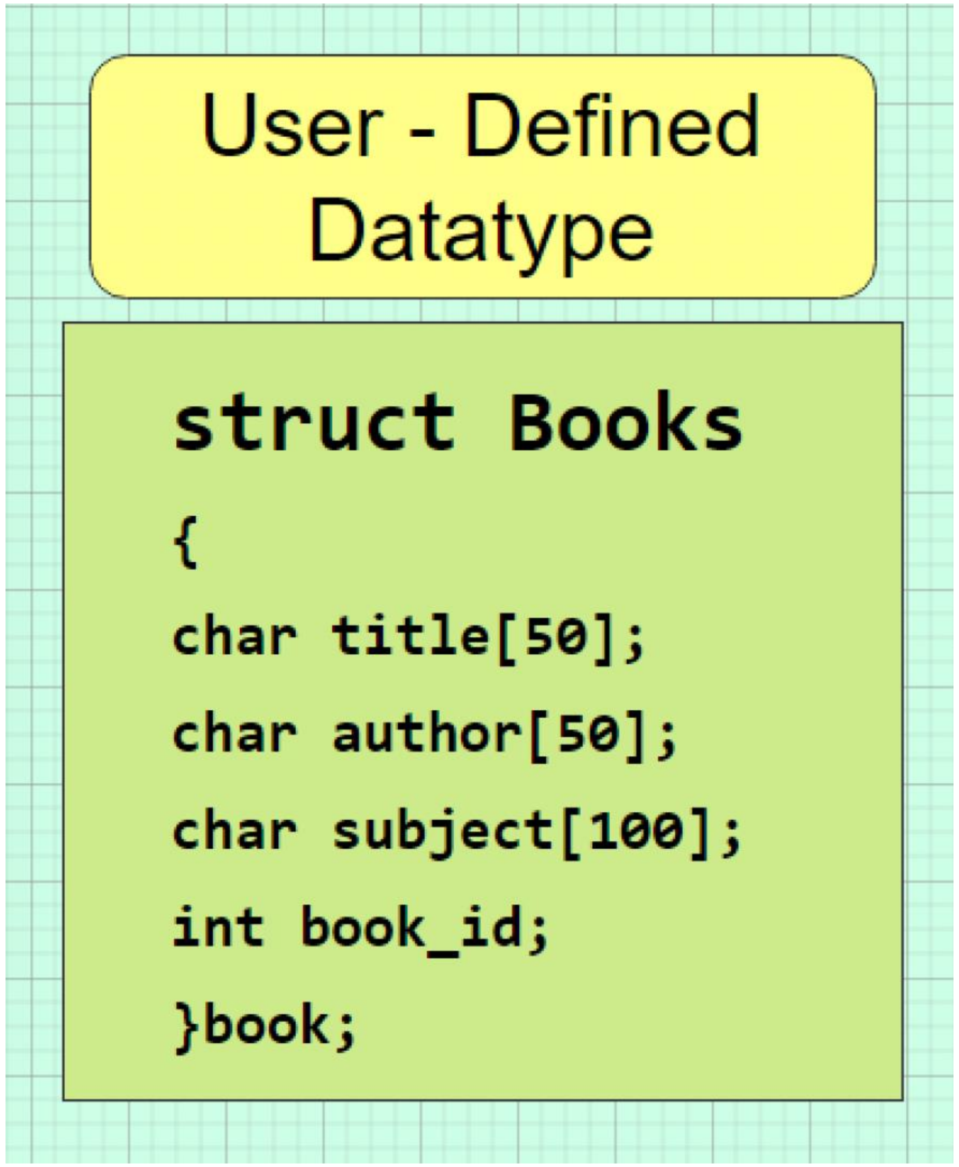
Fundamentals of Computer Programming

CS-110

Course Instructor: Dr. Momina Moetesum

Introduction to Structures

Week 14



```
struct Books
{
char title[50];
char author[50];
char subject[100];
int book_id;
}book;
```

Learning Objectives

01

To understand the concept of objects and attributes

02

To familiarize with syntax of creating an object using structures in C++

03

To use C++ structures to solve real world problems

Structures in C++

- A structure is a user-defined data type in C/C++.
- A structure creates a data type that can be used to group items of possibly different types into a single type.

