Structured Programming
CSE 103

Professor Dr. Mohammad Abu Yousuf

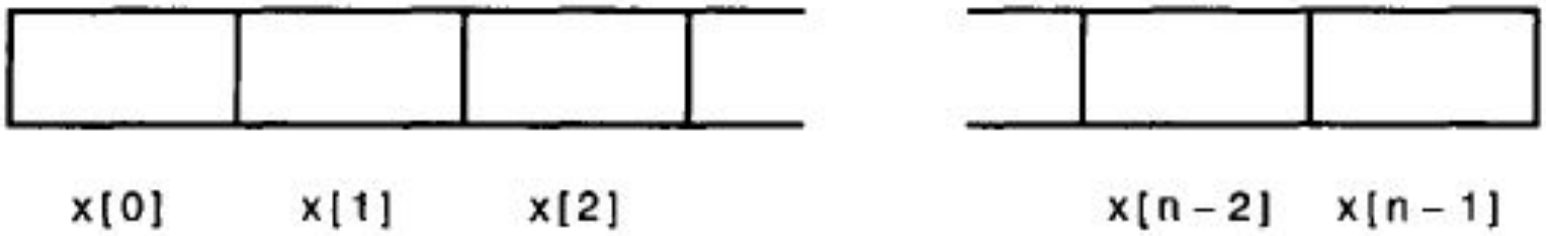
ARRAY

What is array ?

- Many applications require the processing of multiple data items that have common characteristics (e.g., a set of numerical data, represented by x1, x2, . . . ,xn). In such situations it is often convenient to place the data items into an array, where they will all share the same name (e.g., x). The individual data items can be characters, integers, floating-point numbers, etc. However, they must all be of the same type.

- Each array element (i.e., each individual data item) is referred to by specifying the array name followed by one or more subscripts, with each subscript enclosed in square brackets. Each subscript must be expressed as a nonnegative integer. In an n-element array, the array elements are x[0], x[1], x[2] . . . ,x [n – 1]

What is array ?



x is an n-element, one-dimensional array

- Defining an array

  type array_name [array_size];

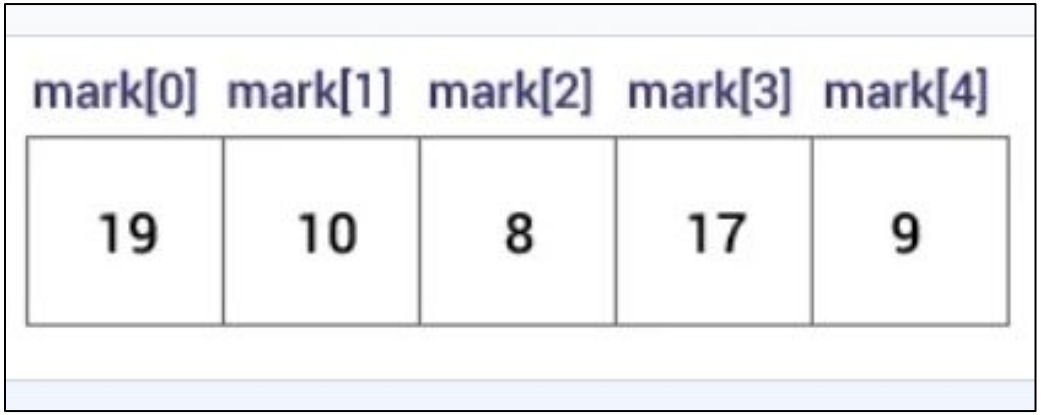
  This is called a single-dimensional array. The array_size must be an integer constant greater than zero and type can be any valid C data type.

  int x[100];

  char text [80] ;

What is array ?

int mark[5] = {19, 10, 8, 17, 9};



What is array ?

- several array definitions that include the assignment of initial values :

  int digits[10] = {1, 2, 3, 4, *5 , 6, 7, 8, 9, 10};*

  char color[3] = { 'R' , 'E', 'D' } ;

  digits[0] = 1
  digits[1] = 2
  digits[2] = 3
  digits[3] = 4
  digits[4] = *5*
  digits[5] = 6
  digits[6] = 7
  digits[7] = *8*
  digits[8] = 9
  digits[9] = 10

  color[0] = 'R'
  color[1] = 'E'
  color[2] = ' D '

What is array ?

