Introduction of C++
Section 7

Dr/Ghada Maher
Eng/ Hossam Medhat

#DEFINE IN C++

- Preprocessor commands are called DIRECTIVES, and begin with a pound or hash symbol (#).
- No white space should appear before the #, and semi colon is NOT required at the end.
- Many things that can be done during preprocessing phase include :

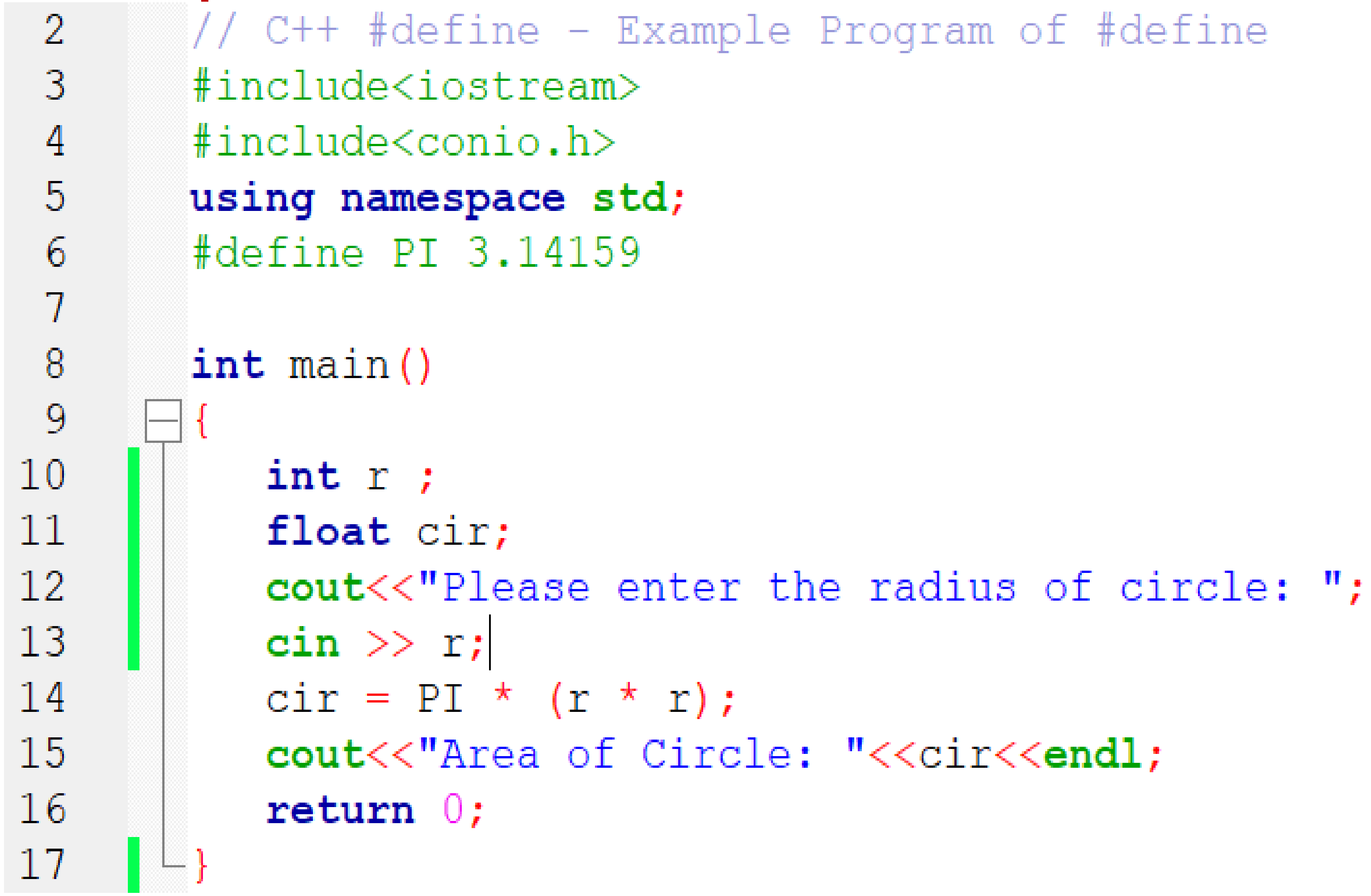
1) Inclusion of other files through #include directive.
2) Definition of symbolic constants through #define directive.

