Abstraction

Abstraction-

Abstraction means providing only some of the information to the user by hiding its internal implementation details. We just need to know about the methods of the objects that we need to call and the input parameters needed to trigger a specific operation, excluding the details of implementation and type of action performed to get the result.

Abstraction is selecting data from a larger pool to show only relevant details of the object to the user. It helps in reducing programming complexity and efforts. It is one of the most important concepts of OOPs.

Real-life example: When you send an email to someone, you just click send, and you get the success message; what happens when you click send, how data is transmitted over the network to the recipient is hidden from you (because it is irrelevant to you).

We can implement Abstraction in C++ using classes. The class helps us to group data members and member functions using available access specifiers. A Class can decide which data members will be visible to the outside world and not. Access specifiers are the main pillar of implementing abstraction in C++. We can use access specifiers to enforce restrictions on class members.

Example:

```
#include <iostream>
using namespace std;
class abstraction {
    private:
        int a, b;
    public:
        // method to set values of private members
        void set(int x, int y) {
            a = x;
            b = y;
        }
    void display() {
        cout << "a = " << a << endl;
        cout << "b = " << b << endl;
```

```
    }
};
int main() {
    implementAbstraction obj;
    obj.set(10, 20);
    obj.display();
    return 0;
}
Output:
a = 10
b = 20
```

Advantages Of Abstraction

- Only you can make changes to your data or function, and no one else can.
- It makes the application secure by not allowing anyone else to see the background details.
- Increases the reusability of the code.
- Avoids duplication of your code.