Array and String

Dr.Pooja Bhatt,

Associate Professor

Artificial Intelligence & Data Science department

Parul® University

Arrays and String

Outline

- ❖Introduction of Array
- ❖Pointer and Arrays
- ❖Array of Functions
- ❖String

Need of Array Variable

- Suppose we need to store `rollno` of the student in the integer variable.

Declaration
```
int rollno;
```

- Now we need to store `rollno` of 100 students.

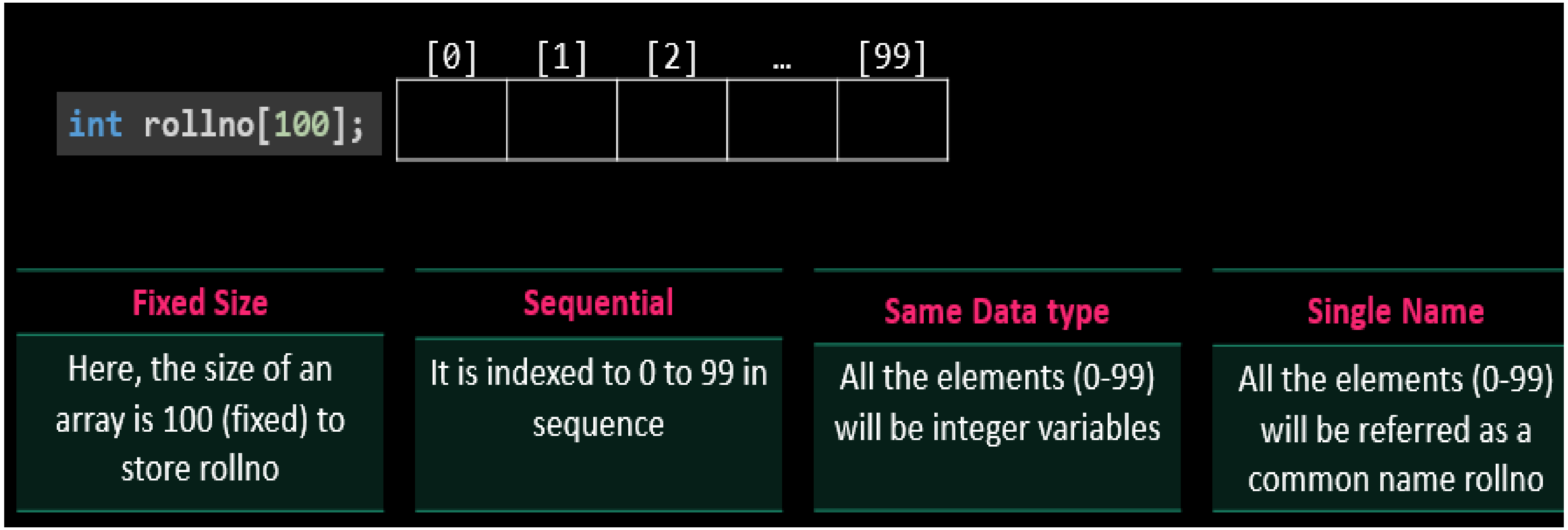
Declaration
```
int rollno101, rollno102, rollno103, rollno104...;
```

- This is not appropriate to declare these many integer variables.
  e.g. 100 integer variables for `rollno`.

- Solution to declare and store multiple variables of similar type is an array.
- An array is a variable that can store multiple values.

Definition: Array

- An array is a fixed size sequential collection of elements of same data type grouped under single variable name.



```
int rollno[100];
```

Fixed Size	Sequential	Same Data type	Single Name
Here, the size of an array is 100 (fixed) to store rollno	It is indexed to 0 to 99 in sequence	All the elements (0-99) will be integer variables	All the elements (0-99) will be referred as a common name rollno

Definition: Array

- An array is defined as the collection of similar type of data items stored at contiguous memory locations. Arrays are the derived data type in C programming language which can store the primitive type of data such as int, char, double, float, etc.
- It also has the capability to store the collection of derived data types, such as pointers, structure, etc.
- The array is the simplest data structure where each data element can be randomly accessed by using its index number.

Advantage of C Array

1) **Code Optimization**: Less code to the access the data.

2) **Ease of traversing**: By using the for loop, we can retrieve the elements of an array easily.

