C++

PROGRAMMING LANGUAGE

L05-FUNCTIONS

Mohammad Shaker
mohammadshaker.com
@ZGTRShaker
2010, 11, 12, 13, 14

Functions

- Why function?
- functions \ procedures in c++
- What's the function prototype (declaration) \ definition?
- Function signature (2010 exam Question)
  - What's it?

```
int foo(int);           // function prototype
```

  - It's just the name of the function with its parameters
    - No return type!!!! (Return type is not considered as part of the function's signature!, guess why?)

```
foo(int)                // function signature
```

Functions

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

int foo(int);                  // function prototype

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

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

foo(int);                          // function prototype


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

```
Compiler error, missing return type
```

Functions

- The following are the same

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

int Mult( int x )                    // note there's no; when we
                                              // write the definition here
{
            x = x * 2;
            return x;
}

void main()
{
}
```

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

int Mult( int x );

void main()
{}

int Mult( int x )
{
            x = x * 2;
            return x;
}
```

Functions

- The following are the same

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

int Mult( int x )
{
            x = x * 2;
            return x;
}

void main()
{
}
```

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

int Mult(int);                    // we didn't write the x
                                  // pirmeted when prototype only
void main()
{}

int Mult( int x )
{
            x = x * 2;
            return x;
 }
```

Functions

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

int Mult(int);

void main()
{
          Mult (3);
}

int Mult( int x )
{
          x = x * 2;
          return x;
}
```

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

int Mult(int);

void main()
{
          cout << Mult(3);
}

int Mult( int x )
{
          x = x * 2;
          return x;
}
```

```
6
```

Functions

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

int Mult(int);

void main()
{
          Mult(3);
}

int Mult( int x )
{
          x = x * 2;
          cout << x;
          return 0;
}
```

```
6
```

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

int Mult(int);

void main()
{
          int c = 3;
          Mult(c);
}

int Mult( int x )
{
          x = x * 2;
          cout << x;
          return 0;
}
```

```
6
```

Functions

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

int Mult(int);

void main()
{
          int x = 3;
          Mult(x);
}

int Mult( int x )
{
          x = x * 2;
          cout << x;
          return 0;
}
```

```
6
```

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

int Mult(int);

void main()
{
          int x = 3;
          Mult(x);
          cout << x;
}

int Mult( int x )
{
          x = x * 2;
          cout << x;
          return 0;
}
```

```
6
```

Functions

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

int Mult(int);

void main()
{
          int x = 3;
          Mult(x);
          cout << endl;
          cout << x;
}

int Mult( int x )
{
          x = x * 2;
          cout << x;
          return 0;
}
```

```
6
3
```

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

int Mult(int);

void main()
{
          int x = 3;
          Mult(x);
          cout << x;
}

int Mult( int x )
{
          x = x * 2;
          cout << x << endl;
          return 0;
}
```

```
6
3
```

Functions

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

int Mult(int);

void main()
{
          int x = 3;
          cout << Mult(x) << endl;
          cout << x;
}

int Mult( int x )
{
          x = x * 2;
          return x;
}
```

```
6
3
```

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

void Mult(void);

void main()
{
          int x = 3;
          Mult() << endl;
          cout << x;
}

void Mult( void )
{
          int x = 3;
          cout << x * 2 << endl;
}
```

```
Compiler error
```

Functions

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

void Mult(void);

void main()
{
          int c = 3;
          Mult() << endl;
          cout << c;
}

void Mult( void )
{
          int x = 3;
          cout << x * 2 << endl;
}
```

Compiler error

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

void Mult(void);

void main()
{
          Mult() << endl;
}

void Mult( void )
{
          int x = 3;
          x = x * 2;
          return 0;
}
```

Compiler error, return 0; with void function

Functions

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

void Mult(void);

void main()
{
          cout << Mult() << endl;
}

void Mult()
{
          int x = 3;
          x = x * 2;
}
```

Compiler error, cout with void return funtion

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

void Mult(void);

void main(void)
{
          Mult();
}

void Mult(void)
{
          int x = 3;
          x = x * 2;
}
```

At last everything’s working

Functions

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

void Mult(void);

void main(void)
{
          Mult();
}

void Mult(void)
{
          int x = 3;
          x = x * 2;
}
```

Compile and run

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

void Mult();

void main()
{
          Mult();
}

void Mult()
{
          int x = 3;
          x = x * 2;
}
```

Compile and run. In the parameter list, void or empty is the same

Functions

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

void Mult(void);

void main()
{
          Mult();
}

Mult()
{
          int x = 3;
          x = x * 2;
}
```

Compiler error, missing return type

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

Mult(void);

void main()
{
          Mult();
}

void Mult()
{
          int x = 3;
          x = x * 2;
}
```

Compiler error, missing return type

Functions

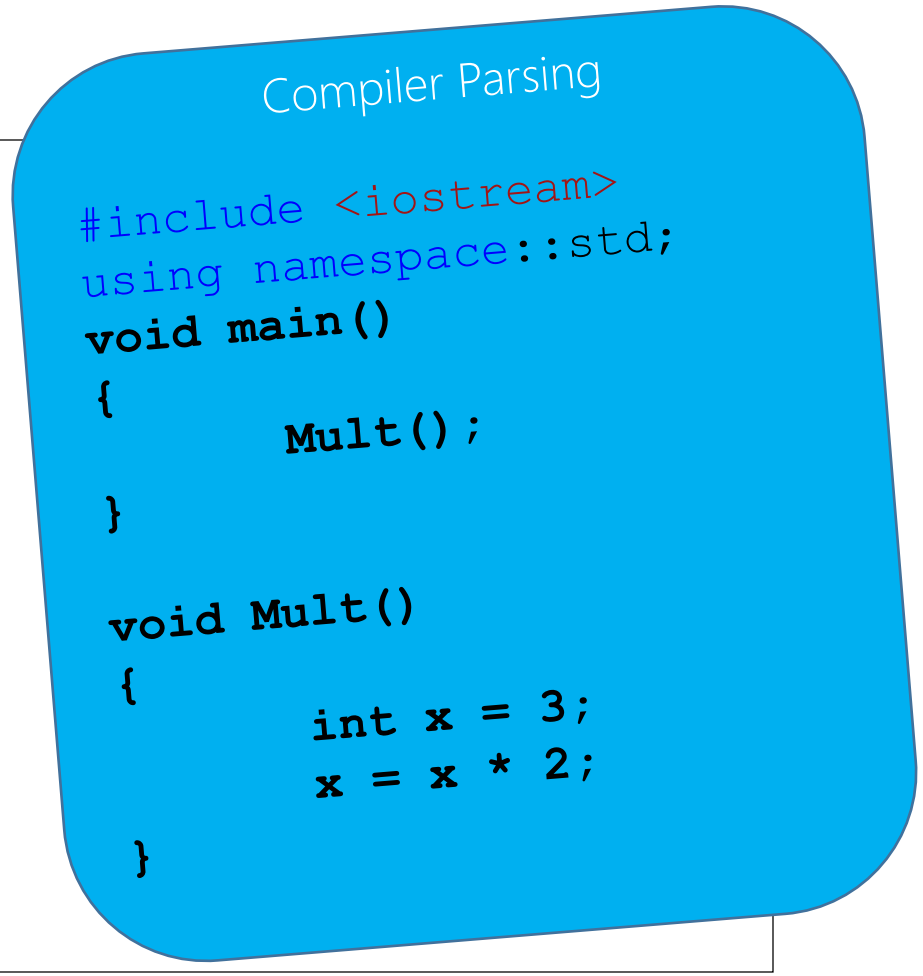
```
#include <iostream>
using namespace::std;

void main()
{
        Mult();
}

void Mult()
{
        int x = 3;
        x = x * 2;
}
```

Compiler error, missing function's prototype

50 pts

Functions

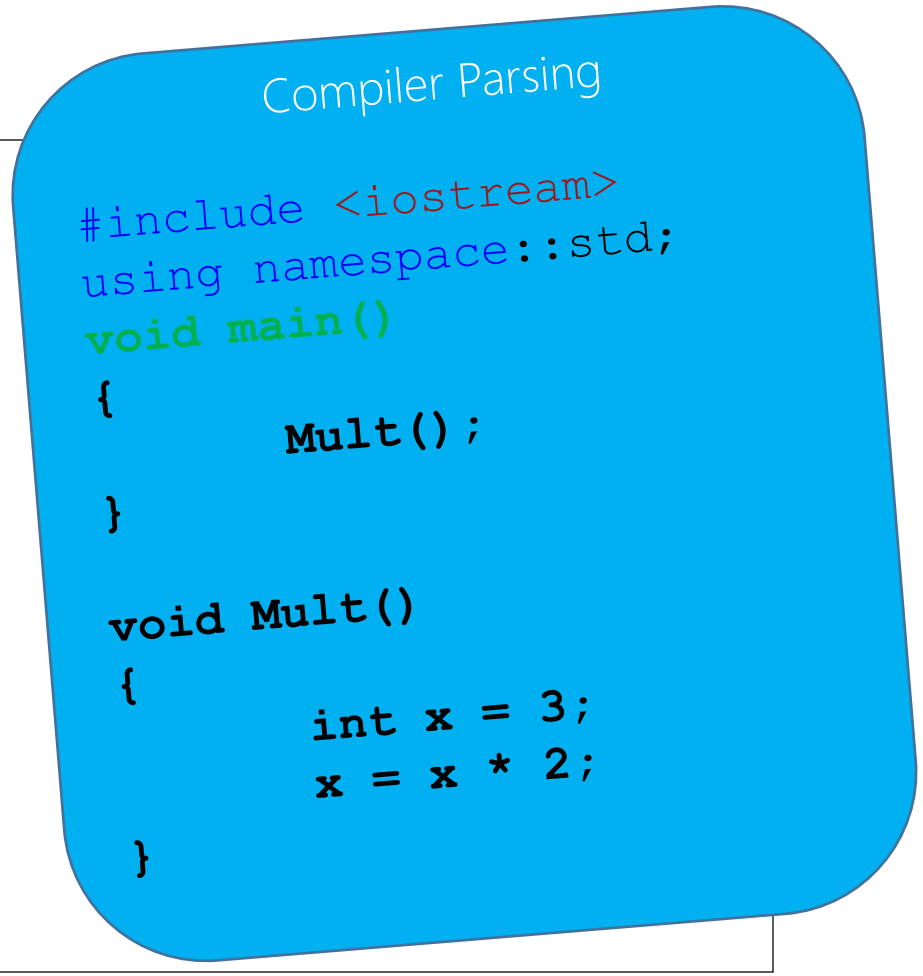
```
#include <iostream>
using namespace::std;

void main()
{
        Mult();
}

void Mult()
{
        int x = 3;
        x = x * 2;
}
```



