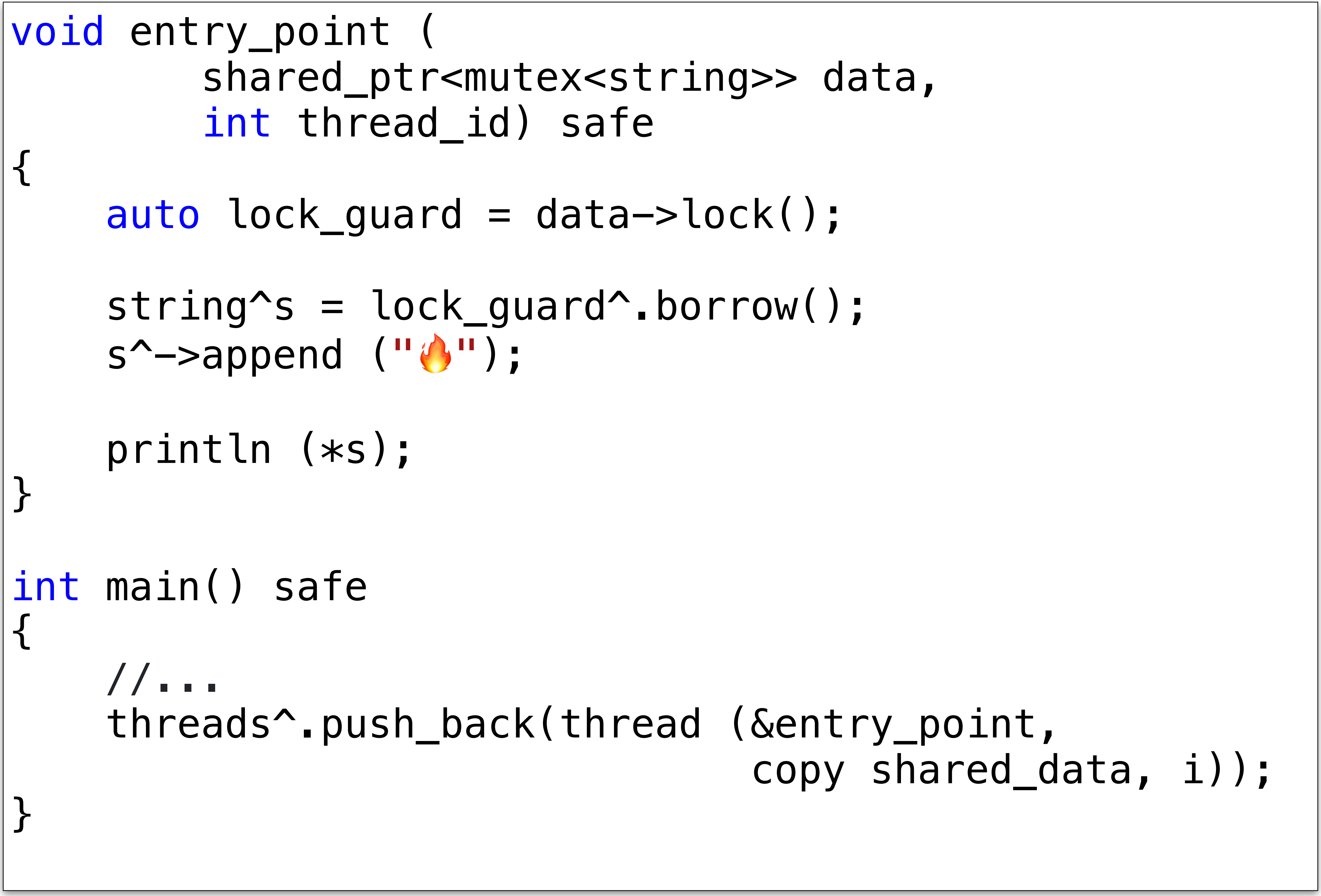
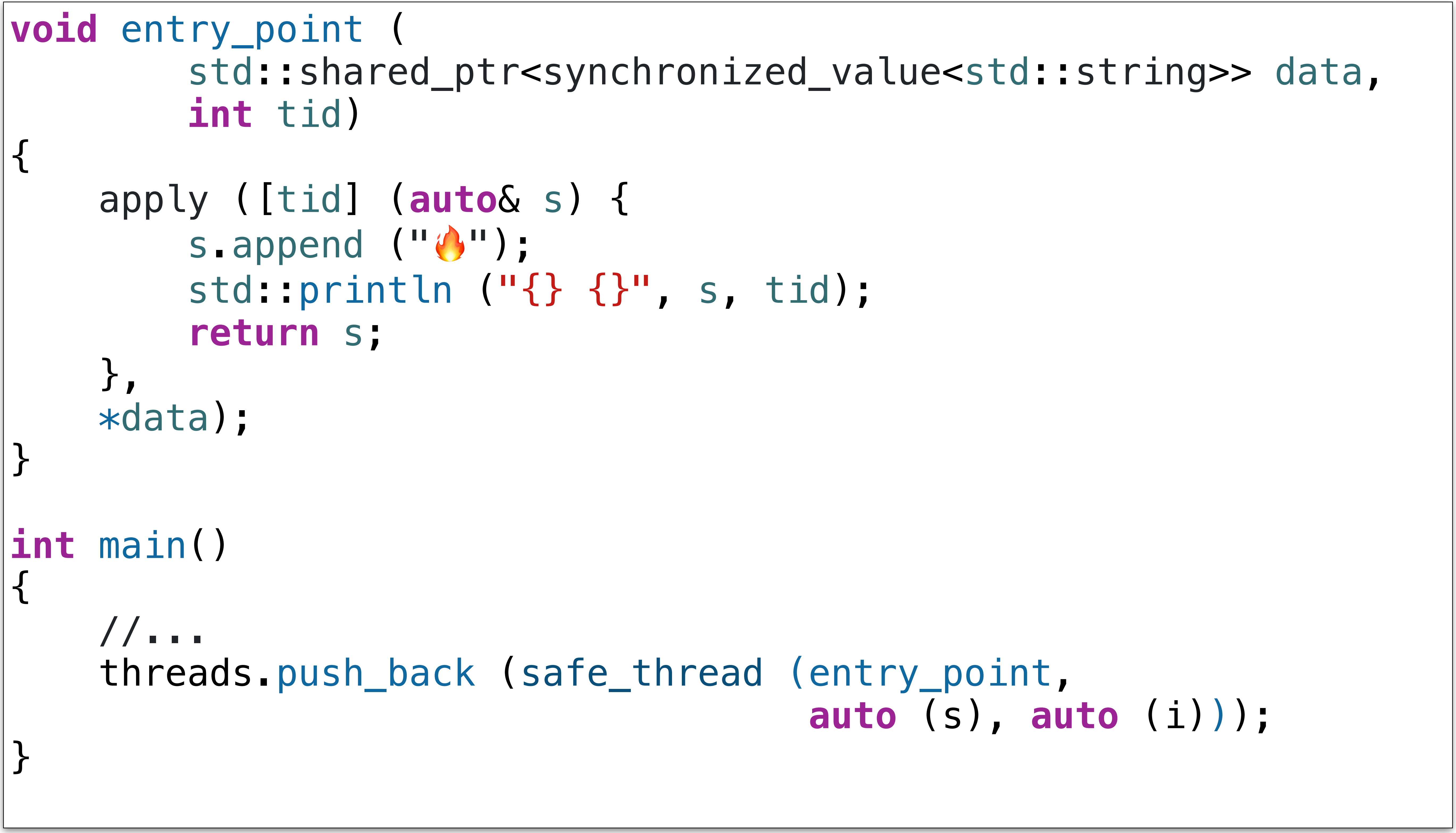
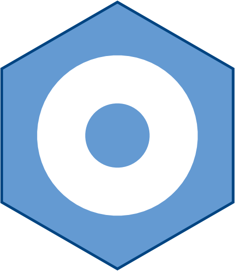
102