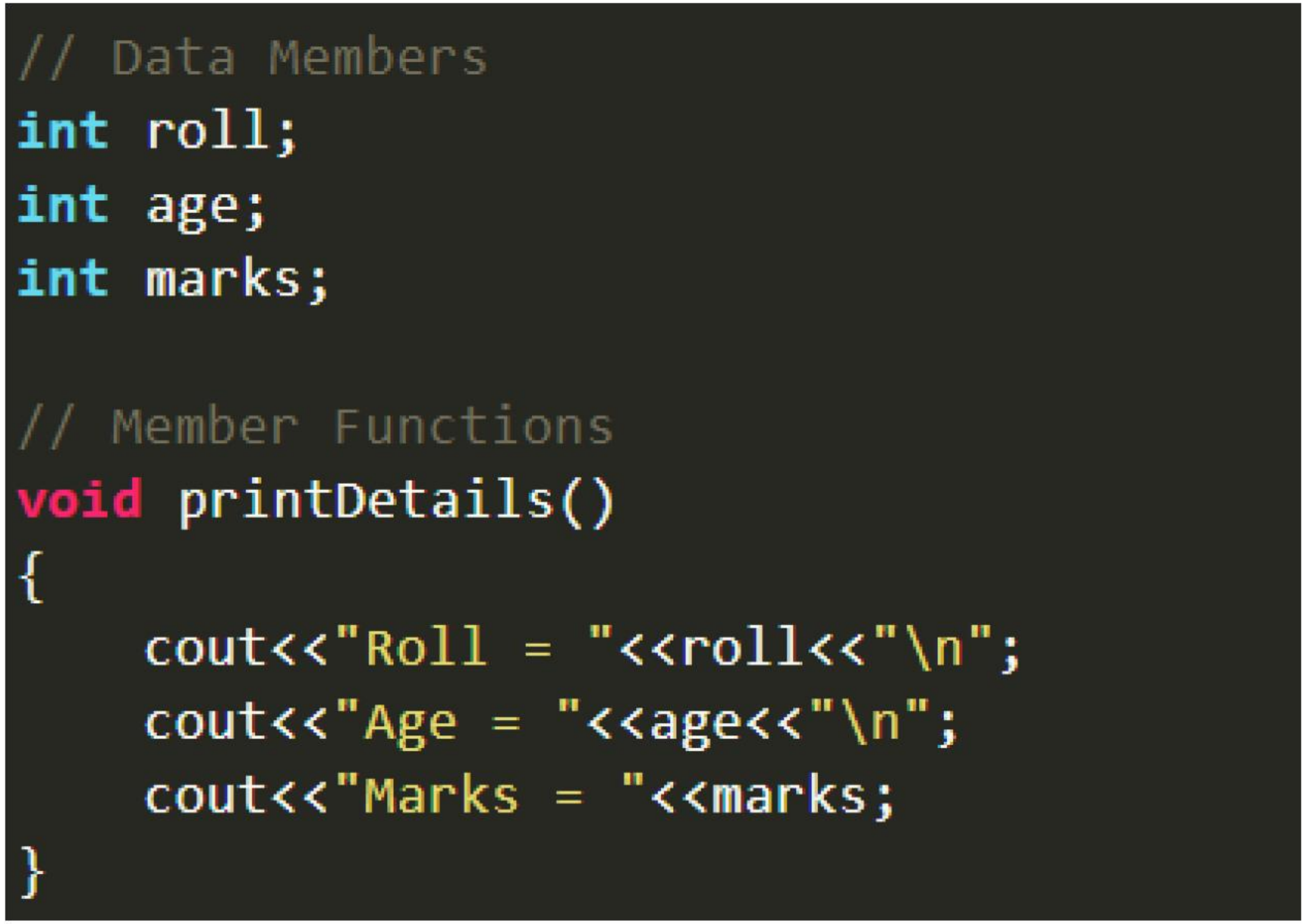
Members

Structures in C++ can contain two types of members:

Data Member: These members are normal C++ variables. We can create a structure with variables of different data types in C++.

Member Functions: These members are normal C++ functions. Along with variables, we can also include functions inside a structure declaration.

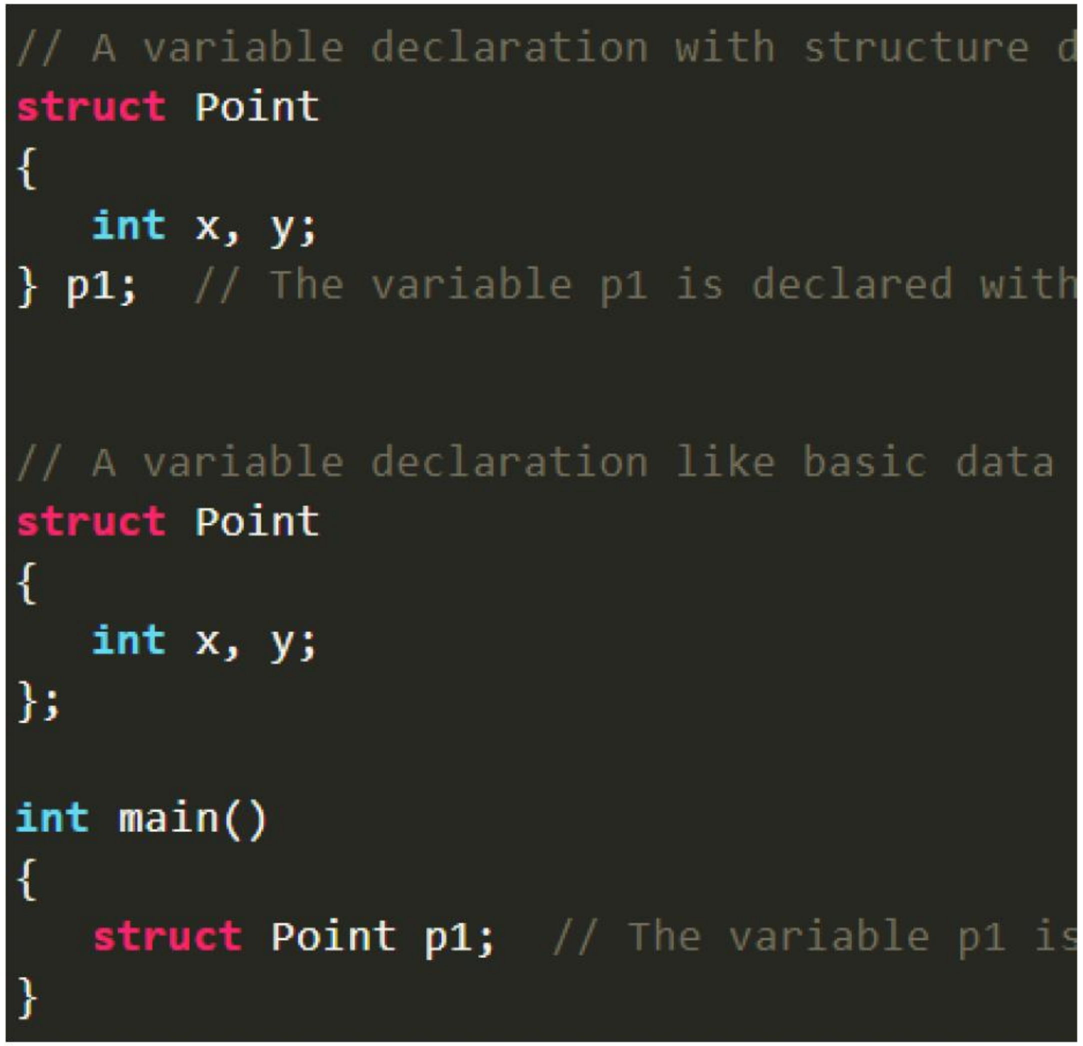
Example



```
// Data Members
int roll;
int age;
int marks;

// Member Functions
void printDetails()
{
    cout<<"Roll = "<<roll<<"\n";
    cout<<"Age = "<<age<<"\n";
    cout<<"Marks = "<<marks;
}
```

How to declare structure variables?

