Программная инженерия

Московский Физико-Технический Институт

Автор курса: к.т.н. Иван Сергеевич Макаров

Контакты: Telegram или i.s.m.mipt@yandex.ru

Оглавление

01 Общее введение и обзор технологий

01.01 [100]

Напишите программу, которая выводит в поток вывода любую строку и при этом обладает функцией `main` с пустым телом. Предложите несколько решений. Впервые я столкнулся с этой задачей на собеседовании в одну крайне назойливую компанию. Для ее решения Вам потребуются определенные средства языка, которые будут рассматриваться позже. Сейчас Вы можете пропустить эту задачу и вернуться к ней спустя некоторое время. Возможно, Вы немного удивились тому, что первая же задача данного кодекса обладает настолько неадекватным уровнем сложности. Это своего рода дань памяти моему детству. Я начал серьезно изучать языки программирования в 12 лет, когда проводил летние школьные каникулы на даче у бабушки с дедушкой. Родители подарили мне две книги: Программирование – принципы и практика с использованием C++ Бьёрна Страуструпа и Язык программирования C Брайана Кернигана и Денниса Ритчи. Также у меня имелся ноутбук со средой разработки Code::Blocks, однако не было ни интернета, ни даже мобильной связи, потому что дача находится в низине, а сеть ловит только на определенном тайном холмике в лесу. Я решил начать с тонкой книги Кернигана, быстро проработать ее, а потом приступить к кирпичу Страуструпа. Опрометчивое решение! В одном из первых заданий просили написать программу, которая удаляла все комментарии из кода другой программы. Наверное, это очень сложно на третий день изучения программирования, но я справился, потому что из-за отсутствия связи с внешним миром я просто не знал, что это сложно. Возможно, именно этот случай помог мне определиться с основным направлением своей жизнедеятельности. Любопытно, что случилось бы со мной, если бы мне тогда попалалась монография Искусство программирования Дональда Кнута? Возможно, я стал бы лучше относиться к математике. Надо будет провести такой эксперимент над собственными детьми.

02 Структурное программирование

02.01 [20]

Реализуйте алгоритм классификации введенных пользователем символов. Выделите четыре следующих класса: десятичная цифра – 10 символов, арифметический оператор – 4 символа, круглая скобка – 2 символа, остальные символы. Используйте ветвление типа `switch` с проваливанием для избежания дублирования кода.

02.02 [40]

Реализуйте алгоритм вычисления всех чисел Армстронга, всех чисел Фибоначчи, всех избыточных чисел и всех дружественных чисел, меньших числа N, заданного пользователем. Рассматривайте оптимальные способы.

02.03 [25]

Реализуйте алгоритм вычисления всех простых множителей некоторого числа, заданного пользователем.

02.04 [40]

Реализуйте алгоритм Евклида в рекурсивном и нерекурсивном виде для вычисления наибольшего общего делителя. Также дополнительно реализуйте алгоритм вычисления наименьшего общего кратного двух чисел. Проверьте правильность работы Ваших реализаций, сравнив их результаты с результатами, полученными в результате использования численных алгоритмов `std::gcd` и `std::lcm` из стандартной библиотеки `numeric`.

02.05 [25]

Реализуйте алгоритм вычисления числа e через сумму ряда Маклорена при `x = 1`. Используйте числа с плавающей точкой типа `double`. Оптимизируйте вычисление членов ряда таким образом, чтобы использовалась только операция деления. Умножение при вычислении факториала может привести к переполнению. Завершайте вычисление, когда очередной член ряда окажется меньше некоторой заданной константы `epsilon`. Сравните Ваше получившееся значение с константой `std::numbers::e` из стандартной библиотеки `numbers`.

02.06 [25]

Реализуйте алгоритм генерации последовательностей Коллатца для всех целых чисел от `1` до `10000` включительно, который также определит наибольшую из них и выведет ее длину и начальное значение. Гипотеза Коллатца – одна из нерешенных проблем современной математики. Возьмем любое положительное целое число `N`. Если число четное, то определим следующий элемент последовательности как `N / 2`, в противном случае определим его как `3 * N + 1`. Утверждается, что вычисляемая таким образом последовательность всегда достигает значения `1`. Оптимизируйте вычисления, используя кэширование проверенных последовательностей.

02.07 [10]

Сформулируйте способ вычисления количества правильных скобочных последовательностей круглых скобок.

02.08 [25]

Реализуйте алгоритм генерации списка инструкций для рекурсивного решения задачи о Ханойских башнях.

02.09 [25]

Реализуйте алгоритм сортировки выбором контейнера `std::vector` целочисленных значений через индексы.

02.10 [25]

Реализуйте алгоритм сортировки Шелла контейнера `std::vector` целочисленных значений через индексы.

02.11 [25]

Реализуйте алгоритм сортировки кучей контейнера `std::vector` целочисленных значений через индексы.

02.12 [25]

Реализуйте алгоритм быстрой сортировки контейнера `std::vector` целочисленных значений через индексы.

02.13 [25]

Реализуйте алгоритм удаления дубликатов из отсортированного по возрастанию контейнера `std::vector` целочисленных значений. В качестве основы используйте приведенную в справочнике реализацию алгоритма `std::unique`, однако не используйте итераторы и семантику перемещения с `std::move`, используйте индексы и копирование. Обратите внимание, данный алгоритм не удаляет дубликаты, он лишь перемещает в начало уникальные элементы и возвращает позицию конца. Постарайтесь реализовать максимально компактный код.

02.14 [25]

Реализуйте алгоритм обращения порядка элементов контейнера `std::vector` целочисленных значений. Используйте функцию `std::swap` для почленного обмена местами элементов с двух противоположных концов.

03 Объектно-ориентированное программирование

03.01 [25]

Реализуйте структуру `Rectangle`, хранящую целочисленные координаты левого нижнего угла прямоугольника и две длины его сторон. Будем предполагать, что все подобные прямоугольники имеют стороны, параллельные координатным осям. Реализуйте функцию, которая будет принимать на вход контейнер `std::vector` экземпляров структуры `Rectangle` и вычислять площадь их пересечения. Оно может быть пустым или вырожденным.

03.02 [30]

Реализуйте классы `Triangle`, `Square` и `Circle`, описывающие геометрические фигуры треугольник, квадрат и окружность. Вы должны реализовать 3 независимых класса. В качестве приватных данных-членов рассматривайте только те, которые необходимы для вычисления периметра и площади фигур. Реализуйте необходимые конструкторы и публичные функции-члены, например, геттеры. Реализуйте публичные функции-члены для вычисления периметра и площади каждой фигуры. Для числа Пи в классе окружносте создайте статическую константу. Убедитесь, что Вы рассмотрели все актуальные для этой задачи аспекты проектирования классов.

