Course Name: A Level (2nd Sem) **Subject:** Data Struture using C++
Topic: Function Overloading **Date:** 26-03-2020

Function overloading

Function overloading means two or more functions can have the same name, but either the number of arguments or the data type of arguments has to be different. In the first example, we create two functions of the same name, one for adding two integers and another for adding two floats. In the second program, we make two functions with identical names but pass them a different number of arguments. Function overloading is also known as compile-time polymorphism.

Function overloading C++ program

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

/* Function arguments are of different data type */
int add(int, int);
float add(float, float);

int main()
{
  int a, b, x;
  float c, d, y;

  cout << "Enter two integers\n";
  cin >> a >> b;
  x = add(a, b);
  cout << "Sum of integers: " << x << endl;
  cout << "Enter two floating point numbers\n";
  cin >> c >> d;
  y = add(c, d);
  cout << "Sum of floats: " << y << endl;
  return 0;
}

int add(int x, int y)
{
  int sum;
  sum = x + y;
  return sum;
}

float add(float x, float y)
{
  float sum;
  sum = x + y;
  return sum;
}
```

We created two functions "add" for integer and float data types, you can create any number of them of the same name as required. Just make sure a compiler will be able to determine which one to call. In the program, you can create add function for long, double, and other data types.

In the functions, the code is the same but data types are different, C++ provides a solution to this problem. We can create one function for different data types which will reduce the size of code by using a feature of C++ known as templates.

Program for function overloading

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

/* Number of arguments are different */

void display(char []);  // print the string passed as argument
void display(char [], char []);

int main()
{
  char first[] = "C programming";
  char second[] = "C++ programming";
  display(first);
  display(first, second);
  return 0;
}

void display(char s[])
{
  cout << s << endl;
}

void display(char s[], char t[])
{
  cout << s << endl << t << endl;
}
```

Output of program:

```
C programming
C programming
C++ programming
```

A function return type has no role because it returns a value when called and at compile time compiler will not be able to decide which one to call.