Compiler error, missing function's prototype

Functions

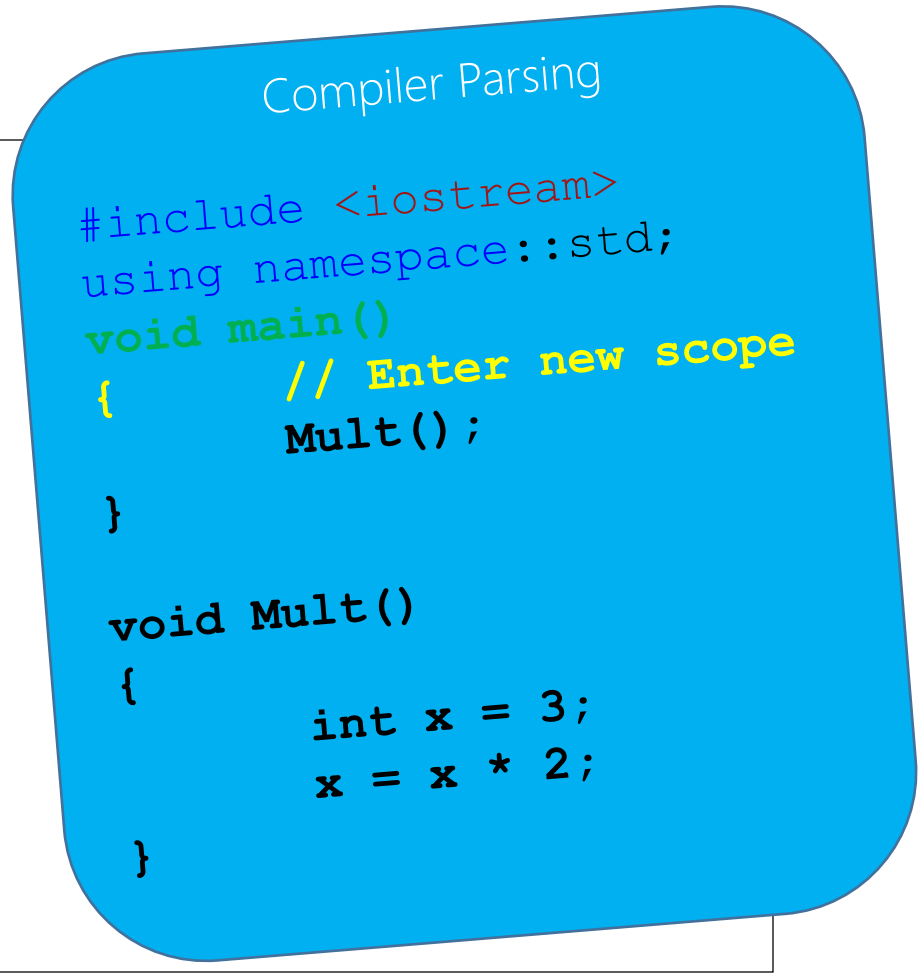
```
#include <iostream>
using namespace::std;

void main()
{
        Mult();
}

void Mult()
{
        int x = 3;
        x = x * 2;
}
```



Compiler error, missing function's prototype

Functions

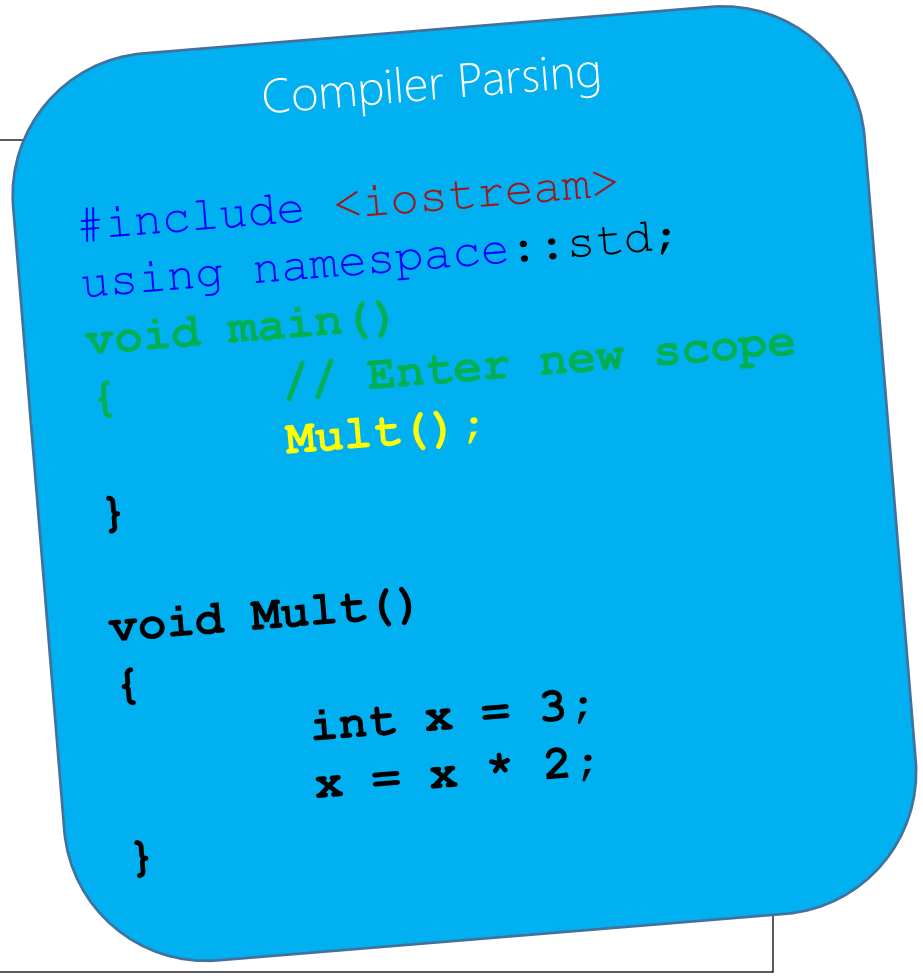
```
#include <iostream>
using namespace::std;

void main()
{
        Mult();
}

void Mult()
{
        int x = 3;
        x = x * 2;
}
```