03.03 [50]

Реализуйте иерархию классов, описывающих геометрические фигуры. Реализуйте классы, описывающие прямоугольник, квадрат, треугольник, эллипс и окружность. Реализуйте наследование квадрата от прямоугольника, окружности от эллипса, а для треугольника и прямоугольника реализуйте дополнительный промежуточный базовый класс многоугольника. Реализуйте во всех классах необходимые конструкторы. В качестве данных-членов добавляйте только те, которые необходимы для вычисления периметра и площади фигур. Например, для эллипса это две полуоси, для прямоугольника две стороны, для треугольника три стороны. Не нужно создавать отдельные данные-члены для радиуса окружности и стороны квадрата. Используйте наследование интерфейса и делегирующие конструкторы. Реализуйте две ветви виртуальных функций-членов – одну для вычисление площади, вторую для вычисление периметра фигур. Не создавайте собственную константу для числа Пи на этот раз. Воспользуйтесь существующей константой `std::numbers::pi` из стандартной библиотеки `numbers`. Реализуйте абстрактный базовый класс `Shape` фигур с интерфейсом чисто виртуальных функций.

03.04 [25]

Продемонстрируйте способ индивидуального замещения в производном классе определений для виртуальных функций с одинаковыми сигнатурами из двух разных базовых классов. Здесь предполагается, что есть два базовых класса с виртуальными функциями, которые имеют одинаковые сигнатуры, но при этом выполняют разные действия. Производный класс наследуется в режиме множественного наследования от этих двух базовых классов. Если в производном классе реализовать обычное замещение для одной из этих виртуальных функций, то автоматически заместится и вторая. Данная проблема также называется проблемой сиамских близнецов и может быть решена посредством внедрения промежуточных классов-посредников в эту иерархию.

03.05 [20]

Продемонстрируйте преобразование типов с помощью оператора `dynamic_cast` в иерархии классов с множественным наследованием. Рассмотрите различные удачные и неудачные случаи down-casting-а и side-casting-а.

03.06 [50]

Реализуйте класс `Container`, который будет являться высокоуровневой оберткой вокруг встроенного динамического массива. Используйте в качестве основы рассмотренную реализацию, в которой реализованы необходимые данные-члены и специальные функции-члены класса. Добавьте в класс дополнительное поле `capacity`, чтобы хранить количество выделенных ячеек памяти. Поле `size` при этом остается для хранения количества элементов массива, т.е. количества занятых ячеек памяти. Реализуйте константную и неконстантную версии функций-членов `front` и `back`, предоставляющих доступ к первому и последнему элементу массива соответственно. Реализуйте константную и неконстантную версии оператора доступа по индексу. Здесь можно не беспокоиться о дублировании кода, реализации будут тривиальными. Реализуйте геттеры для получения текущих значений полей `size` и `capacity`. Реализуйте функцию-член `empty`, проверяющую, является ли массив пустым. Реализуйте функцию-член `clear`, удаляющую все элементы массива, но не освобождающую выделенную память. Реализуйте функцию-член `push_back`, добавляющую новый элемент в первую свободную ячейку памяти внутреннего динамического массива. Тщательно продумайте последовательность действий на тот случай, когда свободных ячеек памяти нет, т.е. когда значения полей `size` и `capacity` равны. Выделяйте дополнительную память оптимальными способами, минимизируя количество операций с диспетчером памяти.

03.07 [25]

Реализуйте класс `IPv4`, представляющий собой сетевые IP адреса в формате IPv4. Реализуйте перегруженные операторы ввода-вывода для экземпляров данного класса. Пользователь должен иметь возможность вводить в консоли компоненты сетевого адреса через точку. Аналогично сетевые адреса должны выводиться в консоль.

03.08 [25]

Доработайте Ваше предыдущее решение задачи `03.07` так, чтобы IP адреса можно было сравнивать, а также переходить к следующему или предыдущему в лексикографическом порядке с помощью перегруженных операторов инкремента и декремента соответственно. Продемонстрируйте цикл перебора возможных адресов.

04 Обобщенное программирование

04.01 [25]

Доработайте пример 02.11 так, чтобы тип обрабатываемых элементов контейнера мог быть произвольным.

04.02 [25]

Доработайте пример 02.13 так, чтобы тип обрабатываемых элементов контейнера мог быть произвольным.

04.03 [25]

Доработайте Ваше предыдущее решение задачи 03.06 так, чтобы контейнер превратился в шаблон класса.

04.04 [25]

Реализуйте вариативный шаблон функции, которая принимает произвольное количество аргументов произвольных типов и возвращает сумму всех аргументов, которые обладают целочисленным типом `int`. Используйте рекурсивное инстанцирование и отдельный шаблон функции для извлечения целочисленных значений.

04.05 [25]

Реализуйте вариативный шаблон функции, которая определяет минимальное значение из пакета своих аргументов, используя для этого оператор меньше. Предполагается, что все аргументы имеют одинаковые типы.

04.06 [25]

Реализуйте вариативный шаблон функции, которая добавляет произвольное количество элементов в конец произвольного контейнера, обладающего функцией-членом `push_back`. Используйте здесь выражение свертки.

04.07 [25]

Реализуйте алгоритм вычисления N-ого числа ряда Фибоначчи, используя метапрограммирование шаблонов.

04.08 [25]

Реализуйте алгоритм вычисления биномиального коэффициента, используя метапрограммирование шаблонов.

04.09 [25]

Реализуйте алгоритм вычисления числа e через ряд Маклорена, используя `constexpr` возможности языка.

04.10 [25]

Реализуйте алгоритм вычисления простого числа с номером N, используя `constexpr` возможности языка.

04.11 [25]

Доработайте пример 04.09 так, чтобы в нем присутствовали все операции умножения, вычитания и деления.

04.12 [20]

Реализуйте преобразующие шаблоны свойств `add_const` и `remove_const`, модифицирующие константность.

04.13 [25]

Реализуйте преобразующий шаблон свойств `decay`, низводящий тип функции, массива и всех типов ссылок.

04.14 [25]

Реализуйте проверочный шаблон свойств `is_class`, проверяющий, является ли заданный тип классом.

04.15 [25]

Доработайте пример 04.15 так, чтобы можно было определить наличие некоторого типа `T` в списке типов.

04.16 [10]

Доработайте пример 04.16 так, чтобы через функцию `get` можно было изменять по ссылке элементы кортежа.

05 Паттерны и технологии проектирования

05.01 [10]

Приведите пример программной задачи, в решении которой будет уместно использование паттерна Builder.

