Functions

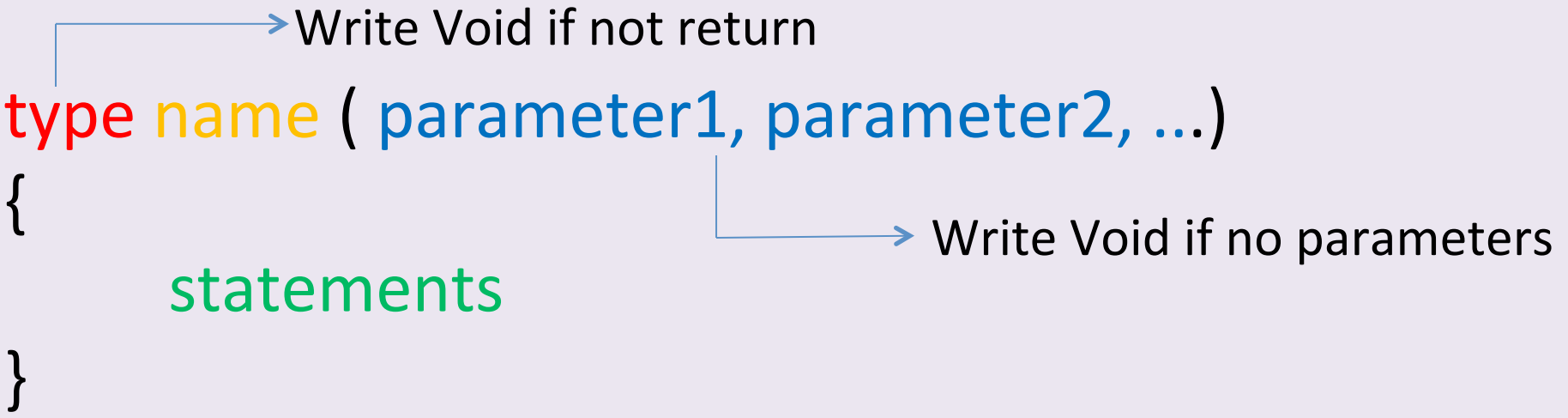
NEW YORK UNIVERSITY
ABU DHABI

- Defining a Function
- Call a Function
- Inline Function

Defining a Function

NEW YORK UNIVERSITY
ABU DHABI

In C++, a function is a group of statements that is given a name, and which can be called from some point of the program. The syntax for definition of a function is:



Defining a Function

NEW YORK UNIVERSITY
ABU DHABI

- Parameters:

  <type> name1, <type> name2, …, <type> nameN

- Return types:
  - It is given by "return" statement, for example, return 1;
  - No need to write "return" statement if no function return

Defining a Function

```
// function example
#include <iostream>
using namespace std;

int multiple (int a, int b)
{
  int r;
  r=a*b;
  return r;
}

int main ()
{
  int z;
  z = multiple(5,3);
  cout << "The result is " << z;
}
```