3) **Ease of sorting**: To sort the elements of the array, we need a few lines of code only.

4) **Random Access**: We can access any element randomly using the array.

Disadvantage of C Array

Fixed Size: We define the size at the time of declaration of the array, we can't exceed the limit. So, it doesn't grow the size dynamically like LinkedList

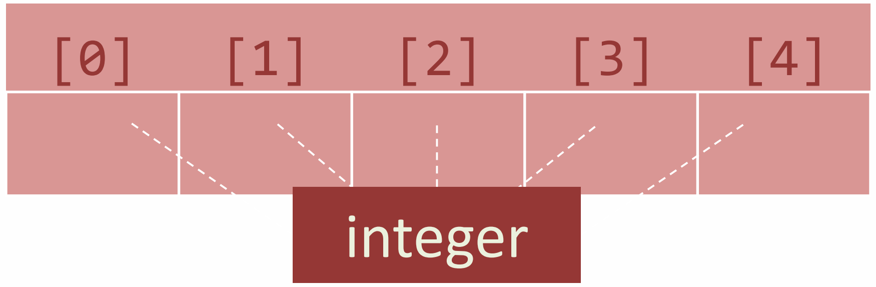
Declaring an array

Syntax

```
data-type variable-name[size];
```

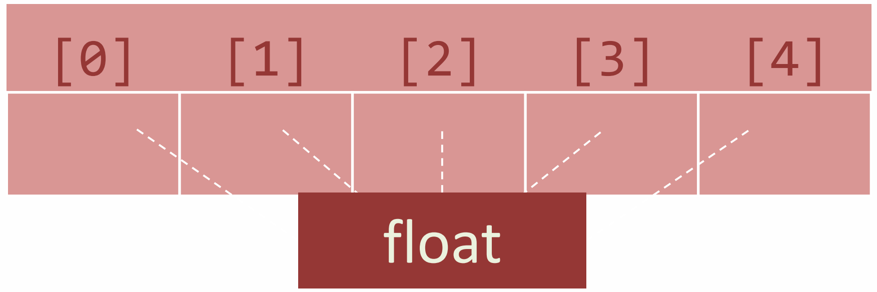
Integer Array

```
int mark[5];
```



Float Array

```
float avg[5];
```



- ❑ By default array index starts with 0.
- ❑ If we declare an array of size 5 then its index ranges from 0 to 4.
- ❑ First element will be store at mark[0] and last element will be stored at mark[4] not mark[5].
- ❑ Like integer and float array we can declare array of type char.

Parul® University

DIGITAL LEARNING CONTENT

Initialing and Accessing an Array

Declaring, initializing and accessing single integer variable

```
int mark=90;      //variable mark is initialized with value 90
printf("%d",mark); //mark value printed
```

Declaring, initializing and accessing integer array variable

```
int mark[5]={85,75,76,55,45}; //mark is initialized with 5 values
printf("%d",mark[0]); //prints 85
printf("%d",mark[1]); //prints 75
printf("%d",mark[2]); //prints 65
printf("%d",mark[3]); //prints 55
printf("%d",mark[4]); //prints 45
```

	[0]	[1]	[2]	[3]	[4]
mark[5]	85	75	65	55	45

Read(Scan) Array Elements

Reading array without loop

```
void main()
{
  int mark[5];
  printf("Enter array element=");
  scanf("%d",&mark[0]);
  printf("Enter array element=");
  scanf("%d",&mark[1]);
  printf("Enter array element=");
  scanf("%d",&mark[2]);
  printf("Enter array element=");
  scanf("%d",&mark[3]);
  printf("Enter array element=");
  scanf("%d",&mark[4]);
  printf("%d",mark[0]);
  printf("%d",mark[1]);
  printf("%d",mark[2]);
  printf("%d",mark[3]);
  printf("%d",mark[4]); }
```

Reading array using loop

```
void main()
{
  int mark[5],i;
  for(i=0;i<5;i++)
  {
   printf("Enter array element=");
   scanf("%d",&mark[i]);
  }
  for(i=0;i<5;i++)
  {
   printf("%d",mark[i]);
  }
}
```

mark[5]

[0]	[1]	[2]	[3]	[4]
85	75	65	55	45

Parul® University

DIGITAL LEARNING CONTENT

Develop a program to count number of positive or negative number from an array of 10 Number

