

Department of Computer Science(BS 6th induction)

Lab Manual(02)

Roll #:24017119-003

subject
code:cs-
103- s25-E



I

<i>Subject Name:</i>	“Object Oriented Programming”
<i>Submitted by:</i>	“Alishba Yaqoob”
<i>Submitted to:</i>	“DR.Usman Ali”
<i>Submission Date:</i>	“8,March,2025”

///

1. Write a Brief explanation of function overloading.

Answer:

Function overloading is a feature in C++ that allows multiple functions to have the same name but different parameter lists. The compiler differentiates these functions based on the number, type, or order of parameters. These enable code reusability and improve readability by allowing to handle various types of input by using a single function name. Function overloading is a form of polymorphism that helps in writing more flexible and maintainable code. However, return types alone cannot be used to distinguish overloaded functions.

Function overloading allows the programmer to assign the same name to all functions with similar tasks. It helps the user to use different functions easily. The user does not need to remember many names for similar types of tasks.

For example, the program can define the following function to calculate average:

Float Average (int, int)

Float Average (float, float)

Float Average (int, float)

2. include the code and output for each example.

Answer:

Example# 01

Overloaded Greeting Function.

Objective:

Create overloaded function to display greetings with and without a name.

```
#include<iostream>
```

```
using namespace std;
```

```
void greeting()
```

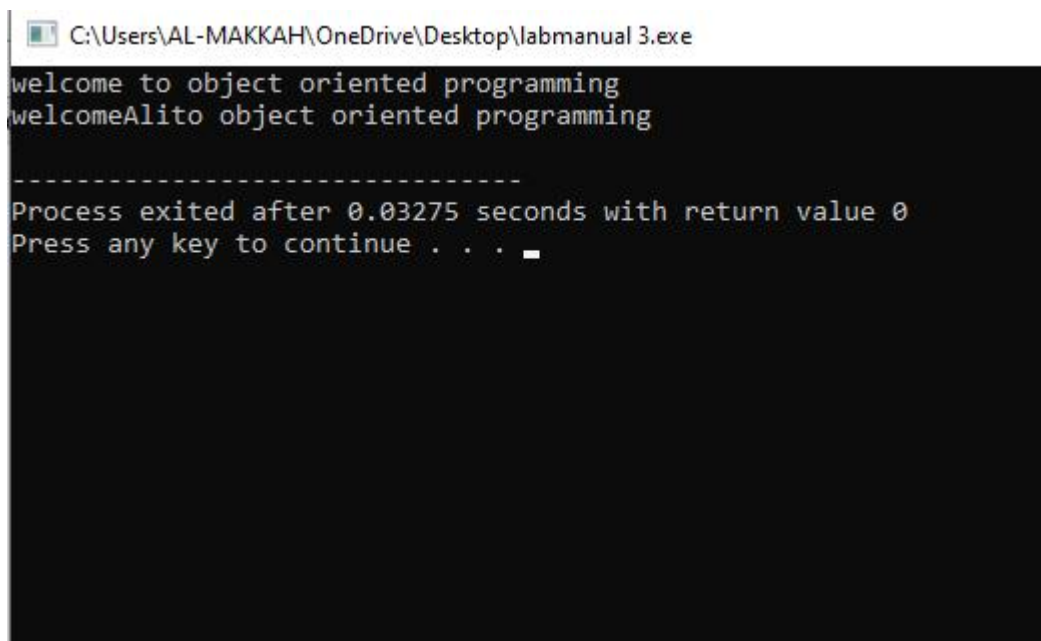
```
{
```

```

        cout<<"welcome to object oriented programming"<<endl;
};
void greetings(string name)
{
    cout<<"welcome"<<name<<"to          object          oriented
programming"<<endl;
};
int main ()
{
    greeting ();
    greetings("Ali");
    return 0;
}

```

Output:



```

C:\Users\AL-MAKKAH\OneDrive\Desktop\labmanual 3.exe
welcome to object oriented programming
welcomeAlito object oriented programming
-----
Process exited after 0.03275 seconds with return value 0
Press any key to continue . . .