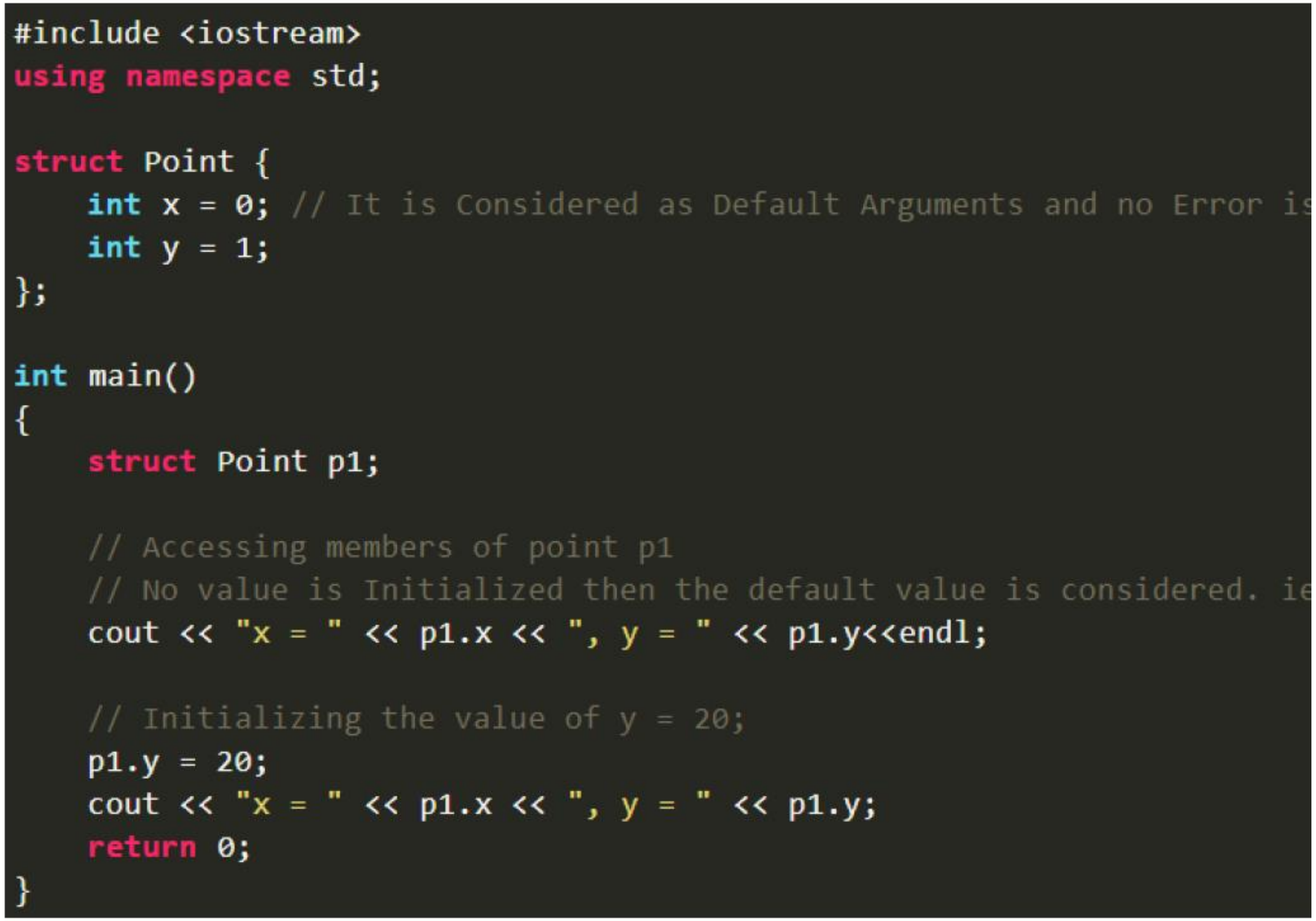
A structure variable can either be declared with structure declaration or as a separate declaration like basic types.



```
// A variable declaration with structure d
struct Point
{
   int x, y;
} p1;  // The variable p1 is declared with


// A variable declaration like basic data
struct Point
{
   int x, y;
};

int main()
{
   struct Point p1;  // The variable p1 is
}
```

How to initialize structure members?