Program

```
void main(){
    int num[10],i,pos,neg;
    pos = 0;
    neg = 0;
    for(i=0;i<10;i++)
    {
        printf("Enter array element=");
        scanf("%d",&num[i]);
    }
    for(i=0;i<10;i++)
    {
        if(num[i]>0)
            pos=pos+1;
        else
            neg=neg+1;
    }
    printf("Positive=%d,Negative=%d",pos,eg);}
```

Output

```
Enter array element=1
Enter array element=2
Enter array element=3
Enter array element=4
Enter array element=5
Enter array element=-1
Enter array element=-2
Enter array element=3
Enter array element=4
Enter array element=5
Positive=8,Negative=2
```

Two dimensional Array in C

Parul® University

DIGITAL LEARNING CONTENT

Two dimensional Array in C

Syntax

```
data-type variable-name[x][y];
```

Declaration

```
int data[3][3]; //This array can hold 9 elements
```

- A two dimensional array can be seen as a table with ‘x’ rows and ‘y’ columns.
- The row number ranges from 0 to (x-1) and column number ranges from 0 to (y-1).

```
int data[3][3];
```

	Column-0	Column-1	Column-2
Row-0	data[0][0]	data[0][1]	data[0][2]
Row-1	data[1][0]	data[1][1]	data[1][2]
Row-2	data[2][0]	data[2][1]	data[2][2]

Initialing and Accessing a 2D Array: Example-1

Program

```
int data[3][3] = {
{1,2,3}, //row 0 with 3 elements
{4,5,6}, //row 1 with 3 elements
{7,8,9}  //row 2 with 3 elements};
printf("%d",data[0][0]); //1
printf("%d",data[0][1]); //2
printf("%d\n",data[0][2]); //3

printf("%d",data[1][0]); //4
printf("%d",data[1][1]);  //5
printf("%d\n",data[1][2]);  //6

printf("%d",data[2][0]);//7
printf("%d",data[2][1]); //8
printf("%d",data[2][2]); //9

// data[3][3] can be initialized like this also
int data[3][3]={{1,2,3},{4,5,6},{7,8,9}};
```

	Column-0	Column-1	Column-2
Row-0	1	2	3
Row-1	4	5	6
Row-2	7	8	9

Parul® University

DIGITAL LEARNING CONTENT

Read(Scan) 2D Array Elements

Program

```
void main(){
    int data[3][3],i,j;
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            printf("Enter array element=");
            scanf("%d",&data[i][j]);
        }
    }
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            printf("%d",data[i][j]);
        }
        printf("\n"); }}
```

	Column-0	Column-1	Column-2
Row-0	1	2	3
Row-1	4	5	6
Row-2	7	8	9

Output

```
Enter array element=1
Enter array element=2
Enter array element=3
Enter array element=4
Enter array element=5
Enter array element=6
Enter array element=7
Enter array element=8
Enter array element=9
1 2 3
4 5 6
7 8 9
```

String

❑The string can be defined as the one-dimensional array of characters terminated by a null ('\0').

❑The character array or the string is used to manipulate text such as word or sentences.

❑Each character in the array occupies one byte of memory, and the last character must always be 0.

❑The termination character ('\0') is important in a string since it is the only way to identify where the string ends.

```c
char name[10];
```

[0]	[1]	[2]	…	[9]

name[10]

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
C	O	M	P	U	T	E	R	\0	

String declaration and initialization

- There are two ways to declare a string in c language.
- By char array
- By string literal

Declaration

```
char name[10];
```

Initialization method 1:

```
char name[10]={‘C',‘O',‘M',‘P',‘U',‘T',‘E',’R’,'\0'};
```

Initialization method 2:

```
char name[10]=“COMPUTER";
//'\0' will be automatically inserted at the end in this type of declaration.
```

name[10]

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
C	O	M	P	U	T	E	R	\0	

Difference between char array and string literal

- There are two main differences between char array and literal.
- We need to add the null character '\0' at the end of the array by ourself whereas, it is appended internally by the compiler in the case of the character array.
- The string literal cannot be reassigned to another set of characters whereas, we can reassign the characters of the array.

