Recap

- Function
- Function Declaration/Prototype
- Function Definition
- Calling a Function
- Return type vs void
- Parameters
- cin vs parameters
- Formal parameters vs Actual Parameters
- Function with default arguments
- Passing an array to the function

Math Library

- Include Library using #include<cmath>
- Few Functions:
  - double sin(double)
  - double cos(double)
  - double tan(double)
  - double sqrt(double)
  - double power(double,double)
  - double floor(double)
  - double ceil(double)
  - double round(double)
  - Int abs(int)

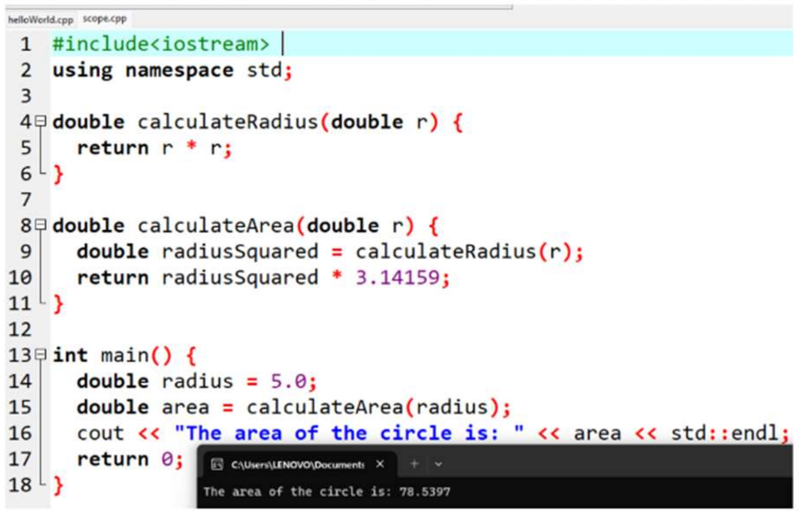
Explore math library

- https://www.programiz.com/cpp-programming/library-function/cmath/round#:~:text=The%20round()%20function%20in,in%20the%20cmath%20header%20file.

Function inside function

- Functions can call other functions and a function can call itself. This allows you to organize your code and reuse functionality.

