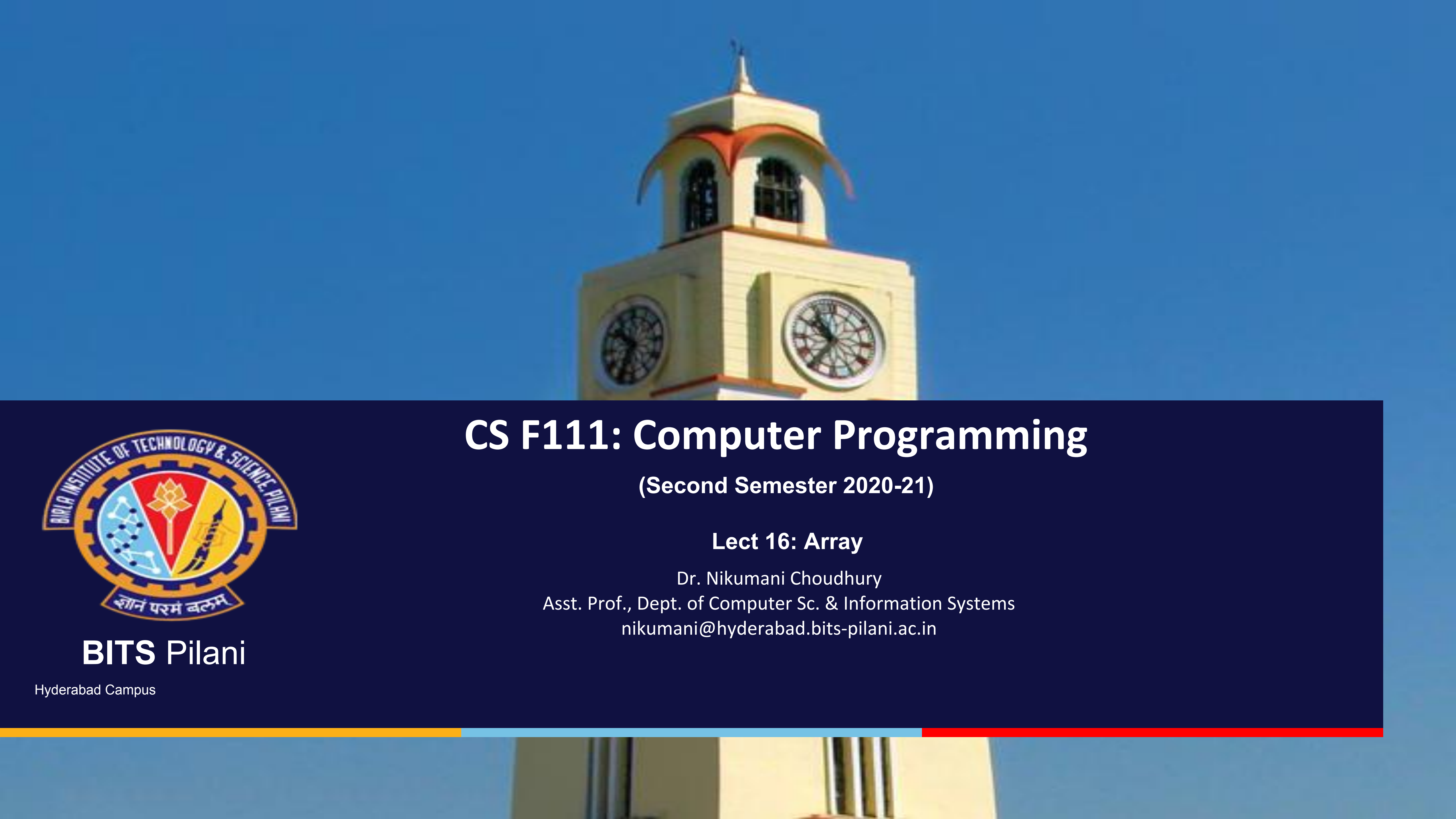


CS F111: Computer Programming

(Second Semester 2020-21)