Compiler error, missing function's prototype

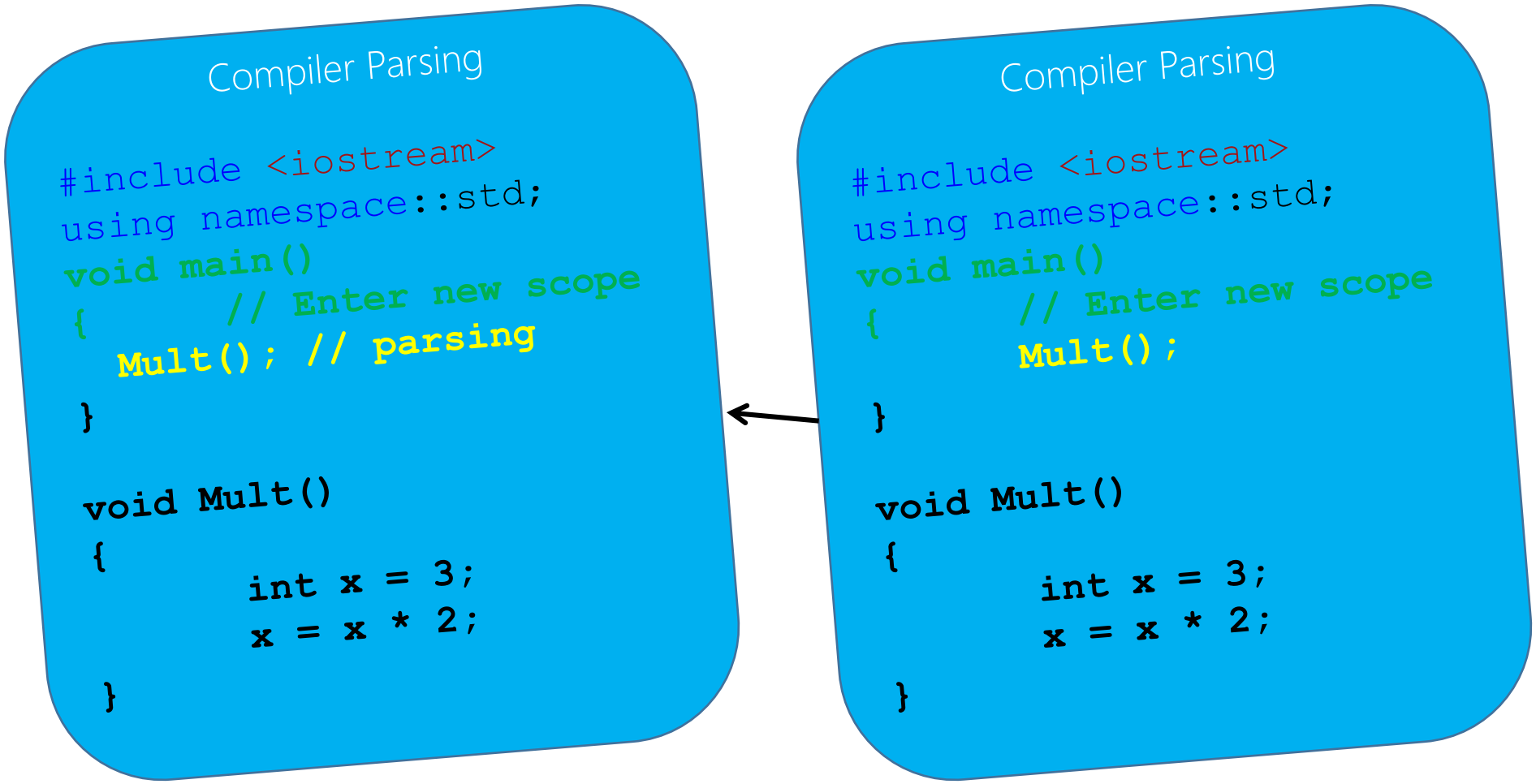
Functions

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

void main()
{
        Mult();
}

void Mult()
{
        int x = 3;
        x = x * 2;
}
```

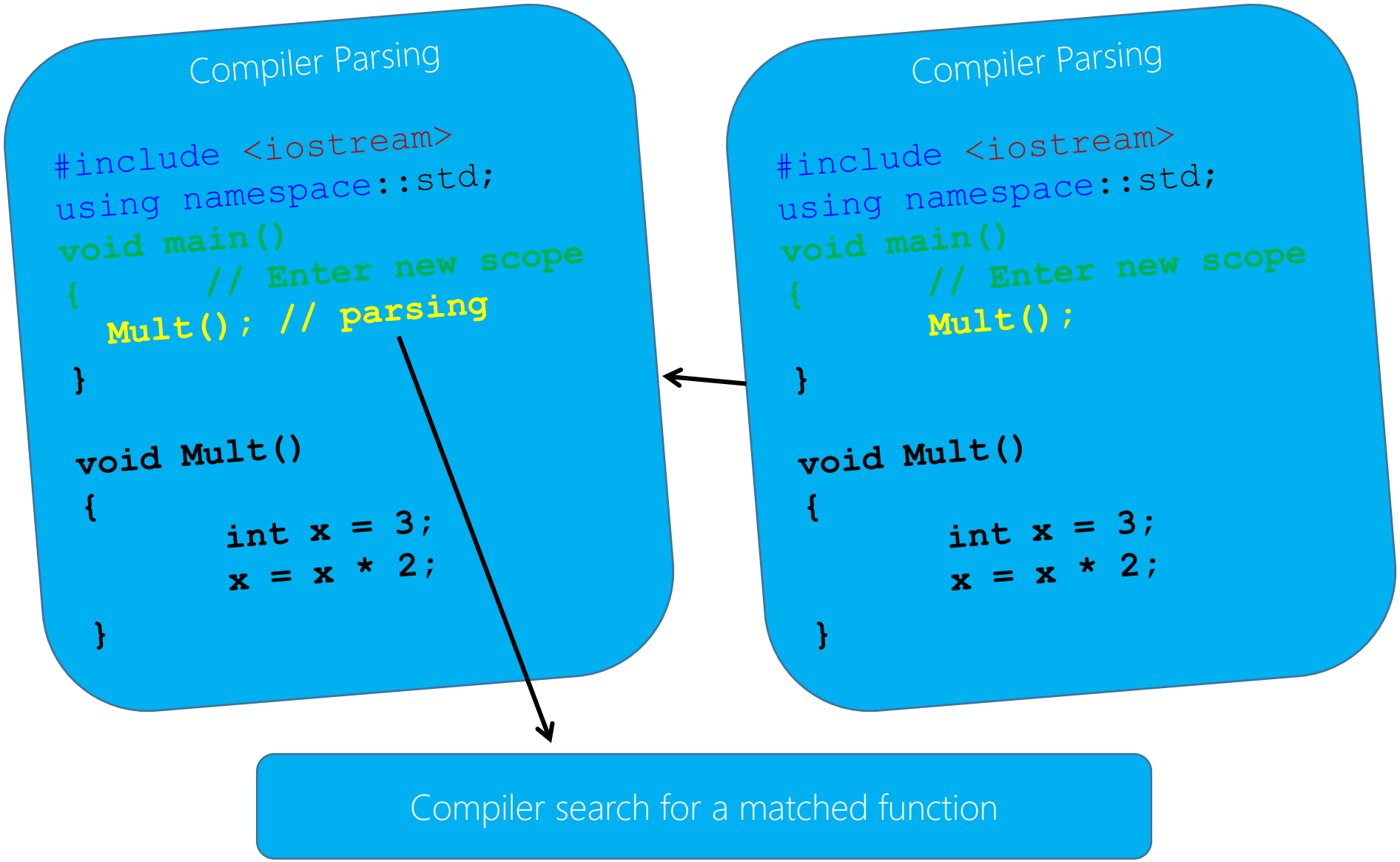


Compiler error, missing function's prototype

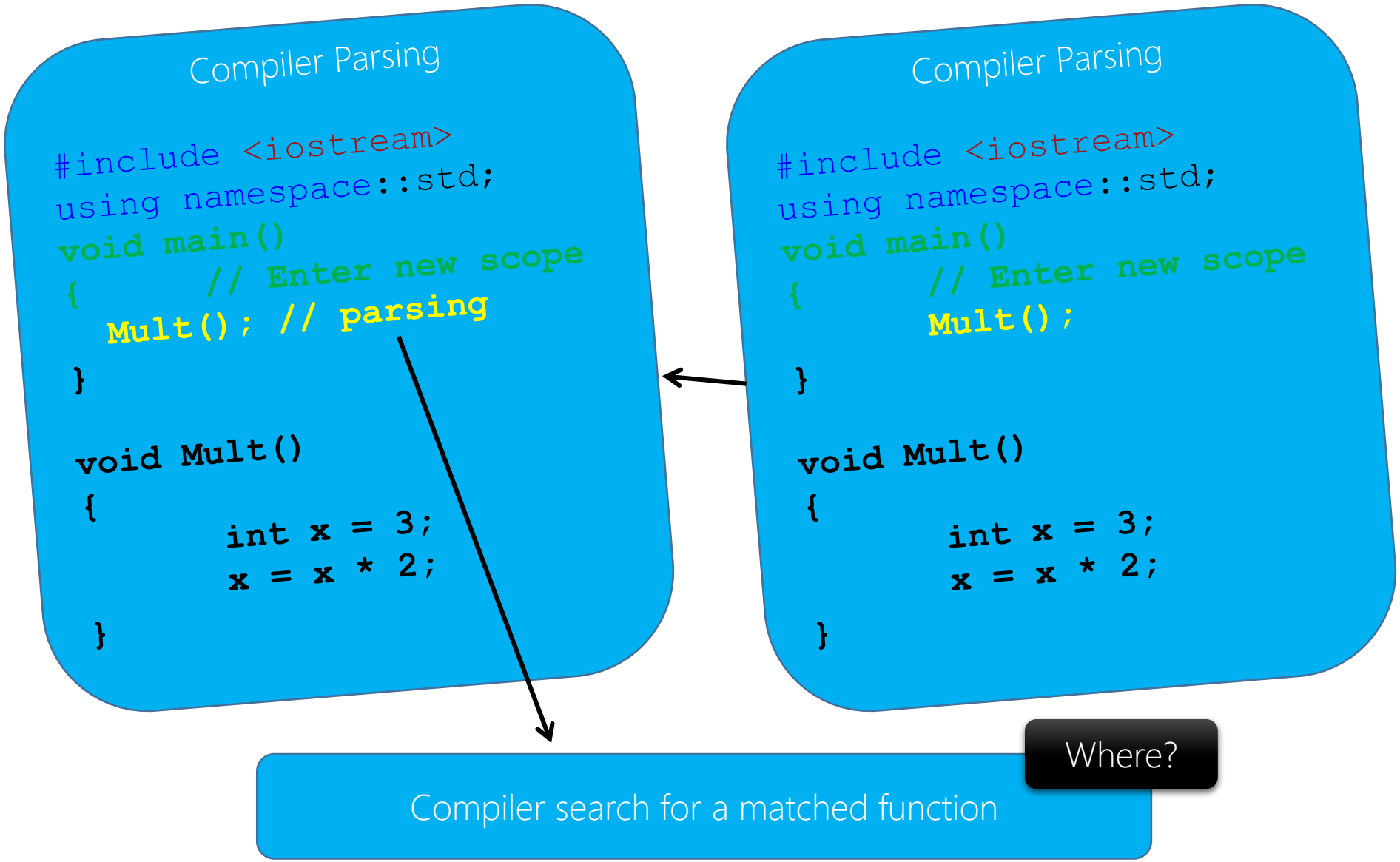
Functions


Compiler Parsing
#include <iostream>
using namespace::std;
void main()
{ // Enter new scope
Mult(); // parsing
}
void Mult()
{
int x = 3;
x = x * 2;
}
Compiler Parsing
#include <iostream>
using namespace::std;
void main()
{ // Enter new scope
Mult();
}
void Mult()
{
int x = 3;
x = x * 2;
}

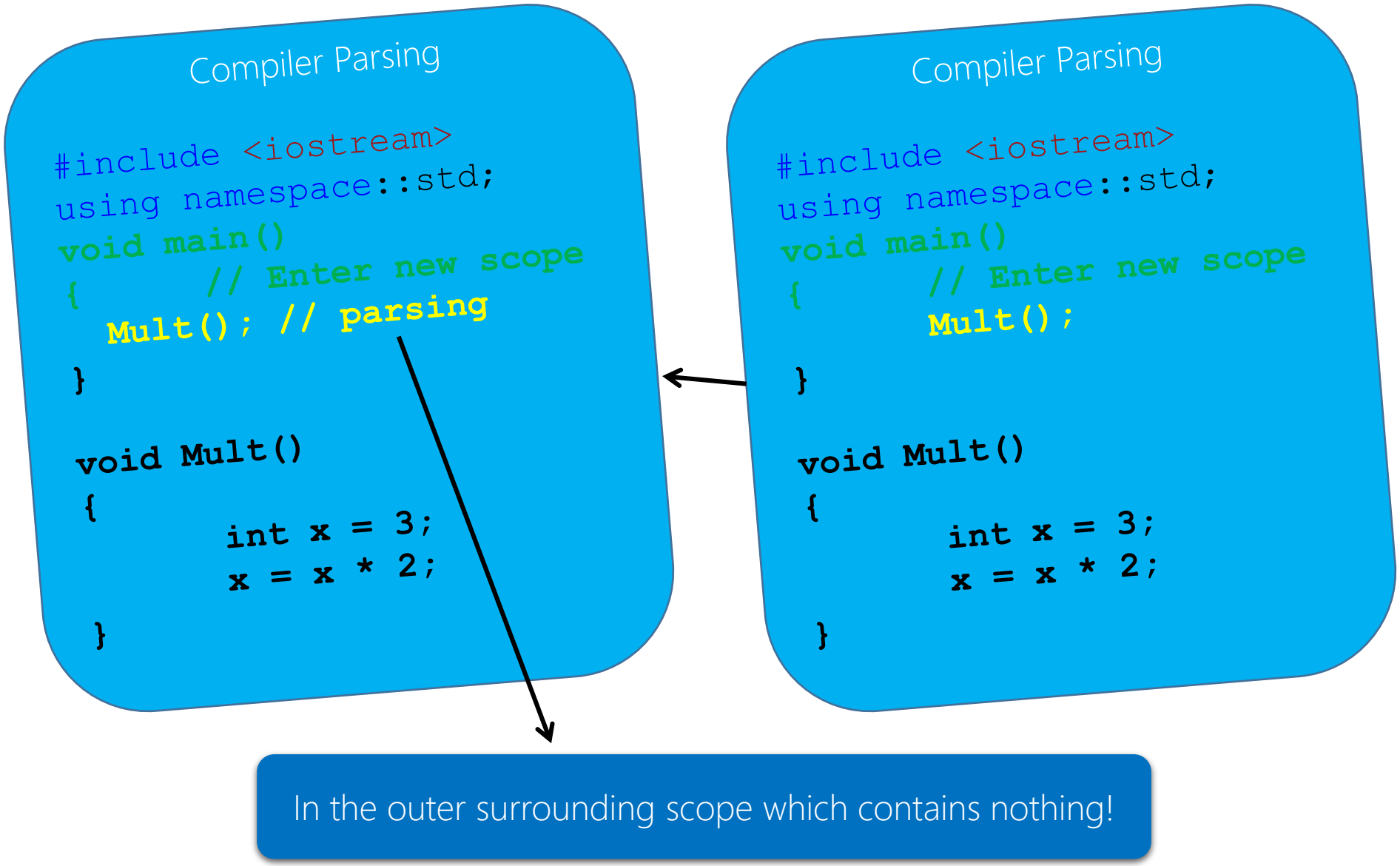
Functions


Compiler Parsing
#include <iostream>
using namespace::std;
void main()
{ // Enter new scope
Mult(); // parsing
}
void Mult()
{
int x = 3;
x = x * 2;
}
Compiler Parsing
#include <iostream>
using namespace::std;
void main()
{ // Enter new scope
Mult();
}
void Mult()
{
int x = 3;
x = x * 2;
}
Compiler search for a matched function

Functions


Compiler Parsing
#include <iostream>
using namespace::std;
void main()
{
// Enter new scope
Mult(); // parsing
}
void Mult()
{
int x = 3;
x = x * 2;
}
Compiler Parsing
#include <iostream>
using namespace::std;
void main()
{
// Enter new scope
Mult();
}
void Mult()
{
int x = 3;
x = x * 2;
}
Compiler search for a matched function
Where?

Functions


Compiler Parsing
#include <iostream>
using namespace::std;
void main()
{ // Enter new scope
Mult(); // parsing
}
void Mult()
{
int x = 3;
x = x * 2;
}
Compiler Parsing
#include <iostream>
using namespace::std;
void main()
{ // Enter new scope
Mult();
}
void Mult()
{
int x = 3;
x = x * 2;
}
In the outer surrounding scope which contains nothing!

Functions

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

void Mult(void);

void main(void)
{
          Mult();
}

int Mult(void)
{
          int x = 3;
          x = x * 2;
          return x;
}
```

```
Compiler error, return type differs
```