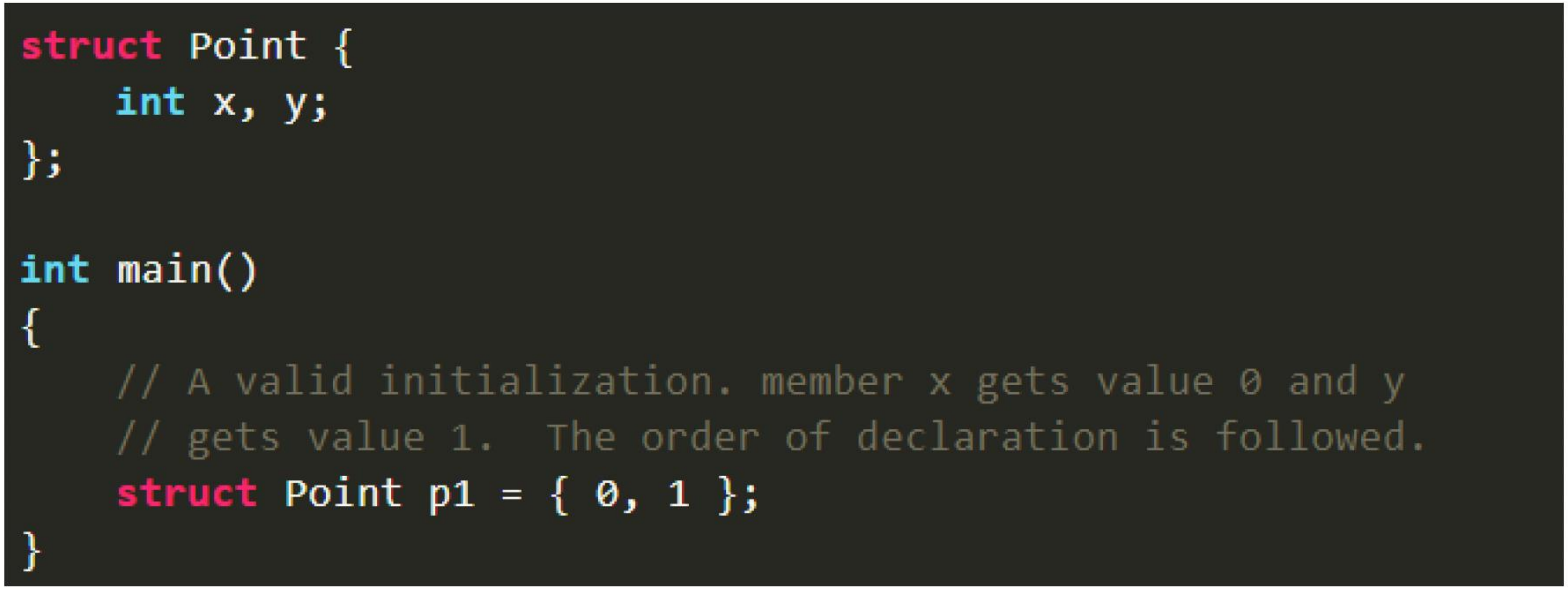
- Structure members **cannot be** initialized with declaration in C.
- But it is considered correct in C++11 and above.



```
#include <iostream>
using namespace std;

struct Point {
    int x = 0; // It is Considered as Default Arguments and no Error is
    int y = 1;
};

int main()
{
    struct Point p1;

    // Accessing members of point p1
    // No value is Initialized then the default value is considered. ie
    cout << "x = " << p1.x << ", y = " << p1.y<<endl;

    // Initializing the value of y = 20;
    p1.y = 20;
    cout << "x = " << p1.x << ", y = " << p1.y;
    return 0;
}
```

How to initialize structure members?

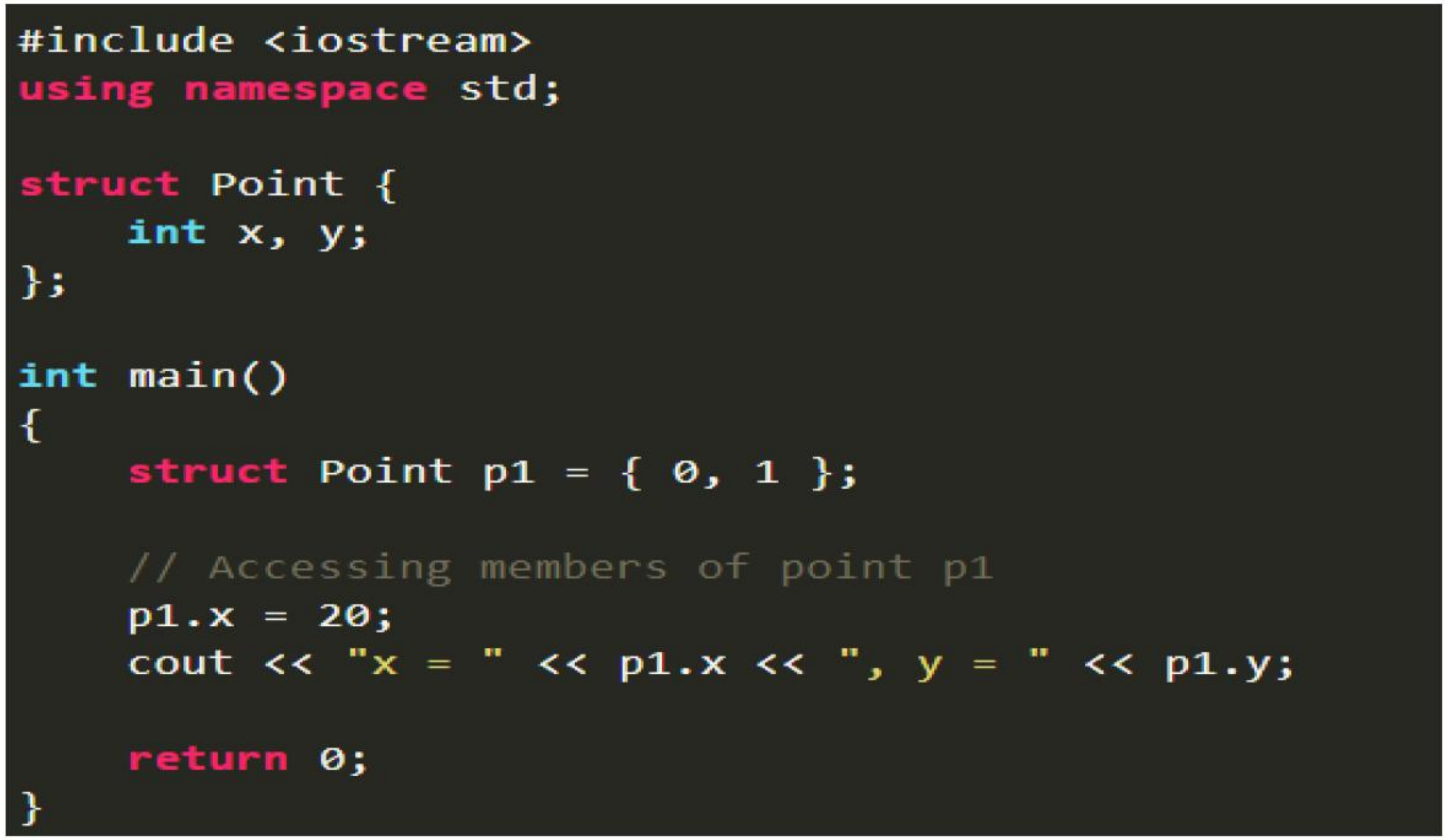
Structure members can be initialized using curly braces ‘{}’. For example, following is a valid initialization.



```
struct Point {
    int x, y;
};

int main()
{
    // A valid initialization. member x gets value 0 and y
    // gets value 1.  The order of declaration is followed.
    struct Point p1 = { 0, 1 };
}
```

How to access structure elements?