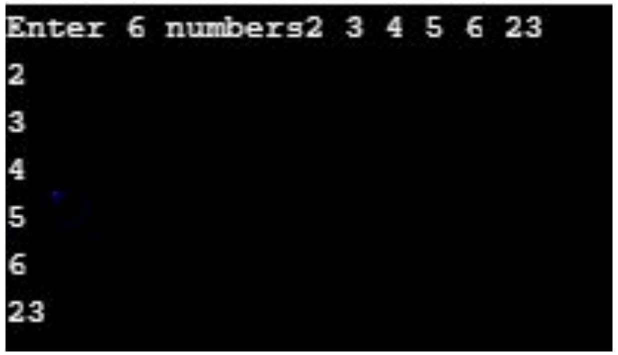
Lect 16: Array

Dr. Nikumani Choudhury
Asst. Prof., Dept. of Computer Sc. & Information Systems
nikumani@hyderabad.bits-pilani.ac.in

BITS Pilani
Hyderabad Campus

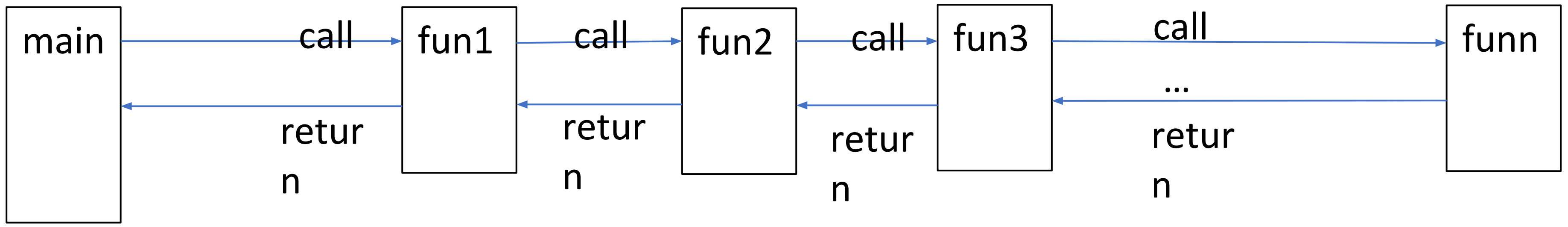
Call-by-Reference: Another example

```c
#include <stdio.h>
#define SIZE 6
void i_array(int f_array[], int num);
int main(void){
  int a[SIZE], i;
  i_array (a, SIZE);
  for (i=0; i<SIZE; i++)
     printf ("%d\n", a[i]);
   printf("\n");
return 0;
}
void i_array (int f_array[], int num) {
  int i;
  printf("Enter %d numbers", num);
  for (i=0; i<num; i++)
    scanf("%d", &f_array[i]);
return;
}
```



```
Enter 6 numbers2 3 4 5 6 23
2
3
4
5
6
23
```

Calling a function from another function



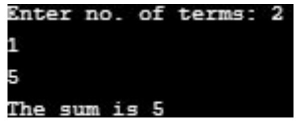
- As each function is allocated separate space in memory (the stack), no conflicts can occur between variable names used in the calling and called functions. Except when used in main, return does not terminate a program.

Ex: $1^1 + 2^2 + 3^3 + \dots + n^n$

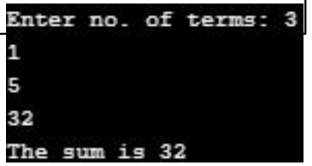
```
int power (int base, int expo);
int sum (int terms);
int main(){
int t;
printf ("Enter no. of terms: ");
scanf("%d", &t);
printf("The sum is %d", sum(t));
return 0;
}
```

```
int sum (int t) {
int i = 0, base = 1;
int sum = 0;
for (i = 0; i < t; i++) {
 sum += power (base, base);
 printf ("%d\n", sum);
 base++;
}
return sum;
}
```

```
int power (int base, int
expo)
{
 int i = 0, result = 1;
 while (++i <= expo)
   result *= base;
 return result;
}
```

