Код:

// main.cpp

#include <iostream>

int hello(int x);

int main() {

int result = hello(5);

std::cout << "Result: " << result << std::endl;

return 0;

}

// f.cpp

int hello(int x);

Получим ошибку "multiple definition of 'hello'", так как компилятор видит два разных объявления для функции hello, но не видит их определения.

// f.cpp

int hello(int x) {

return x \* 2;

}

Компиляция пройдет успешно

// f.cpp

static int hello(int x) {

return x \* 2;

}

Получим ошибку "undefined reference to 'hello'", потому что static делает функцию видимой только в пределах файла, и компилятор не может найти определение функции hello в main.cpp.

// main.cpp

#include <iostream>

static int hello(int x);

int main() {

int result = hello(5);

std::cout << "Result: " << result << std::endl;

return 0;

}

Теперь компиляция пройдет успешно, так как функция hello с модификатором static является локальной для каждого файла и не конфликтует между файлами.

В финальном исполняемом файле будет только одна копия тела функции hello, потому что каждый файл (main.cpp и f.cpp) содержит свою собственную локальную копию функции hello.

static\_1 and static\_2

В обоих примерах static 1 и static 2 используется статическая переменная в функции staticMemory(). Однако, существует различие в области видимости переменных и их связывании:

static 1:

В данном случае переменная a имеет локальную область видимости внутри функции staticMemory(). Она инициализируется только один раз при первом вызове функции и сохраняет свое значение между вызовами. При этом, она не видна за пределами функции. Это называется локальной статической переменной.

static 2:

Здесь переменная a объявлена в глобальной области видимости (вне функций). Она имеет глобальную область видимости и связывание. Такая переменная видна во всех функциях данного файла (или, если объявлена с ключевым словом extern, в других файлах компиляции). Это глобальная переменная.

Чтобы сделать static 2 эквивалентным static 1 в плане невозможности импортировать переменную в другом файле, вы можете сделать переменную a в static 2 статической (что ограничит её область видимости только для данного файла) и добавить ключевое слово static:

// static2.cpp

#include <iostream>

static int a = 0; // Теперь a является локальной статической переменной

void staticMemory()

{

std::cout << "a: " << a << std::endl;

a += 1;

}

int main()

{

staticMemory(); // prints 0

staticMemory(); // prints 1

staticMemory(); // prints 2

return 0;

}