- Example :

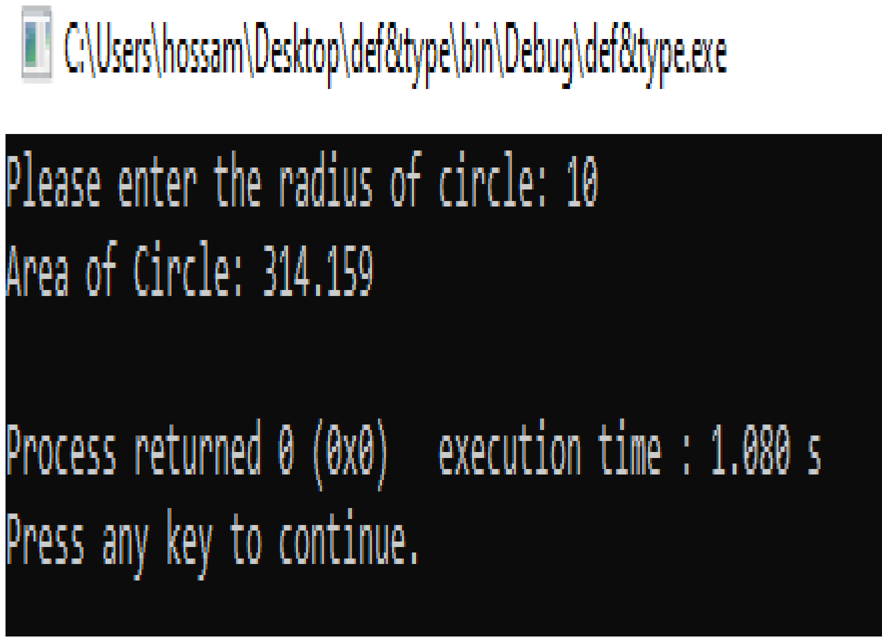
```
#define PI 3.14159
```

#DEFINE IN C++ "CONT"

➢ **Example :**



```
// C++ #define - Example Program of #define
#include<iostream>
#include<conio.h>
using namespace std;
#define PI 3.14159

int main()
{
    int r ;
    float cir;
    cout<<"Please enter the radius of circle: ";
    cin >> r;
    cir = PI * (r * r);
    cout<<"Area of Circle: "<<cir<<endl;
    return 0;
}
```



```
C:\Users\hossam\Desktop\def&type\bin\Debug\def&type.exe

Please enter the radius of circle: 10
Area of Circle: 314.159


Process returned 0 (0x0)   execution time : 1.080 s
Press any key to continue.
```

TYPEDEF IN C++

- Using typedef doest not actually create a new data class, rather it defines a new name for an existing type. This can increase the portability & Readability of a programs only the typedef statements would have to be changed.

- C++ typedef Syntax

```
typedef type
name;
```

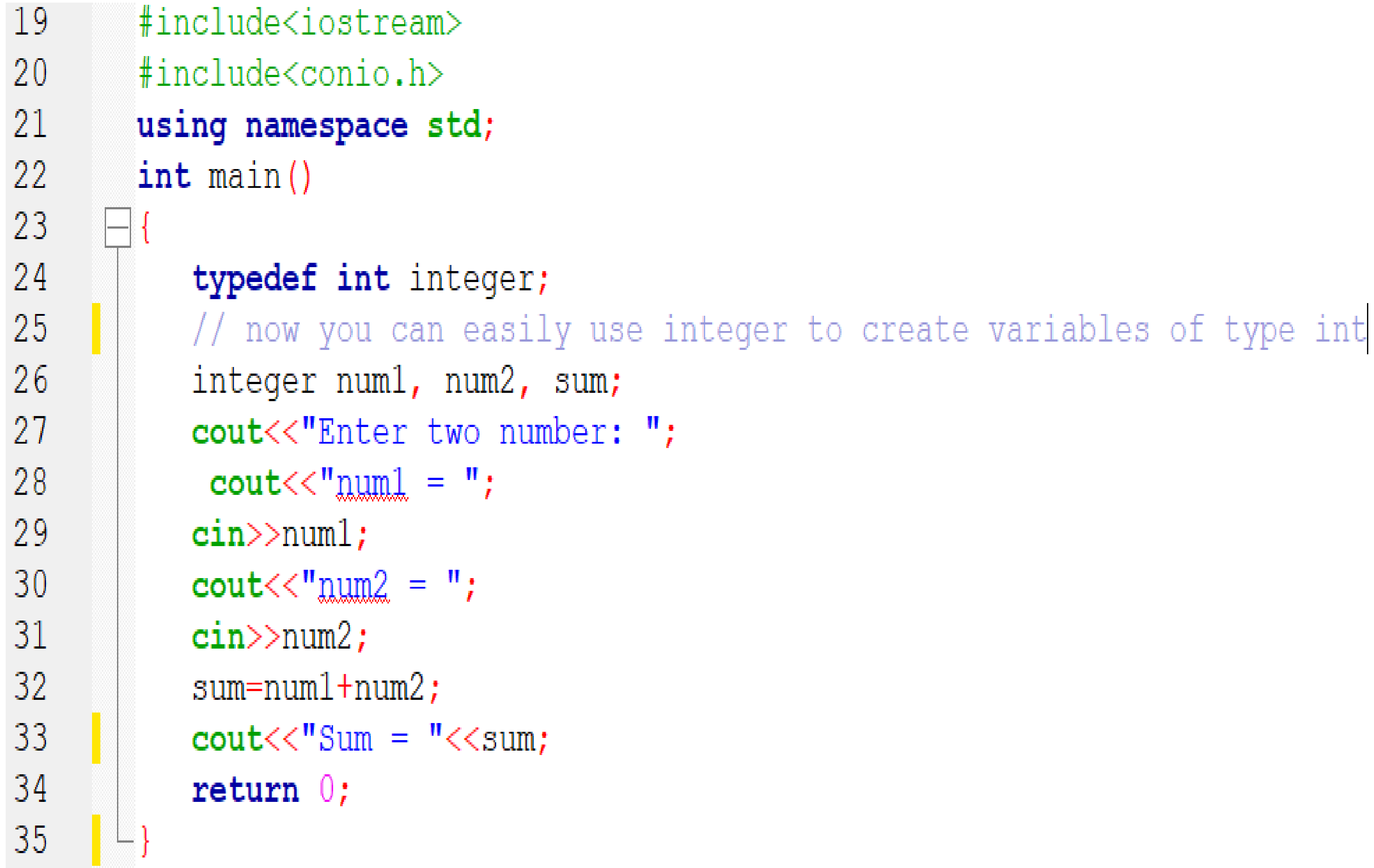
- Example :

```
typedef float amount;
```