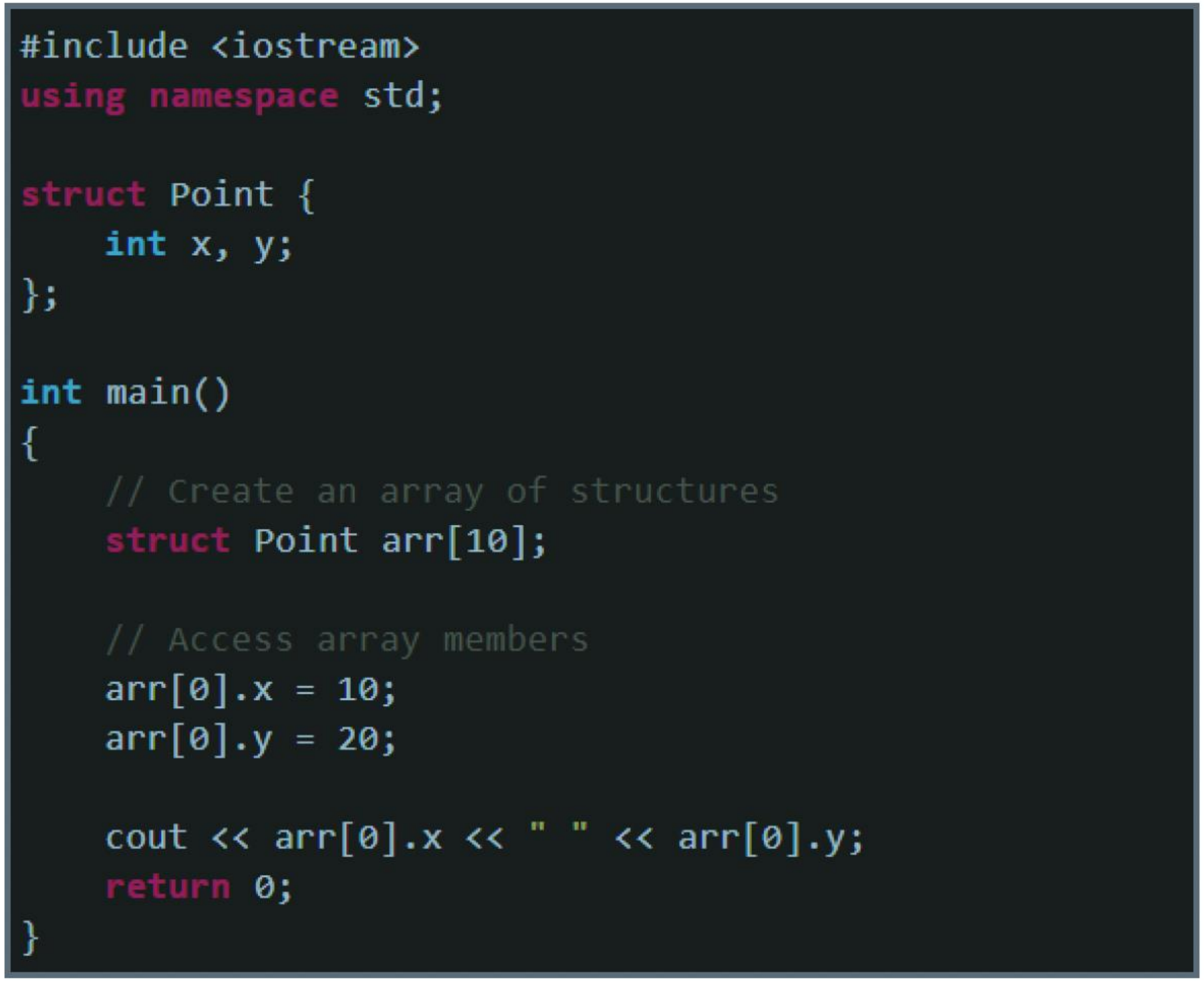
- Structure members are accessed using dot (.) operator.



