Abstraction

Abstraction means displaying only essential information and hiding the details.

Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.

We can implement Abstraction in C++ **using classes**. Class helps us to group data members and member functions using available access specifiers. A Class can decide which data member will be visible to outside world and which is not.

Access specifiers are the main pillar of implementing abstraction in C++. We can use access specifiers to enforce restrictions on class members. For example:

- Members declared as **public** in a class, can be accessed from anywhere in the program.
- Members declared as **private** in a class, can be accessed only from within the class. They are not allowed to be accessed from any part of code outside the class.

We can easily implement abstraction using the above two features provided by access specifiers. Say, the members that defines the internal implementation can be marked as private in a class. And the important information needed to be given to the outside world can be marked as public. And these public members can access the private members as they are inside the class.

Example:

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

class implementAbstraction
{
    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
```

You can see in the above program we are not allowed to access the variables a and b directly, however one can call the function set() to set the values in a and b and the function display() to display the values of a and b.

Advantages of Data Abstraction:

- Helps the user to avoid writing the low level code
- Avoids code duplication and increases reusability.
- Can change internal implementation of class independently without affecting the user.
- Helps to increase security of an application or program as only important details are provided to the user.

<u>Abstraction separates code into interface and implementation. So while designing your component, you must keep interface independent of the implementation so that if you change underlying implementation then interface would remain intact.</u>