Passing arrays to functions

In this lecture, the following topics are covered:

- ❑ Passing One Dimensional Arrays to Functions
- ❑ Passing Two Dimensional Arrays to Functions

1. Passing One Dimensional Arrays to Functions

Like variables of other data types, we can also pass an array to a function. In some situations, you may want to pass individual elements of the array; while in other situations, you may want to pass the entire array. In this section, we will discuss both cases.

1.1 Passing Individual Elements

The individual elements of an array can be passed to a function by passing either their data values or addresses.

Passing Data Values

Individual elements can be passed in the same manner as we pass variables of any other data type. The condition is just that the data type of the array element must match with the type of the function parameter. Look at Fig. 3.21(a) which shows the code to pass an individual array element by passing the data value.

Calling function

```
main()
{
        int arr[5] ={1, 2, 3, 4, 5};
        func(arr[3]);
}
```

Called function

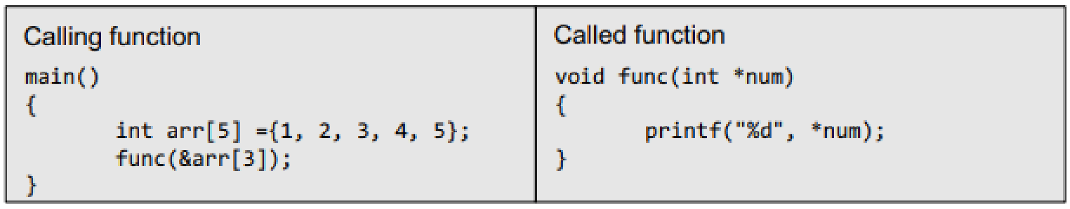
```
void func(int num)
{
        printf("%d", num);
}
```

Figure 3.21(a) Passing values of individual array elements to a function

In the above example, only one element of the array is passed to the called function. This is done by using the index expression. Here, **arr [3]** evaluates to a single integer value.

Passing Addresses

Like ordinary variables, we can pass the address of an individual array element by preceding the indexed array element with the address operator. Therefore, to pass the address of the fourth element of the array to the called function, we will write **&**arr [3]. However, in the called function, the value of the array element must be accessed using the indirection (*) operator. Look at the code shown in Fig. 3.21(b).



Calling function

```
main()
{
        int arr[5] ={1, 2, 3, 4, 5};
        func(&arr[3]);
}
```

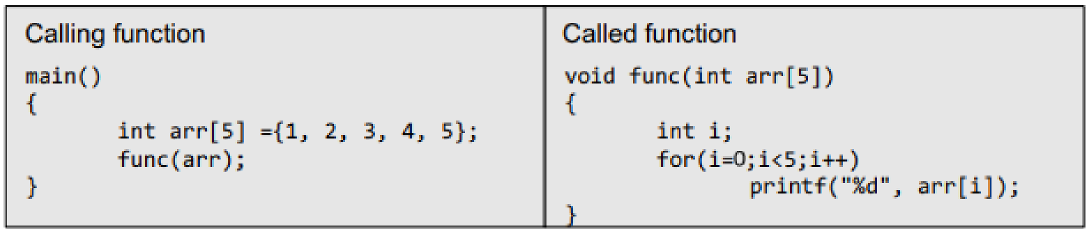
Called function

```
void func(int *num)
{
        printf("%d", *num);
}
```

Figure 3.21(b) Passing addresses of individual array elements to a function

1.2 Passing the Entire Array

- In **C**, the array name refers to the first byte of the array in the memory.
- The address of the remaining elements in the array can be calculated using the array name and the index value of the element. Therefore, **when we need to pass an entire array to a function, we can simply pass the name of the array**.
- Figure 3.22 illustrates the code which passes the entire array to the called function.