Call a Function

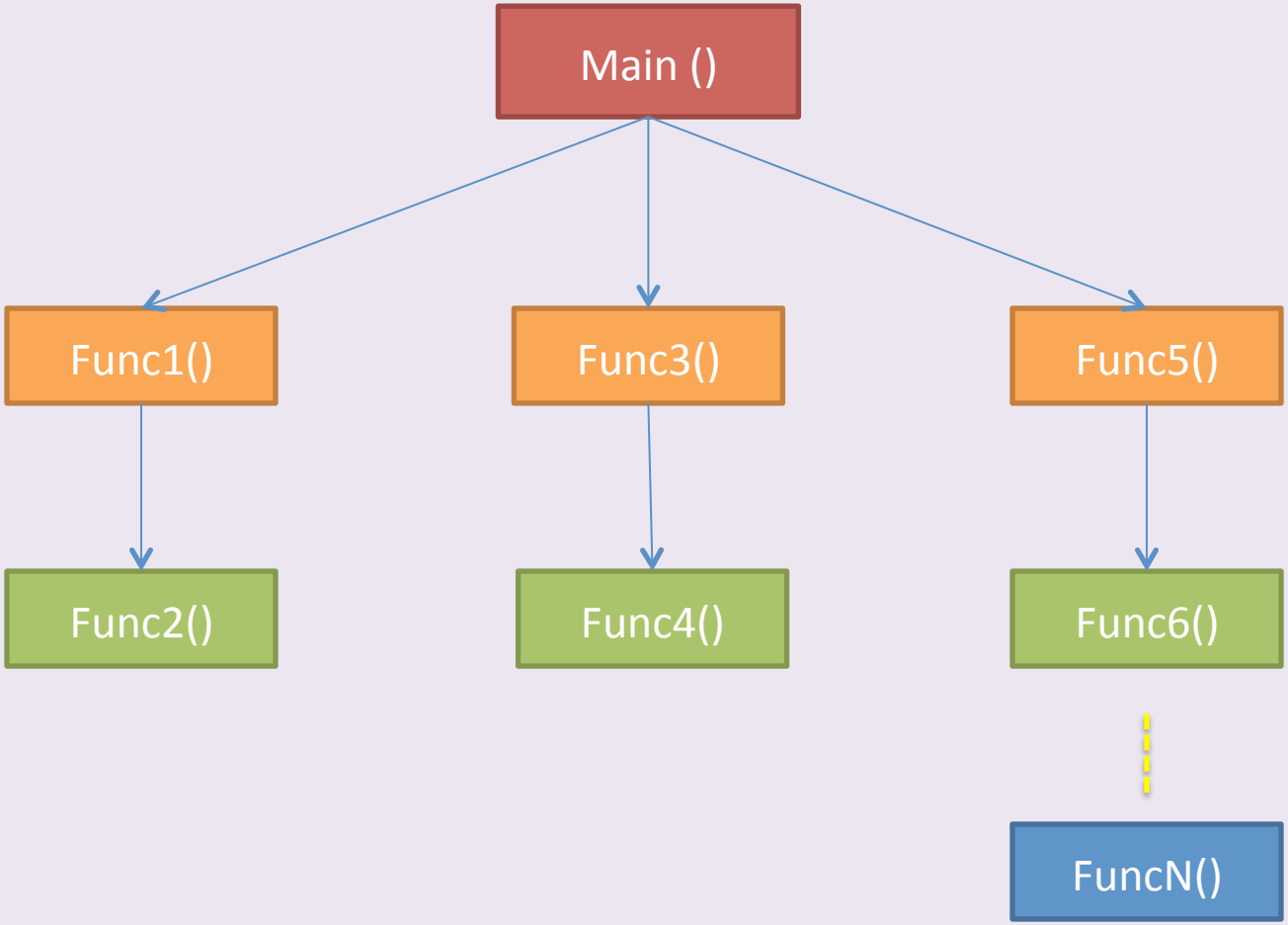
NEW YORK UNIVERSITY
ABU DHABI

- Defining a Function
- Call a Function
- Inline Function

Call a Function

NEW YORK UNIVERSITY
ABU DHABI

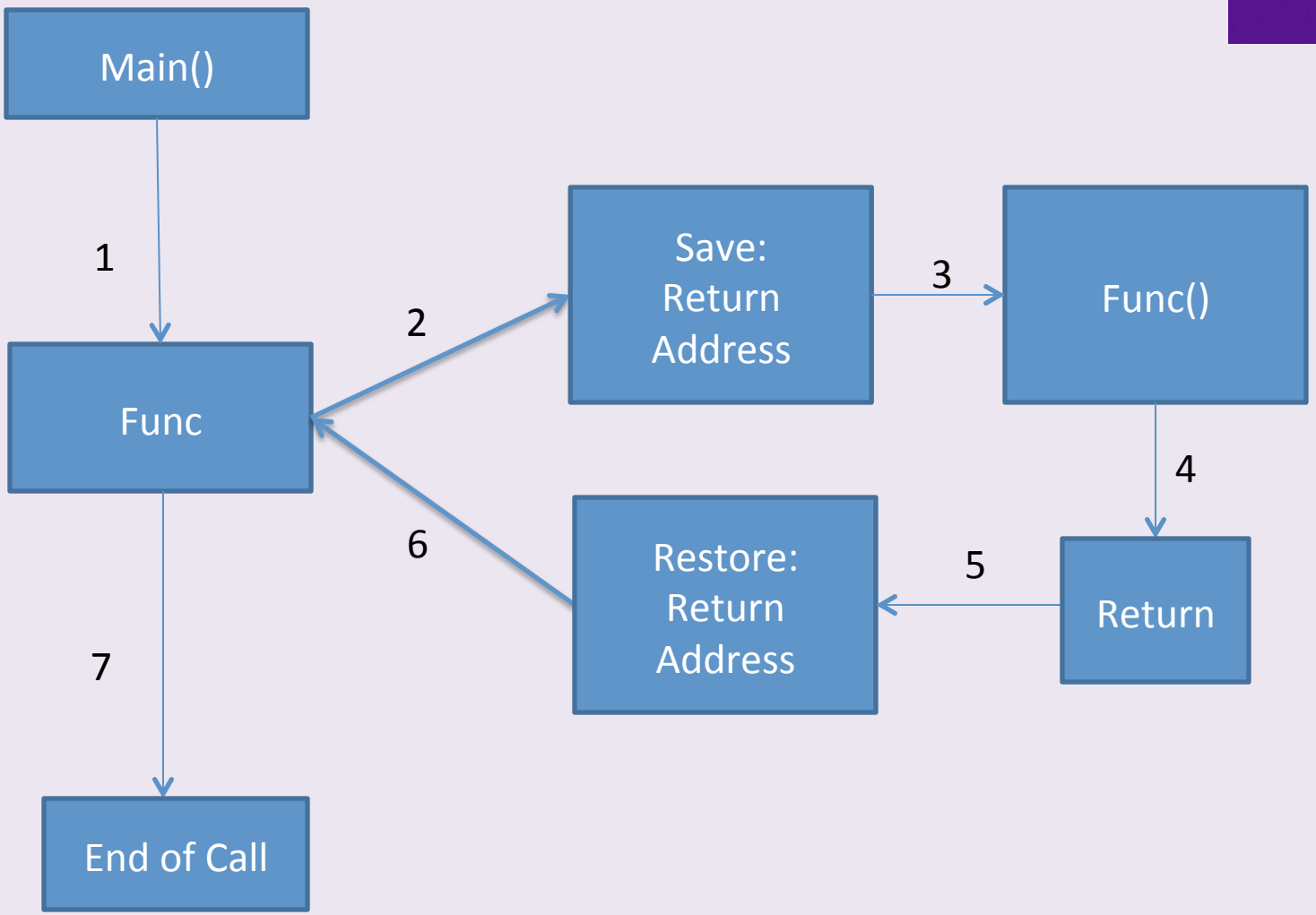
- To use a function, you will have to call or invoke the function to perform defined task.


Main ()
Func1()
Func3()
Func5()
Func2()
Func4()
Func6()
FuncN()

Main()


Call Func5
Call Func5
Call FuncN

Diagram of calling a Function



Parameters vs Arguments

In common usage, the terms parameter and argument are often interchanged. However, there is a distinction between the two:

- A function parameter is a variable declared in the prototype or declaration of a function

```
int example(int p)
```

P is the parameter

- An argument is the value that is passed to the function in place of a parameter

```
example(10)
```

10 is the argument passed to parameter

Passing Arguments

- Arguments passed by Value
- Arguments passed by Reference

Arguments passed by Value

Arguments can be passed by value. This means that, when calling a function, what is passed to the function are the values of these arguments on the moment of the call, which are copied into the variables represented by the function parameters.

For example, take:

```
int x=6, y=7, z;
z = multiple ( x, y );
```

In this case, function multiple is passed 6 and 7, which are copies of the values of x and y, respectively. These values (6 and 7) are used to initialize the variables set as parameters in the function's definition, but any modification of these variables within the function has no effect on the values of the variables x and y outside it, because x and y were themselves not passed to the function on the call, but only copies of their values at that moment.

Arguments Passed by Reference

In certain cases,

- It may be useful to access an external variable from within a function.

To do that, arguments can be passed by reference, instead of by value

- The memory of the arguments is too big therefore copying the

Arguments to parameters is difficult and infeasible

For example, the function swap in this code swaps the value of its two arguments, causing the variables used as arguments to actually be modified by the call:

```
void swap(int& a, int& b);
```

To gain access to its arguments, the function declares its parameters as references. In C++, references are indicated with an ampersand (&) following the parameter type.

Example:

```
void fun(int &y) // y is now a reference
{
    cout << "y = " << y << endl;
    y = 10;
    cout << "y = " << y << endl;
}

int main()
{
    int a = 5;
    cout << "a = " << a << endl;
    fun(a);
    cout << "a = " << a << endl;
    return 0;
}
```

Output:

```
a = 5
y = 5
y = 10
a = 10
```

Example

- Write a function to find biggest one among three numbers.

