the arrays that receive static memory allocation are called static arrays.
- In c it is possible to allocate memory to arrays at run time. This feature is known as dynamic memory allocation and arrays created at run time are dynamic arrays.
- Dynamic arrays are created using what are known as pointer variables and memory management functions malloc, calloc, realloc.
- The concept of dynamic arrays Is used in creating and manipulating data structers such as linked lists, stacks and queues.

An ‘c’ program of array : Find the sum of 5 numbers

```
#include<stdio.h>
Void main()
{
   int a[5],i,sum=0;
   printf(“enter 5 numbers”);
   for(i=0;i<5;i++)
   {
```

```
        scanf("%d",&a[i]);
        sum=sum+a[i];
        }
        printf("sum=%d",sum);
    }
```

A 'c' program of two-dimensional array: matrix

```
#include<stdio.h>
void main()
{
        int i,j,a[2][2];
        for(i=0;i<2;i++)
        {
        for(j=0;j<2;j++)
        {scanf("%d",&a[i][j]);}}
        for(i=0;i<2;i++)
        {
        for(j=0;j<2;j++)
        {printf("%d ",a[i][j]);}
        {printf("\n");
}
```

Chapter 9: Strings

- C implements the string data structure using arrays of type char.
- You have already used the string extensively.
    - printf("This program is terminated!\n");
    - #define ERR_Message "Error!!"
- Since string is an array, the declaration of a string is the same as declaring a char array.
    - char string_var[30];
    - char string_var[20] = "Initial value";
- Simple String Program:-

```
#include<stdio.h>
void main()
{
        char str1[3],str2[5];
        Printf(“Enter  city”);
        Scanf(“%s”,str1);
        Printf(“Enter second city “);
        Scanf(“%s”,str2);
        Printf(“%s %s”,str1,str2);
}

O/P:-Enter city
New
Enter second city
York
New York
```

- Memory Storage for a String
- The string is always ended with a null character ‘\0’.
- The characters after the null character are ignored.
- e.g., char str[20] = “Initial value”;

- Arrays of Strings
- An array of strings is a two-dimensional array of characters in which each row is one string.
    - char names[People][Length];
- char month[5][10] = {“January”, “February”, “March”, “April”, “May”};

- Input/Output of a String
- The placeholder %s is used to represent string arguments in printf and scanf.

- printf(“Topic: %s\n”, string_var);

➢ The string can be right-justified by placing a positive number in the placeholder.

- printf(“%8s”, str);

➢ The string can be left-justified by placing a negative number in the placeholder.

- Printf(“%-8s”, str);

Right and Left Justification of Strings

The “**%8s**” placeholder displays a string which is right-justified and in 8-columns width.

If the actual string is longer than the width, the displayed field is expanded with no padding.

An Example of Manipulating String with scanf and printf

```
1.  #include <stdio.h>
2.
3.  #define STRING_LEN  10
4.
5.  int
6.  main(void)
7.  {
8.        char dept[STRING_LEN];
9.        int  course_num;
10.       char days[STRING_LEN];
11.       int  time;
12.
13.       printf("Enter department code, course number, days and ");
14.       printf("time like this:\n> COSC 2060 MWF 1410\n> ");
15.       scanf("%s%d%s%d", dept, &course_num, days, &time);
16.       printf("%s %d meets %s at %d\n", dept, course_num, days, time);
17.
18.       return (0);
19. }

Enter department code, course number, days and time like this:
> COSC 2060 MWF 1410
> MATH 1270 TR 800
MATH 1270 meets TR at 800
```

The dept is the initial memory

- String Library Functions
- The string can not be copied by the assignment operator '='.
  - e..g, "str = "Test String"" is not valid.
  - C provides string manipulating functions in the "string.h" library.
  - The complete list of these functions can be found in Appendix B of the textbook.
- Some String Functions from String.h

Function	Purpose	Example

➢ strcpy	➢ Makes a copy of a string	➢ strcpy(s1, “Hi”);
➢ strcat	➢ Appends a string to the end of another string	➢ strcat(s1, “more”);
➢ strcmp	➢ Compare two strings alphabetically	➢ strcmp(s1, “Hu”);
➢ strlen	➢ Returns the number of characters in a string	➢ strlen(“Hi”) returns 2.
➢ strtok	➢ Breaks a string into tokens by delimiters.	➢ strtok(“Hi, Chao”, “ ,”);

- ➢ Functions strcpy and strncpy
- ➢ Function strcpy copies the string in the second argument into the first argument.
    - o e.g., strcpy(dest, “test string”);
    - o The null character is appended at the end automatically.
    - o If source string is longer than the destination string, the overflow characters may occupy the memory space used by other variables.
    - o Function strncpy copies the string by specifying the number of characters to copy.

- You have to place the null character manually.
- e.g., strncpy(dest, “test string”, 6); dest[6] = ‘\0’;
- If source string is longer than the destination string, the overflow characters are discarded automatically.

➢ Functions strcat and strlen
➢ Functions strcat and strncat concatenate the fist string argument with the second string argument.

- strcat(dest, “more..”);
- strncat(dest, “more..”, 3);

➢ Function strlen is often used to check the length of a string (i.e., the number of characters before the fist null character).