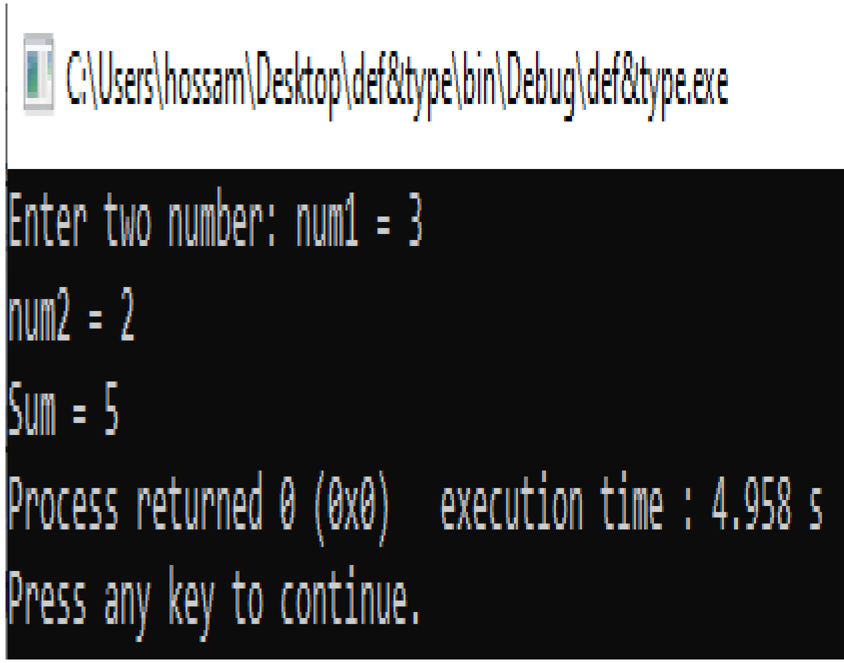
```
amount loan, saving,
instalment;
```

TYPEDEF IN C++ "CONT"

➢ **Example :**



```cpp
#include<iostream>
#include<conio.h>
using namespace std;
int main()
{
    typedef int integer;
    // now you can easily use integer to create variables of type int
    integer num1, num2, sum;
    cout<<"Enter two number: ";
     cout<<"num1 = ";
    cin>>num1;
    cout<<"num2 = ";
    cin>>num2;
    sum=num1+num2;
    cout<<"Sum = "<<sum;
    return 0;
}
```