Inline Function

NEW YORK UNIVERSITY
ABU DHABI

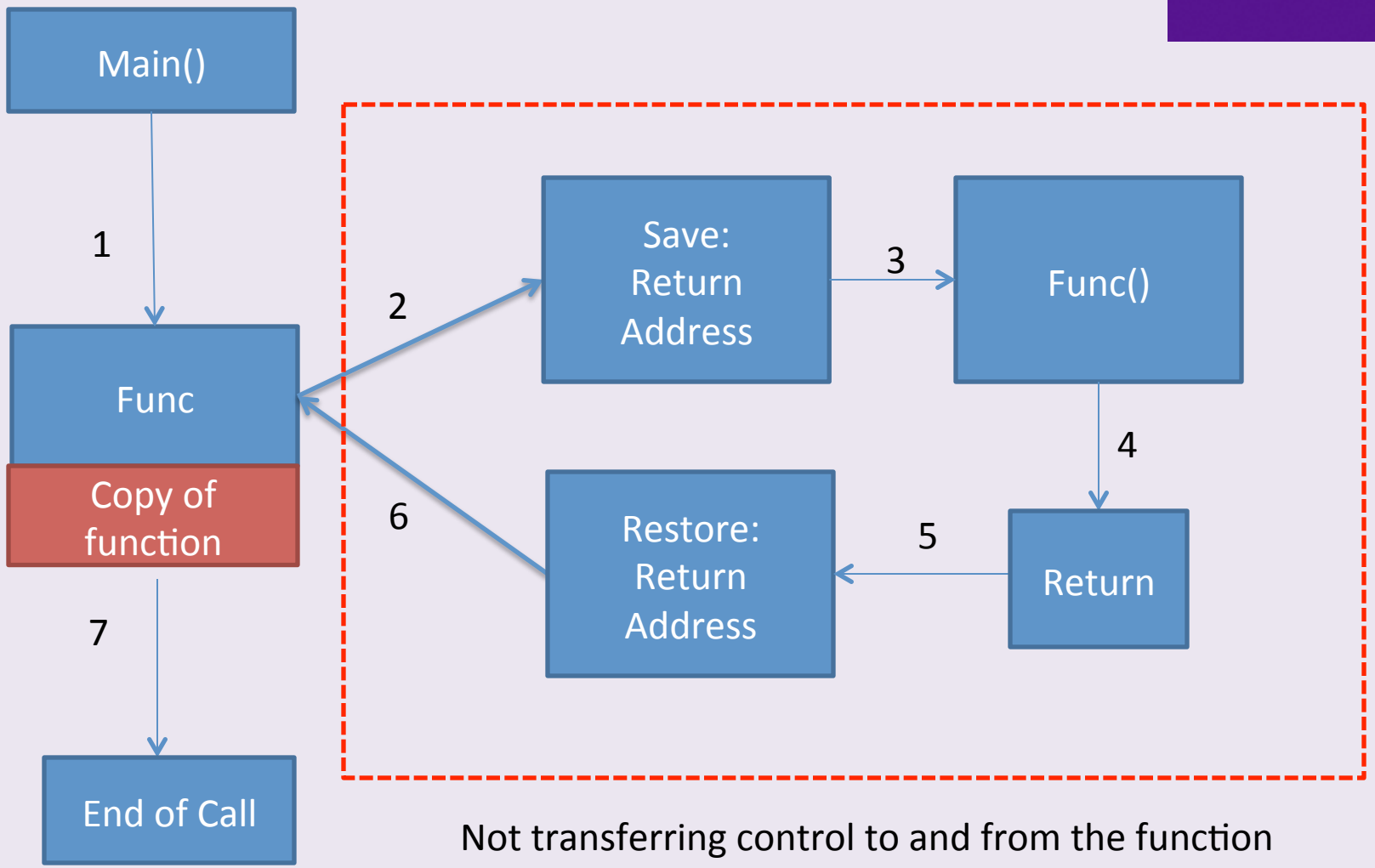
- Defining a Function
- Call a Function
- Inline Function

Inline Function

NEW YORK UNIVERSITY
ABU DHABI

An inline function is one for which the compiler copies the code from the function definition directly into the code of the calling function rather than creating a separate set of instructions in memory.

Inline Function



Not transferring control to and from the function

Example

```
#include <iostream>

using namespace std;

inline int Max(int x, int y)
{
   return (x > y)? x : y;
}

// Main function for the program
int main( )
{

   cout << "Max (20,10): " << Max(20,10) << endl;
   cout << "Max (0,200): " << Max(0,200) << endl;
   cout << "Max (100,1010): " << Max(100,1010) << endl;
   return 0;
}
```