05.02 [25]

Продемонстрируйте вариацию паттерна Builder, выполняющего поэтапное конструирование следующим образом: `Person p = builder.name("Ivan").age(26).height(180).get()`. Здесь предполагается, что `builder` является экземпляром некоторого класса `Builder`, предназначенного для поэтапного конструирования экземпляров класса `Person`. Изначально в объекте `builder` создается «пустой» экземпляр класса `Person`, который последовательно дополняется необходимыми атрибутами в процессе вызова функций-членов класса `Builder`.

05.03 [10]

Приведите пример программной задачи, в решении которой будет уместно использование паттерна Factory.

05.04 [10]

Приведите пример программной задачи, в решении которой будет уместно использование паттерна Prototype.

05.05 [10]

Сформулируйте недостатки паттерна Singleton, а также перечислите и прокомментируйте его альтернативы.

05.06 [10]

Приведите пример программной задачи, в решении которой будет уместно использование паттерна Adapter.

05.07 [20]

Продемонстрируйте вариацию паттерна Adapter, используя закрытое наследование вместо композиции.

05.08 [10]

Приведите пример программной задачи, в решении которой будет уместно использование паттерна Bridge.

05.09 [10]

Приведите пример программной задачи, в решении которой будет уместно использование паттерна Composite.

05.10 [10]

Приведите пример программной задачи, в решении которой будет уместно использование паттерна Decorator.

05.11 [20]

Продемонстрируйте вариацию паттерна Decorator с реализацией по умолчанию чисто виртуальной функции.

05.12 [10]

Приведите пример программной задачи, в решении которой будет уместно использование паттерна Memento.

05.13 [10]

Приведите пример программной задачи, в решении которой будет уместно использование паттерна Observer.

05.14 [10]

Приведите пример программной задачи, в решении которой будет уместно использование паттерна State.

05.15 [10]

Приведите пример программной задачи, в решении которой будет уместно использование паттерна Strategy.

05.16 [10]

Приведите пример программной задачи, в решении которой будет уместно использование паттерна Template.

05.17 [20]

Продемонстрируйте вариацию паттерна Bridge, используя статический полиморфизм вместо динамического.

05.18 [20]

Продемонстрируйте вариацию паттерна Strategy, используя статический полиморфизм вместо динамического.

05.19 [20]

Продемонстрируйте вариацию паттерна Abstract Factory, заменив виртуальные функции паттерном CRTP.

05.20 [20]

Продемонстрируйте вариацию паттерна Decorator, сделав класс декоратора шаблоном и используя наследование от параметра шаблона класса. Указатель или ссылка на декорируемый объект заменится базовым классом.

05.21 [50]

Напишите программу, которая будет рассчитывать время, затрачиваемое для получения желаемых оценок на различных учебных курсах. В данном алгоритме будут присутствовать общие компоненты, например, учет халявности преподавателя, однако для разных курсов будут и отличающиеся условия, например, на одних курсах необходимо сдавать еженедельные домашние задания, а на других можно сдать тетрадь соседа. Для начала реализуйте несколько классов, каждый из которых будет соответствовать определенному курсу и будет по-своему определять вычисление некоторой компоненты алгоритма. Далее создайте общий для них базовый класс, который будет задавать структуру всего алгоритма и вычислять общие компоненты. В данной ситуации Вы должны реализовать паттерн шаблонный метод и идиому NVI. Далее переработайте реализованную Вами иерархию при помощи паттерна CRTP. В результате должен получиться так называемый перевернутый миксин. Сдавать следует оба решения: шаблонный метод на виртуальных функциях и миксин на основе паттерна CRTP.

05.22 [25]

Доработайте пример `03.18` так, чтобы все необходимые операторы подмешивались за счет использования базовых классов на основе миксинов из Boost.Operators. Из собственных реализаций должен остаться минимальный набор операций, например, из шести операций сравнения должна быть реализована только одна, т.е. оператор меньше. Остальные пять операций сравнения должны быть подмешаны в класс с помощью миксинов.

05.23 [25]

Доработайте Ваше предыдущее решение задачи 03.08 так, чтобы все необходимые для обслуживания IP адресов операторы подмешивались за счет использования базовых классов на основе миксинов из Boost.Operators.

05.24 [25]

Доработайте пример `03.18` так, чтобы для сравнения использовался оператор `<=>` трехстороннего сравнения.

06 Организация проектов и библиотек

06.01 [30]

Доработайте пример 03.02 так, чтобы компоненты класса были корректно разделены между заголовочным файлом и файлом исходного кода. В сумме в Вашем проекте должны получиться два файла исходного кода и один заголовочный файл. Помимо этого поместите компоненты Вашего класса в отдельное пространство имен.

06.02 [10]

Сформулируйте проблемы неоптимальности идиомы Pimpl и способы их решения через идиому Fast Pimpl.

06.03 [30]

Доработайте Ваше предыдущее решение задачи 06.01 так, чтобы вместо заголовков использовались модули.

07 Обработка ошибок и исключений

07.01 [25]

Реализуйте алгоритм решения квадратного уравнения, принимающий в качестве аргументов коэффициенты уравнения a, b и c. Гарантируется, что коэффициент a не равен нулю. Квадратное уравнение может иметь 2, 1 или 0 решений. Для хранения решений используйте класс-хранилище `std::variant`, в котором для представления 0 решений будет использоваться тип `std::monostate`. Для представления двух решений рекомендуется использовать тип `std::pair`, а для представления одного решения достаточно одиночного `double`. Точность сравнения чисел с плавающей точкой в данной задаче можно не затрагивать, в случае с дискриминантом допускается проверка условия `d == 0.0`. Также не используйте функцию `std::visit` из публичных решений.

07.02 [25]

Реализуйте алгоритм нахождения точки пересечения двух прямых на плоскости. Каждая прямая задается каноническим образом через три коэффициента a, b и c. Как известно, две прямые могут пересекаться, совпадать или быть параллельны, таким образом можно получить 0, 1 или бесконечно много точек пересечения. Для хранения результата используйте класс-хранилище `std::variant`. Подумайте, какие альтернативные варианты стоит организовать в данном хранилище. Для лучшей организации Вашего кода рекомендую создать вспомогательные классы `Point` и `Line`, но также не используйте функцию `std::visit` из публичных решений.

07.03 [25]

Реализуйте класс `Person`, обладающий опциональными данными на основе класса-хранилища `std::optional`.

07.04 [25]

Продемонстрируйте на простых примерах обработку исключений `std::bad_alloc`, `std::bad_variant_access`, `std::bad_optional_access`, а также исключений `std::length_error` и `std::out_of_range`, генерируемых функциями-членами класса-контейнера `std::vector`. Объясните, кто и почему выбрасывает эти исключения.

07.05 [25]

Доработайте пример `03.18` так, чтобы места возникновения ошибок обрабатывались с помощью исключений.