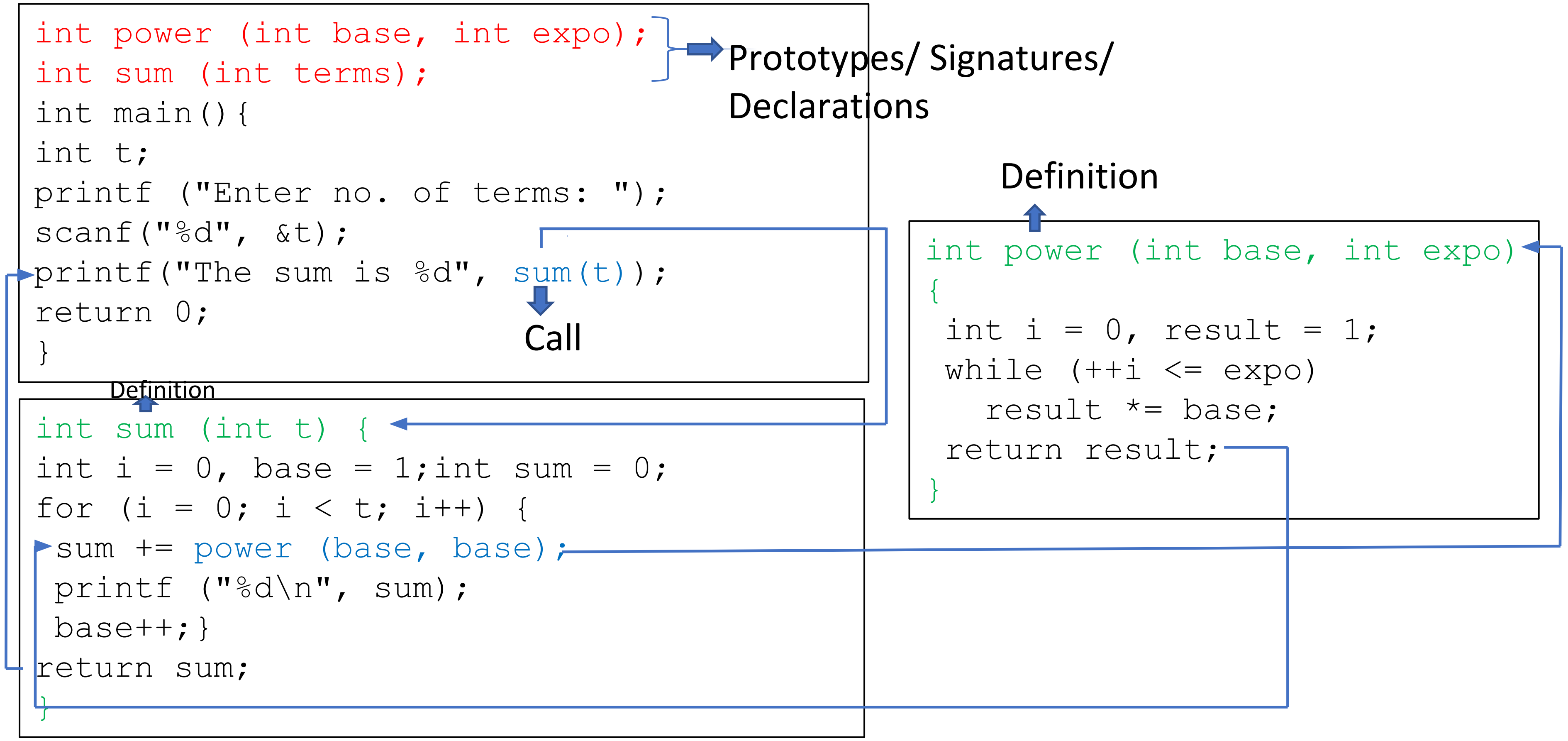


```
Enter no. of terms: 2
1
5
The sum is 5
```



```
Enter no. of terms: 3
1
5
32
The sum is 32
```

Recap: Multi-function program/ Nesting



```
int power (int base, int expo);
int sum (int terms);
int main(){
int t;
printf ("Enter no. of terms: ");
scanf("%d", &t);
printf("The sum is %d", sum(t));
return 0;
}
```

```
int sum (int t) {
int i = 0, base = 1;int sum = 0;
for (i = 0; i < t; i++) {
 sum += power (base, base);
 printf ("%d\n", sum);
 base++;}
return sum;
}
```

```
int power (int base, int expo)
{
 int i = 0, result = 1;
 while (++i <= expo)
   result *= base;
 return result;
}
```

Recap: Can we return multiple values from a function?

