

Abstraction in C++ with example

Abstraction is one of the feature of Object Oriented Programming, where you show only relevant details to the user and hide irrelevant details. For example, when you send an email to someone you just click send and you get the success message, what actually happens when you click send, how data is transmitted over network to the recipient is hidden from you (because it is irrelevant to you).

Let's see how this can be achieved in a C++ program using access specifiers:

Abstraction Example

```
#include <iostream>
using namespace std;
class AbstractionExample{
private:
    /* By making these data members private, I have
    * hidden them from outside world.
    * These data members are not accessible outside
    * the class. The only way to set and get their
    * values is through the public functions.
    */
    int num;
    char ch;

public:
    void setMyValues(int n, char c) {
        num = n; ch = c;
    }
    void getMyValues() {
        cout<<"Numbers is: "<<num<< endl;
        cout<<"Char is: "<<ch<<endl;
    }
};
int main(){
    AbstractionExample obj;
    obj.setMyValues(100, 'X');
    obj.getMyValues();
    return 0;
}
```

Output:

```
Numbers is: 100
Char is: X
```

Advantage of data abstraction

The major advantage of using this feature is that when the code evolves and you need to make some adjustments in the code then you only need to modify the high level class where you have declared the members as private. Since none class is accessing these data members directly, you do not need to change the low level(user level) class code. Imagine if you had made these data members public, if at some point you want to change

the code, you would have to make the necessary adjustments to all the classes that are accessing the members directly.

Other advantages of data abstraction are:

- 1) Makes the application secure by making data private and avoiding the user level error that may corrupt the data.
- 2) This avoids code duplication and increases the code reusability.