Data Abstraction in C++

Data abstraction refers to providing only essential information to the outside world and hiding their background details, i.e., to represent the needed information in program without presenting the details.

Data abstraction is a programming (and design) technique that relies on the separation of interface and implementation.

Let's take a real life example of AC, which can be turned ON or OFF, change the temperature, change the mode, and other external components such as fan, swing. But, we don't know the internal details of the AC, i.e., how it works internally. Thus, we can say that AC seperates the implementation details from the external interface.

In C++ program if we implement class with private and public members then it is an example of data abstraction.

Data Abstraction can be achieved in two ways:

Abstraction using Classes: 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.

Abstraction in Header files: One more type of abstraction in C++ can be header files. For example, consider the pow() method present in math.h header file. Whenever we need to calculate power of a number, we simply call the function pow() present in the math.h header file and pass the numbers as arguments without knowing the underlying algorithm according to which the function is actually calculating power of numbers.

Access Specifiers Implement Abstraction:

- **Public specifier:** When the members are declared as public, members can be accessed anywhere from the program.
- **Private specifier:** When the members are declared as private, members can only be accessed only by the member functions of the class.

Advantages Of Abstraction

1. Only you can make changes to your data or function and no one else can.
2. Makes the application secure by not allowing anyone else to see the background details.
3. Increases reusability of the code.
4. Avoids duplication of your code.

Let's see a simple example of abstraction in header files.

main.cpp

```cpp
//  program to calculate the power of a number
 #include <iostream>
#include<math.h>
using namespace std;
int main()
{
   int n = 4;
   int power = 3;
   int result = pow(n,power);         // pow(n,power) is the  power function
   std::cout << "Cube of n is : " <<result<< std::endl;
   return 0;
}
```

input

```
Cube of n is : 64
```

In the above example, pow() function is used to calculate 4 raised to the power 3. The pow() function is present in the math.h header file in which all the implementation details of the pow() function is hidden

Let's see a simple example of data abstraction using classes.

main.cpp (Ctrl+M)

```cpp
//  data abstraction using classes
#include <iostream>
using namespace std;
class Sum
{
    private: int x, y, z; // private variables
    public:
    void add()
    {
        cout<<"Enter two numbers: ";
        cin>>x>>y;
        z= x+y;
        cout<<"Sum of two number is: "<<z<<endl;
    }
};
int main()
{
    Sum sm;
    sm.add();
    return 0;
}
```

In the above example, abstraction is achieved using classes. A class 'Sum' contains the private members x, y and z are only accessible by the member functions of the class.