- **char color[3] = "RED”;**

- The number of values between braces { } can not be larger than the number of elements that we declare for the array between square brackets [ ].

What is array ?

- **int digits[5] = {3, 3, 3};**

  digits[0] = 3

  digits[1] = 3

  digits[2] = 3

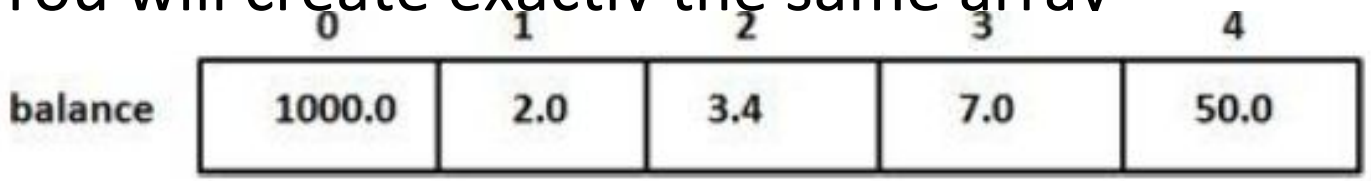
  digits[3] = 0

  digits[4] = 0

  If you omit the size of the array, an array just big enough to hold the initialization is created. Therefore, if you write:

  double balance[ ] = {1000.0, 2.0, 3.4, 17.0, 50.0};

  You will create exactlv the same arrav



```
1.#include<stdio.h>
2.int main()
3.{
4.   int i=0;
5.   int marks[5]={20,30,40,50,60};//declaration and initialization of array
6.   //traversal of array
7.   for(i=0;i<5;i++){
8.         printf("%d \n",marks[i]);
9.   }
10.  return 0;
11.}
```

```
// Program to take 5 values from the user and store them in an array
// Print the elements stored in the array
#include <stdio.h>
int main()
{
  int values[5];
  printf("Enter 5 integers: ");
  // taking input and storing it in an array
  for(int i = 0; i < 5; ++i)
  {
     scanf("%d", &values[i]);
  }

  printf("Displaying integers: ");
  // printing elements of an array
  for(int i = 0; i < 5; ++i)
 {
    printf("%d\n", values[i]);
 }
 return 0;
}
```

Array Input/Output

Example

- **Calculate Average Using Arrays**

  This program takes n number of element from user(where, n is specified by user), stores data in an array and calculates the average of those numbers.

```c
int main(){
    int n, i;
    float num[100], sum=0.0, average;
    printf("Enter the numbers of data: ");
    scanf("%d",&n);
    while (n>100 || n<=0)
    {
        printf("Error! number should in range of (1 to 100).\n");
        printf("Enter the number again: ");
        scanf("%d",&n);
    }
   for(i=0; i<n; ++i)
   {
       printf("%d. Enter number: ",i+1);
       scanf("%f",&num[i]);
       sum+=num[i];
   }
   average=sum/n;
   printf("Average = %.2f",average);
   return 0;
}
```

- Output:

```
Enter the numbers of data: 6
1. Enter number: 45.3
2. Enter number: 67.5
3. Enter number: -45.6
4. Enter number: 20.34
5. Enter number: 33
6. Enter number: 45.6
Average = 27.69
```

- **Display Largest Element of an array**

  This program takes n number of element from user(where, n is specified by user) and stores data in an array. Then, this program displays the largest element of that array using loops.

```c
#include <stdio.h>
int main(){
     int i,n;
     float arr[100];
     printf("Enter total number of elements(1 to 100): ");
     scanf("%d",&n);
     printf("\n");
     for(i=0;i<n;++i)   /* Stores number entered by user. */
     {
         printf("Enter Number %d: ",i+1);
         scanf("%f",&arr[i]);
     }
     for(i=1;i<n;++i)   /* Loop to store largest number to arr[0] */
     {
         if(arr[0]<arr[i]) /* Change < to > if you want to find smallest
              arr[0]=arr[i];
     }
     printf("Largest element = %.2f",arr[0]);
     return 0;
}
```

- Output:

```
Enter total number of elements(1 to 100): 8

Enter Number 1: 23.4
Enter Number 2: -34.5
Enter Number 3: 50
Enter Number 4: 33.5
Enter Number 5: 55.5
Enter Number 6: 43.7
Enter Number 7: 5.7
Enter Number 8: -66.5
```