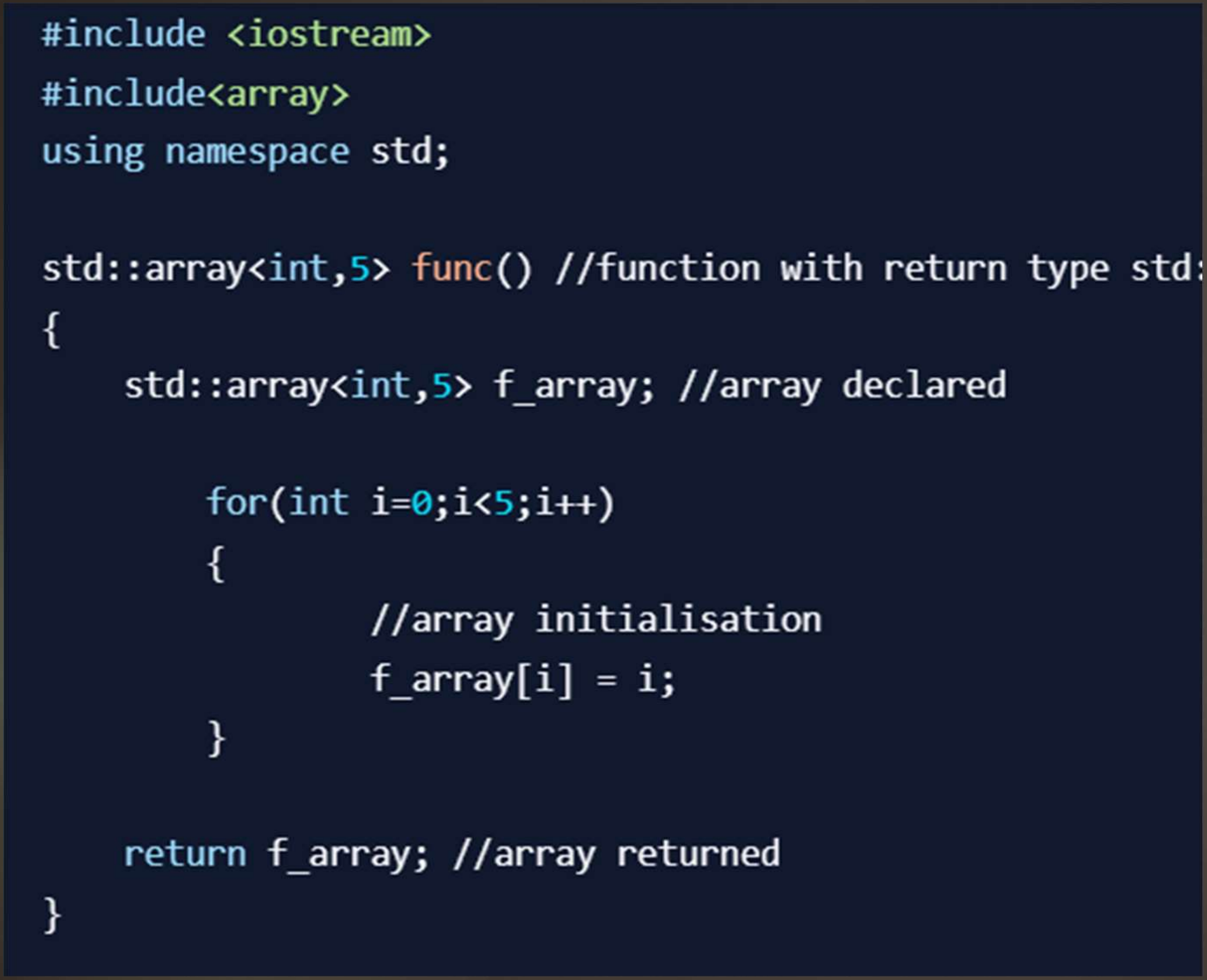
Function inside function



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

double calculateRadius(double r) {
  return r * r;
}

double calculateArea(double r) {
  double radiusSquared = calculateRadius(r);
  return radiusSquared * 3.14159;
}

