**Polymorphism**

The word “polymorphism” means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. A real-life example of polymorphism is a person who at the same time can have different characteristics. A man at the same time is a father, a husband, and an employee. So the same person exhibits different behavior in different situations. This is called polymorphism. Polymorphism is considered one of the important features of Object-Oriented Programming.

**Types of Polymorphism**

* **Compile-time Polymorphism**
* **Runtime Polymorphism**

**1. Compile-time Polymorphism**

This type of polymorphism is achieved by function overloading

**Function Overloading**

When there are multiple functions with the same name but different parameters, then the functions are said to be **overloaded,**hence this is known as Function Overloading. Functions can be overloaded by **changing the number of arguments** or/and **changing the type of arguments**.

And, depending upon the number/type of arguments, different functions are called. For example

// C++ program to overload sum() function

#include <iostream>

using namespace std;

// Function with 2 int parameters

int sum(int num1, int num2) {

return num1 + num2;

}

// Function with 2 double parameters

double sum(double num1, double num2) {

return num1 + num2;

}

// Function with 3 int parameters

int sum(int num1, int num2, int num3) {

return num1 + num2 + num3;

}

int main() {

// Call function with 2 int parameters

cout << "Sum 1 = " << sum(5, 6) << endl;

// Call function with 2 double parameters

cout << "Sum 2 = " << sum(5.5, 6.6) << endl;

// Call function with 3 int parameters

cout << "Sum 3 = " << sum(5, 6, 7) << endl;

return 0;

}

**2. Runtime Polymorphism**

This type of polymorphism is achieved by Function Overriding.

In [C++ inheritance](https://www.programiz.com/cpp-programming/inheritance), we can have the same function in the base class as well as its derived classes.

When we call the function using an object of the derived class, the function of the derived class is executed instead of the one in the base class.

So, different functions are executed depending on the object calling the function.

This is known as **function overriding.**

**Example :**

#include <iostream>

using namespace std;

class Animal {

public:

void eat(){

cout<<"Eating...";

}

};

class Dog: public Animal

{

public:

void eat()

{

cout<<"Eating bread...";

}

};

int main(void) {

Dog d = Dog();

d.eat();

return 0;

}

**o/p**

*Eating bread...*