University of Anbar
College of Computer Science and Information Technology
Department of Computer Networks

C++ Functions II

Lecture 5

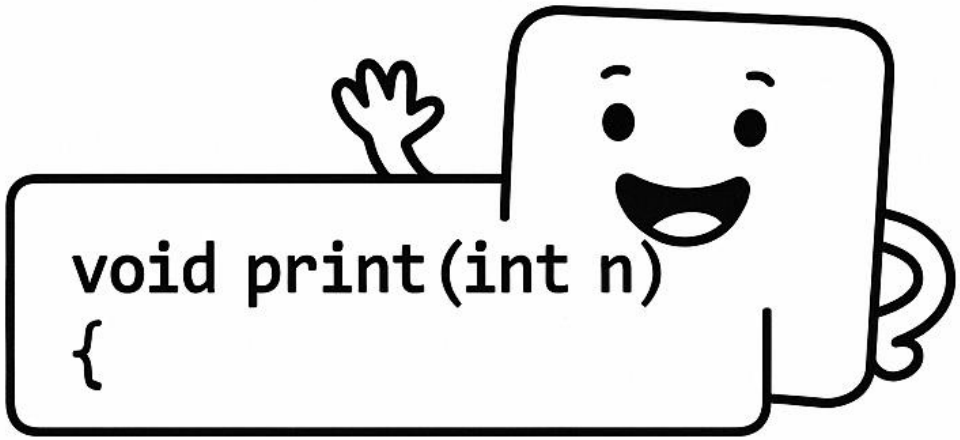
Dr. Ali Al-Kubaisi

Asst.Lect. Abdulrahman Abbas Mukhlif

Outlines

- C++ Function Overloading
- C++ Variable Scope
  - Local Scope
  - Global Scope
  - Naming Variables
- C++ Recursion

Function Overloading



```
void print(int n)
{
```

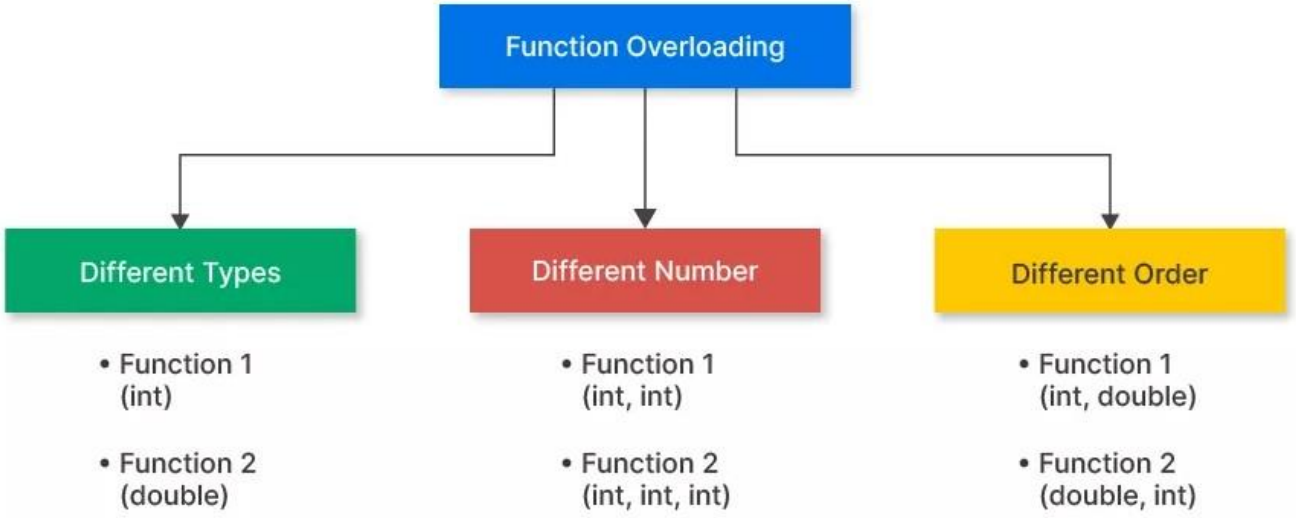
```
void print(double d)
{
```

```
void print(string str)
{
```