int main() {
  double radius = 5.0;
  double area = calculateArea(radius);
  cout << "The area of the circle is: " << area << std::endl;
  return 0;
}
```

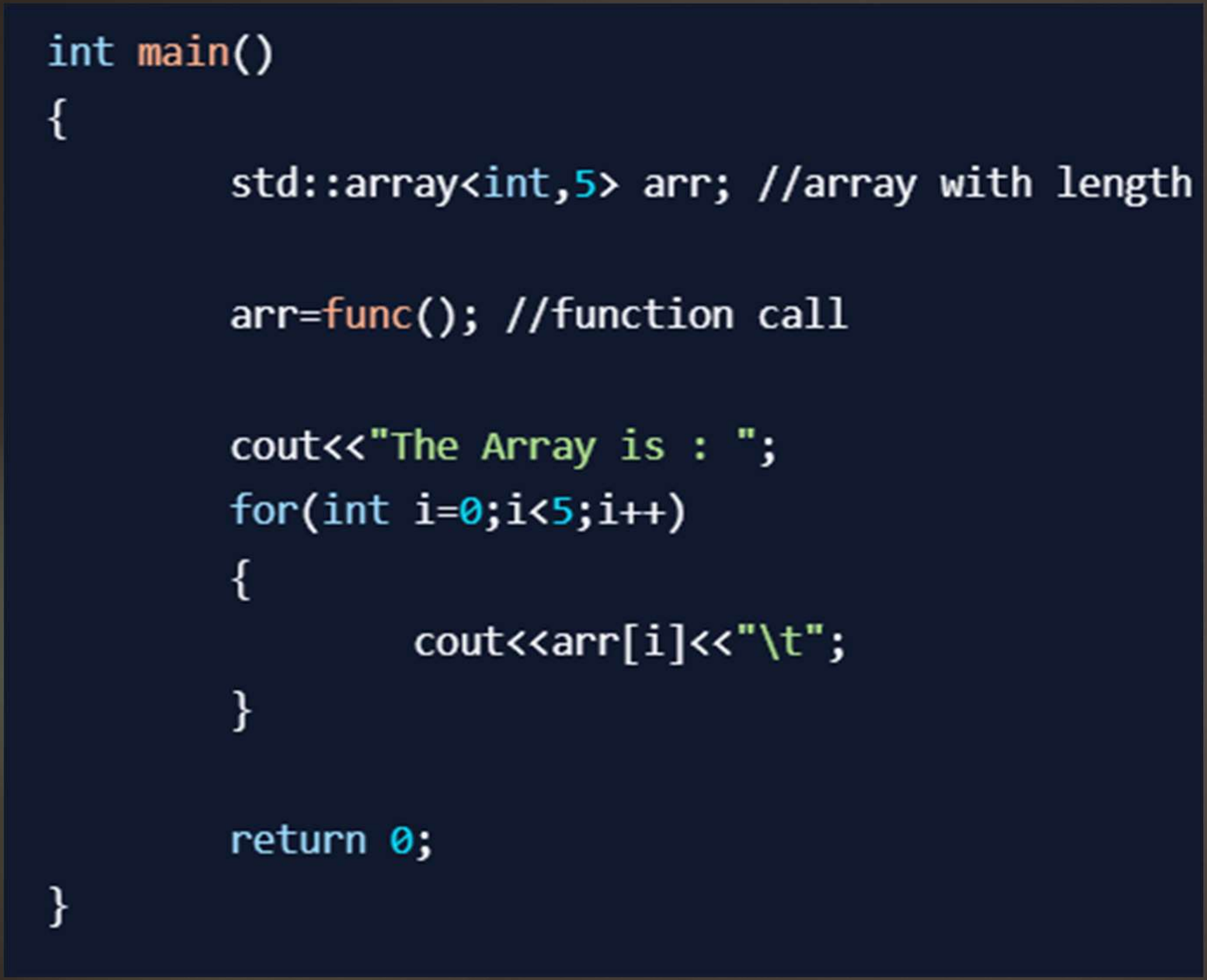
```
The area of the circle is: 78.5397
```

Return an array from the function



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

std::array<int,5> func() //function with return type std:
{
    std::array<int,5> f_array; //array declared

        for(int i=0;i<5;i++)
        {
                //array initialisation
                f_array[i] = i;
        }

    return f_array; //array returned
}
```

Return an array from the function



```
int main()
{
        std::array<int,5> arr; //array with length

        arr=func(); //function call

        cout<<"The Array is : ";
        for(int i=0;i<5;i++)
        {
                cout<<arr[i]<<"\t";
        }

        return 0;
}
```