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

int Mult(void);

void main(void)
{
          Mult();
}

void Mult(void)
{
          int x = 3;
          x = x * 2;
}
```

```
Compiler error, return type differs
```

Functions

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

int Mult(void);

void main(void)
{
          cout << Mult() << endl;
}

int Mult(void)
{
          int x = 3;
          x = x * 2;
          return x;
}
```

```
6
```

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

int Mult(void);

void main(void)
{
          cout << Mult() << endl;
}

int Mult(void)
{
          int x = 3;
          return x * 2;
}
```

```
6
```

Functions

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

int Mult(void);
void main(void)
{
          cout << Mult() << endl;
}

int Mult(void)
{
          int x = 3;
          return x * 2;
          cout << x << end;
}
```

```
6
```

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

int Mult(int x );

void main(void)
{
          cout << Mult() << endl;
}

int Mult(int x )
{
          return x * 2;
}
```

```
Compiler error, no variable passed to function
```

Nothing got excuted after the return statement

Functions

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

int Mult(int x );

void main(void)
{
          cout << Mult(3) << endl;
}

int Mult(int x )
{
          return x * 2;
}
```

```
0
```

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

int Mult(int x );

void main(void)
{
          cout << Mult(3) << endl;
}

int Mult(int x )
{
          int x = 2;
          return x * 2;
}
```

```
Compiler error, redefinition of variable x
```

Functions

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

int Mult(int, int );
void main(void)
{
          int i1, i2;
          cout << Mult(i1,i2) << endl;
}

int Mult(int x, int y)
{
          return x * 2;
}
```

```
6
```

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

int Mult(int, int );
void main(void)
{
          int i1, i2;
          cout << Mult(i1,i2) << endl;
}

int Mult(int x, y)
{
          return x * 2;
}
```

```
Compiler error, missing type for y
```

Functions

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

int Mult(int, int );
void main(void)
{
            int i1, i2;
            cout << Mult(i1,i2) << endl;
}

int Mult(int x, int y)
{
            if (x = 2 )
            {
                        return 3;
            }
            else
            {
                        return x * 2;
            }
}
```

```
3
```

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

int Mult(int, int );
void main(void)
{
            int i1, i2;
            cout << Mult(i1,i2) << endl;
}

int Mult(int x, int y)
{
            int MeMe (int z)
            {
                        return 0;
            }
}
```

```
Compiler error, can't define a function inside another one
```

Functions

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

int Mult(int, int );
void main(void)
{
            float i1= 3.4, i2;
            cout << Mult(i1,i2) << endl;
}

int Mult(int x, int y)
{
            return 2*x;
}
```

```
6
```

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

int Mult(int);

void main(void)
{
            for (int i = 0; i<=10; i++)
            {
                        cout << Mult(i) << "-";
            }
            cout << “hehe” << endl;
            cout<< endl;

}

int Mult(int x )
{
            return 2*x;
}
```

```
0-2-4-6-8-10-12-14-16-18-20-hehe

Press any key to continue
```

Functions

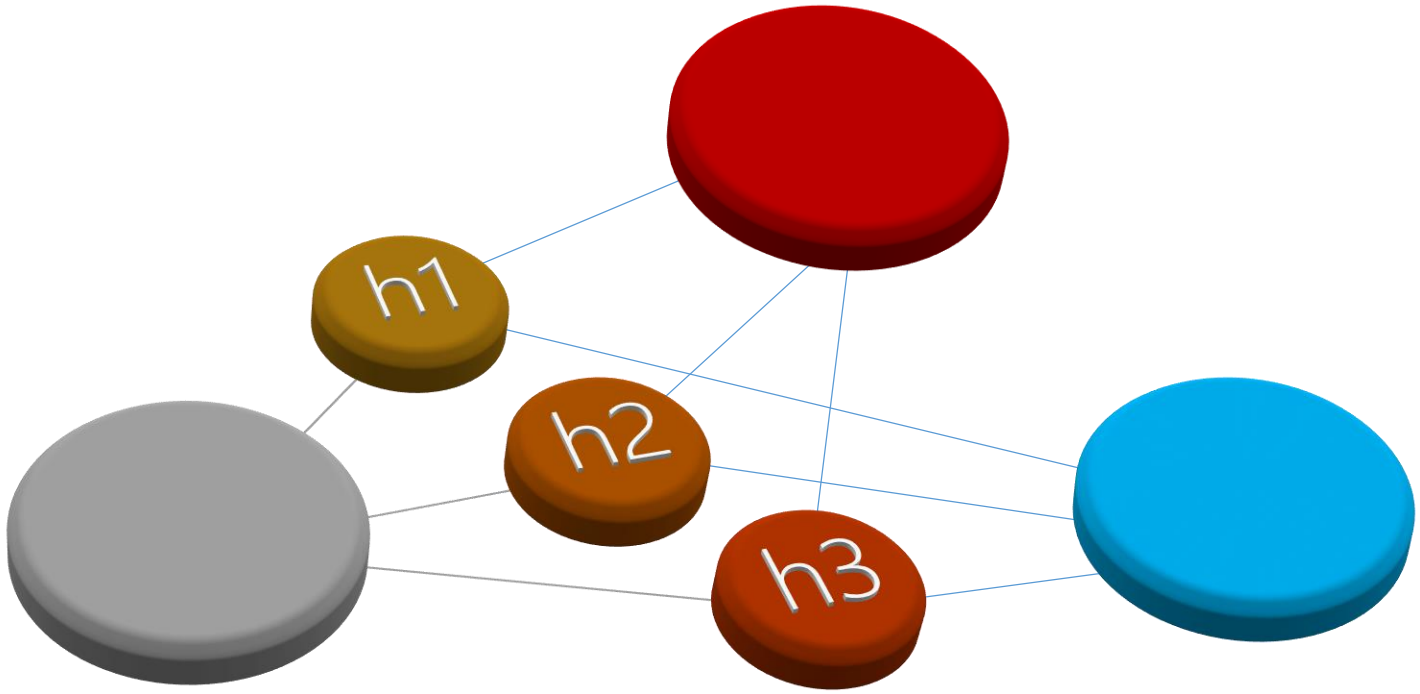
```
#include <iostream>
using namespace::std;

void Crazy1();
void Crazy2(int);

void main(void)
{           Crazy1();   }

void Crazy1()
{
            int x = 3;
            Crazy2(x);
            cout << x;
}

void Crazy2(int x )
{
            x+=6;
            cout << x << endl;
}
```

```
9
3
```

Headers

Headers

- The Concept of headers



Headers File

- What's a header?
  - Moving functions out side the main program
    - Logical groups
      - Stored in their own files

- How to do it?
  - By Moving
    - function declarations
      - Into headers files
        - » (.h) files // header files
    - function definitions
      - Into source files
        - » (.cpp) files // source files
      - Note: function definitions can't be split up into multiple files!

Functions

- To use it
  - Include <cmath>

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


void main(void)
{
            cout << sqrt(100.0) << endl;
}
```

```
10
```

```
#include <iostream>

using namespace::std;


void main(void)
{
            cout << sqrt(100.0) << endl;
}
```

```
Compiler error identifier not found
```

Functions

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


void main(void)
{
            cout << sqrt(100) << endl;
}
```

Compiler error, can't use integer "no overloaded function"

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


void main(void)
{
            cout << sqrt(sqrt(100.0)) << endl;
}
```

3.16228

Functions

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


void main(void)
{
            int x;
            cout << sqrt(6*x) << endl;
}
```

Compiler error, can't use integer "no overloaded function"

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


void main(void)
{
            double x = 3;
            cout << sqrt(3*x) << endl;
}
```

3

Functions

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


void main(void)
{
            float x = 3;
            cout << sqrt(3*x) << endl;
}
```

```
3
```

```
rand()
```

rand()

- Random number
  - rand()
  - Needs to include <cstdlib>
    - Generates unsigned integers
      - Between
        - 0 (Zero :D)
        - Rand_Max number (usually 32767)

rand()

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


void main(void)
{
        int x = 3;
        x = rand();
        cout << x << endl;
}
```

```
41
```

Now, when compiling many times
Every time we have the same random value!

```
41
```

```
41
```

```
41
```

rand()

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


void main(void)
{
            int x = 3;
            x = rand();
            cout << x << endl;
}
```

```
54
```

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


void main(void)
{
            int x = 3;
            x = rand();
            cout << x << endl;
            cout << x << endl;
}
```

```
156
156
```

Works without <cstlib>

rand()

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


void main(void)
{
            int x = 3;
            x = rand();
            cout << x << endl;
            x = rand();
            cout << x << endl;
}
```

```
156
12356
```

srand()

srand()

- `srand()`
  - `#include <cstdlib>`
  - Give a starting value for the rand()function
  - Usually used once in program

srand()

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


void main(void)
{
            int x = 3;
            srand (time(0));
            cout << x << endl;
}
```

```
3
```

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


void main(void)
{
            int x = 3;
            srand (time());
            cout << x << endl;
}
```

```
Compiler error, need value for time()
```

srand()

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


void main(void)
{
        int x = 3;
        srand (time(0));
        x = rand ();
        cout << x << endl;
}
```

What happened when compiling many times?

```
456
```

```
12345
```

```
189
```

Now, when compiling many times. Every time we have a new different value

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


void main(void)
{
        int x = 3;
        srand (5);
        x = rand ();
        cout << x << endl;
}
```

What happened when compiling many times?

