Abstraction

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.

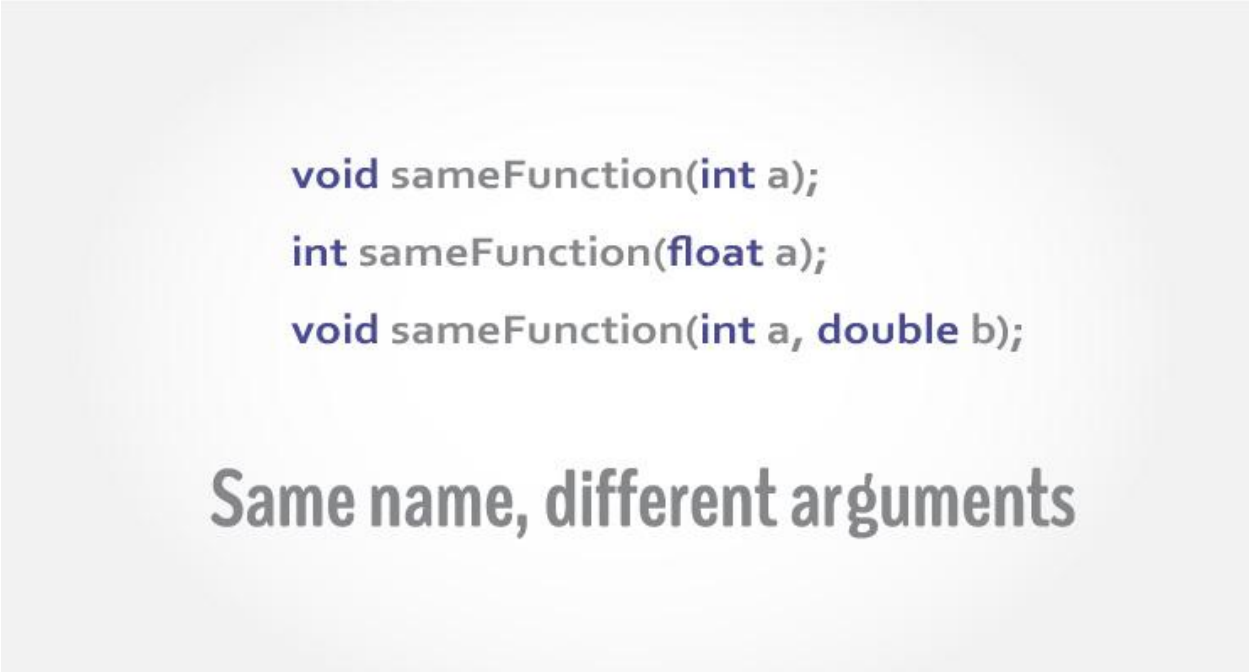
Data abstraction is a programming (and design) technique that relies on the separation of interface and implementation.

Let's take one real life example of a TV, which you can turn on and off, change the channel, adjust the volume, and add external components such as speakers, VCRs, and DVD players, BUT you do not know its internal details, that is, you do not know how it receives signals over the air or through a cable, how it translates them, and finally displays them on the screen.

```
/* C++ Data Abstraction - Example Program */

#include<iostream>
using namespace std;
class ADD
{
        public:
                ADD(int i=0)
                {
                        tot=i;
                }
                void addnumber(int num)
                {
                        tot=tot+num;
                }
                int gettotal()
                {
                        return tot;
                }
        private:
                // hidden from outside the world
                int tot;
};
int main()
```

```
{

        ADD aob;
        int a, b, c;
        cout<<"Enter any three numbers: ";
        cin>>a>>b>>c;
        aob.addnumber(a);
        aob.addnumber(b);
        aob.addnumber(c);
        cout<<"Total = "<<aob.gettotal();

}
```

Function Overloading



When a several function declarations are specified for a single function name in the same scope, the (function) name is said to be overloaded. C++ allows functions to have the same name if it can distinguish them by their number and type of arguments. For example, following four functions are different in C++

```
float divide(int a, int b);
float divide(int a, int b, int c);
float divide(float x, float y);
float divide(float x, float y, float z);
```

That is, divide() taking two int type arguments is different from divide() taking three int type arguments is different from divide() taking two float type arguments is different from divide() taking three float type arguments.

This is known as function overloading.

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

void display(int);
void display(float);
void display(int, float);

int main() {

   int a = 5;
   float b = 5.5;

   display(a);
   display(b);
   display(a, b);

   return 0;
}

void display(int var) {
   cout << "Integer number: " << var << endl;
}

void display(float var) {
   cout << "Float number: " << var << endl;
}

void display(int var1, float var2) {
   cout << "Integer number: " << var1;
   cout << " and float number:" << var2;
}
```

Encapsulation in C++

Encapsulation is a process of wrapping of data and methods in a single unit. It is achieved in C++ language by class concept.

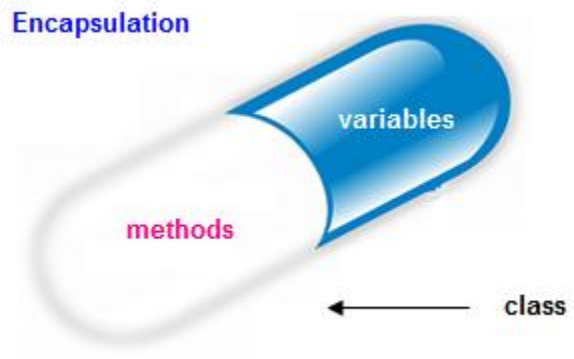
Combining of **state** and **behavior** in a single container is known as encapsulation. In C++ language encapsulation can be achieve using **class** keyword, state represents declaration of variables on attributes and behavior represents operations in terms of method.

Advantage of Encapsulation

The main advantage of using of encapsulation is to secure the data from other methods, when we make a data private then these data only use within the class, but these data not accessible outside the class.

Real Life Example of Encapsulation in C++

The common example of encapsulation is **Capsule**. In capsule all medicine are encapsulated in side capsule.



Benefits of encapsulation

- Provides abstraction between an object and its clients.
- Protects an object from unwanted access by clients.

- Example: A bank application forbids a client to change an Account's balance.

Example of Encapsulation in C++

```
#include<iostream.h>
#include<conio.h>

class sum
{
private: int a,b,c;

public:
void add()
{
clrscr();
cout<<"Enter any two numbers: ";
cin>>a>>b;
c=a+b;
cout<<"Sum: "<<c;
}
};
void main()
{
sum s;
s.add();
getch();
}
```

Output

```
Enter any two number:

4

5

Sum: 9
```

In above example all data and function are bind inside class sum.