07.06 [25]

Рассмотрим представленную ниже функцию. Вам необходимо выделить в ней все точки, в которых возможно ветвление пути выполнения. Например, в данной функции представлен оператор `if`, который может выполняться по одному или другому пути. Эти пути мы будем называть нормальными путями выполнения. На первый взгляд кажется, что их два, но на самом деле больше. Далее рассмотрим, например, оператор вывода. В соответствии со стандартом C++ каждый оператор вывода может выбрасывать исключения, таким образом каждый оператор вывода в приведенном коде порождает точку ветвления пути выполнения – один путь будет являться нормальным, а другой ненормальным с исключением. Всего в данном коде представлено 6 операторов вывода, которые в совокупности обеспечивают 6 точек ветвления. Составьте полный список точек ветвления пути выполнения для данного кода. Примечание: `String` и `Employee` – это какие-то пользовательские классы.

```
String evaluate_salary_and_return_name(Employee e)
{
  if (e.title() == "CEO" || e.salary() > 100000)
  {
    std::cout << e.name() << " " << e.surname() << " is overpaid.\n";
  }
  else
  {
    std::cout << e.name() << " is not overpaid.\n";
  }
  return e.name() + " " + e.surname();
}
```

07.07 [25]

Доработайте пример `02.11` так, чтобы алгоритм тестировался с помощью инструментов библиотеки Boost.Test.

07.08 [25]

Доработайте пример `02.13` так, чтобы алгоритм тестировался с помощью инструментов библиотеки Boost.Test.

07.09 [25]

Сравните среднее время выполнения виртуальной функции и обычной функции с аналогичной реализацией.

07.10 [25]

Доработайте пример `02.11` так, чтобы алгоритм тестировался с помощью инструментов библиотеки Google.Test.

07.11 [25]

Доработайте пример `02.13` так, чтобы алгоритм тестировался с помощью инструментов библиотеки Google.Test.

08 Особенности математических вычислений

08.01 [25]

Реализуйте вариативный шаблон функции, вычисляющей суммарный объем памяти в байтах, занимаемый пакетом ее параметров. Внутри функции используйте выражение свертки, обычный оператор `sizeof` и оператор сложения в выражении свертки. Продемонстрируйте альтернативу без использования выражения свертки.

08.02 [10]

Продемонстрируйте мошеннический способ изменения приватного поля экземпляра некоторого класса внешним пользователем не являющимся другом посредством приведения `reinterpret_cast` или `std::bit_cast`.

08.03 [25]

Доработайте пример `03.02` так, чтобы можно было задавать месяц через именованную константу перечисления с областью видимости. Это позволит избежать возможной путаницы при указании дня и месяца в разном порядке. Реализуйте перечисление с областью видимости для месяцев и продемонстрируйте его использование.

08.04 [20]

Доработайте Ваши предыдущие решения задач `07.01` и `07.02` так, чтобы в реализациях алгоритмов учитывались все необходимые аспекты правильной работы с числами с плавающей точкой, в частности, точность, значение бесконечности и знаковый ноль. Позаботьтесь о том, чтобы в консоль не выводился минус для нуля.

08.05 [10]

Доработайте пример `08.07` так, чтобы выполнялось обратное дискретное преобразование Фурье для сигнала.

08.06 [75]

Рассмотрим окружность и случайный вписанный в нее треугольник. С какой вероятностью центр окружности окажется внутри данного треугольника? Вычислите приблизительную вероятность, используя метод Монте-Карло. Для этого сгенерируйте большое количество случайных вписанных в окружность треугольников и определите, сколько треугольников содержат центр окружности. Рассмотрим окружность O с радиусом `r = 1` и центром в точке P с координатами `(1; 1)`. Благодаря симметрии можно зафиксировать одну из вершин каждого случайно сгенерированного вписанного в окружность O треугольника ABC. Пусть точка A имеет фиксированные координаты `(1; 0)`. Для вершин треугольника B и C достаточно случайно сгенерировать величину угла w на отрезке `[0; 2Pi]`, а затем вычислить координаты по следующим формулам: `x = 1 + cos(w)` и `y = 1 + sin(w)`. Для проверки принадлежности точки P треугольнику ABC следует использовать метод барицентрических координат. Площадь S со знаком треугольника ABC можно вычислить как `[AB * AC] / 2`, где `[AB * AC]` означает величину векторного произведение векторов AB и AC. Барицентрические координаты точки P вычисляются следующим образом: `a = [PA * PB] / 2S`, `b = [PB * PC] / 2S` и `c = [PC * PA] / 2S`. Точка P находится внутри треугольника ABC, если все три барицентрические координаты a, b и c неотрицательны. Постарайтесь максимально оптимизировать код. Допускается использование типа данных `float`. Для генерации случайных чисел используйте вихрь Мерсенна `std::mt19937`. Программу следует запускать в режиме Release. После вычисления приблизительной вероятности с помощью метода Монте-Карло предоставьте точный математически обоснованный ответ. В трехмерном случае рассмотрим сферу и случайный вписанный в нее тетраэдр. С какой вероятностью центр сферы окажется внутри данного тетраэдра? Вычислите точную вероятность, используя математические рассуждения. В N-мерном случае рассмотрим гиперсферу и случайный вписанный в нее симплекс. С какой вероятность центр гиперсферы окажется внутри данного симплекса? Достаточно предъявить без обоснования общую формулу вероятности, параметризованную только числом N.

08.07 [25]

Доработайте пример `08.11` так, чтобы с помощью хронометра можно было проводить серии измерений. Предполагается, что хронометр можно будет останавливать и перезапускать, а внутри себя он будет содержать контейнер `std::vector`, в котором будут храниться несколько временных интервалов, соответствующих отдельным измерениям. Добавьте функции-члены `start` и `stop`, которые будут вызываться в начале и конце каждого измерения в серии. Функция `start` сохраняет время запуска, функция `stop` вызывает функцию `elapsed` и записывает замеренный интервал во внутренний контейнер. Также добавьте булевский флаг в хронометр, показывающий, идет ли сейчас замер или нет. В функциях `start` и `stop` следует проверять и обновлять данный флаг. Если функциями хронометра пользуются неправильно, то необходимо генерировать исключение. В качестве итогового результата хронометр должен выдавать для серии измерений среднее значение времени.

08.08 [25]

Напишите программу, которая сможет вычислить временной интервал в днях между двумя заданными датами.

09 Низкоуровневое управление памятью

09.01 [10]

Приведите пример программной задачи, в решении которой будет уместно использование идиомы RAII.