- Directly, No. Indirectly, Yes. By using arrays, pointers or structures one can pass multiple values.

```
#include <stdio.h>
#define SIZE 6
void i_array(int f_array[], int num);
int main(void){
  int a[SIZE], i;
  i_array (a, SIZE);
  for (i=0; i<SIZE; i++)
     printf ("%d\n", a[i]);
   printf("\n");
return 0;}
void i_array (int f_array[], int num) {
  int i;
  printf("Enter %d numbers", num);
  for (i=0; i<num; i++)
    scanf("%d", &f_array[i]);
return;
}
```

```
#include <stdio.h>
void compare(int a, int b, int
                  *p, int *q) {
 if (a > b) {
   *p = a; *q = b;
 }
 else {*p = b; *q = a;}
}
int main() {
int g, s, x, y;
printf("Enter two numbers:\n");
scanf("%d%d", &x, &y);
compare(x, y, &g, &s);
printf("\nThe greater is: %d
and smaller is: %d", g, s);
return 0;
}
```

Structures later …

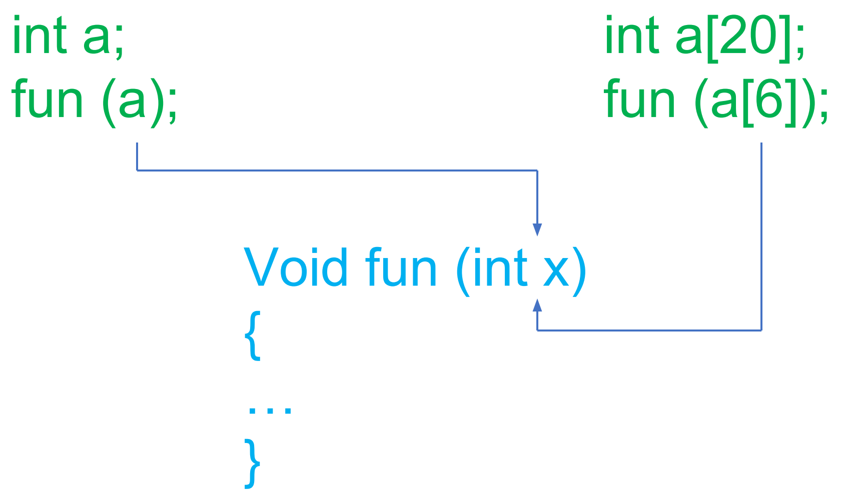
Passing Array elements to a function

- The array is passed by value or by reference
- Examples:

```
/* call by value*/
main ( )
{
  int i;
  int marks[ ] = {12, 34, 65, 45};
  for ( i = 0; i <= 3; i++)
    display (marks [i]);
}
display (int m)
{
  printf (“%d”, m);
}
```

Output:

```
12 34 65 45
```



```
int a;
fun (a);
```

```
int a[20];
fun (a[6]);
```

```
Void fun (int x)
{
…
}
```

Continued…

```
/* call by reference */
main ( )
{
 int i;
 int marks[ ] = {12, 34, 65, 45};
 for ( i = 0; i <= 3; i++)
     display (&marks[i]);
}
display (int *n)
{
   printf (“%d”, *n);
}
```

Output:

```
12 34 65 45
```

$\xrightarrow{\quad}$ N is a pointer variable

$\xrightarrow{\quad}$ value at address (indirection) operator

Passing an entire array to a function

```
/* passing entire array using call by reference */
main ( )
{
 int marks[ ] = {12, 34, 65, 45};
 display (&marks[0], 4);
}
display (int *j, int n)
{
   int i;
   for ( i = 0; i <= n-1; i++)
     {
        printf (“\n element = %d”, *j);
        j++;
     }
}
```

Note: The address of the 0th element (called as base address) can also be passed by just passing the name of the array.