Calling function

```
main()
{
        int arr[5] ={1, 2, 3, 4, 5};
        func(arr);
}
```

Called function

```
void func(int arr[5])
{
        int i;
        for(i=0;i<5;i++)
                printf("%d", arr[i]);
}
```

Figure 3.22 Passing entire array to a function

- A function that accepts an array can declare the formal parameter in either of the two following ways:

```
func(int arr[]); or func(int *arr);
```

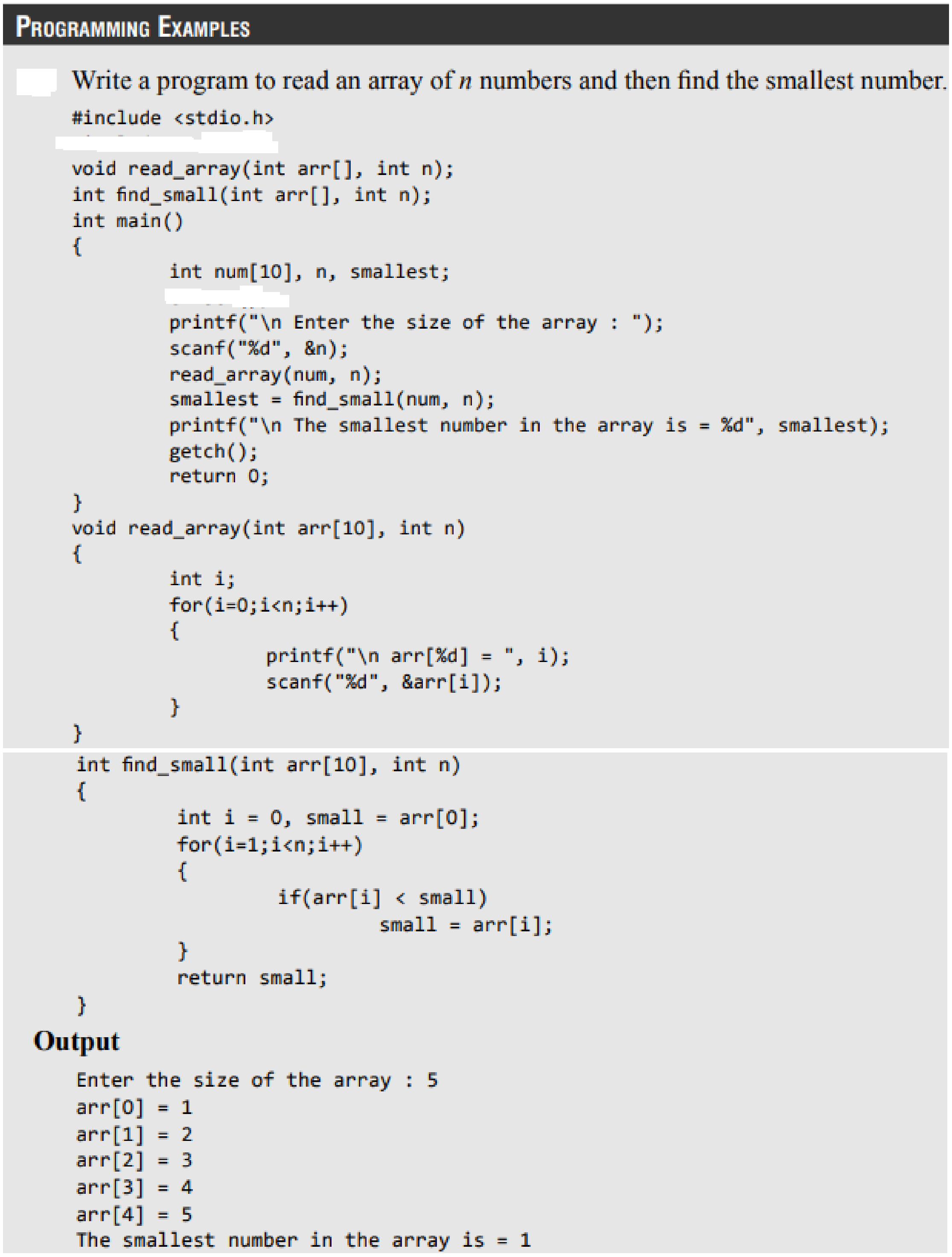
- When we pass the name of an array to a function, the address of the zeroth element of the array is copied to the local pointer variable in the function.
- When a formal parameter is declared in a function header as an array, it is interpreted as a pointer to a variable and not as an array.
- With this pointer variable you can access all the elements of the array by using the expression: **array_name + index**.
- You also have to pass the size of the array as another parameter to the function. So for a function that accepts an array as parameter, the declaration should be as follows.

```
func(int arr[], int n); or func(int *arr, int n);
```