Function Overloading

- With **function overloading**, multiple functions can have the **same name** with:
  - ✓different parameters,
  - ✓different types
  - ✓and/or different order

```
int myFunction(int x)
float myFunction(float x)
double myFunction(double x, double y)
```



Function Overloading (Cont.)

```
int plusFuncInt(int x, int y)
{
  return x + y;
}

double plusFuncDouble(double x, double y)
{
  return x + y;
}

int main()
{
  int myNum1 = plusFuncInt(8, 5);
  double myNum2 = plusFuncDouble(4.3, 6.26);
  cout << "Int: " << myNum1 << "\n";
  cout << "Double: " << myNum2;
  return 0;
}
```

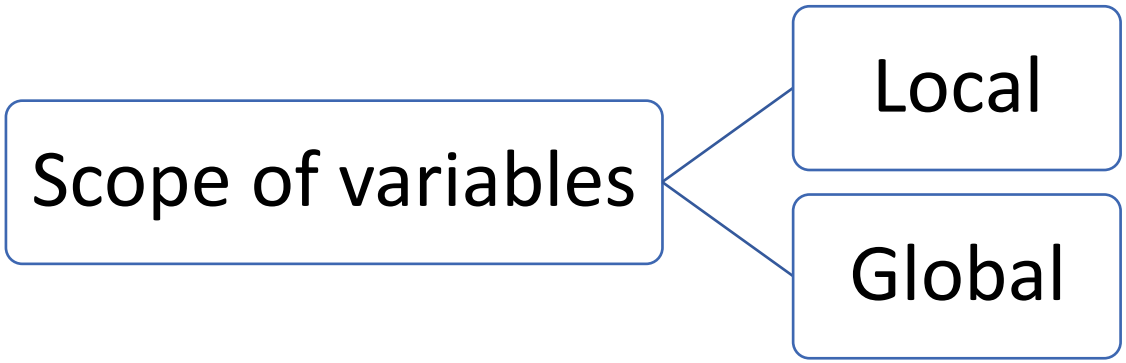
```
int plusFunc(int x, int y)
{
  return x + y;
}

double plusFunc(double x, double y)
{
  return x + y;
}

int main()
{
 int myNum1 = plusFunc(8, 5);
 double myNum2 = plusFunc(4.3, 6.26);
 cout << "Int: " << myNum1 << "\n";
 cout << "Double: " << myNum2;
 return 0;
}
```

C++ Variable Scope

- In C++, variables are only accessible inside the region they are created. This is called **scope**.



Local Scope

- **Local Scope**: A variable created inside a function belongs to the *local scope* of that function, and can only be used inside that function.

```
void myFunction()
{
  // Local variable that belongs to myFunction
  int x = 5;

  // Print the variable x
  cout << x;
}

int main() {
  myFunction();
  return 0;
}
```

Local Scope (cont.)

- A **local variable** cannot be used outside the function it belongs to.
- If you try to access it outside the function, an error occurs:

```
void myFunction() {
  // Local variable that belongs to myFunction
  int x = 5;
}

int main() {
  myFunction();

  // Print the variable x in the main function
  cout << x;
  return 0;
}
```

Global Scope

- A variable created outside of a function, is called a **global variable** and belongs to the *global scope*.
- Global variables are available from within any scope, global and local:

```
// Global variable x
int x = 5;

void myFunction() {
  // We can use x here
  cout << x << "\n";
}

int main() {
  myFunction();

  // We can also use x here
  cout << x;
  return 0;
}
```

Naming Variables

- If you operate with the same variable name inside and outside of a function, C++ will treat them as two separate variables; One available in the global scope (outside the function) and one available in the local scope (inside the function):

```
// Global variable x
int x = 5;

void myFunction() {
  // Local variable with the same name as the global variable (x)
  int x = 22;
  cout << x << "\n"; // Refers to the local variable x
}

int main() {
  myFunction();

  cout << x; // Refers to the global variable x
  return 0;
}
```

Recursion

- **Recursion** is the technique of making a function call itself.
- This technique provides a way to break complicated problems down into simple problems which are easier to solve.

```
int sum(int k) {
  if (k > 1) {
    return k + sum(k - 1);
  } else {
    return 1;
  }
}

int main() {
  int result = sum(5);
  cout << result;
  return 0;
}
```