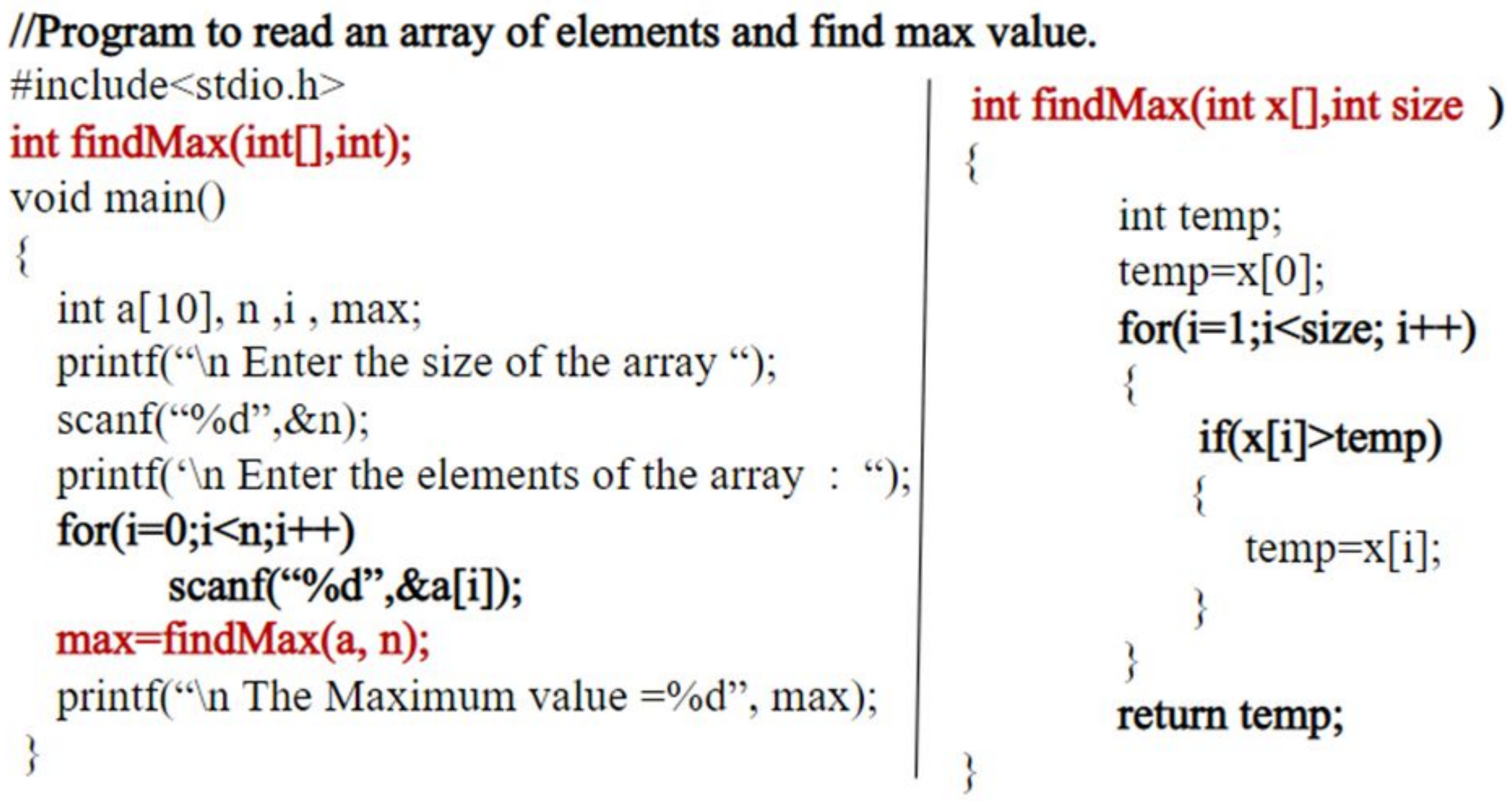
display (&marks[0], 4);

⇕

display (marks, 4);

Passing an array as a constant:

double average (const int ary [ ], int size);



```
//Program to read an array of elements and find max value.
#include<stdio.h>
int findMax(int[],int);
void main()
{
   int a[10], n ,i , max;
   printf(“\n Enter the size of the array “);
   scanf(“%d”,&n);
   printf(‘\n Enter the elements of the array  :  “);
   for(i=0;i<n;i++)
          scanf(“%d”,&a[i]);
   max=findMax(a, n);
   printf(“\n The Maximum value =%d”, max);
 }
```

```
int findMax(int x[],int size  )
{
            int temp;
            temp=x[0];
            for(i=1;i<size; i++)
            {
                 if(x[i]>temp)
                 {
                     temp=x[i];
                 }
            }
            return temp;
}
```

Output:

```
Enter the size of the array 5
Enter the elements of the array: 10  4   56   7   8
The Maximum value = 56
```

Food for thought

Marks [0] $\Leftrightarrow$ *(marks+0)

marks[1] $\Leftrightarrow$ *(marks+1)

…

marks[i] $\Leftrightarrow$ *(marks + i)

marks[i] $\Rightarrow$ *(marks + i) $\Rightarrow$ *(i + marks) $\Rightarrow$ i[marks]

marks[0]	12
marks[1]	34