Read String: scanf()

```
void main()
{
    char name[10];
    printf("Enter name:");
    scanf("%s",name);
    printf("Name=%s",name);
}
```

Output

```
Enter name: Parul
Name=Parul
```

Output

```
Enter name: CSE Parul
Name=CSE
```

- ❑ There is no need to use address of (&) operator in scanf to store a string.
- ❑ As string name is an array of characters and the name of the array, i.e., name indicates the base address of the string (character array).
- ❑ scanf() terminates its input on the first whitespace(space, tab, newline etc.) encountered.

Read String: gets()

```
#include<stdio.h>
void main()
{
    char name[10];
    printf("Enter name:");
    gets(name); //read string including white spaces
    printf("Name=%s",name);
}
```

Output

```
Enter name:Parul University
Name=Parul University
```

- gets(): Reads characters from the standard input and stores them as a string.
- puts(): Prints characters from the standard.
- scanf(): Reads input until it encounters whitespace, newline or End Of File(EOF) whereas gets() reads input until it encounters newline or End Of File(EOF).
- gets(): Does not stop reading input when it encounters whitespace instead it takes whitespace as a string.

Parul® University

DIGITAL LEARNING CONTENT

String Handling Functions : strlen()

- ❑C has several inbuilt functions to operate on string. These functions are known as string handling functions.
- ❑`strlen(s1):` returns length of a string in integer

Program

```
#include <stdio.h>
#include <string.h> //header file for string functions
void main()
{
    char s1[10];
    printf("Enter string:");
    gets(s1);
    printf("%d",strlen(s1)); // returns length of s1 in integer
}
```

Output

```
Enter string: Parul
University
15
```

Parul® University

DIGITAL LEARNING CONTENT

String Handling Functions: strcmp()

- ❑ `strcmp(s1,s2):` Returns 0 if s1 and s2 are the same.
- ❑ Returns less than 0 if `s1<s2`.
- ❑ Returns greater than 0 if `s1>s2`.

Program

```
void main()
{
    char s1[10],s2[10];
    printf("Enter string-1:");
    gets(s1);
    printf("Enter string-2:");
    gets(s2);
    if(strcmp(s1,s2)==0)
       printf("Strings are same");
    else
       printf("Strings are not same");}
```

Output

```
Enter string-1:Computer
Enter string-2:Computer
Strings are same
```

Output

```
Enter string-1:Computer
Enter string-2:Computer
Strings are same
```

Parul® University

DIGITAL LEARNING CONTENT

String Handling Functions

For examples consider: `char s1[]="Their",s2[]="There";`

Syntax	Description
`strstr(s1,s2)`	Returns a pointer to the first occurrence of a given string s2 in string s1. `printf("%s",strstr(s1,"he"));` Output : heir
`strcat(s1,s2)`	Appends 2nd string at the end of 1st string. `strcat(s1,s2);` a copy of string s2 is appended at the end of string s1. Now s1 becomes “TheirThere”
`strchr(s1,c)`	Returns a pointer to the first occurrence of a given character in the string s1. `printf("%s",strchr(s1,'i'));` Output : ir
`strcpy(s1,s2)`	**Copies 2nd string to 1st string.** **`strcpy(s1,s2)` copies the string s2 in to string s1 so s1 is now “There”. s2 remains unchanged.**

For examples consider: `char s1[]="Their",s2[]="There";`

Syntax	Description
`strrev(s1)`	**Reverses given string.** **`strrev(s1);` makes string s1 to "riehT"**
`strlwr(s1)`	Converts string s1 to lower case. `printf("%s",strlwr(s1));` Output : their
`strncpy(s1,s2,n)`	Copies first n character of string s2 to string s1 `s1=""; s2="There";` `strncpy(s1,s2,2);` `printf("%s",s1);` Output : Th
`strncat(s1,s2,n)`	Appends first n character of string s2 at the end of string s1. `strncat(s1,s2,2);` `printf("%s", s1);` Output : TheirTh

For examples consider: `char s1[]="Their",s2[]="There";`

ntax	Description
`strncmp(s1,s2,n)`	**Compares first n character of string s1 and s2 and returns similar result as `strcmp()` function.**
`strrchr(s1,c)`	Returns the last occurrence of a given character in a string s1. `printf("%s",strrchr(s2,'e'));` Output : ere

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in