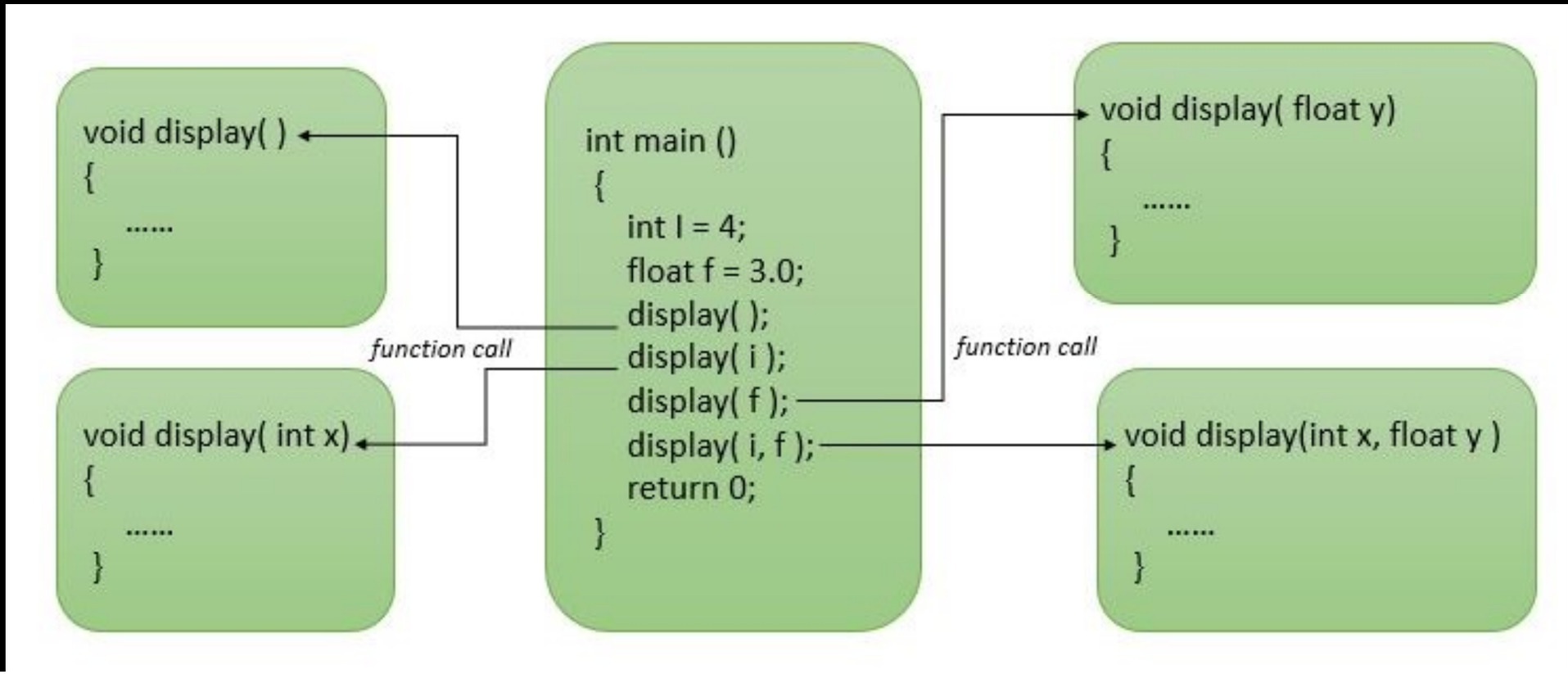
C++ POLYMORPHISM

C++ Function Overloading

- As we know that functions are the piece of code that can be used anywhere in the program with just calling it multiple times to reduce the complexity of the code.
- In POP, we can use as many functions as per need, however, the names of the function shouldn’t match.
- In the case of OOP, we can use a function name as many times following the condition that the number of arguments or the type of arguments must differ.
- So the method of using the same function name for different functions is simply called **function overloading**.
- When an overloaded function is called from main program the function with matching arguments is invoked.

Examples

```
void display();              //function with no arguments
void display( int );         //function with one integer type arguments
void display( float );       //function with one floating point arguments
void display( int, float );  //function with one floating and one integer type arguments
```



```
#include <iostream.h>
  void display ( )
{
  int a = 3;
  cout << a << endl;
}
void display (int a )
{
  cout << a << endl;
}
void display (double a )
{
   cout << a << endl;
}
void display(int a, float b)
{
   cout<< a << " , " << b << endl;
}
```

```
int main()
{
  display();
  display(5);
  display(2.3);
  display(5,4.0);
   getch();
 }
```

C++ Function Overriding

- As we know, inheritance is a feature of OOP that allows us to create derived classes from a base class.
- The derived classes inherit features of the base class.
- Suppose, the same function is defined in both the derived class and the based class.
- Now if we call this function using the object of the derived class, the function of the derived class is executed.
- This is known as **function overriding** in C++.
- The function in derived class overrides the function in base class.

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

class Base {
   public:
    void print() {
        cout << "Base Function" << endl;
    }
};

class Derived : public Base {
   public:
    void print() {
        cout << "Derived Function" << endl;
    }
};
```

```cpp
int main() {
    Derived derived1;
    derived1.print();
    return 0;
}
```

Output

```
Derived Function
```

```
class Base {
    public:
      void print() {
          // code
      }
};

class Derived : public Base {
    public:
      void print() {
          // code
      }
};

int main() {
    Derived derived1;
    derived1.print();
    return 0;
}
```