```

Example# 03

Overloaded add function.

Objective:

Create overloaded functions to added two or three integer.

```
#include<iostream>
```

```
using namespace std;
```

```
int add(int a, int b)
```

```
{
```

```
    return a+b;
```

```
}
```

```
int add(int a, int b, int c)
```

```
{
```

```
    return a + b + c;
```

```
}
```

```
int main()
```

```
{
```

```
    cout<<"Result:" << add (10, 20)<<endl;
```

```
    cout<<"Result:" << add (10, 20, 30)<<endl;
```

```
    return 0;
```

```
}
```

Out put:

C:\Users\AL-MAKKAH\OneDrive\Desktop\labmanual 3.exe

```
Result:30
Result:60

-----
Process exited after 0.03244 seconds with return value 0
Press any key to continue . . .
```

Example# 03

Objective:

Create overloaded functions to display text with different colors and repetition.

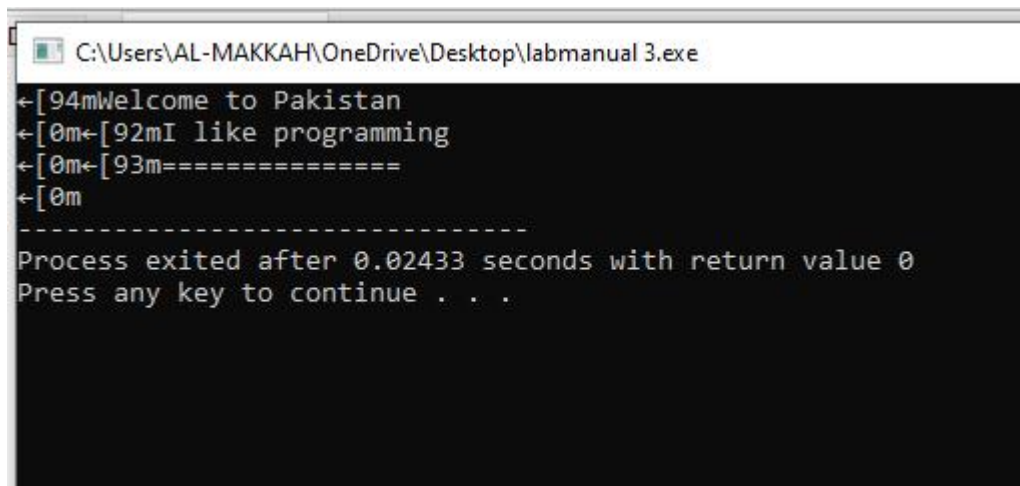
```
#include <iostream>
using namespace std;
```

```
#define RED 91
#define GREEN 92
#define YELLOW 93
#define BLUE 94
#define PURPLE 95
#define AQUA 96
```

```
void setColor(int colorCode) {
    cout << "\033[" << colorCode << "m";
}
```

```
void resetColor() {  
    cout << "\033[0m";  
}  
  
void displayText(string text) {  
    cout << text << endl;  
}  
  
void DisplayText(string text, int color) {  
    setColor(color);  
    cout << text << endl;  
    resetColor();  
}  
  
int main() {  
    DisplayText("Welcome to Pakistan", BLUE);  
    DisplayText("I like programming", GREEN);  
    DisplayText("=====", YELLOW);  
  
    return 0;  
}
```

Output:



```
C:\Users\AL-MAKKAH\OneDrive\Desktop\labmanual 3.exe
Welcome to Pakistan
I like programming
=====
-----
Process exited after 0.02433 seconds with return value 0
Press any key to continue . . .
```

3. Answer the following question:

1. Write a advantage of function overloading.

There are many Advantages of function overloading:

- An advantage of function overloading is code reusability and readability.
- It allows the use of the same function name for different types or numbers of parameters. It is making the code more intuitive and easier to maintain.
- Instead of creating multiple function names for similar operations, function overloading enables a more structured and organized approach, reducing redundancy and improving program efficiency
- It speeds up the program execution it provide readability and consistency in the program.
- It helps in code to maintenance.