- It is not necessary to pass the whole array to a function.
- We can just pass a part of the array known as a sub-array. A pointer to a sub-array is also an array pointer. For example, if we want to send the array

starting from the third element then we can pass the address of the third element and the size of the sub-array, i.e., if there are 10 elements in the array, and we want to pass the array starting from the third element, then only eight elements would be part of the sub-array. So the function call can be written as:

```
func(&arr[2], 8);
```



PROGRAMMING EXAMPLES

Write a program to read an array of *n* numbers and then find the smallest number.

```
#include <stdio.h>

void read_array(int arr[], int n);
int find_small(int arr[], int n);
int main()
{
        int num[10], n, smallest;

        printf("\n Enter the size of the array : ");
        scanf("%d", &n);
        read_array(num, n);
        smallest = find_small(num, n);
        printf("\n The smallest number in the array is = %d", smallest);
        getch();
        return 0;
}
void read_array(int arr[10], int n)
{
        int i;
        for(i=0;i<n;i++)
        {
                printf("\n arr[%d] = ", i);
                scanf("%d", &arr[i]);
        }
}
int find_small(int arr[10], int n)
{
        int i = 0, small = arr[0];
        for(i=1;i<n;i++)
        {
                if(arr[i] < small)
                        small = arr[i];
        }
        return small;
}
```

Output

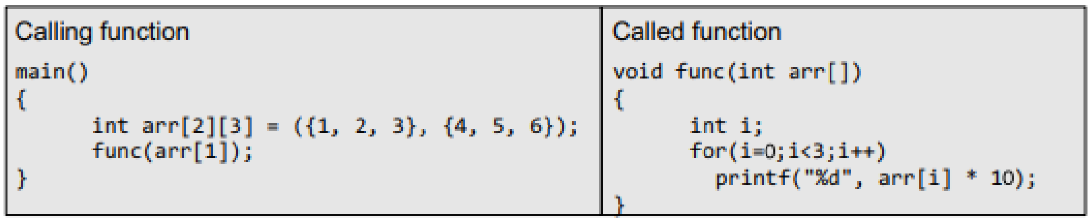
```
Enter the size of the array : 5
arr[0] = 1
arr[1] = 2
arr[2] = 3
arr[3] = 4
arr[4] = 5
The smallest number in the array is = 1
```

2. Passing Two Dimensional Arrays to Functions

- There are three ways of passing a two-dimensional array to a function.
    - First, we can pass individual elements of the array. This is exactly the same as passing an element of a one-dimensional array.
    - Second, we can pass a single row of the two-dimensional array. This is equivalent to passing the entire one-dimensional array to a function that has already been discussed in a previous section.
    - Third, we can pass the entire two-dimensional array to the function.

Passing a Row

A row of a two-dimensional array can be passed by indexing the array name with the row number. Look at Fig. 3.32 which illustrates how a single row of a two-dimensional array can be passed to the called function.