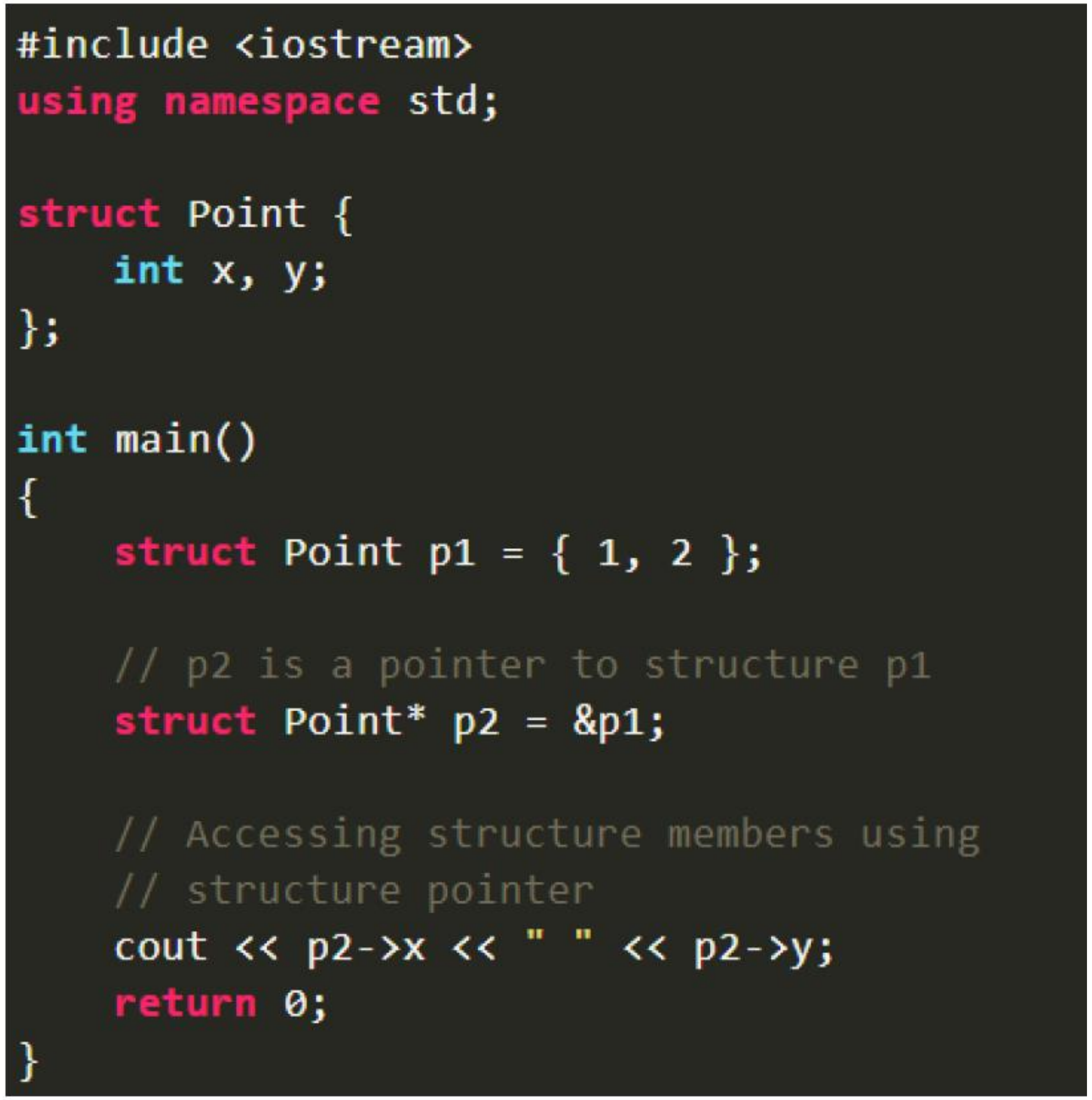
```
#include <iostream>
using namespace std;

struct Point {
    int x, y;
};

int main()
{
    struct Point p1 = { 0, 1 };

    // Accessing members of point p1
    p1.x = 20;
    cout << "x = " << p1.x << ", y = " << p1.y;

    return 0;
}
```

Array of Structures

- Like other primitive data types, we can create an array of structures.



```
#include <iostream>
using namespace std;

struct Point {
    int x, y;
};

int main()
{
    // Create an array of structures
    struct Point arr[10];

    // Access array members
    arr[0].x = 10;
    arr[0].y = 20;

    cout << arr[0].x << " " << arr[0].y;
    return 0;
}
```

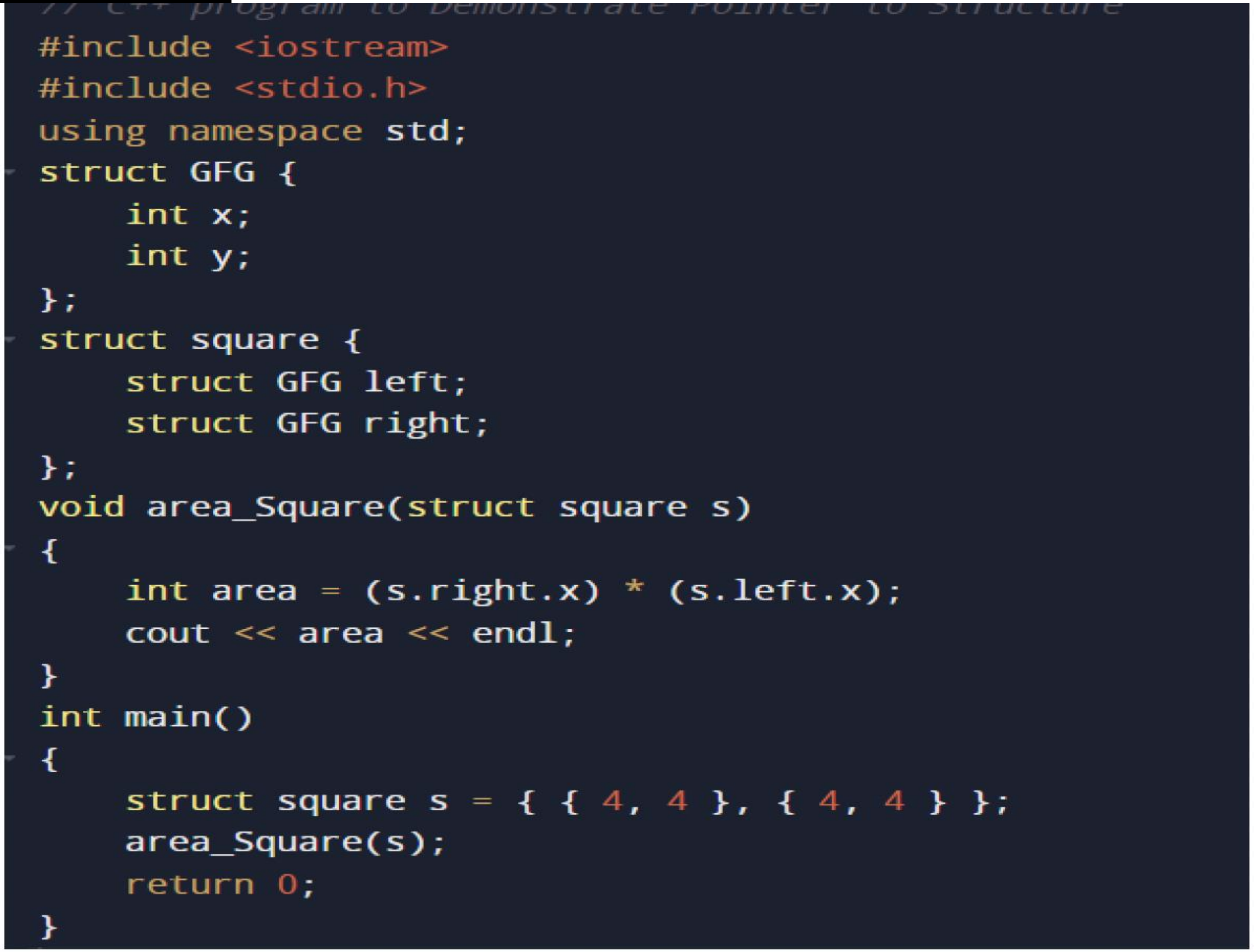


```cpp
#include <iostream>
using namespace std;

struct Point {
    int x, y;
};

int main()
{
    struct Point p1 = { 1, 2 };

    // p2 is a pointer to structure p1
    struct Point* p2 = &p1;

    // Accessing structure members using
    // structure pointer
    cout << p2->x << " " << p2->y;
    return 0;
}
```