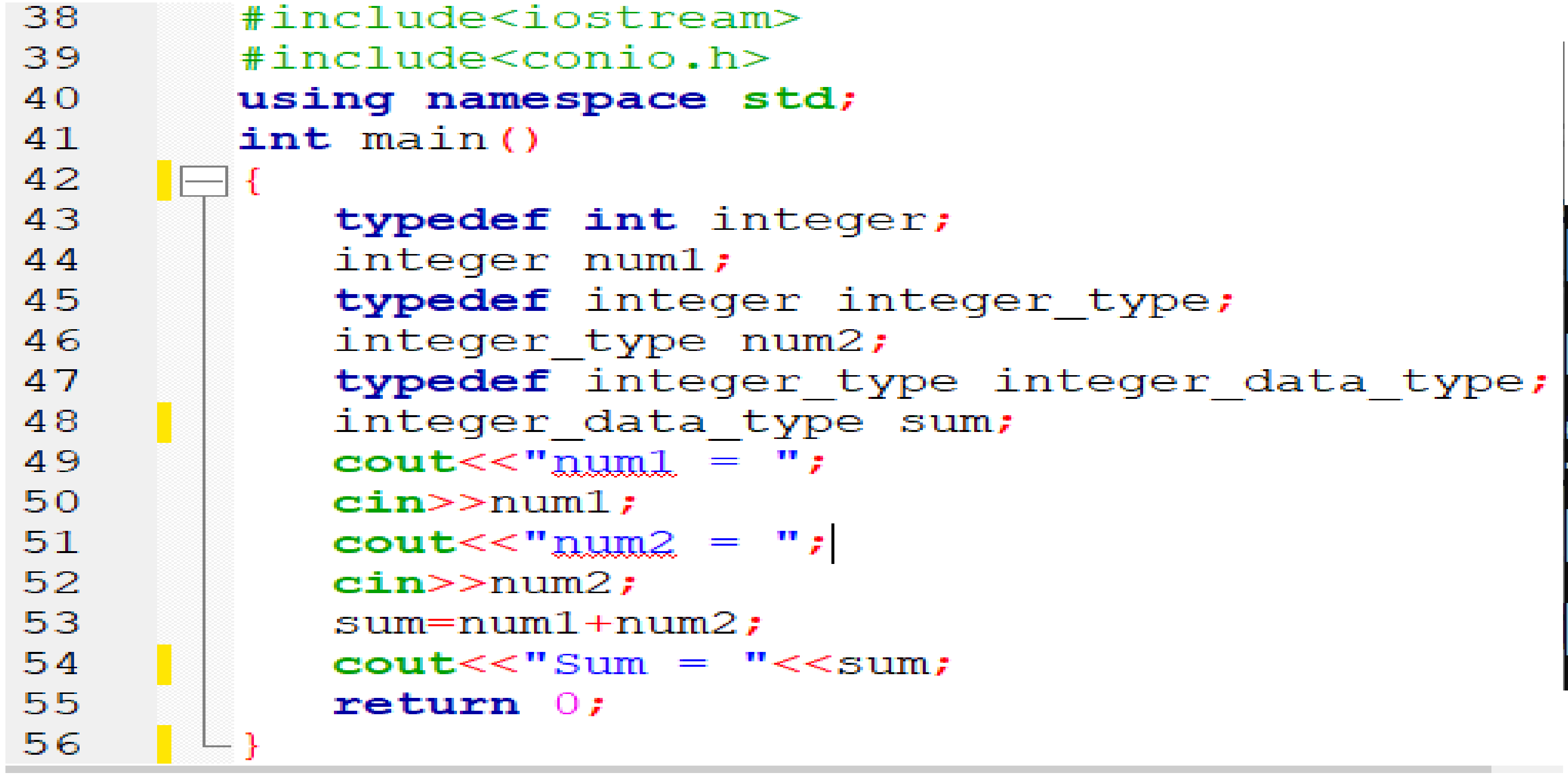


C:\Users\hossam\Desktop\def&type\bin\Debug\def&type.exe

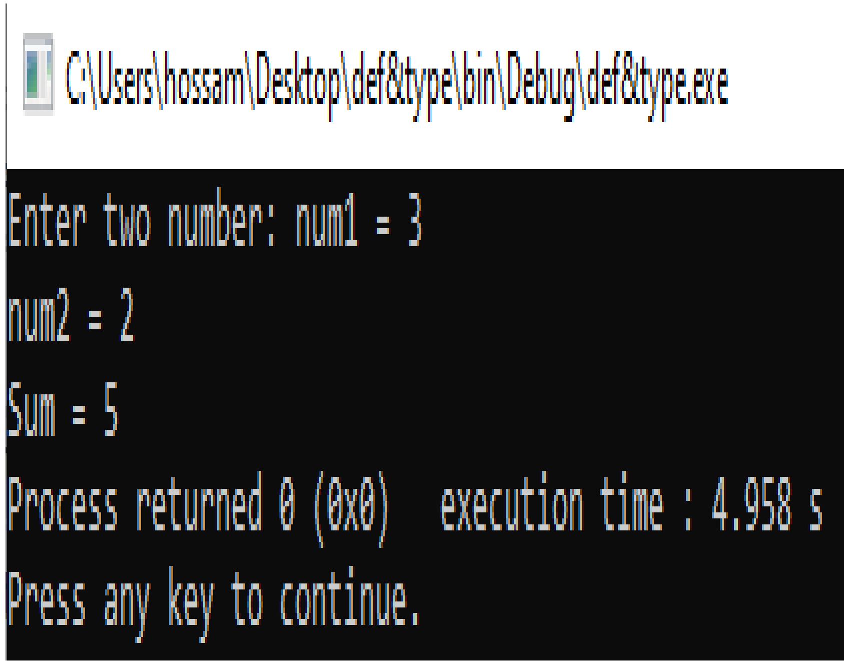
```
Enter two number: num1 = 3
num2 = 2
Sum = 5
Process returned 0 (0x0)   execution time : 4.958 s
Press any key to continue.
```

TYPEDEF IN C++ "CONT"

➢ **Example :**



```
#include<iostream>
#include<conio.h>
using namespace std;
int main()
{
    typedef int integer;
    integer num1;
    typedef integer integer_type;
    integer_type num2;
    typedef integer_type integer_data_type;
    integer_data_type sum;
    cout<<"num1  =  ";
    cin>>num1;
    cout<<"num2  =  ";
    cin>>num2;
    sum=num1+num2;
    cout<<"Sum  =  "<<sum;
    return 0;
}
```



```
C:\Users\hossam\Desktop\def&type\bin\Debug\def&type.exe
Enter two number: num1 = 3
num2 = 2
Sum = 5
Process returned 0 (0x0)   execution time : 4.958 s
Press any key to continue.
```

➢ **The names are integer, integer_type and then integer_data_type, all the three names uses to create variable of type int, as shown in this C++ program.**

C++ Structure Array

- The structure and the array both are C++ derived types. While arrays are collections of analogous elements, structures assemble dissimilar elements under one roof.
- Both the array and the structure allow several values to be treated together as a single data object.
- The arrays and structures can be combined together to form complex data objects.
- There may be structures contained within an array ; also there may be an array as an element of a structure.