2. Can functions be overloaded based on return type? Why or why not?

No, functions cannot be overloaded based on return type alone in this language c++.

Reason:

The function signature used for overloading includes the function name and the parameter list, but not the return type. The compiler determines which function to call based on the number and type of arguments passed. If two functions have the same name and parameter list but different return types, the compiler cannot distinguish between them, leading to an ambiguity and error.

For ex;

```
#include <iostream>
```

```
using namespace std;
```

```
class Example {
```

```
public:
```

```
    int func() { return 10; } // Function 1
```

```
    double func() { return 10.5; } // Function 2 (Error: Same name and  
parameters)
```

```
};
```

```
int main() {
```

```
    Example obj;
```

```
    cout << obj.func(); // Compiler error: Ambiguity
```

```
    return 0;
```

```
}
```

3. How does the compiler differentiate between overloaded function.

The compiler differentiates between overloaded functions based on their function signature, which includes the function name and the parameter list. The parameter list is number, type, or order of parameters. The return type is not considered for distinguishing overloaded functions.

Differentiate number of parameter:

```
void show(int a);
```

```
void show(int a, int b); // Different number of parameters
```

Different Types of Parameters:

```
void display(int a);
```

```
void display(double b); // Different parameter types
```

different order of parameter:

```
void print(int a, double b);
```

```
void print(double b, int a); // Different order of parameters
```

example:

```
#include <iostream>
```

```
using namespace std;
```

```
void greet(string name) {
```

```
    cout << "Hello, " << name << "!" << endl;
```

```
}
```

```
void greet(string name, int age) { // Different parameter list
```

```
    cout << "Hello, " << name << "!" << " You are " << age << " years old." << endl;
```

```
}
```

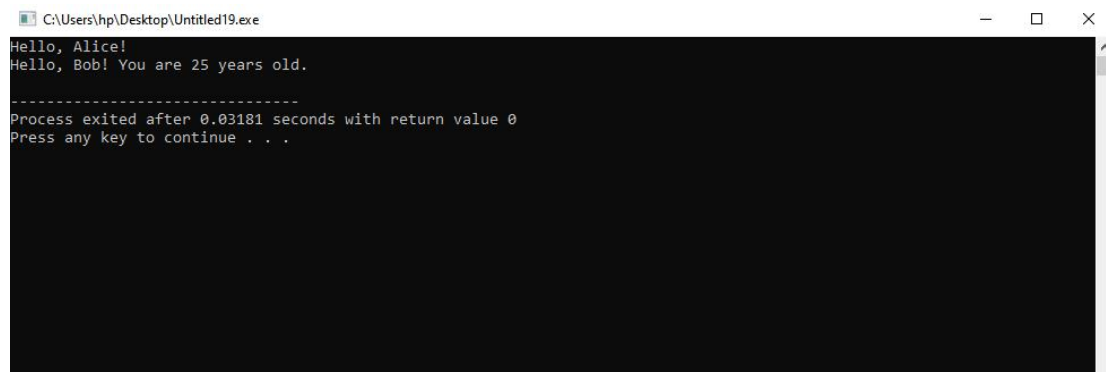
```
int main() {
```

```
    greet("Alice"); // Calls the first function
```

```
    greet("Bob", 25); // Calls the second function

    return 0;

}
```



```
C:\Users\hp\Desktop\Untitled19.exe
Hello, Alice!
Hello, Bob! You are 25 years old.
-----
Process exited after 0.03181 seconds with return value 0
Press any key to continue . . .
```