```
48
```

```
48
```

```
48
```

Not an appropriate srand() paramater leads to insufficient using of srand()!!!

srand() with clock()

```
#include <iostream>
#include <ctime>
using namespace::std;
void main(void)
{
          srand(clock());
          // clock is from <ctime>
          // no needs to value for clock ()
          for (int i = 0; i < 10; i++)
                    cout << rand() << endl;
}
```

```
714
4415
16337
2807
14052
5430
30050
16163
20477
3146
Press any key to continue
```

rand()

- Scaling & shifting
  - %
    - x % y // returns a value between 0 to y-1
    - let's see an example

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


void main(void)
{
          int i;
          i = rand() % 6 + 1;
          // rand() % 6: number between 0 to 5
          // +1: makes a shift of the range we have
          // 0 to 5 becomes 1 to 6
          cout << i << endl;
          // here will print a number absolutely between 1 to 6
}
```

```
6
```

Or any other number between 1 and 6

Variables and Storage

Variables and Storage

- Name, type, size, values
- Storage class
  - Period variable living in memory
- Linkage
  - Multiple-file processing, which files can use it.
- Scope
  - Variable can be referenced in program

Storage Classes

- `auto` keyword
  - Can used Explicitly
  - `auto double x;`
- `register` keyword
  - High speed register
  - `register double x = 4;`
- `static` keyword (Very important)
  - function's local variables
  - Keeps values between functions call
  - Known ONLY in ITS OWN function
  - `Static double x = 3;`
- `extern` key word
  - Global variables accessible to functions in Same file, Other files
  - Must be declared in each file which has been used in.
  - Used to access Global variable in another file

Storage Classes

```
#include <iostream>
using namespace::std;
void main(void)
{
        auto double x = 4;
}
```

Compile and run

```
#include <iostream>
using namespace::std;
void main(void)
{
        auto register double x = 4;
}
```

Compiler error, can't use both at the same time

```
#include <iostream>
using namespace::std;
void main(void)
{
        register double x = 4;
}
```

Compile and run

```
#include <iostream>
using namespace::std;
void main(void)
{
        register auto double x = 4;
}
```

Compiler error, can't use both at the same time

`Static` Storage (Very important)

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

void StaticMyHeadache()
{
          static int x = 8;
          cout << "In function, x = " <<  x << endl;
          x*=2;
          cout << "In function, x = " <<  x << endl;
}

int main()
{
          int x = 3;
          cout << x << endl;
          StaticMyHeadache();
          cout << x << endl;
          StaticMyHeadache();
}
```

```
3
In function, x = 8
In function, x = 16
3
In function, x = 16
In function, x = 32
Press any key to continue
```

Defined and initialized at the same time in the ***first time of calling the function*** then the compiler no longer parse this line of code

extern Storage

- 1st file

```
#include <iostream>
using namespace::std;
void main(void)
{
          int MyGlobalVar;
}
```

- 2nd file

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

extern int MyGlobalVar;
```

extern Storage

- 1st file

```
#include <iostream>
using namespace::std;
void main(void)
{
          int MyGlobalVar;
}
```

- 2nd file

```
#include <iostream>
using namespace::std;
void main(void)
{
          extern int MyGlobalVar;
}
```

```
Compiler error, why?
```

2 “main”s in the same project! that’s wrong! Any project in the world will only have one main

inline

Inline

For small, common-used functions
No function call, just copy the code into the program
Compiler can ignore Inline

Inline functions

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

int IWannaSleep(int);

void main( )
{
          int x;
          cout << "Enter the Input Value: ";
          cin>>x;
          cout<<"\n The Output is: " << IWannaSleep(x) << endl;
}

inline int IWannaSleep (int x1)
{
          return 5*x1;
}
```

```
Enter the Input Value: 2


 The Output is: 10

Press any key to continue
```

Blocks and Scopes

Blocks and Scopes

- Scopes
  - Local Scopes
  - Global Scopes

- The Golden Rule
  - Life time of block scope variables is lifetime of block

Blocks and Scopes

```
#include <iostream>

using namespace::std;


void main(void)
{
          int x = 9, y = 3;
          {
          cout << "This is inner block " << endl;
          }
}
```

```
This is inner block
```

```
#include <iostream>

using namespace::std;


void main(void)
{
          int x = 9, y = 3;
          {
                    cout << x << endl;
          }
}
```

```
9
```

Blocks and Scopes

```
#include <iostream>

using namespace::std;


void main(void)
{
		int x = 9, y = 3;
		{
			cout << x << endl;
		}
		cout << x << endl;
}
```

```
9
9
```

```
#include <iostream>

using namespace::std;


void main(void)
{
		int x = 9, y = 3;
		{
			int x = 4;
			cout << x << endl;
		}
}
```

```
4
```

Blocks and Scopes

```
#include <iostream>

using namespace::std;


void main(void)
{
          int x = 9, y = 3;
          {
                    int x;
                    x = 4;
                    cout << x << endl;
          }
           cout << x << endl;
}
```

```
4
9
```

```
#include <iostream>

using namespace::std;


void main(void)
{
          int x = 9, y = 3;
          {
                    int x = 4;
                    cout << x << endl;
          }
           cout << y << endl;
}
```

```
4
3
```

Blocks and Scopes

```
#include <iostream>

using namespace::std;


void main(void)
{
          int x = 9;
          {
                    int x = 4, y = 3;
                    cout << x << endl;
          }
           cout << y << endl;
}
```

```
Compiler error, y undeclared identifier
```

Blocks and Scopes

```
#include <iostream>

using namespace::std;
int x = 35;                                    // Global Variable

void bar()
{
            int x = 3;                         // local Variable
}

void main(void)
{
   int x = 9, y = 3;
            {
                        int x = 6;             // local Variable
            }
    cout << x << endl;
}
```

```
9
```

```
#include <iostream>

using namespace::std;
int x = 35;                                    //Global Variable

void bar()
{
            int x = 3;                         // local Variable
}

void main(void)
{
            int x = 9, y = 3;
            {
                        int x = 6;             // local Variable
            }
            cout <<::x << endl;
}
```

```
35
```

Blocks and Scopes

```
#include <iostream>

using namespace::std;
int x = 34;                         // Global Variable

void bar()
{
            int x = 3;              // local Variable
            cout <<::x << endl;
}

void main(void)
{
            int x = 9, y = 3;
            {
                        int x = 6;  // local Variable
            }
}
```

```
Compile & Run
```

```
#include <iostream>

using namespace::std;
int x = 34;                                     // Global Variable

void bar()
{
            int x = 3;                          // local Variable
            cout << x << endl;
}

void main(void)
{
            int x = 9, y = 3;
            {
                        int x = 6;              // local Variable
            }
            bar();
}
```

```
3
```

Blocks and Scopes

```
#include <iostream>

using namespace::std;
int x = 34;                                    // Global Variable

void bar()
{
            int x = 3;                         // local Variable
            cout << ++::x << endl;
}

void main(void)
{
            int x = 9, y = 3;
            {
                        int x = 6;             // local Variable
            }
            bar();
}
```

```
35
```

```
#include <iostream>

using namespace::std;
int x = 34;                                    // Global Variable

void bar()
{
            int x = 3;                         // local Variable
            cout <<::x++ << endl;
}

void main(void)
{
            int x = 9, y = 3;
            {
                        int x = 6;             // local Variable
            }
            bar();
            cout <<::x << endl;
}
```

```
34
35
```

Math Library Functions

Method	Description	Example
`ceil( x )`	rounds *x* to the smallest integer not less than *x*	`ceil( 9.2 )` is `10.0` `ceil( -9.8 )` is `-9.0`
`cos( x )`	trigonometric cosine of *x* (*x* in radians)	`cos( 0.0 )` is `1.0`
`exp( x )`	exponential function *ex*	`exp( 1.0 )` is `2.71828` `exp( 2.0 )` is `7.38906`

`fabs( x )`	absolute value of *x*	`fabs( 5.1 )` is `5.1`
		`fabs( 0.0 )` is `0.0`
		`fabs( -8.76 )` is `8.76`

`floor( x )`	rounds *x* to the largest integer not greater than *x*	`floor( 9.2 )` is `9.0` `floor( -9.8 )` is `-10.0`
`fmod( x, y )`	remainder of *x/y* as a floating-point number	`fmod( 13.657, 2.333 )` is `1.992`
`log( x )`	natural logarithm of *x* (base *e*)	`log( 2.718282 )` is `1.0` `log( 7.389056 )` is `2.0`
`log10( x )`	logarithm of *x* (base 10)	`log10( 10.0 )` is `1.0` `log10( 100.0 )` is `2.0`
`pow( x, y )`	*x* raised to power *y* (*xy*)	`pow( 2, 7 )` is `128` `pow( 9, .5 )` is `3`
`sin( x )`	trigonometric sine of *x* (*x* in radians)	`sin( 0.0 )` is `0`
`sqrt( x )`	square root of *x*	`sqrt( 900.0 )` is `30.0` `sqrt( 9.0 )` is `3.0`
`tan( x )`	trigonometric tangent of *x* (*x* in radians)	`tan( 0.0 )` is `0`

Quiz

Quiz - 1

```
#include <iostream>

using namespace::std;
int x = 34;

void foo()
{
          static int x = 3;   // initialized only ONE time
          cout << "first x in foo = " << x <<  endl;
          x++;
          cout << "second x in foo = " << x <<  endl;
}
```

```
void main(void)
{
          int x = 9;
          cout << x << endl;
          {
                    int x = 5;
                    cout << x << endl;
          }
          cout << x << endl;
          foo();
          foo();
          cout << x << endl;
          cout <<::x << endl;
}
```

```
9
5
9
first x in foo = 3
second x in foo = 4
first x in foo = 4
second x in foo = 5
9
34
Press any key to continue
```

Quiz - 2

```
#include <iostream>

using namespace::std;
int x = 34;

void foo()
{
            static int x = 3;     // initialized only ONE time
            x = 5;
            cout << "first x in foo = " << x <<  endl;
            x++;
            cout << "second x in foo = " << x <<  endl;
}
```

```
void main(void)
{
            int x = 9;
            cout << x << endl;
            {
                        int x = 5;
                        cout << x << endl;
            }
            cout << x << endl;
            foo();
            foo();
            cout << x << endl;
            cout <<::x << endl;
}
```

```
9
5
9
first x in foo = 5
second x in foo = 6
first x in foo = 5
second x in foo = 6
9
34
Press any key to continue
```

150 pts

Quiz – 3, 4

```
#include <iostream>

using namespace::std;
int x = 34;


void foo()
{
          static int x = 3;
          int x = 3;
          cout << "first x in foo = " << x <<  endl;
          x++;
          cout << "second x in foo = " << x <<  endl;
}
```

Does this function runs?