09.02 [25]

Реализуйте класс `Logger` для трассировки, используя идиому RAII и информацию из `std::source_location`. Предполагается, что пользователь данного класса в начале каждой функции будет создавать экземпляр логгера, конструктор которого выведет в консоль сообщение о начале выполнения функции с указанием всей необходимой информации. Деструктор совершит аналогичное действие после завершения работы данной функции. Оформите инструкции вывода трассировочных сообщений в отдельной функции, доступной для пользователя.

09.03 [25]

Реализуйте класс `Scope_Guard`, обеспечивающий откат эффектов первого действия при неудавшемся из-за возникшего исключения втором. Предположим, в некоторой функции необходимо последовательно выполнить два действия – A и B. Действие B может завершиться неудачно и привести к генерации исключения. В таком случае необходимо откатить действие A, обеспечив тем самым транзакционное поведение и строгую гарантию безопасности исключений в этом месте. Можно оформить `try`-секцию вокруг действия B, перехватить и обработать исключение. Однако есть и более элегантный подход, основанный на идиоме RAII. Можно создать экземпляр некоторого класса `Scope_Guard` между действиями A и B. В деструкторе этого класса следует прописать откат действия A. В случае возникновения исключения деструктор будет вызван автоматически, а откат будет произведен. Если же действие B осуществится нормально без исключений, то после него следует с помощью дополнительной инструкции вручную отключить откат действия A в деструкторе `Scope_Guard`. Обратите внимание, что откат действия A, а также деструктор класса `Scope_Guard` не должны генерировать исключений.

09.04 [30]

Доработайте примеры `05.02`, `05.03`, `05.07`, `05.08`, `05.10`, `05.12` так, чтобы использовались интеллектуальные указатели вместо обычных. Подумайте, какой тип интеллектуальных указателей, а именно `std::shared_ptr` или `std::unique_ptr`, следует использовать. Указатель `std::weak_ptr` не потребуется Вам в данной задаче.

09.05 [25]

Доработайте пример `09.03` так, чтобы дополнительные шаблонные версии перемещающего конструктора и оператора перемещающего присваивания позволяли передавать в качестве своего аргумента `other` другие объекты типа `Unique < U >` , то есть указатели на другой тип `U`, отличный от типа `T`. Предполагается, что тип `U` является производным от типа `T`, поэтому допускается сохранять указатель на производный тип в указателе на базовый тип. Для более надежной работы шаблонов добавьте им ограничения через `requires`. В ограничениях должно быть указано, что тип `U` обязан являться производным от типа `T`. Используйте шаблон свойств `std::is_convertible` через шаблон переменной. Также дополните интерфейс класса `Unique` оператором стрелочка, функцией-членом `get` и оператором неявного приведения класса `Unique` к типу `bool`.

09.06 [25]

Доработайте пример `09.04` так, чтобы дополнительные шаблонные версии перемещающего конструктора и оператора перемещающего присваивания позволяли передавать в качестве своего аргумента `other` другие объекты типа `Shared < U >` , то есть указатели на другой тип `U`, отличный от типа `T`. Предполагается, что тип `U` является производным от типа `T`, поэтому допускается сохранять указатель на производный тип в указателе на базовый тип. Для более надежной работы шаблонов добавьте им ограничения через `requires`. В ограничениях должно быть указано, что тип `U` обязан являться производным от типа `T`. Используйте шаблон свойств `std::is_convertible` через шаблон переменной. Также дополните интерфейс класса `Shared` оператором стрелочка, функцией-членом `get` и оператором неявного приведения класса `Shared` к типу `bool`.

09.07 [25]

Реализуйте класс `Tree` бинарного дерева, используя интеллектуальные указатели для связи родителей и потомков. Реализуйте структуру `Node`, содержащую указатели типа `std::shared_ptr` на правого и левого потомков, указатель типа `std::weak_ptr` на родителя и данные типа `T`. Продемонстрируйте отсутствие неразрушимых циклических связей и корректную работу деструкторов всех узлов по аналогии с изученным примером `09.01`.

09.08 [10]

Продемонстрируйте вариант использования интеллектуальных указателей типа `std::shared_ptr`, при котором указатели указывают на член крупного общего объекта, например, на одно из полей экземпляра некоторой структуры, но при этом также выполняют автоматическое управление памятью для всего исходного объекта.

09.09 [25]

Доработайте пример `02.11` так, чтобы вместо обычных указателей можно было использовать итераторы контейнера. Для работы с итераторами вместо арифметических операторов следует использовать обобщенные функции, например, `std::next` и `std::prev`. Итераторы могут быть любой категории, не обязательно произвольного доступа. Также итераторы могут задавать начало и конец коллекции, хранящейся в произвольном контейнере, поэтому функция должна стать шаблоном. В качестве подсказки можете использовать приведенную в справочнике реализацию алгоритма `std::lower_bound`. Вместо `nullptr` возвращайте итератор конца.

09.10 [25]

Доработайте пример `02.13` так, чтобы вместо обычных указателей можно было использовать итераторы контейнера. Для работы с итераторами вместо арифметических операторов следует использовать обобщенные функции, например, `std::distance` и `std::iter_swap`. Итераторы могут быть любой категории, не обязательно произвольного доступа. Также итераторы могут задавать начало и конец коллекции, хранящейся в произвольном контейнере, поэтому функции должны стать шаблонами. В качестве подсказки можете использовать приведенную в справочнике реализацию алгоритма `std::sort`. Пользовательский компаратор не нужен.

09.11 [25]

Доработайте пример `09.06` так, чтобы односвязный список превратился в двусвязный, а итераторы стали двунаправленными. Для этого в структуру `Node` необходимо добавить указатель на предыдущий узел, а для итератора дополнительно создать перегруженные версии префиксного и постфиксного операторов декремента.

09.12 [10]

Приведите пример программной задачи, в решении которой будет уместно использование паттерна Iterator.

09.13 [10]

Приведите пример программной задачи, в решении которой будет уместно использование паттерна Facade.

09.14 [25]

Реализуйте класс итератора концепции Forward Iterator для генерации последовательности нечетных чисел.

09.15 [10]

Сформулируйте принципы функционирования и использования таблиц страниц для отображения адресов.

09.16 [10]

Сравните среднее время выделения диспетчером памяти блоков памяти размерами в 1 Кб, 1 Мб и 1 Гб.

09.17 [25]

Доработайте пример `09.21` так, чтобы данный аллокатор можно было использовать в контейнере `std::vector`.

09.18 [25]

Доработайте пример `09.21` так, чтобы уменьшилась степень фрагментации памяти. Для этого алгоритм `find_first` следует заменить на алгоритм `find_best`, определяющий среди блоков списка свободный фрагмент равный или наиболее близкий по размеру требуемому. Сравните производительность для обеих версий.

09.19 [10]

Исследуйте публичные реализации Buddy аллокатора и опишите его предназначение и особенности работы.