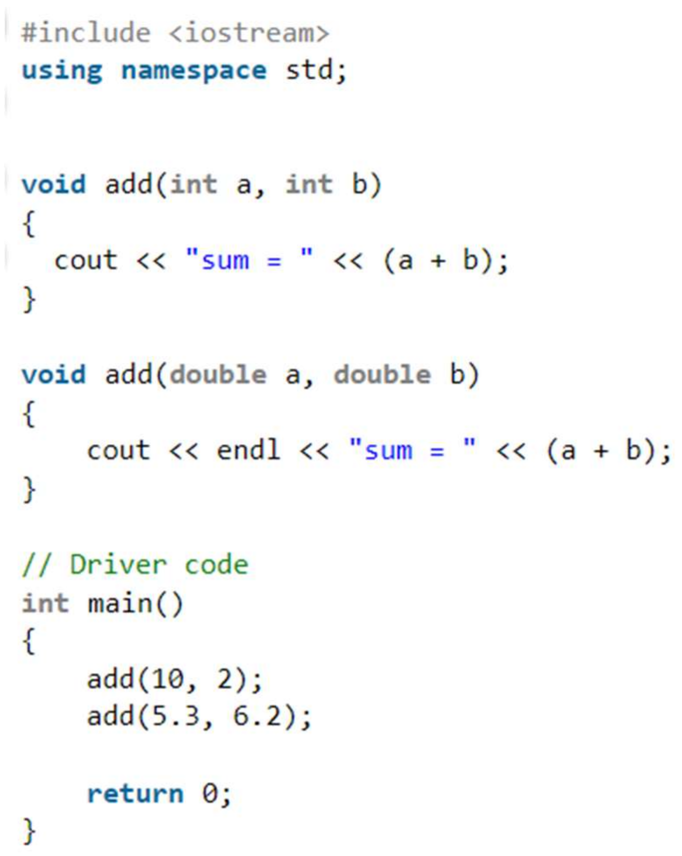
Function Overloading

- Function overloading means when two or more functions have same name but different parameters.
- Parameters can be different in terms of:
  - Number
  - Data Type
  - Sequence

Different number of parameters

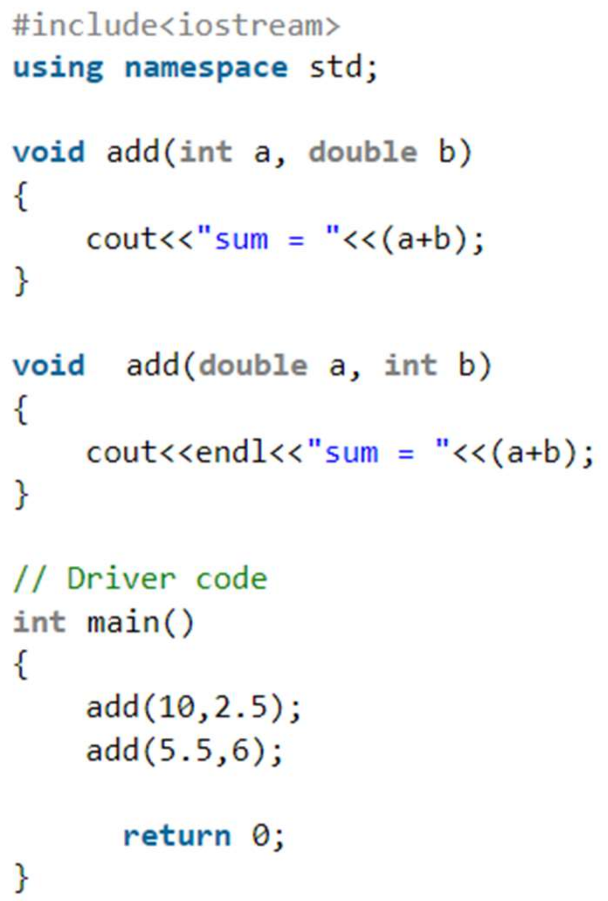
```cpp
#include <iostream>
using namespace std;

void add(int a, int b)
{
  cout << "sum = " << (a + b);
}

void add(int a, int b, int c)
{
    cout << endl << "sum = " << (a + b + c);
}

// Driver code
int main()
{
    add(10, 2);
    add(5, 6, 4);

    return 0;
}
```

Different Data Type of parameters



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


void add(int a, int b)
{
  cout << "sum = " << (a + b);
}

void add(double a, double b)
{
    cout << endl << "sum = " << (a + b);
}

// Driver code
int main()
{
    add(10, 2);
    add(5.3, 6.2);

    return 0;
}
```

Different Sequence of parameters



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

void add(int a, double b)
{
    cout<<"sum = "<<(a+b);
}

void  add(double a, int b)
{
    cout<<endl<<"sum = "<<(a+b);
}

// Driver code
int main()
{
    add(10,2.5);
    add(5.5,6);

      return 0;
}
```

Type conversion will also work with functions