ARRAYS OF STRUCTURES

- C++ Structure Array Example :-

```
#include<iostream>
#include<conio.h>
using namespace std;
struct emp {
   int empno;
   char name[20];
};
void main()
{
   emp evar[5];
```

C++ Arrays Within Structures

❖ **Example :-**

```
struct student {
   int rollno ;
   char name[21] ;
   float marks[5] ;     // Array marks is now member element
of
} ;
```

```
student learner ;
learner.marks[2] ;
```

C++ Arrays within structures “Cont”

- the array in a structure may even be two-dimensional as it is shown below :-

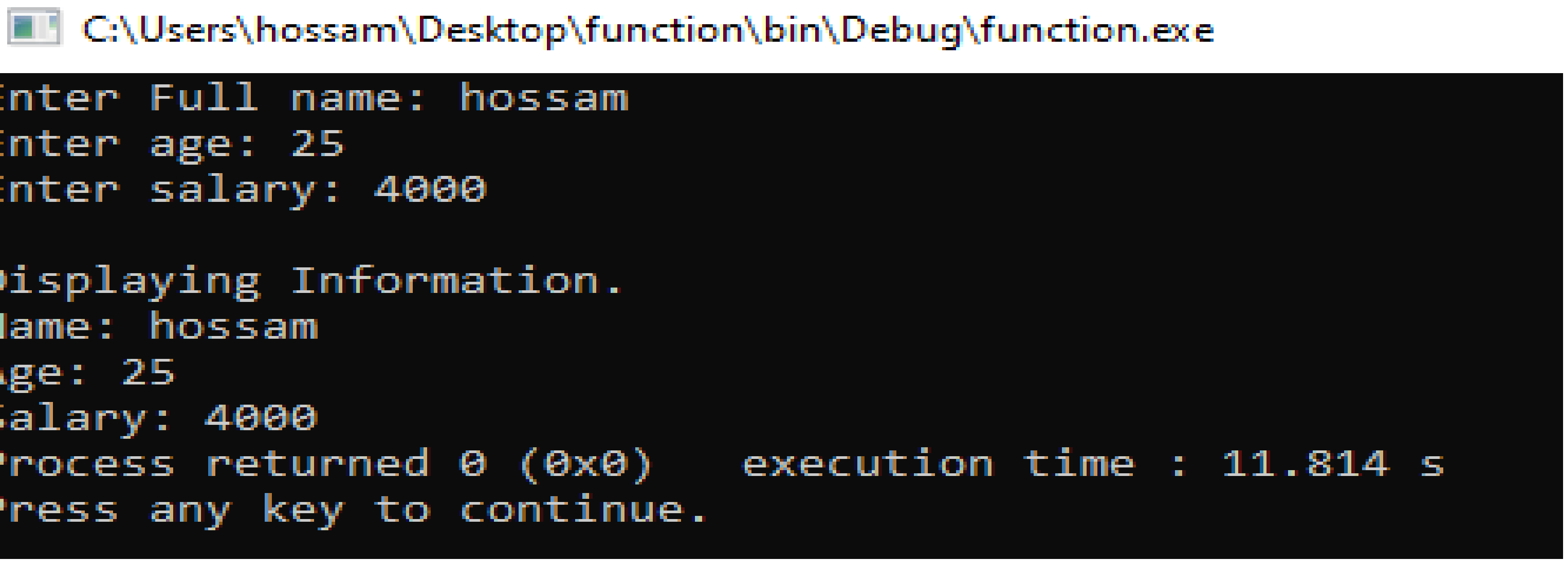
```
struct type {
   int x[5][5] ;     // 5 × 5 array of ints
   float y ;
} var ;
```

To reference integer 2, 4 in x of structure var, we shall write :

```
var.x[2][4] ;
```