09.20 [40]

Доработайте примеры `09.18`, `09.19`, `09.20`, `09.21` так, чтобы все аллокаторы представлены в виде иерархии в составе статической или динамической библиотеки. Для этого следует организовать наследование от абстрактного полиморфного базового класса, внедрить механизм виртуальных функций и разбить интерфейсы и реализации на заголовочные файлы и файлы исходного кода. Продемонстрируйте использование библиотеки.

10 Коллекции объектов и контейнеры

10.01 [10]

Сформулируйте способ хранения полухода шахматной партии с минимально возможными затратами памяти.

10.02 [30]

Исследуйте систему выделения памяти внутри вектора. Для этого определите с помощью вызовов функции-члена `capacity`, во сколько раз изменяется емкость вектора при нехватке памяти для размещения новых элементов. Также определите, как увеличивается емкость вектора, если задать ее начальное значение вручную с помощью вызова функции-члена `reserve`. Дополнительно определите, как осуществляется выделение памяти в предельном случае, когда вектор уже запросил большой объем памяти и на выполнение следуещего запроса у ОС может не хватить ресурсов. В комментариях в коде подробно опишите полученные наблюдения.

10.03 [10]

Сформулируйте идею однопроходного алгоритма поиска среднего узла односвязного или двусвязного списков.

10.04 [10]

Доработайте пример `10.11` так, чтобы контейнеры могли дополнительно хранить и обрабатывать максимумы.

10.05 [20]

Сформулируйте определения кучи для максимума, а также двоичной, биномиальной и фибоначчиевой кучи.

10.06 [20]

Продемонстрируйте опасные способы изменения ключа в контейнерах `std::map` без создания нового узла.

10.07 [90]

Исследуйте 9 хэш-функций из этой статьи на равномерность и количество возникающих коллизий. Данные хэш-функции предназначены для строк, поэтому Вам потребуется написать генератор множества случайных строк без дубликатов. Рекомендую ограничиться английским алфавитом в нижнем регистре и строками из 10 символов. Множество строк нужно подготовить один раз. Для всех хэш-функций оно должно быть одним и тем же. Для исключения наличия дубликатов можно добавлять новые строки в контейнеры типа `std::set` или `std::unordered_set` в процессе генерации. Определите количество коллизий дял каждой функции на различных количествах хэшируемых строк, например, от 100000 до 2000000 с шагом 100000 и постройте графики полученных зависимостей. В данной задаче используйте режим Release x86 из-за особенностей хэш-функций.

10.08 [25]

Реализуйте класс `Phonebook` для хранения записей телефонного справочника, которым разные клиенты смогут пользоваться по-разному. Например, городская типография собирается напечатать справочник, и ей нужны записи в отсортированном по фамилии человека порядке. Рекламное агентство нуждается в произвольном доступе к записям справочника. Регулярный пользователь хочет за максимально короткое время находить нужную ему запись. Удовлетворите данные желания клиентов, реализовав контейнер на базе Boost.MultiIndex.

11 Итераторы и алгоритмы на диапазонах

11.01 [25]

Реализуйте функцию, которая возвращает указатель на саму себя. Используйте дополнительного посредника.

11.02 [30]

Доработайте пример 02.13 так, чтобы пользователь мог задавать критерий сортировки через функцию, передаваемую в алгоритм сортировки как дополнительный аргумент, тип которого является указателем на функцию. Продемонстрируйте использование собственных критериев сортировки по возрастанию и по убыванию. Доработайте тот же пример так, чтобы пользователь мог задавать критерий сортировки через функциональный объект с перегруженным оператором вызова, передаваемый в алгоритм сортировки как дополнительный аргумент, тип которого задается дополнительным параметром шаблона. Продемонстрируйте использование собственных критериев сортировки по возрастанию и по убыванию, а также функциональных объектов из стандартной библиотеки `std::less` и `std::greater`. Доработайте тот же пример так, чтобы пользователь мог задавать критерий сортировки через лямбда-выражение, передаваемое в алгоритм сортировки как дополнительный аргумент, тип которого задается дополнительным параметром шаблона. Продемонстрируйте использование собственных критериев сортировки по возрастанию и по убыванию. Сдавайте 3 файла решений.

11.03 [25]

Реализуйте паттерн Factory в виде класса, содержащего ассоциативный контейнер, связывающий тип или код создаваемого объекта и пакет инструкций для создания этого объекта. Пакет инструкций следует оформить в виде лямбда-выражения. Необходимо реализовать способ хранения лямбда выражений по аналогии с примером 11.03. Разместите ассоциативный контейнер в приватной секции, инициализируйте его и реализуйте необходимый публичный интерфейс. Продемонстрируйте несколько способов использования Вашей фабрики.

11.04 [10]

Приведите пример программной задачи, в решении которой будет уместно использование паттерна Command.

11.05 [10]

Приведите пример программной задачи, в решении которой будет уместно использование паттерна Visitor.

11.06 [25]

Доработайте Ваше предыдущее решение задачи 03.02 так, чтобы информация о фигурах сериализовалась, т.е. выводилась в строковом виде в консоль, с помощью отдельного класса посетителя и двойной диспетчеризации.

11.07 [10]

Исследуйте публичные реализации компьютерной игры Asteroids. Во время игры программа должна обрабатывать столкновения объектов, например, астероидов различных типов, космических кораблей и пуль. Объясните, из-за чего может возникнуть дублирование кода и как оно устраняется за счет использования паттерна Visitor.

11.08 [20]

Доработайте Ваши предыдущие решения задач 07.01 и 07.02 так, чтобы упакованный в `std::variant` результат работы алгоритма выводился в консоль одним из посетителей. Посетителей можно реализовать как функциональные объекты с перегруженными операторами вызова или как лямбда-выражения. Обязательно продемонстрируйте в решении использование функции `std::visit` по аналогии с изученным примером 11.06.

11.09 [10]

Доработайте пример 11.08 так, чтобы через функцию `get` можно было изменять по ссылке элементы варианта.

11.10 [30]

Реализуйте вариативные шаблоны функций `any_of`, `all_of` и `none_of`, проверяющие наличие в коллекции объектов хотя бы одного из заданных объектов, всех заданных объектов и ни одного из заданных объектов соответственно. Количество заданных объектов может быть произвольным. Используйте выражения свертки. Для сокращения дублирования кода проверок рекомендуется реализовать дополнительный невариативный шаблон функции `has_one`, которая будет проверять наличие в коллекции объектов одного заданного значения.

11.11 [25]

Реализуйте алгоритм `transform_if` на итераторах как синтез алгоритмов `std::transform` и `std::copy_if`.