What is a structure pointer?

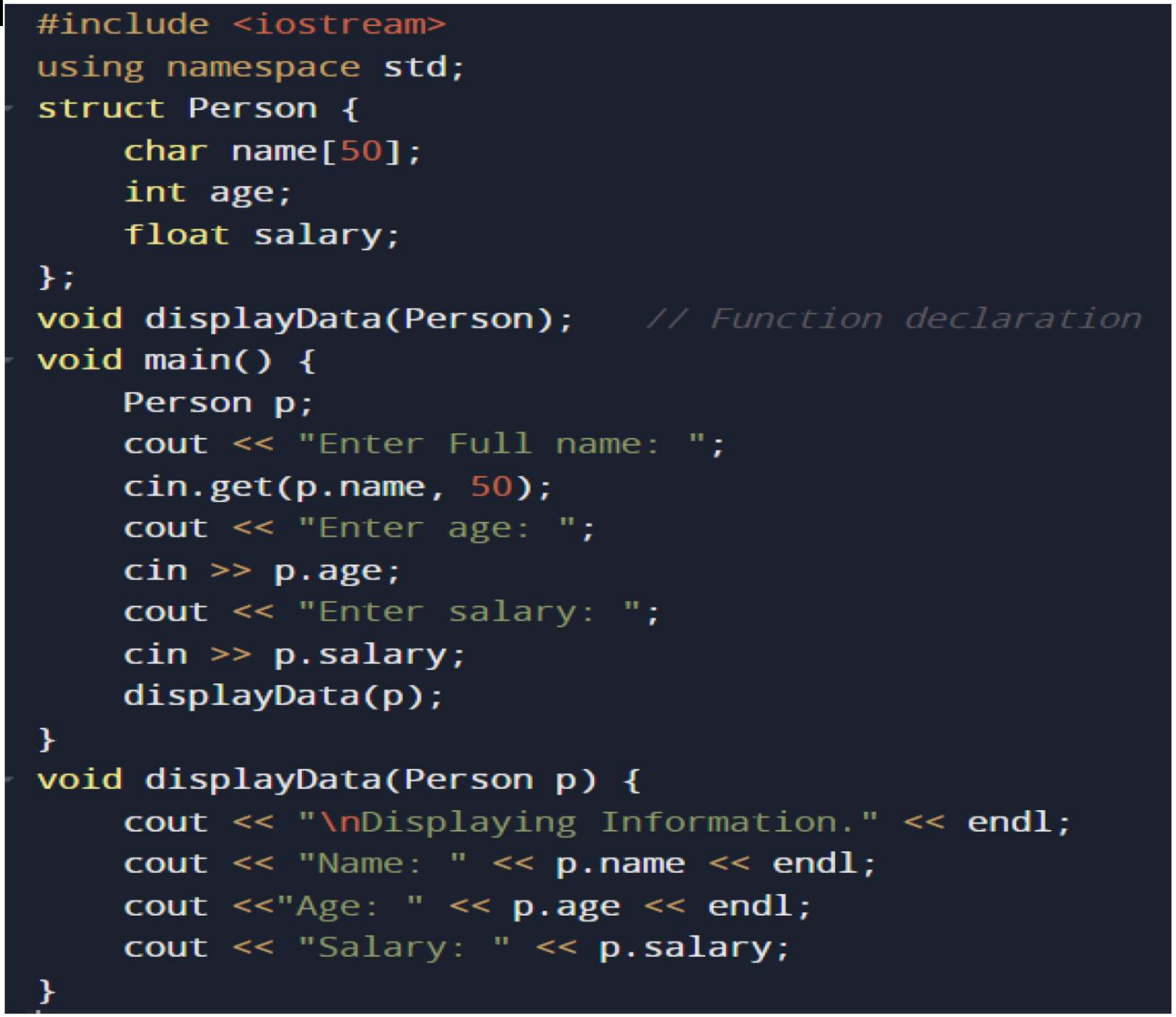
- Like primitive types, we can have pointer to a structure. If we have a pointer to structure, members are accessed using arrow (->) operator instead of the dot (.) operator.
- (*p).x is another way to access similar to p->x
- Pointer to structure in C++ can also be referred to as Structure Pointer. A **structure Pointer** in C++is defined as the pointer which points to the address of the memory block that stores a structure.

