

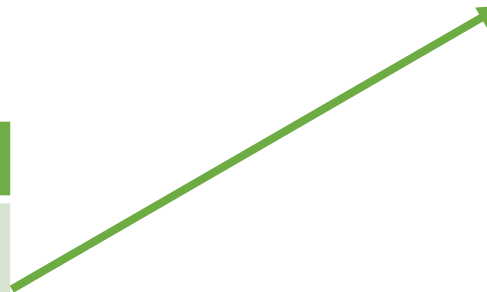
Заголовочные файлы (*.h)

*Предварительное объявление
функции add (прототип)*

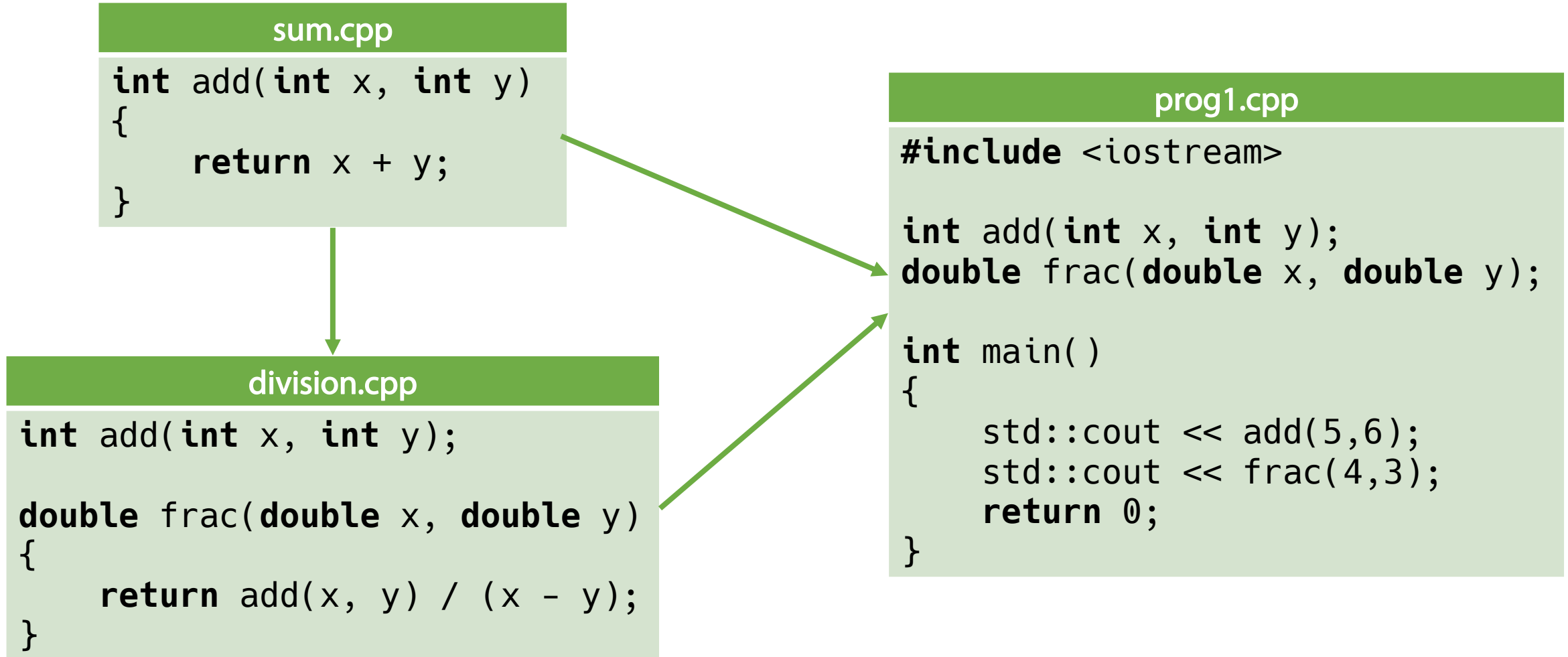


```
sum.cpp
int add(int x, int y)
{
    return x + y;
}
```

```
prog1.cpp
#include <iostream>
int add(int x, int y);
int main()
{
    std::cout << add(5,6);
    return 0;
}
```



Заголовочные файлы (*.h)



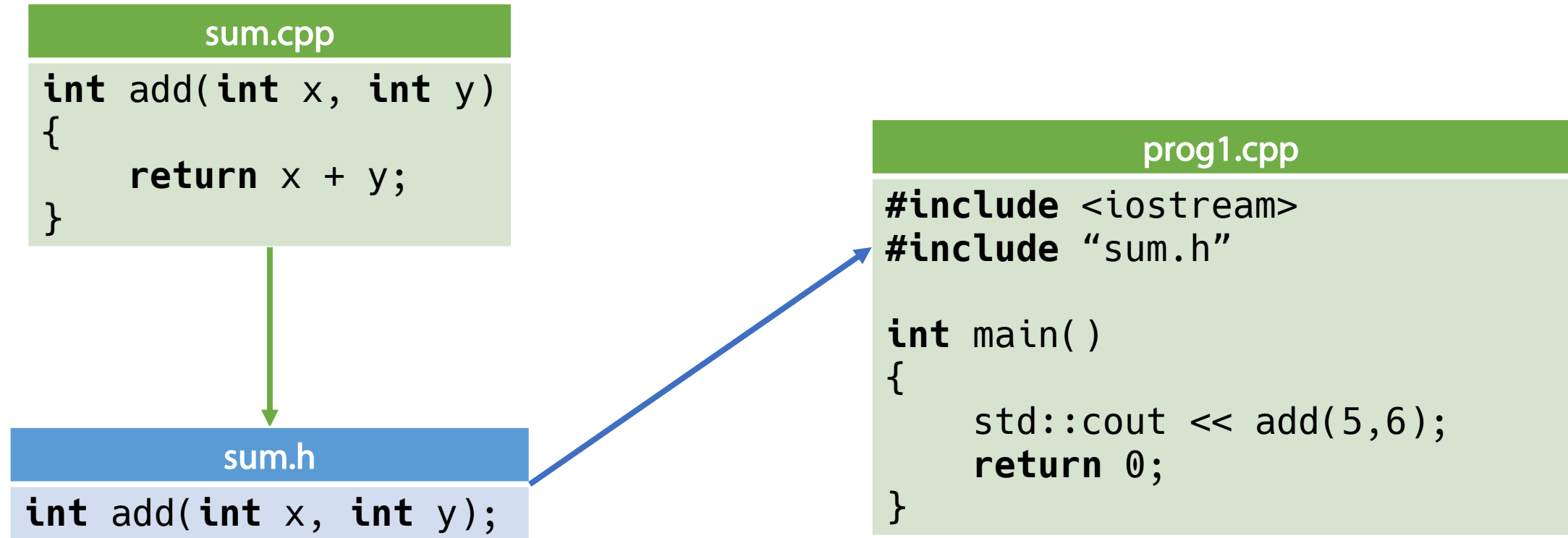
Заголовочные файлы (*.h)