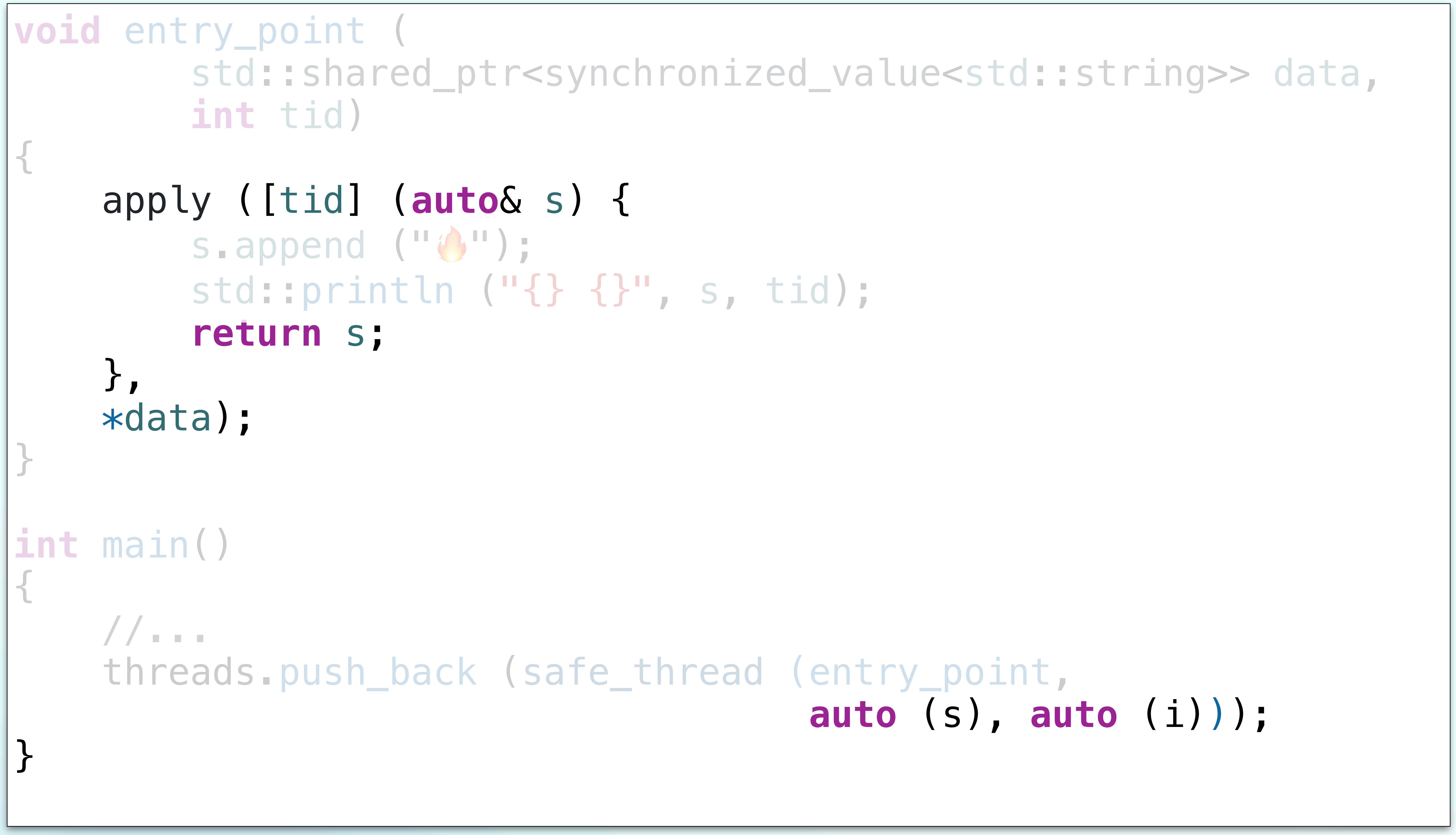
```
void entry_point (
        shared_ptr<mutex<string>> data,
        int thread_id) safe
{
    auto lock_guard = data->lock();

    string^s = lock_guard^.borrow();
    s^->append ("🔥");

    println (*s);
}

int main() safe
{
    //...