- e.g., dest[6] = “Hello”;
  strncat(dest, “more”, 5-strlen(dest));
  dest[5] = ‘\0’;

➢ Example Of Strlen():-

```
#include<stdio.h>
#include<string.h>
void main()
{
        Char str[20];
        int n;
        Printf(“Enter  the word \n”);
        Scanf(“%s”,str);
        n=strlen(str);
        Printf(“Length of  word is %d \n”,n);
}
O/P:-
Enter the word
Hello
Length of word is 5
```

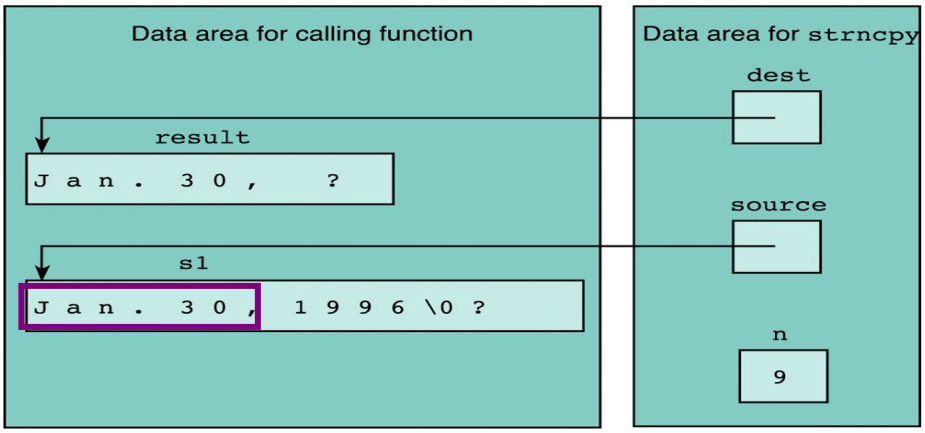
➢ Example of Strcat():-

```
#include<stdio.h>
#include<conio.h>
void main()
{
        char  str1[20],str2[20],n[40];
        Printf("Enter  any  word  \n");
        Scanf("%s",str1);
        Printf("Enter second word \n");
        Scanf("%s",str2);
        strcat(str1,str2);
        Printf("Concat string is %s",str1);
}
O/P:-Enter any word
abc
Enter second word
def
```

➢ Concat string is abcdef

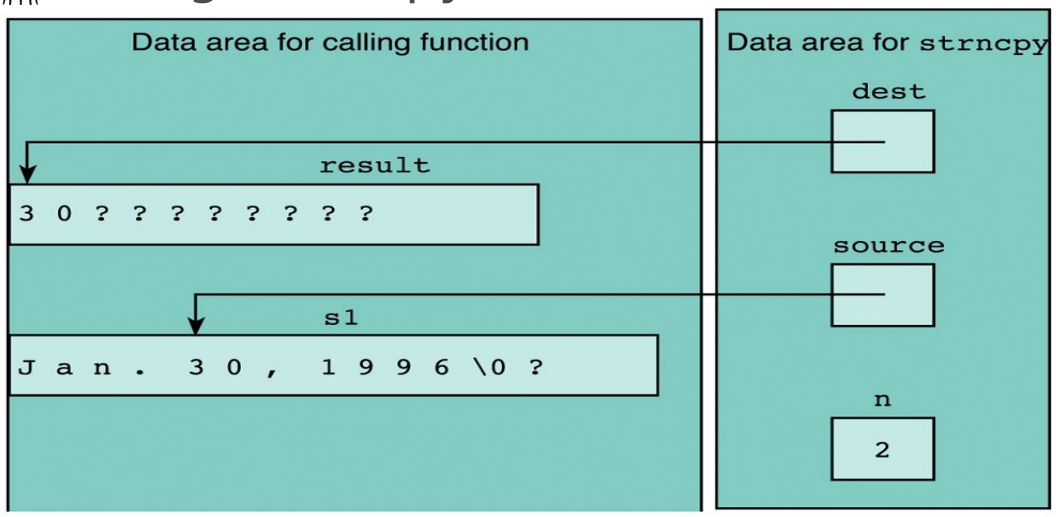
Extracting Substring of a String (1/2)

We can use strncpy to extract



Extracting Substring of a String (2/2)

e.g., strncpy(result, **&s1[5]**, 2);



- Distinction Between Characters and Strings
- The representation of a char (e.g., 'Q') and a string (e.g., "Q") is essentially different.
- A string is an array of characters ended with the null character

Character 'Q' ≠ String "Q"

- String Comparison (1/2)
- Suppose there are two strings, str1 and str2.
  - The condition str1 < str2 compare the initial memory address of str1 and of str2.
- The comparison between two strings is done by comparing each corresponding character in them.
  - The characters are comapared against the ASCII table.
  - "thrill" < "throw" since 'i' < 'o';

- “joy” < joyous“;

➢ The standard string comparison uses the strcmp and strncmp functions

➢ Relationship	➢ Returned Value	➢ Example
➢ str1 < str2	➢ Negative	➢ “Hello”< “Hi”
➢ str1 = str2	➢ 0	➢ “Hi” = “Hi”
➢ str1 > str2	➢ Positive	➢ “Hi” > “Hello”

➢ One can check if two strings are the same by

➢ if(strcmp(str1, str2) != 0)
printf(“The two strings are different!”);