Calling function

```
main()
{
      int arr[2][3] = ({1, 2, 3}, {4, 5, 6});
      func(arr[1]);
}
```

Called function

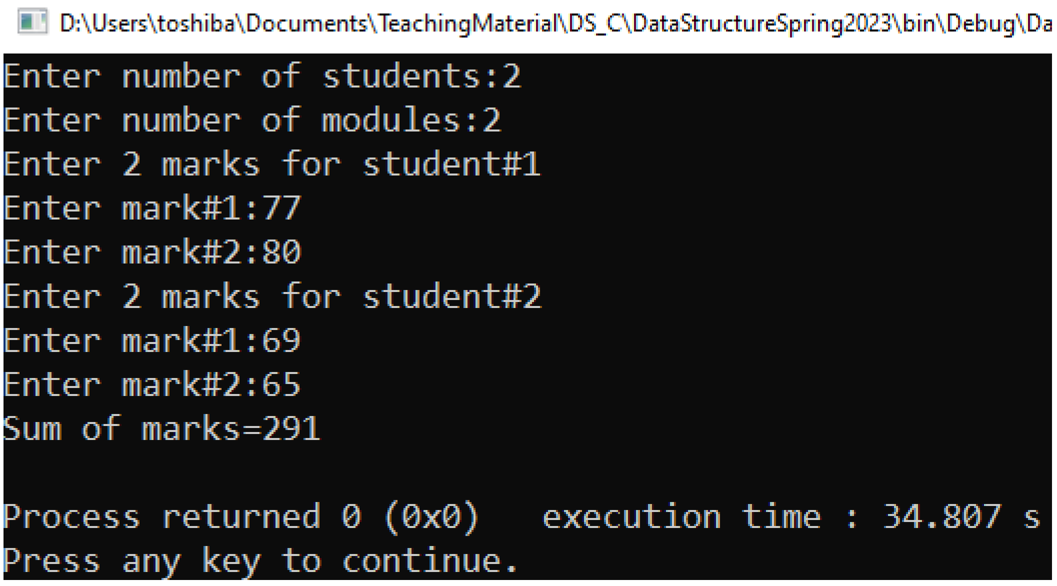
```
void func(int arr[])
{
      int i;
      for(i=0;i<3;i++)
        printf("%d", arr[i] * 10);
}
```

Figure 3.32 Passing a row of a 2D array to a function

Passing the Entire 2D Array

To pass a two-dimensional array to a function, we use the array name as the actual parameter (the way we did in case of a 1D array). However, the parameter in the called function must indicate that the array has two dimensions. The following C program shows how to pass an entire 2D array (**students_marks**) to a function **(sumOf2DArray).**

```c
#include <stdio.h>
#include <stdlib.h>

int sumOf2DArray( int rows, int cols, int arr[rows][cols]  );

int main()
{
    int students, modules;
    printf("Enter number of students:");
    scanf("%d", &students);
    printf("Enter number of modules:");
    scanf("%d", &modules);
    int students_marks [students][modules];
    for(int i=0; i<students; i++){
            printf("Enter %d marks for student#%d\n",modules,(i+1));
        for(int j=0; j<modules; j++){
             printf("Enter mark#%d:",(j+1));
             scanf("%d", &students_marks [i][j]);
        }
    }

    int sumOfMarks =sumOf2DArray( students, modules, students_marks);
    printf("Sum of marks=%d\n", sumOfMarks);


    return 0;
}
int sumOf2DArray( int rows, int cols, int arr[rows][cols] ){
    int sum=0;

    for (int i=0; i< rows; i++){
            for (int j = 0; j< cols; j++){
                sum = sum + arr[i][j];
            }

    }
    return sum;
}
```



D:\Users\toshiba\Documents\TeachingMaterial\DS_C\DataStructureSpring2023\bin\Debug\Da

```
Enter number of students:2
Enter number of modules:2
Enter 2 marks for student#1
Enter mark#1:77
Enter mark#2:80
Enter 2 marks for student#2
Enter mark#1:69
Enter mark#2:65
Sum of marks=291

Process returned 0 (0x0)   execution time : 34.807 s
Press any key to continue.
```