```
Compiler error, redefinition of x
```

```
#include <iostream>

using namespace::std;
int x = 34;


void foo()
{
          static x = 3;
          x = 5;
          cout << "first x in foo = " << x <<  endl;
          x++;
          cout << "second x in foo = " << x <<  endl;
}
```

Does this function runs?

```
Compiler error, missing type after static
```

Functions Default Parameters

Default Parameters

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

void print(int x=0, int n=4 )
{
          cout << x << endl;
}

void main(void)
{
          print(2,3);
}
```

```
2
```

```
#include <iostream>
using namespace::std;
int s = 0;

void print(int x=0, int n )
{
          cout << x << endl;
}

void main(void)
{
          print(2,3);
}
```

```
Compiler error, missing default value for parameter 2
```

When defaulting a parameter, all other parameters - if exist - on the right side should be defaulted too

Default Parameters

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

void print(int x, int n=0 )
{
          cout << x << endl;
}

void main(void)
{
          print(2,3);
}
```

```
2
```

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

void print(int x, int n, int y  )
{
          cout << x << endl;
}

void main(void)
{
          print(2,3,4);
}
```

```
2
```

Default Parameters

```
#include <iostream>
using namespace::std;
int s = 0;

void print(int x, int n, int y = 0    )
{
          cout << x << endl;
}

void main(void)
{
          print(2,3,4);
}
```

```
2
```

```
#include <iostream>
using namespace::std;
int s = 0;

void print(int x, int n, int y = 0    )
{
          cout << x << endl;
}

void main(void)
{
          print(2,3);
}
```

```
2
```

Default Parameters

```
#include <iostream>
using namespace::std;
int s = 0;

void print (int x, int n, int y=10 )
{
          cout << x << endl;
          cout << n << endl;
          cout << y << endl;
}

void main(void)
{
          print(2,3);
}
```

```
2
3
10
```

Default Parameters

```
#include <iostream>
using namespace::std;
int s = 0;

void print(int x, int n=10, int y )
{
        cout << x << endl;
        cout << n << endl;
        cout << y << endl;
}

void main(void)
{
        print(2,3);
}
```

```
Compiler error. y is not defaulted.
```

Default Parameters

```
#include <iostream>
using namespace::std;
int s = 0;

void print(int x, int n=10,
{
        cout << x << endl;
        cout << n << endl;
        cout << y << endl;
}

void main(void)
{
        print(2,3);
}
```

Compiler Parsing

```
void print(int x, int n=10, int y);

void main(void)
{
        print(2,3);
}
```

```
Compiler error. y is not defaulted.
```

Default Parameters

```
#include <iostream>
using namespace::std;
int s = 0;

void print(int x, int n=10,
{
        cout << x << endl;
        cout << n << endl;
        cout << y << endl;
}

void main(void)
{
        print(2,3);
}
```

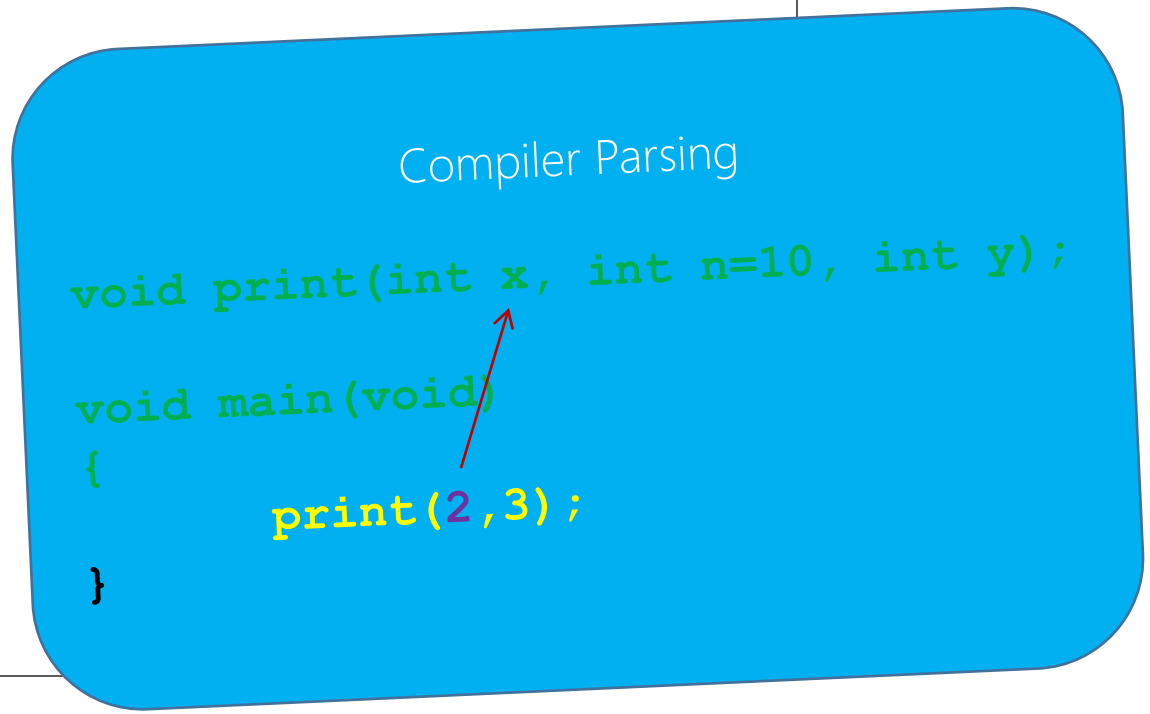
Compiler Parsing

```
void print(int x, int n=10, int y);

void main(void)
{
        print(2,3);
}
```

```
Compiler error. y is not defaulted.
```

Default Parameters

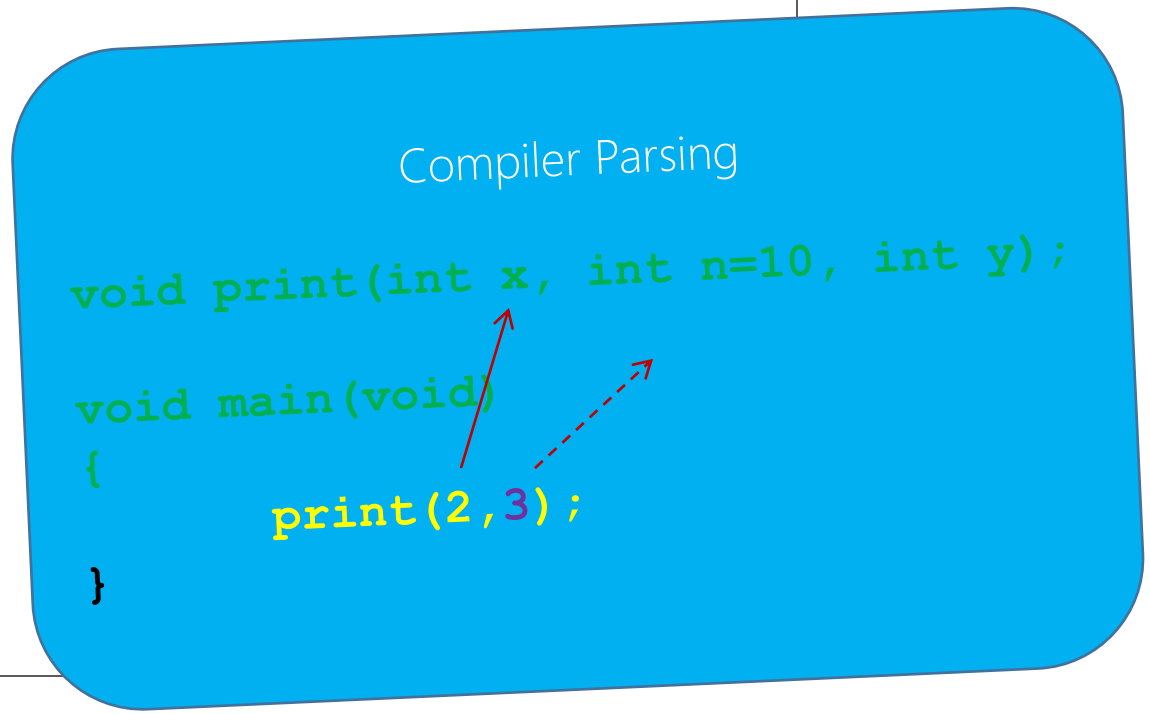
```
#include <iostream>
using namespace::std;
int s = 0;

void print(int x, int n=10,
{
        cout << x << endl;
        cout << n << endl;
        cout << y << endl;
}

void main(void)
{
        print(2,3);
}
```




```
Compiler error. y is not defaulted.
```