C++ PASSING STRUCTURE TO FUNCTION

- **Example :**

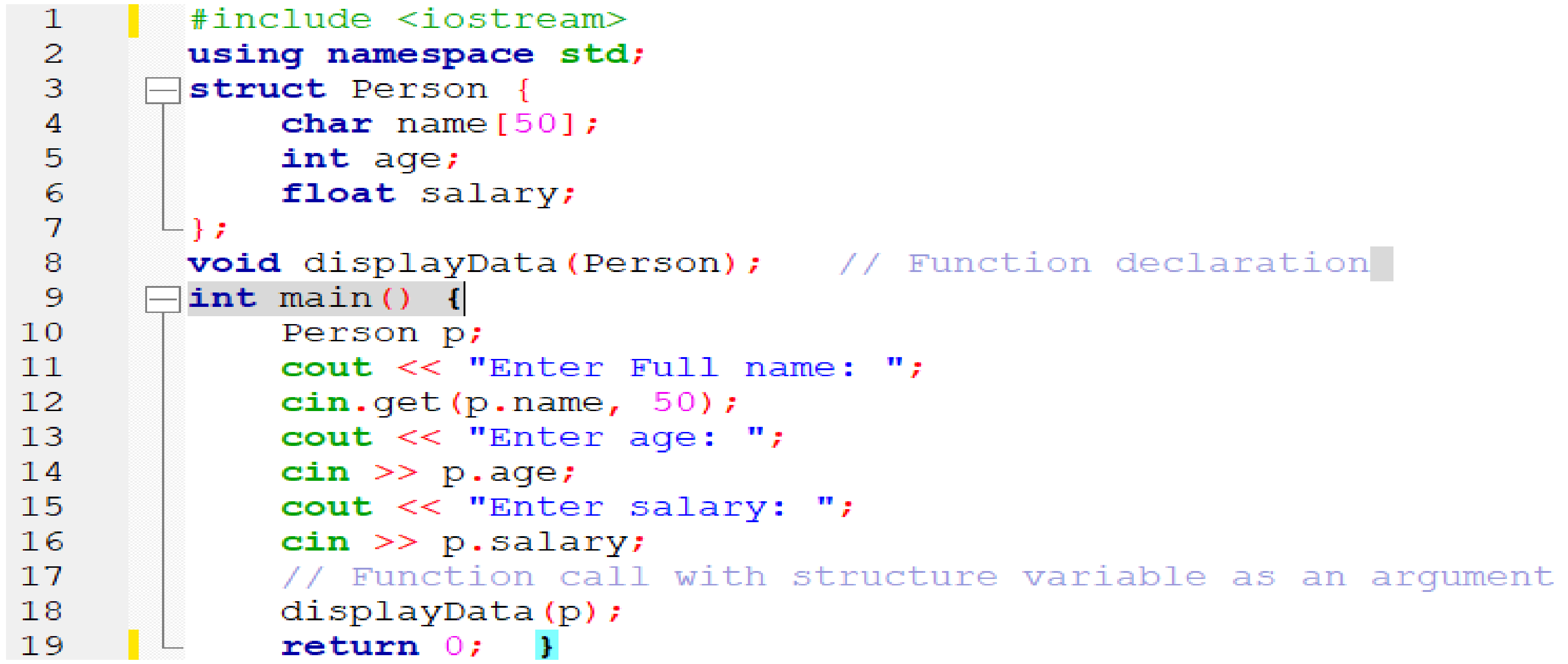


C:\Users\hossam\Desktop\function\bin\Debug\function.exe

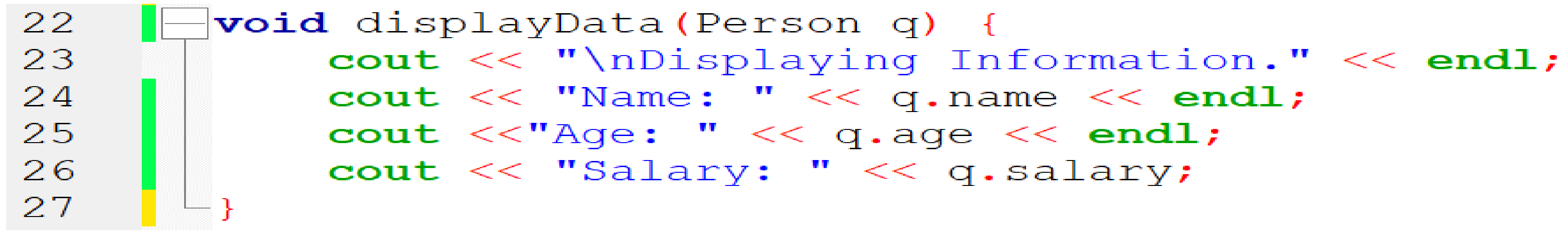
```
Enter Full name: hossam
Enter age: 25
Enter salary: 4000

Displaying Information.
Name: hossam
Age: 25
Salary: 4000
Process returned 0 (0x0)   execution time : 11.814 s
Press any key to continue.
```

C++ PASSING STRUCTURE TO FUNCTION “ CALL BY VALUE”