11.12 [50]

Продемонстрируйте использование алгоритмов `std::count`, `std::mismatch`, `std::equal`, `std::search`, `std::replace`, `std::remove`, `std::unique`, `std::rotate`, `std::partition` и `std::nth_element` из стандартной библиотеки на простейших тестовых примерах. Не используйте в данной задаче диапазоны и представления.

11.13 [10]

Сформулируйте принципы работы и сложность для всех алгоритмов сортировки из стандартной библиотеки.

11.14 [20]

Продемонстрируйте использование алгоритмов `std::accumulate`, `std::inner_product`, `std::partial_sum` и `std::adjacent_difference` из стандартной библиотеки работы с числами на простейших тестовых примерах.

11.15 [20]

Доработайте Ваше предыдущее решение задачи 11.12 так, чтобы использовались алгоритмы на диапазонах.

11.16 [20]

Продемонстрируйте использование инструментов просмотра `views::istream`, `views::counted`, `views::join` и `views::stride` из стандартной библиотеки диапазонов и представлений на простейших тестовых примерах.

11.17 [25]

Продемонстрируйте использование алгоритма поиска путей Беллмана-Форда из библиотеки Boost.Graph.

11.18 [25]

Продемонстрируйте использование алгоритма поиска путей Флойда-Уоршелла из библиотеки Boost.Graph.

12 Кодирование символов и парсинг текстов

12.01 [10]

Сформулируйте способы ослабления монолитности и перегруженности класса для строк `std::basic_string`.

12.02 [20]

Исследуйте оптимизацию малых строк, реализованную для класса `std::basic_string` в стандартной библиотеке Вашей версии компилятора. Определите размер буфера N, использующегося для хранения малых строк без динамического выделения памяти. Сравните среднее время создания строк длины N и строк длины N+1.

12.03 [30]

Сформулируйте алгоритмы Бойера-Мура и Бойера-Мура-Хорспула поиска подстроки в строке и продемонстрируйте их работу, используя возможности стандартной библиотеки шаблонов, а именно алгоритм `std::search`. Сравните производительность этих алгоритмов с умолчательной версией алгоритма поиска подстроки в строке.

12.04 [10]

Сформулируйте полные алгоритмы формирования кодировок символов UTF-16 в версиях big и little endian.

12.05 [25]

Реализуйте алгоритм преобразования строки шестнадцатиричных цифр в коллекцию 8-битных целых чисел. Например, строка `"BAAD"` должна быть преобразована в коллекцию `0xBA, 0xAD`. Для хранения коллекции можно использовать контейнер `std::vector`. Предполагается, что в исходной строке четное количество цифр.

12.06 [25]

Реализуйте алгоритм преобразования начальных символов слов некоторого текста в верхний регистр, а остальных символов в нижний регистр. Например, текст `"heLLo, woRLD"` должен превратиться в `"Hello, World"`.

12.07 [25]

Реализуйте алгоритм конкатенации коллекции строк в одну строку через заданный пользователем разделитель.

12.08 [25]

Реализуйте алгоритм разбиения строки на лексемы по заданному списку возможных символов-разделителей.

12.09 [25]

Реализуйте алгоритм сжатия строк вида aaabbbccc, состоящих только из латинских букв, в строки вида a3b3c3.

12.10 [25]

Реализуйте алгоритм поиска внутри строки подстроки-палиндрома наибольшей длины. Если таковых подстрок несколько, достаточно найти первую из них. Для уменьшения сложности алгоритма используйте кэширование.

12.11 [25]

Продемонстрируйте использование пяти алгоритмов для обработки строк из библиотеки Boost.StringAlgorithms.

12.12 [25]

Реализуйте алгоритм конвертации валют, обеспечивающий ввод и вывод денежных сумм с учетом национальных стандартов. Например, Вы можете реализовать перевод EUR в RUB. В таком случае Вам потребуется использовать недружественную европейскую, например, немецкую локаль для настройки ввода суммы в EUR и русскую локаль для настройки вывода суммы в RUB. Для наиболее удобной реализации предлагаю воспользоваться манипуляторами из стандартной библиотеки `std::get_money` и `std::put_money`. Основная сложность в данной задаче заключается в выявлении корректных названий доступных в рамках операционной системы локалей. На операционных системах семейства Windows локали с высокой долей вероятности будут иметь названия вида `ru_RU.cp1251` и `de_DE.cp1251`. Для установки кодировки cp1251 в консоли на Windows следует в начале функции `main` добавить инструкцию `system("chcp 1251")`. На Linux по умолчанию используется кодировка UTF-8, поэтому дополнительные действия не потребуются. Список установленных в системе локалей можно просмотреть, набрав в терминале команду `locale -a`. Их названия будут записаны в виде `ru_RU.utf8`.

12.13 [25]

Реализуйте алгоритм побуквенной транслитерации кириллических символов русского алфавита. Предположим, что пользователь вводит символы в консоли в кодировке системной локали по умолчанию, например, cp1251. Вы должны конвертировать кодировку символов из cp1251 в UTF-8, а затем в UTF-32 наиболее удобным способом. Далее замените все кириллические символы латинскими в соответствии с правилами транслитерации. Для программной реализации используйте заранее подготовленное отображение в виде хэш-таблицы. Конвертируйте кодировку обратно в UTF-8, а затем в cp1251 и выведите результат преобразования в консоль. Преобразование кодировки cp1251 актуально только для Windows, на Linux данные действия не потребуются.

12.14 [25]

Напишите программу, которая принимает на вход текст и извлекает из него все российские автомобильные номера. В демонстрационном тестовом тексте для избежания проблем с кириллицей допускается использовать латиницу. Текст подготовьте самостоятельно и внедрите его в исходный код с помощью сырых строк. В тексте должны присутствовать как корректные, так и некорректные автомобильные номера, а также мусорный текст.

12.15 [25]

Напишите программу, которая принимает на вход текст и извлекает из него все адреса электронных почт. Допускается использовать публичные регулярные выражения из интернета, но, желательно, некоторые упрощенные варианты, чтобы не получилось спагетти-кода и чтобы Вы понимали все составляющие их компоненты и логику. Текст подготовьте самостоятельно и внедрите его в исходный код с помощью сырых строк. В тексте должны присутствовать как корректные, так и некорректные адреса электроных почт, а также мусорный текст.

12.16 [25]

Напишите программу, которая принимает на вход строку с URL и извлекает из него отдельные компоненты, такие как протокол, домен, порт и путь. Используйте группы в круглых скобках внутри регулярного выражения.