Default Parameters

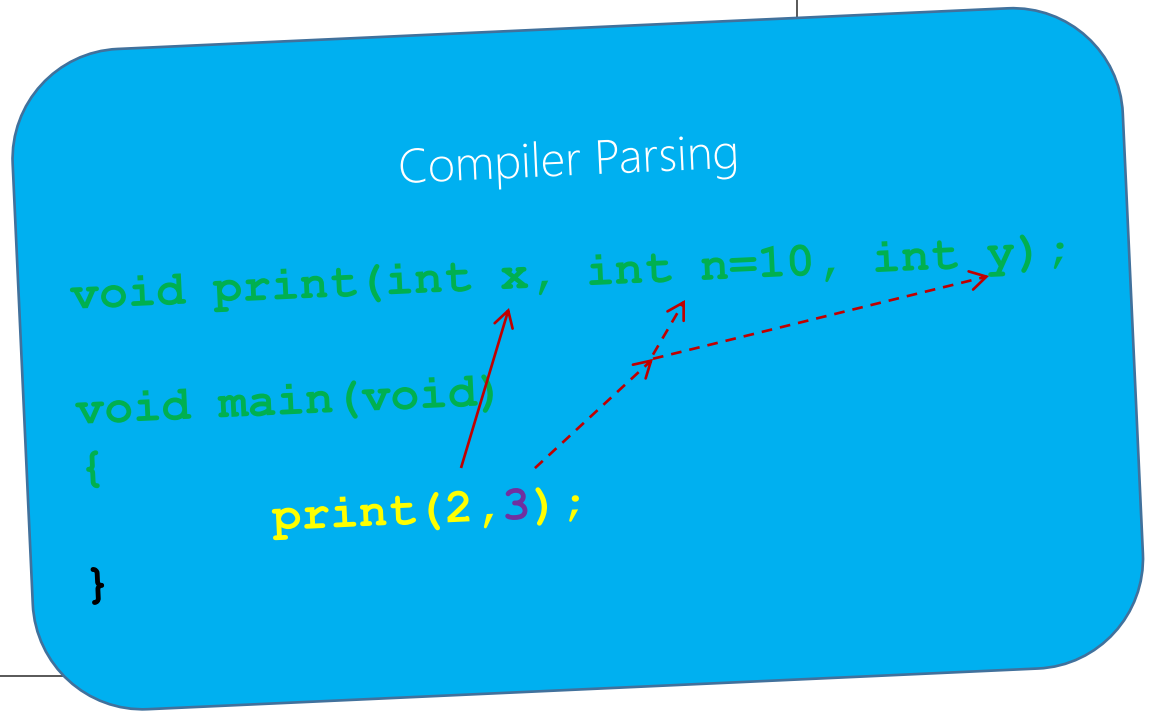
```
#include <iostream>
using namespace::std;
int s = 0;

void print(int x, int n=10,
{
        cout << x << endl;
        cout << n << endl;
        cout << y << endl;
}

void main(void)
{
        print(2,3);
}
```




```
Compiler error. y is not defaulted.
```

Default Parameters

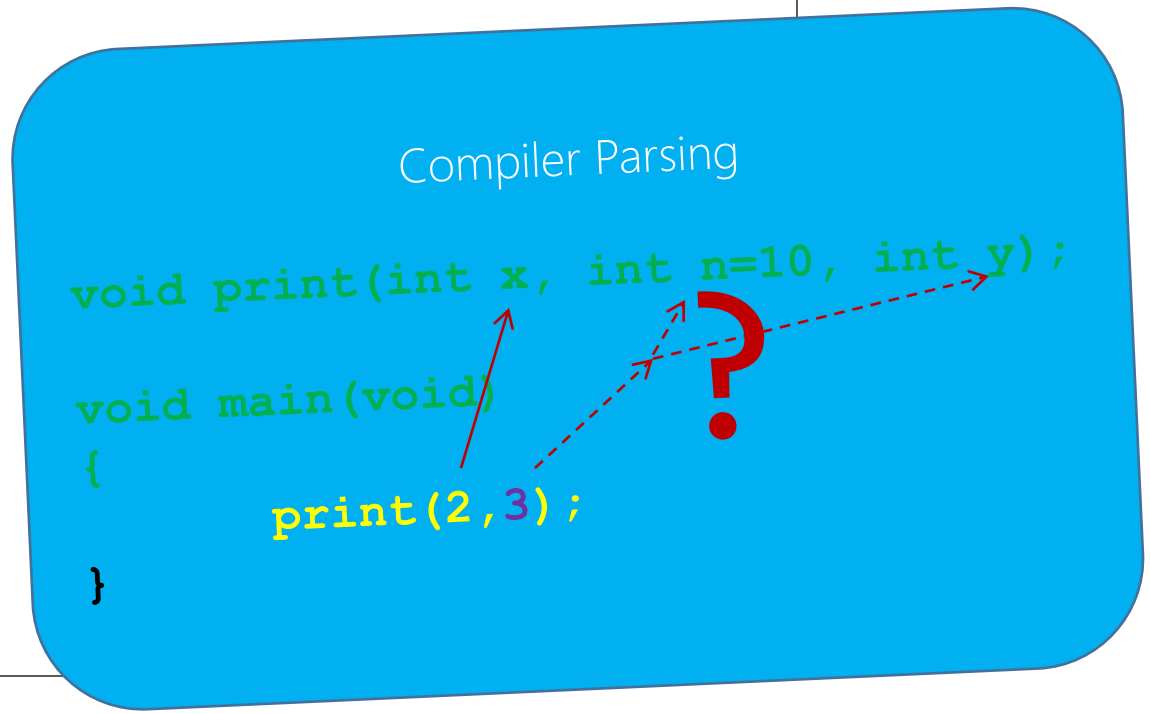
```
#include <iostream>
using namespace::std;
int s = 0;

void print(int x, int n=10,
{
        cout << x << endl;
        cout << n << endl;
        cout << y << endl;
}

void main(void)
{
        print(2,3);
}
```




```
Compiler error. y is not defaulted.
```

Default Parameters

```
#include <iostream>
using namespace::std;
int s = 0;

void print(int x, int n=10,
{
        cout << x << endl;
        cout << n << endl;
        cout << y << endl;
}

void main(void)
{
        print(2,3);
}
```

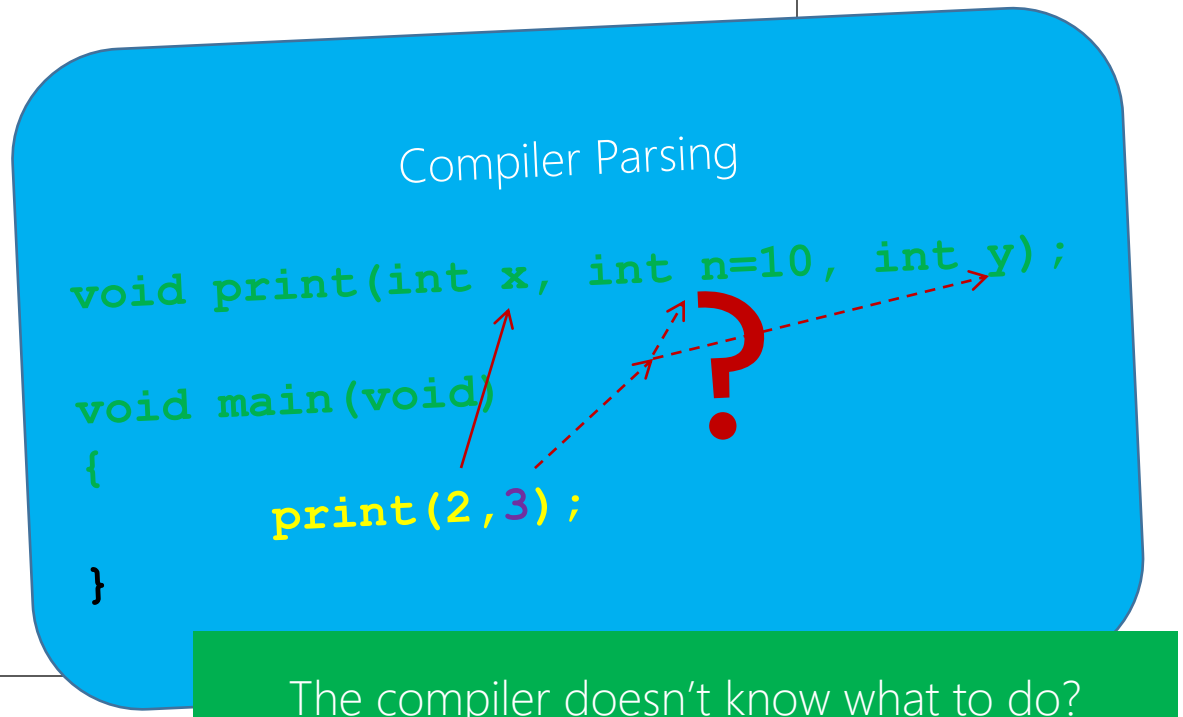



```
Compiler error. y is not defaulted.
```

Default Parameters

```
#include <iostream>
using namespace::std;
int s = 0;

void print(int x, int n=10,
{
        cout << x << endl;
        cout << n << endl;
        cout << y << endl;
}

void main(void)
{
        print(2,3);
}
```

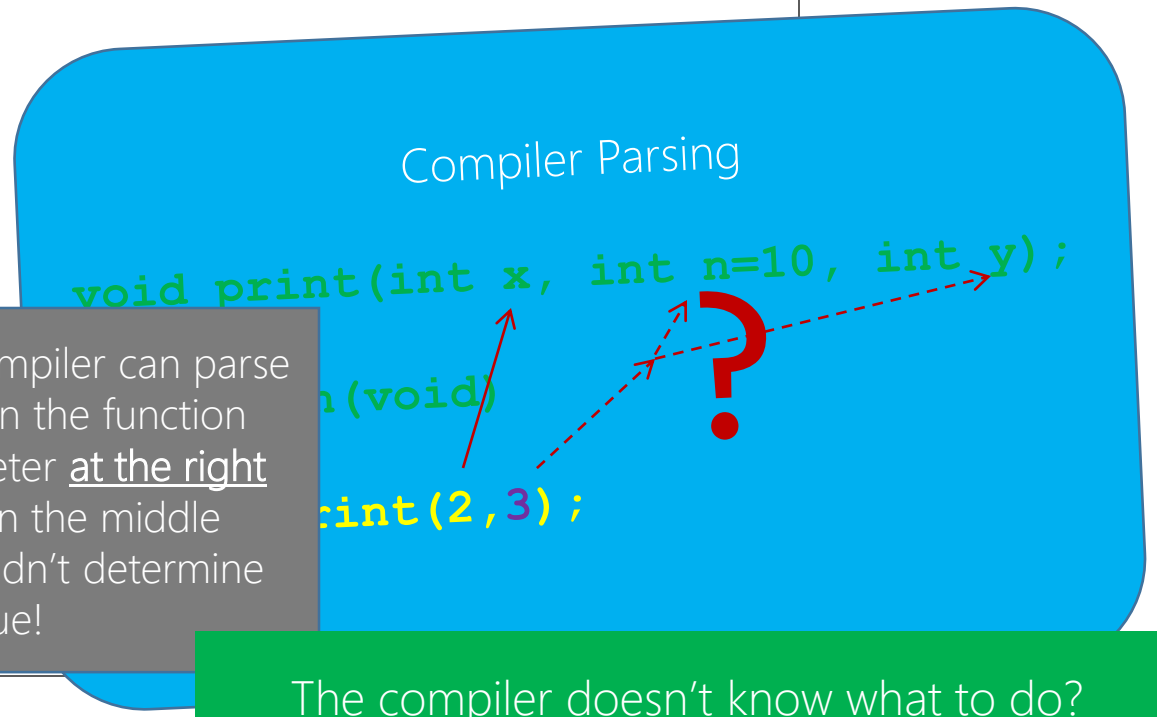
```
Compiler error. y is not defaulted.
```