sum.cpp

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

sum.h

```
int add(int x, int y);
```



prog1.cpp

```
#include <iostream>
#include "sum.h"

int main()
{
    std::cout << add(5,6);
    return 0;
}
```

Заголовочные файлы (*.h)

Проблема дублирования объявлений

m1.h

```
int square(int a, int b)
{
    return a * b;
}
```

m2.h

```
#include "m1.h"
```

prog1.cpp

```
#include <iostream>
#include "m1.h"
#include "m2.h"

int main( )
{
    std::cout << "square = ";
    std::cout << square(5,6);
    return 0;
}
```

Заголовочные файлы (*.h)

Проблема дублирования объявлений – **header guards**

```
m1.h
#pragma once
int square(int a, int b)
{
    return a * b;
}
```

```
m2.h
#include "m1.h"
```

```
prog1.cpp
#include <iostream>
#include "m1.h"
#include "m2.h"

int main()
{
    std::cout << "square = ";
    std::cout << square(5,6);
    return 0;
}
```

Заголовочные файлы (*.h)

- Использовать директивы препроцессора
- Избегать определения переменных и функций
- Придерживаться соответствия имен заголовочных файлов и файлов исходного кода (`sum.h` ↔ `sum.cpp`)
- Избегать подключения одних заголовочных файлов из других
- Не подключать файлы исходного кода с помощью `#include`

Пространства имен (namespaces)

Конфликт имен

sumr.h

```
int doIt(int a, int b)
{
    return a + b;
}
```

subtracr.h

```
int doIt(int a, int b)
{
    return a - b;
}
```

prog1.cpp

```
#include "sumr.h"
#include "subtracr.h"

int main()
{
    int sum = doIt(5, -8);
    return 0;
}
```

Какую версию вызвать?

Пространства имен (namespaces)

sumr.h

```
namespace sumr {  
    int doIt(int a, int b)  
    {  
        return a + b;  
    }  
}
```

subtrctr.h

```
namespace subtrctr {  
    int doIt(int a, int b)  
    {  
        return a - b;  
    }  
}
```

prog1.cpp

```
#include "sum.h"  
#include "subtract.h"  
  
int main()  
{  
    int s = doIt(5,-8);  
    return 0;  
}
```

Без указания пространства имен компилятор не находит определение doIt

Пространства имен (namespaces)

sumr.h

```
namespace sumr {  
    int doIt(int a, int b)  
    {  
        return a + b;  
    }  
}
```

subtract.h

```
namespace subtractr {  
    int doIt(int a, int b)  
    {  
        return a - b;  
    }  
}
```

prog1.cpp

```
#include "sum.h"  
#include "subtract.h"  
  
int main()  
{  
    int s;  
    s = sumr::doIt(5, -8);  
    return 0;  
}
```

Доступ к функции doIt из пространства имен sumr с помощью оператора разрешения области видимости ::

Пространства имен (namespaces)

sumr.h

```
namespace sumr {  
    int doIt(int a, int b)  
    {  
        return a + b;  
    }  
}
```

subtract.h

```
namespace subtractr {  
    int doIt(int a, int b)  
    {  
        return a - b;  
    }  
}
```

prog1.cpp

```
#include "sum.h"  
#include "subtract.h"
```

```
using sumr::doIt;
```

```
int main()  
{  
    int s;  
    s = doIt(5, -8);  
    return 0;  
}
```

using-объявление говорит, что используется doIt из пространства sumr

Пространства имен (namespaces)

sumr.h

```
namespace sumr {  
    int doIt(int a, int b)  
    {  
        return a + b;  
    }  
}
```

subtract.h

```
namespace subtractr {  
    int doIt(int a, int b)  
    {  
        return a - b;  
    }  
}
```

prog1.cpp

```
#include "sum.h"  
#include "subtract.h"
```

```
using namespace sumr;
```

```
int main()  
{  
    int s;  
    s = doIt(5, -8);  
    return 0;  
}
```

using-директива говорит, что должны быть подключены все имена из пространства sumr

Пространства имен (namespaces)

- Пространство имен – это область кода, внутри которой гарантируется уникальность используемых идентификаторов
- Пространство имен может быть описано в нескольких файлах или в разных местах одного файла
- Пространства имен могут быть вложены друг в друга, но этого лучше избегать