➢ **Example :**



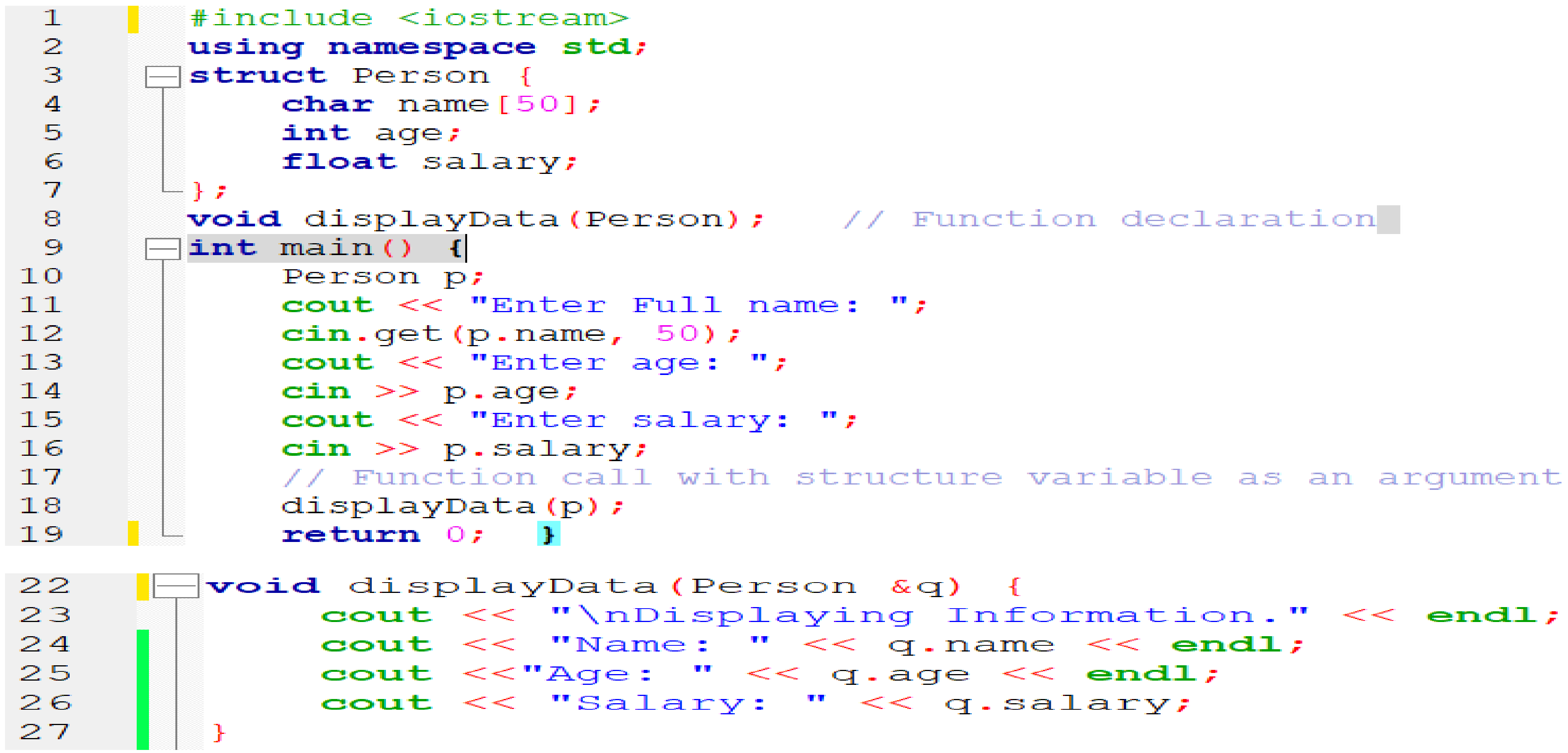
```cpp
#include <iostream>
using namespace std;
struct Person {
    char name[50];
    int age;
    float salary;
};
void displayData(Person);    // Function declaration
int main()  {
    Person p;
    cout << "Enter Full name: ";
    cin.get(p.name, 50);
    cout << "Enter age: ";
    cin >> p.age;
    cout << "Enter salary: ";
    cin >> p.salary;
    // Function call with structure variable as an argument
    displayData(p);
    return 0;   }
```



```cpp
void displayData(Person q)  {
    cout << "\nDisplaying Information." << endl;
    cout << "Name: " << q.name << endl;
    cout <<"Age: " << q.age << endl;
    cout << "Salary: " << q.salary;
}
```

C++ Passing Structure to Function “Call by Reference”

➢ **Example :**



```cpp
#include <iostream>
using namespace std;
struct Person {
    char name[50];
    int age;
    float salary;
};
void displayData(Person);    // Function declaration
int main()  {
    Person p;
    cout << "Enter Full name:  ";
    cin.get(p.name, 50);
    cout << "Enter age: ";
    cin >> p.age;
    cout << "Enter salary: ";
    cin >> p.salary;
    // Function call with structure variable as an argument
    displayData(p);
    return 0;   }
```

```cpp
void displayData(Person &q)  {
    cout << "\nDisplaying Information." << endl;
    cout << "Name: " << q.name << endl;
    cout <<"Age: " << q.age << endl;
    cout << "Salary: " << q.salary;
}
```

C++ NESTED DATA STRUCTURE

- A structure element may be either complex or simple. The simple elements are any of the fundamental data types of C++ i.e., int, float, char, double.
- A structure may consist of an element that itself is complex i.e., it is made up of fundamental types e.g., arrays, structures etc.
- A structure can be nested inside another structure.

