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. Return type has no role because the function will return a value only when it is called.

1) First example

In this example, we make two functions one for adding two integers and other for adding two floats, but they have the same name.

```
#include <stdio.h>
/* Two functions with same name but different data type in input
arguments*/
int add(int, int);
float add(float, float);

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

   printf("Enter two integers\n");
   scanf("%d",&a);
   scanf("%d",&b);
   x = add(a, b); //Function for integers
   printf("The sum of integers is %d\n",x);

   printf ("\nEnter two floating point numbers\n");
   scanf("%f",&c);
   scanf("%f",&d);
   y = add(c, d); //Function for floats
   printf("The sum of floats is %f\n",y);

   return 0;
}

int add(int x, int y) //Function for integers
{
   int sum;
   sum = x + y;
   return sum;
}

float add(float x, float y) //Function for floats
{
   float sum;
   sum = x + y;
   return sum;
}
```

2) Second example

In this example, we make two functions with same name but differing in only the number of input arguments.

```
#include <stdio.h>

/* Number of arguments are different */

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

void display(char [], char []);


int main()

{

   char first[] = "C programming";

   char second[] = "C++ programming";

   printf("Calling function 1...\n");

   display(first);

   printf("\nCalling function 2...\n");

   display(first, second);


   return 0;

}


void display(char s[])

{

   printf("%s\n",s);

}


void display(char s[], char t[])

{

   printf("%s\n%s",s,t);

}
```