Largest element= 55.5

```c
/* read in a line of lowercase text to uppercase */

#include <stdio.h>
#include <ctype.h>

#define  SIZE  80

main()
{
    char letter[SIZE];
    int count;

    /* read in the line */

    for (count = 0; count < SIZE; ++count)
        letter[count] = getchar();

    /* display the line in upper case */

    for (count = 0; count < SIZE; ++count)
        putchar(toupper(letter[count]));
}
```

PASSING ARRAYS TO FUNCTIONS

- An entire array can be passed to a function as an argument. The manner in which the array is passed differs markedly, however, from that of an ordinary variable.
- To pass an array to a function, the array name must appear by itself, without brackets or subscripts, as an actual argument within the function call. The corresponding formal argument is written in the same manner, though it must be declared as an array within the formal argument declarations.
- When declaring a one dimensional array as a formal argument, the array name is written with a pair of empty square brackets. The size of the array is not specified within the formal argument declaration.
- Some care is required when writing function prototypes that include array arguments. An empty pair of square brackets must follow the name of each array argument, thus indicating that the argument is an array. If argument names are not included in a function declaration, then an empty pair of square brackets must follow the array argument data type.

PASSING ARRAYS TO FUNCTIONS

```
1.#include <stdio.h>
2.void getarray(int arr[])
3.{
4.    printf("Elements of array are : ");
5.    for(int i=0;i<5;i++)
6.    {
7.        printf("%d ", arr[i]);
8.    }
9.}
10.int main()
11.{
12.   int arr[5]={45,67,34,78,90};
13.   getarray(arr);
14.   return 0;
15.}
```

In this program, we have first created the array **arr[]** and then we pass this array to the function getarray(). The **getarray()** function prints all the elements of the array arr[].

Example

- Write a C program that passes a three-element integer array to a function, where the array elements are altered. The values of the array elements are displayed at three different places in the program, thus illustrating the effects of the alterations.

```c
#include <stdio.h>

void modify(int a[]);          /* function prototype */

main()
{
    int count, a[3];           /* array definition */

    printf("\nFrom main, before calling the function:\n");
    for (count = 0; count <= 2; ++count)   {
        a[count] = count + 1;
        printf("a[%d] = %d\n", count, a[count]);
    }

    modify(a);

    printf("\nFrom main, after calling the function:\n");
    for (count = 0; count <=2; ++count)
        printf("a[%d] = %d\n", count, a[count]);
}
void modify(int a[])           /* function definition */
{
    int count;

    printf("\nFrom the function, after modifying the values:\n");
    for (count = 0; count <= 2; ++count)   {
        a[count] = -9;
        printf("a[%d] = %d\n", count, a[count]);
    }
    return;
}
```

PASSING ARRAYS TO FUNCTIONS

When the program is executed, the following output is generated.

```
From main, before calling the function:
a[0] = 1
a[1] = 2
a[2] = 3

From the function, after modifying the values:
a[0] = -9
a[1] = -9
a[2] = -9

From main, after calling the function:
a[0] = -9
a[1] = -9
a[2] = -9
```

PASSING ARRAYS TO FUNCTIONS

```
// how to pass an array to a function as a pointer

#include <stdio.h>
void printarray(char *arr)
{
   printf("Elements of array are : ");
   for(int i=0;i<5;i++)
   {
      printf("%c ", arr[i]);
   }
}
int main()
{
  char arr[5]={'A','B','C','D','E'};
  printarray(arr);
  return 0;
}
```

Find output

```
#include <stdio.h>

main()
{
    int a, b = 0;
    static int c[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0};

    for (a = 0; a < 10; ++a)
        if ((c[a] % 2) == 0) b += c[a];
    printf("%d", b);
}
```

Find output

- Output of previous program:

  20 (sum of the array elements whose values are even)

Find output

```
#include <stdio.h>

main()
{
    int a, b = 0;
    static int c[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0};

    for (a = 0; a < 10; ++a)
        if ((a % 2) == 0) b += c[a];
    printf("%d", b);
}
```

Find output

- Output of previous program:

  25 (sum of the even array elements)

Thank you