12.17 [25]

Напишите программу, которая принимает на вход текст с датами в одном формате, например, с использованием слэша в качестве разделителя, а возвращает копию этого текста с датами, оформленными в другом формате. В регулярном выражении для дат организуйте дополнительные проверки. Например, должен допускаться максимум 12-ый месяц, для каждого месяца должно допускаться определенное количество дней, например, не может быть 30 или более дней в феврале или 32 или более дня в декабря. Високосные года в регулярном выражении можно не учитывать. Также не являются допустимыми нулевой день и нулевой месяц.

12.18 [30]

Доработайте пример `12.10` так, чтобы пользователь имел возможность вычислять факториал числа при помощи унарного оператора восклицательный знак, остаток от деления с помощью бинарного оператора процент, а также использовать квадратные и фигурные скобки в арифметических выражениях. При вычислении факториала числа можно не беспокоиться о проблеме переполнения типа `double`. При разборе скобок в выражении необходимо проверять, что открывающей скобке одного типа соответствует закрывающая скобка того же типа.

12.19 [30]

Доработайте пример `12.15` так, чтобы пользователь имел возможность вычислять факториал числа при помощи унарного оператора восклицательный знак, остаток от деления с помощью бинарного оператора процент, а также использовать квадратные и фигурные скобки в арифметических выражениях. При вычислении факториала числа можно не беспокоиться о проблеме переполнения типа `double`. При разборе скобок в выражении необходимо проверять, что открывающей скобке одного типа соответствует закрывающая скобка того же типа.

12.20 [25]

Реализуйте алгоритм калькулятора на основе обратной польской нотации и стеков для операторов и операндов.

13 Потоки ввода-вывода и сериализация

13.01 [25]

Реализуйте алгоритм преобразования коллекции 8-битных целых чисел в строку шестнадцатиричных цифр. Например, коллекция `0xBA, 0xAD` должна быть преобразована в строку `"BAAD"`. Для хранения коллекции можно использовать контейнер `std::vector`. Используйте строковый поток вывода и манипуляторы потоков.

13.02 [25]

Напишите программу, которая выведет корректно отформатированные первые 10 строк треугольника Паскаля.

13.03 [25]

Напишите программу, которая выведет в консоль отформатированную таблицу с информацией о запущенных в системе процессах. Для каждого процесса необходимо вывести его название, идентификатор, статус и владельца. Создайте структуру для представления процесса и контейнер для хранения процессов.Данные для нескольких процессов можно сгенерировать самостоятельно, не требуется обращаться к операционной системе.

13.04 [25]

Доработайте пример `13.06` так, чтобы алгоритм корректно обрабатывал сырые строковые литералы в коде.

13.05 [25]

Напишите программу, которая удалит все пустые и пробельные строки из пользовательского текстового файла.

13.06 [10]

Доработайте пример `12.10` так, чтобы тестовые арифметические выражения считывались из готового файла.

13.07 [20]

Доработайте пример `12.10` так, чтобы алгоритм тестировался с помощью инструментов библиотеки Google.Test.

13.08 [25]

Напишите программу, которая удалит все элементы директории, измененные ранее заданной временной точки.

13.09 [20]

Доработайте пример `13.11` так, чтобы находились и выводились в консоль только те элементы в заданной пользователем директории, названия которых соответствуют заданному пользователем регулярному выражению.

13.10 [25]

Напишите программу, которая переименует все файлы в заданной директории. Например, Вы можете реализовать изменение регистра символов в названиях файлов. Для переименования используйте функцию-член `replace_filename` класса `std::filesystem::path`. Рекурсивный обход директории выполнять не требуется.

13.11 [25]

Напишите программу, которая рекурсивно обойдет заданную пользователем директорию и в процессе обхода найдет файлы с одинаковым содержимым. Среди таких файлов должны быть удалены все, кроме одного. Все удаленные файлы следует заменить символьными ссылками, которые указывают на оставшийся уникальный файл. Данный подход позволяет сэкономить место на диске, не выполняя сжатия и сохраняя данные. Для создания символьных ссылок используйте свободную функцию `create_symlink` библиотеки `std::filesystem`.

13.12 [50]

Доработайте пример `13.12` так, чтобы вместо формата данных JSON использовался формат данных XML.

13.13 [25]

Доработайте пример `13.13` так, чтобы в PDF документе орисовывалась демонстрационная таблица с данными.

14 Параллельное программирование

14.01 [20]

Определите, какое максимальное теоретическое и практическое количество потоков можно создать в x32 процессе с виртуальным адресным пространством размером 4 Гб. Учтите, что потоки имеют собственные стеки.

14.02 [10]

Сформулируйте аспекты, которые стоит учитывать при внедрении потоков для повышения производительности.

14.03 [25]

Реализуйте алгоритм вычисления числа π методом Монте-Карло, используя несколько параллельных потоков.

14.04 [25]

Доработайте пример `14.05` так, чтобы количество потоков можно было задавать посредством дополнительного аргумента функции. Исследуйте реализацию на масштабируемость. Постройте график зависимости времени работы алгоритма от количества потоков. Определите оптимальное количество потоков и объясните полученные результаты. Следует добавить больше реалистичности в реализацию алгоритма и сценарий его использования. Например, следует подготовить контейнер с большим количеством элементов. Также операцию сложения можно попробовать заменить на иное более сложное с вычислительной точки зрения действие, например, задействовав числа с плавающей точкой и сложные математические тригонометрические функции.

14.05 [25]

Продемонстрируйте использование разделяемого будущего результата в двух потоках на `std::shared_future`.

14.06 [25]

Реализуйте алгоритм `for_each` в параллельном варианте, используя для этого рекурсивное разбиение входной последовательности данных, асинхронные задачи на основе `std::async` и механизм будущих результатов на основе `std::future`. Вспомните, что возвращает стандартная непараллельная версия алгоритма `std::for_each` и объясните, почему в параллельной версии так поступить нельзя. Используйте пример `14.04` в роли основы.

14.07 [25]

Продемонстрируйте использование параллельной версии алгоритма `std::inclusive_scan` из стандартной библиотеки, а также такой сценария использования, при котором достигается превосходство параллельной версии данного алгоритма над стандартным последовательным алгоритмом `std::partial_sum` при условии наличия на Вашем компьютере нескольких аппаратных потоков. В алгоритмах следует использовать сложную с вычислительной точки зрения пользовательскую бинарную функцию, которую можно оформить в виде лямбда выражения. Рекомендуется использовать числа с плавающей точкой и сложные математические тригонометрические функции. Превосходство параллельного алгоритма должно быть достигнуто в режиме Release.

14.08 [25]

Доработайте пример `14.05` так, чтобы наблюдатели уведомлялись параллельно через условную переменную.

15 Компьютерные сети и сетевые технологии

16 Встраивание технологий других языков

16.01 [25]

Доработайте пример `16.06` так, чтобы визуализировались исходный и преобразованный сигналы примера `08.07`.