The compiler doesn't know what to do? Should it replace the value (3) with the default value of the defaulted parameter (**n**) or go on to the next parameter (**y**) and pass it through

Default Parameters

```
#include <iostream>
using namespace::std;
int s = 0;

void print(int x, int n=10,
{
        cout << x << endl;
```



And **now** you can know **why** the compiler can parse the function calling correctly when the function header have the defaulted parameter <u>at the right most</u> of the prototype and not in the middle coz as you saw, the compiler couldn't determine where to pass the value!

```
Compiler error. y is not defaulted.
```

The compiler doesn't know what to do? Should it replace the value (3) with the default value of the defaulted parameter (**n**) or go on to the next parameter (**y**) and pass it through

Default Parameters

```
#include <iostream>
using namespace::std;
int s = 0;

void print(int x, int n=10, int y = 5 )
{
          cout << x << endl;
          cout << n << endl;
          cout << y << endl;
}

void main(void)
{
          print(2,3);
}
```

```
2
3
5
```

Calling by value
vs
calling by reference

Calling by value VS calling by reference

- Calling by value
  - Copy of data
  - Changes to copy don't affect original
  - Prevent an unwanted side effect

- Calling by reference (&)
  - Function directly access data
  - Changes affect original (no copy exist here!)
  - Using & after data type
    - `void foo(double & x)`

Calling by value VS calling by reference

```
#include <iostream>
using namespace::std;
int s = 0;

void FuncByValue (int x )
{
            cout << "In function before multiplication, x = "<< x << endl;
            x = x*4;
            cout << "In function after multiplication, x = " << x << endl;
}

void main(void)
{
            int x = 2;
            cout << "In main, before calling the function " << x << endl;
            FuncByValue(x);
            cout << "In main, after calling the function " << x << endl;

}
```

```
In main, before calling the function 2

In function before multiplication, x = 2

In function after multiplication, x = 8

In main, after calling the function 2

Press any key to continue
```

```
#include <iostream>
using namespace::std;
int s = 0;

void FuncByRef (int & x )
{
            cout << "In function before multiplication, x = "<< x << endl;
            x = x*4;
            cout << "In function after multiplication, x = " << x << endl;
}

void main(void)
{
            int x = 2;
            cout << "In main, before calling the function " << x << endl;
            FuncByRef (x);
            cout << "In main, after calling the function " << x << endl;

}
```

```
In main, before calling the function 2

In function before multiplication, x = 2

In function after multiplication, x = 8

In main, after calling the function 8

Press any key to continue
```

Calling by value VS calling by reference

```
#include <iostream>
using namespace::std;
int s = 0;

void FuncByRef (& int  x )
{
            cout << "In function before multiplication, x = "<< x << endl;
            x = x*4;
            cout << "In function after multiplication, x = " << x << endl;
}

void main(void)
{
            int x = 2;
            cout << "In main, before calling the function " << x << endl;
            FuncByRef (x);
            cout << "In main, after calling the function " << x << endl;

}
```

Compiler error, & before type

```
#include <iostream>
using namespace::std;
int s = 0;

void FuncByRef ( int  x & )
{
            cout << "In function before multiplication, x = "<< x << endl;
            x = x*4;
            cout << "In function after multiplication, x = " << x << endl;
}

void main(void)
{
            int x = 2;
            cout << "In main, before calling the function " << x << endl;
            FuncByRef (x);
            cout << "In main, after calling the function " << x << endl;

}
```

Compiler error, & after variable

Calling by value VS calling by reference

```
#include <iostream>
using namespace::std;
int s = 0;

int FuncByValue (  int x )
{
            cout << "In function before multiplication, x = "<< x << endl;
            x = x*4;
            cout << "In function after multiplication, x = " << x << endl;
            return x;
}

void main(void)
{
            int x = 2;
cout << "In main, before calling the function, x = " << x << endl;
cout << FuncByValue(x) << endl;
cout << "In main, after calling the function, x = " << x << endl;

}
```

```
In main, before calling the function, x = 2

In function before multiplication, x = 2

In function after multiplication, x = 8

8

In main, after calling the function, x = 2

Press any key to continue
```

```
#include <iostream>
using namespace::std;
int s = 0;

int FuncByRef (  int & x )
{
            cout << "In function before multiplication, x = "<< x << endl;
            x = x*4;
            cout << "In function after multiplication, x = " << x << endl;
            return x;
}

void main(void)
{
            int x = 2;
cout << "In main, before calling the function, x = " << x << endl;
cout << FuncByRef(x) << endl;
cout << "In main, after calling the function, x = " << x << endl;

}
```

```
In main, before calling the function, x = 2

In function before multiplication, x = 2

In function after multiplication, x = 8

8

In main, after calling the function, x = 8

Press any key to continue
```

Function Overloading

Function Overloading

- Is a function with
  - same name
  - different parameters
  - (Not the return type!)

- That means that the overloaded functions are distinguished in Signature
  - (Not the return type!)

Function Overloading

```
#include <iostream>
using namespace::std;
int s = 0;

int f (  int &x )
{
            x = x * 2;
            return x;
}

int f (  float &x )
{
            x = x * 3;
            return x;
}

void main(void)
{
            int y1 = 2;
            float y2 = 2;

            cout << f(y1) << endl;
            cout << f(y2) << endl;
}
```

```
4
6
```

Oveloading but with warning (casting)

```
#include <iostream>
using namespace::std;
int s = 0;

int f (  int &x )
{
            x = x * 2;
            return x;
}

float f (  float &x )
{
            x = x * 3;
            return x;
}

void main(void)
{
            int y1 = 2;
            float y2 = 2;

            cout << f(y1) << endl;
            cout << f(y2) << endl;
}
```

```
4
6
```

No warnings

Function Overloading

```
#include <iostream>
using namespace::std;
int s = 0;

int f (  int x )
{
            x = x * 2;
            return x;
}

float f (  float x )
{
            x = x * 3;
            return x;
}

void main(void)
{
            int y1 = 2;
            float y2 = 2;

            cout << f(y1) << endl;
            cout << f(y2) << endl;
}
```

```
4
6
```

```
#include <iostream>
using namespace::std;
int s = 0;

int f (  int x )
{
            x = x * 2;
            return x;
}

float f (  float x )
{
            x = x * 0;
            return x;
}

void main(void)
{
            int y1 = 2;
            float y2 = 2;

            cout << f(y1) << endl;
            cout << f(y2) << endl;
}
```

```
4
0
```

Function Overloading

```
#include <iostream>
using namespace::std;
int s = 0;

int f (  int x, int z)
{
            x = x * 2;
            cout << x << endl;
            cout << z << endl;
            return x;
}

float f (float x, int z)
{
            x = x * 3;
            cout << x << endl;
            cout << z << endl;
            return x;
}

void main(void)
{
            int y1 = 2;
            float y2 = 2;
            int z1 = 3, z2 = 2;

            f(y1,z1);
            f(y2,z2);
}
```

```
4
3
6
2
```

No warnings

```
#include <iostream>
using namespace::std;
int s = 0;

int f (  int x, int z)
{
            x = x * 2;
            cout << x << endl;
            cout << z << endl;
            return x;
}

float f (float x, int z)
{
            x = x * 3;
            cout << x << endl;
            cout << z << endl;
            return x;
}

void main(void)
{
            int y1 = 2;
            float y2 = 2;
            float z1 = 3, z2 = 2;

            f(y1,z1);
            f(y2,z2);
}
```

```
4
3
6
2
```

Warnings

Function Overloading

```
#include <iostream>
using namespace::std;
int s = 0;

int f (  int x, int z)
{
            x = x * 2;
            cout << x << endl;
            cout << z << endl;
            return x;
}

float f (int z, float x)
{
            x = x * 3;
            cout << x << endl;
            cout << z << endl;
            return x;
}

void main(void)
{
            int y1 = 2;
            float y2 = 2;
            int z1 = 3, z2 = 2;

            f(y1,z1);
            f(y2,z2);
}
```

```
4
3
4
2
```

Warnings

```
#include <iostream>
using namespace::std;
int s = 0;

int f (  int x, int z)
{
            x = x * 2;
            cout << x << endl;
            cout << z << endl;
            return x;
}

float f (int z, float x)
{
            x = x * 3;
            cout << x << endl;
            cout << z << endl;
            return x;
}

void main(void)
{
            int y1 = 2;
            float y2 = 2;
            float z1 = 3, z2 = 2;

            f(y1,z1);
            f(y2,z2);
}
```

```
9
2
6
2
```

Warnings

Function Overloading

```
#include <iostream>
using namespace::std;
int x = 0;

float f2 (int z )
{
            cout << z << endl;
            z*=2;
            return z;
}

void f1 (  int &x )
{
            x+2;
            cout << x << endl;
            f2(x);
            cout << x << endl;
}


void main(void)
{
            f1(::x++);
            cout << ++::x << endl;
}
```

Compiler error Can't convert from int to &int

```
#include <iostream>
using namespace::std;
int x = 0;


void f1 (  int &x )
{
            x+2;
            cout << x << endl;
            f2(x);
            cout << x << endl;
}

float f2 (int z )
{
            cout << z << endl;
            z*=2;
            return z;
}


void main(void)
{
            f1(::x);
            cout <<::x << endl;
}
```

Compiler error. f1 can't know what f2 is!

Quiz

Quiz 1

```
#include <iostream>

using namespace::std;
int x = 34;

void foo()
{
            static int x = 3;       // initialized only ONE time
            x = 5;
            cout << "first x in foo = " << x <<  endl;
            x++;
            cout << "second x in foo = " << x <<  endl;
}
```

```
void main(void)
{
            int x = 9;
            cout << x << endl;
                        {
            {
                        int x = 5;
                        cout << x++ << endl;
                        }
            cout << x << endl;
            ::x = x+2;
                        }
            cout << x << endl;
            {
            foo();
            foo();}
            cout << x << endl;
            cout <<::x << endl;
}
```

```
9
5
9
9
first x in foo = 5
second x in foo = 6
first x in foo = 5
second x in foo = 6
9
11
Press any key to continue
```

Quiz 2

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

void TryMe();

void main()
{
	cout << "In main" << endl;
	TryMe();
}

void TryMe()
{
	cout << "In function" << endl;
	main();
}
```

```
In main
In function
In main
In function
In main
In function
In main
In function

In main
In function
```

Quiz 3

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

void TryMe()
{
      cout << "In function" << endl;
      main();
}

void main()
{
     TryMe();
      cout << "In main" << endl;
}
```

Compiler error!

`TryMe()` can't know where the main function is