C++ NESTED DATA STRUCTURE "CONT"

❖ Example :-

```
struct addr {
   int houseno ;
   char area[26] ;
   char city[26] ;
   char state[26] ;  };
struct emp {
   int empno ;
   char name[26] ;
addr address ;    /*  address is a structure variable itself (of type addr)
                       and it is member of another structure, the emp
structure.*/
} ;
emp worker ;      // create structure variable
```

C++ ACCESSING NESTED STRUCTURE MEMBER

- The members of structures are accessed using dot operator.
- Example :-

➢ To access the city member of address structure which is an element of another structure worker
, we write :

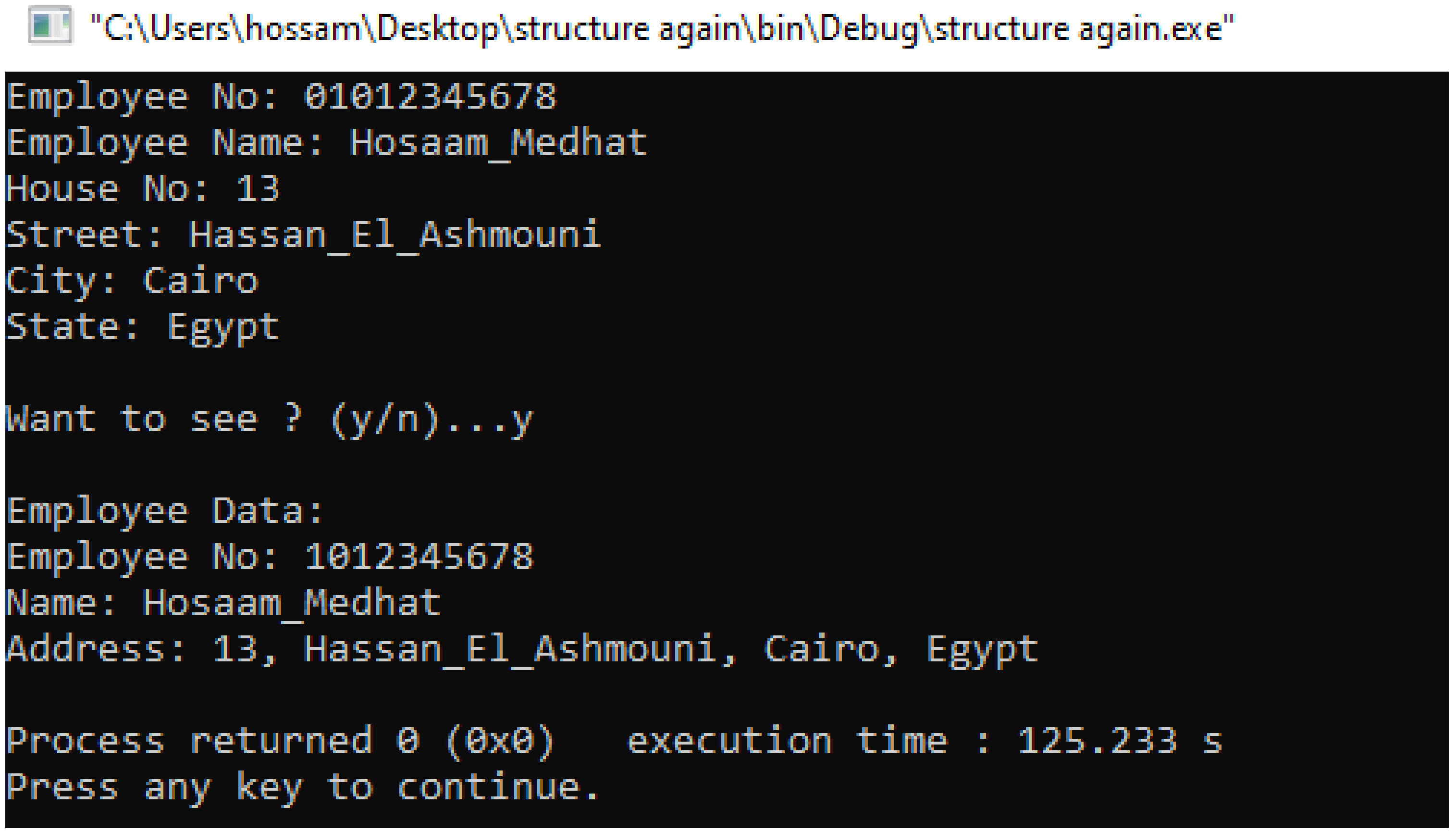
```
worker.address.city
```

➢ To initialize houseno member of address structure element of worker structure, we can write :

```
worker.address.houseno = 1693
```

C++ NESTED DATA STRUCTURE “CONT”

- **Quiz : Write the program that output this data by using C++ Nested Data Structure**



"C:\Users\hossam\Desktop\structure again\bin\Debug\structure again.exe"

```
Employee No: 01012345678
Employee Name: Hosaam_Medhat
House No: 13
Street: Hassan_El_Ashmouni
City: Cairo
State: Egypt

Want to see ? (y/n)...y

Employee Data:
Employee No: 1012345678
Name: Hosaam_Medhat
Address: 13, Hassan_El_Ashmouni, Cairo, Egypt

Process returned 0 (0x0)   execution time : 125.233 s
Press any key to continue.
```

THANKS

Dr/Ghada Maher
Eng/ Hossam Medhat