CodInClub

Powered By BridgeLabz

Wk5 /S5/ Lecture # : DSOOPS-24

Topics Covered

- What is a Structure?
- Defining and Using Structures
- What is a Class?
- Basics of Class Syntax and Access Specifiers
- Differences Between Struct and Class
- Creating and Using Objects
- Creating Objects: Stack vs. Heap

What is a Structure?

A structure (`struct`) lets you group different variables together under one name—like a custom datatype.

Example:

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

int main() {
    Point p;          // Create a Point (object)
    p.x = 3;
    p.y = 7;
    std::cout << p.x << ", " << p.y << std::endl; // Output: 3, 7
    return 0;
}
```

Defining and Using Structures

- Use the `struct` keyword.

- Members are public by default (can be accessed directly).
- Access members with a dot (.).

Example:

```
struct Student {
    int id;
    double marks;
};

int main() {
    Student s1;
    s1.id = 101;
    s1.marks = 93.5;
    std::cout << s1.id << " " << s1.marks << std::endl;
    return 0;
}
```

What is a Class?

A class is like a struct but with private members by default. It's a blueprint for objects, and is the basis for object-oriented programming.

Example:

```
class Rectangle {
public:
    int width;
    int height;
};

int main() {
    Rectangle box;
    box.width = 10;
    box.height = 5;
    std::cout << box.width * box.height << std::endl; // Output:
50
```

```
    return 0;
}
```

- `public:` allows outside code to access members.
- By default, class members are private.

Differences Between Struct and Class

Aspect	**struct**	**class**
Default Access	public	private
Syntax	`struct Name {...};`	`class Name {...};`
Usage	Group data	Data + control/hiding

Creating and Using Objects

An object is a variable created from a struct or class.
Use the type name and the object name:

```
Student s1;        // Object of struct Student
Rectangle box;     // Object of class Rectangle

s1.id = 7;
std::cout << s1.id << std::endl;
```

Creating Objects: Stack vs. Heap

Objects can be created in two major places: the stack or the heap.

Stack Allocation (Automatic Storage)

- Object is created with a normal declaration, e.g. `Student s1;`.
- The object is *automatically* created when the function starts and destroyed when it ends.
- Fast, safe, and easiest to use.

Example:

```
void fun() {
    Point p;  // p lives on the stack (auto)
    p.x = 5;
}
```

- No need to manage memory yourself!

Heap Allocation (Dynamic Storage)

- Object is created with `new`.
- Stays until you delete it yourself with `delete`.
- Useful for larger objects or objects that need to "outlive" a function.

Example:

```
Point* p = new Point;   // p points to a new object on the heap
p->x = 10;              // Use arrow (->) to access
p->y = 5;
delete p;               // Must free memory!
```

Key points:

- Use `*` pointer, `->` member access, and always `delete` when finished.
- Heap objects let you decide when the object is destroyed.

Comparing Stack and Heap Objects

Place	How to Create	How to Use	Who Destroys	When Destroyed
Stack	`Point p;`	`p.x`, `p.y`	C++	End of block/func
Heap	`Point* p = new Point;`	`p->x`, `p->y`	You	When you `delete`

Tip: For most programs, prefer stack allocation for simple/small objects. Use heap (`new`/`delete`) only if you really need dynamic control.

Practice Problems and Activities

Easy 1

Define a struct called Book with two members: `int pages;` and `double price;`. Create a variable of type Book (on the stack), assign values, and print them.

Easy 2

Define a class named Circle with a public member `double radius;`. Create an object (on the heap) and set its radius to 4.2. Print the radius, then delete the object.

Medium

Given this struct:

```
struct Date {
```

```
    int day;
    int month;
    int year;
};
```

Write a main function that creates a Date object for 15th August 1947 and prints “15-8-1947”. Then, create a Date object on the heap, set it to 26-1-1950, print it, and free memory.

Hard

Explain the output of the code: Why is there an error if you try to access `secret` directly? Fix the code to allow printing `secret`. Also demonstrate creating the object both on the stack and heap.

```
class Box {
    int secret;
public:
    int visible;
};

int main() {
    Box b;
    b.visible = 10;
    b.secret = 42;      // Error!
    std::cout << b.visible << std::endl;
    // Print secret here (after fixing)
    return 0;
}
```

Hint: What’s the default access for class members? How can you let code outside the class access private members?

Wrap-Up & Key Takeaways

- Structures and classes let you group and control variables.
- Objects can be created on the stack (simple, safe, automatic) or on the heap (manual lifetime control, must remember to free).
- Prefer stack for most cases; use heap only when you need objects to live longer than a function or to create many at runtime.
- Understanding the difference between stack and heap is essential for writing bug-free, efficient C++ code.