Промежуточный контроль 2

Источник знаний

Энтони Уильямс

Глава 3

Разделение данных между потоками

Вопрос 1

Одно из основных достоинств применения потоков для реализации параллелизма – …

Вопрос 2

Одна из основных причин ошибок, связанных с параллелизмом, – …

Вопрос 3

Все проблемы разделения данных между потоками связаны с …

Вопрос 4

Инвариант данных – это …

Вопрос 5

В параллельном программировании под состоянием гонки понимается …

Вопрос 6

Своей сложностью параллельные программы в немалой степени обязаны …

Вопрос 7

Существует несколько способов борьбы с проблематичными гонками. Перечислить

Вопрос 8

Самый простой механизм защиты разделяемых данных в C++11 – …

Вопрос 9

В C++ для создания мьютекса следует сконструировать объект типа `std::mutex`, для захвата мьютекса служит функция-член `lock()`, а для освобождения – функция-член `unlock()`. Однако вызывать эти функции напрямую не рекомендуется, потому что в этом случае

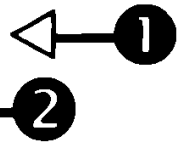
...

И что делать?

Вопрос 10

```
std::list<int> some_list;
std::mutex some_mutex;

void add_to_list(int new_value)
{
      std::lock_guard<std::mutex> guard(some_mutex);
      some_list.push_back(new_value);
}
```



Хотя иногда такое использование глобальных переменных уместно, в большинстве случает мьютекс и защищаемые им данные …

Вопрос 11

Определите проблему в коде

```
class some_data
{
      int a;
      std::string b;
public:
      void do_something();
};

class data_wrapper
{
private:
      some_data data;
      std::mutex m;
public:
      template<typename Function>
      void process_data(Function func)
      {
            std::lock_guard<std::mutex> l(m);
            func(data);
      }
};
```

Вопрос 12

Приведите пример, демонстрирующий «состояние гонки, внутренне присущее интерфейсам»

Вопрос 13

Что понимается под гранулярностью блокировки

Вопрос 14

Какая проблема! Решение

```
class X
{
private:
      some_big_object some_detail;
      std::mutex m;
public:
      X(some_big_object const& sd):some_detail(sd){}
      friend void swap(X& lhs, X& rhs)
      {
         if(&lhs==&rhs)
               return;

           ...
```

Вопрос 15

3.2.5. Дополнительные рекомендации, как избежать взаимоблокировок

Вопрос 16

На что влияет гранулярность блокировки

Вопрос 17

Что такое отложенная (ленивая) инициализация?

Решения для многопоточной реализации

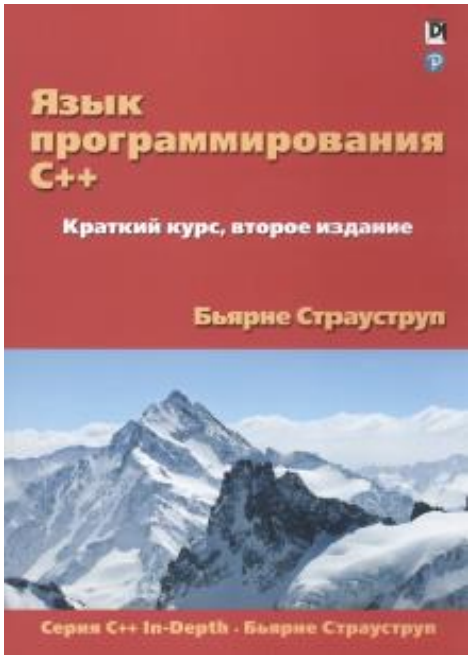
Вопрос 18

Рекурсивная блокировка:

- Что
- Зачем
- Правила использования

Happy End!!!

Бьярне Страуструп



Страуструп, Бьярне.

Язык программирования C++. Краткий курс, 2-е изд. : Пер. с англ. -

СПб.: ООО "Диалектика", 2019. - 320 с.

Бьярне Страуструп

Оглавление

15. 7. Обмен информацией с заданиями

Стандартная библиотека предоставляет несколько средств, позволяющих программистам работать на концептуальном уровне заданий (потенциально работающих параллельно), а не непосредственно на низком уровне потоков и блокировок.

- ***future*** и ***promise*** для возврата значения из задания, запущенного в отдельном потоке.
- ***packaged_task*** для помощи в запуске задач и подключении механизмов для возврата результата.
- ***async*** *()* для запуска заданий способом, очень похожим на вызов функции.

Все эти средства находятся в заголовочном файле <**future**>.