    threads^.push_back(thread (&entry_point,
                               copy shared_data, i));
}
```



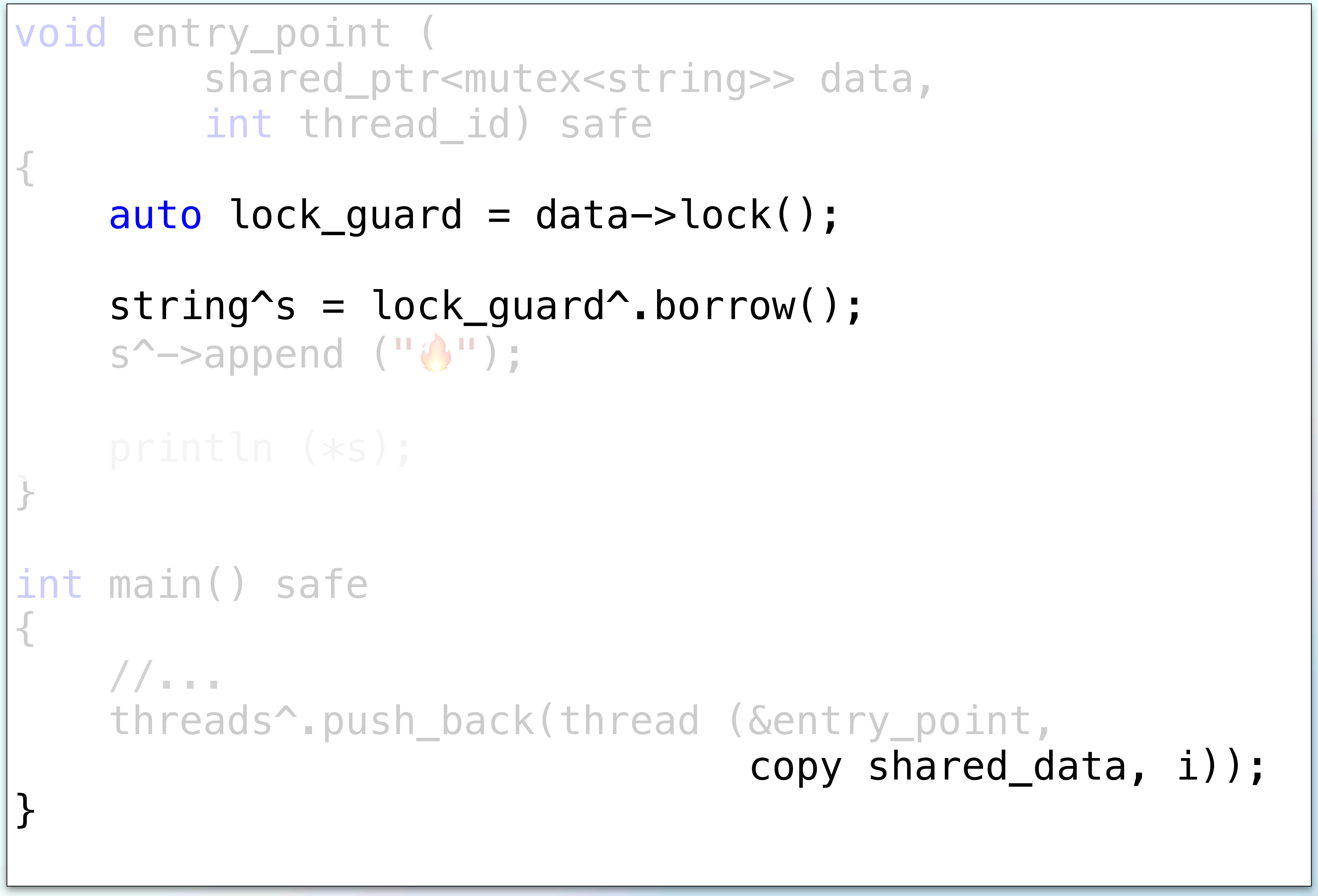
```cpp
void entry_point (
        std::shared_ptr<synchronized_value<std::string>> data,
        int tid)
{
    apply ([tid] (auto& s) {
        s.append ("🔥");
        std::println ("{} {}", s, tid);
        return s;
    },
    *data);
}

int main()
{
    //...
    threads.push_back (safe_thread (entry_point,
                                    auto (s), auto (i)));
}
```



C++



```
void entry_point (
        std::shared_ptr<synchronized_value<std::string>> data,
        int tid)
{
    apply ([tid] (auto& s) {
        s.append ("🔥");
        std::println ("{} {}", s, tid);
        return s;
    },
    *data);
}

int main()
{
    //...
    threads.push_back (safe_thread (entry_point,
                                    auto (s), auto (i)));
}
```



```
void entry_point (
        shared_ptr<mutex<string>> data,
        int thread_id) safe
{
    auto lock_guard = data->lock();

    string^s = lock_guard^.borrow();
    s^->append ("🔥");

    println (*s);
}

int main() safe
{
    //...
    threads^.push_back(thread (&entry_point,
                               copy shared_data, i));
}
```

C++ Reflection to the Rescue?

- **Recursive Sync/Send Type Trait Checking**