Nested Structures



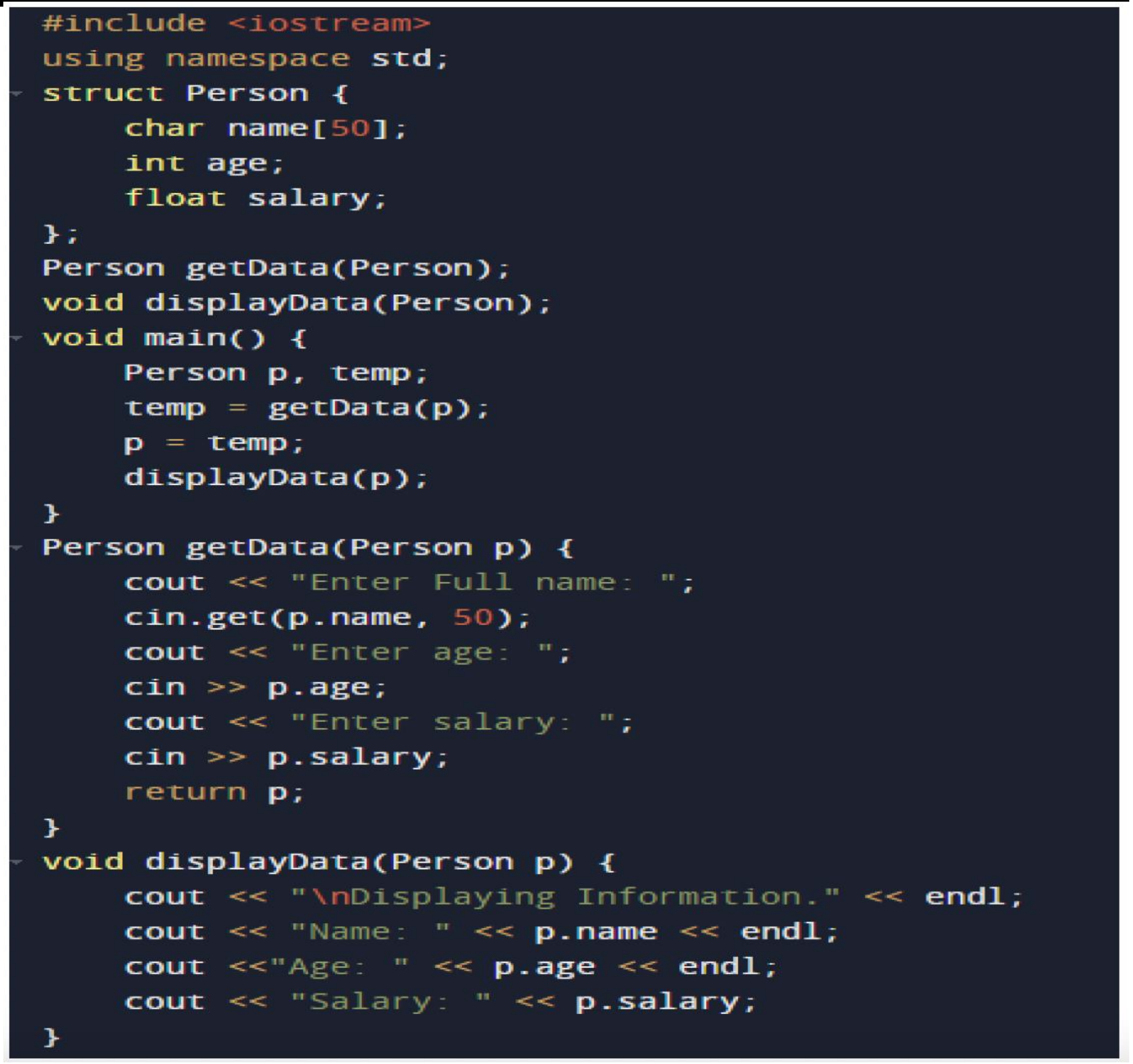
```cpp
// C++ program to Demonstrate Pointer to Structure
#include <iostream>
#include <stdio.h>
using namespace std;
struct GFG {
    int x;
    int y;
};
struct square {
    struct GFG left;
    struct GFG right;
};
void area_Square(struct square s)
{
    int area = (s.right.x) * (s.left.x);
    cout << area << endl;
}
int main()
{
    struct square s = { { 4, 4 }, { 4, 4 } };
    area_Square(s);
    return 0;
}
```

Passing Structures as Arguments to Functions



```cpp
#include <iostream>
using namespace std;
struct Person {
    char name[50];
    int age;
    float salary;
};
void displayData(Person);   // Function declaration
void main() {
    Person p;
    cout << "Enter Full name: ";
    cin.get(p.name, 50);
    cout << "Enter age: ";
    cin >> p.age;
    cout << "Enter salary: ";
    cin >> p.salary;
    displayData(p);
}
void displayData(Person p) {
    cout << "\nDisplaying Information." << endl;
    cout << "Name: " << p.name << endl;
    cout <<"Age: " << p.age << endl;
    cout << "Salary: " << p.salary;
}
```

Returning a structure from function



```
#include <iostream>
using namespace std;
struct Person {
    char name[50];
    int age;
    float salary;
};
Person getData(Person);
void displayData(Person);
void main() {
    Person p, temp;
    temp = getData(p);
    p = temp;
    displayData(p);
}
Person getData(Person p) {
    cout << "Enter Full name: ";
    cin.get(p.name, 50);
    cout << "Enter age: ";
    cin >> p.age;
    cout << "Enter salary: ";
    cin >> p.salary;
    return p;
}
void displayData(Person p) {
    cout << "\nDisplaying Information." << endl;
    cout << "Name: " << p.name << endl;
    cout <<"Age: " << p.age << endl;
    cout << "Salary: " << p.salary;
}
```

Acknowledgment

- Content of these slides are taken from:
  - https://www.geeksforgeeks.org/
  - https://www.tutorialspoint.com/
  - https://www.programiz.com/
  - https://www.w3schools.com/