Критичность комментария будет в градации от 0 до 4:

0 – не критично

4 – очень критично

|  |  |  |
| --- | --- | --- |
| **Номер строки** | **Описание проблемы, способ решения** | **Критичность, комментарии** |
| 8 | FileProcessor являеется базовым классом, как я понял. Судя по использованию в мейне – используются фишки полиморфизма: создаётся указатель на базовый класс, чтобы потом туда положить экземпляр наследника, который сделан динамически.  Чтобы память не потекла при освобождении по созданному указателю на базовый класс, нужно чтобы в базовом классе был определен виртуальный деструктор. | 4 |
| 105-133 | Данный синглтон FileProcessorFactorySingleton в своей текущей реализации имеет две проблемы: он не потокобезопасный и он источник утечки памяти. Память утекает, потому что для статичного поля instance никто не вызывает free.  Не потокобезопасно, потому что может случится так, что два поток будут обращаться к методу getInstance и нарваться на то, что они будут пытаться в один и тот же указатель сложить выделенный в динамической памяти ресурс.  Лучше переделать это в синглтон Мейерса: убрать поле instance, создавать статический экземпляр этого класса(фабрики) внутри статичного метода getInstance(), сделать приватные конструктор и деструктор. | 4 |
| 39, 58-60 | Список инициализации не совпадает с тем, в каком порядке описаны поля в классе. Поля будут конструироваться в том порядке, в котором они описаны в классе.  Это может привести к неожиданным последствиям в виде, того, поле compressionParams будет пытаться конструироваться от еще несконструированных полей param1, param2  Либо нужно привести список инициализации с соответствие с порядком полей в классе. Либо наоборот поменять порядок полей в классе так, чтобы соответсвовать списку инициализации Лучше еще в конструктор compressionParams передавать конкатенацию не поелй param1, param2, а конкатенацию переданных параметров p1, p2 | 4 |
| 142 | После того как создали FileProcess нужно проверить fp на nullptr А то на 151 словим сегфолт | 4 |
| 90 | delete -> delete[] | 4 |
| 26, 47, 80 | По идеи в эту стрингу fileContent попадут символы до первого пробела или символа перехода на новую строку. Возможно задумка была в том, чтобы получить всё содержимое файла. В таком случае, я бы предложил заполнить строку fileContent посимвольно в цикле: ``` while(in.good()) {  fileContent.push\_back(in.get()); }  ``` | 4 |
| 24, 45, 78 | После конструирования потока ввода из файла следует проверить, что по указанному пути файл вообще существовал: добавить проверку if (!in) и что собсна делать в этом случае | 4 |
| 68 | Условие цикла надо поменять на строго меньше 16. В противном случае выйдем за пределы массива encryptionKey | 4 |
| 72 | Обращение к элементу по индексу 16 – это выход за пределы массива – неопределенное поведение. Надо заменить на 15. | 4 |
| 75 | 1. Лучше не использовать виртуальные методы в конструкторах/деструкторах, потому что на момент конструирования, например наследника – сначала будет конструироваться самый базовый класс и соответственно вызов виртуального метода может привести к непредвиденным последствиям в виде например pure virtual call если метод в базов классе чисто виртуальный.  Нужно подумать какова цель вообще метода clearKey() – сейчас будучи публичным методом можно подумать, что он нужен для внешнего пользователя. Однако это было бы корректно если бы мы могли сами вызывать то, что происходит в теле конструктора класса FileEncryptor – инициализировать ключ.  Поэтому я предлагаю перенести тело метода clearKey() в деструктор и сделать его виртуальным(по идеи, т.к. нужно добавить виртуальный деструктор в базовый класс, то тут будет переопределение деструктора)  Но есть нюанс: в теле clearKey() кидается исключение. Сейчас получается, что исключение кидается в деструкторе, а это не желательно. Обычно деструкторы неявно помечены как noexcept. Но даже если явно пропишем noexcept(false), то рискуем нарваться на double exception – когда например, где то в рандомном месте полетело исключение и начнет раскручиваться стек, сработает вот у экземпляра этого класса деструктор и полетит новое исключения, при этом потеряв старое, которое инициировало раскрутку стека.  Поэтому, я думаю, следует убрать бросание этого исключения. Возможно если это важно, то можно попробовать подумать в сторону кодов ошибок или логирования. | 4 |
| 75 | Нужно по правилу 5 дописать копирующие и перемещающие конструкторы и операторы присваивания.  Плюс на строках 102, 103 проверяется что у данного класса перемещение не кидает исключений, поэтому нужно будет сделать их не кидающими исключений и пометить их как noexcept. | 4 |
|  |  |  |
| 115 | В методе FileProcessorFactorySingleton::createFileProcessor есть две проблемы: 1. нет возвращаемого значения по умолчанию. есть куча ретернов внутри ифов, но если придет строка, которая не совпадает ни с одним из вариантов, то получается, что не найдется на такой случай возвращаемого значения. Надо добавить в конце метода *return nullptr* 2. mode – используется только для чтения. можно принимать его по конст ссылке или если это с++17 то std::string\_view | 3 |
| 24-26, 45-47, 78-80 | Я заметил ,что данный код повторяется в переопределении наследниками метода processFile(). Предлагаю выделить эту часть в отдельный метод внутри родительского класса в секцию protected и назвать например: std::string getFileContent(const std::string& filePath) | 3 |
| 41-42 | Перемещающие конструктор и оператор присваивания надо пометить как noexcept Предлагаю также сделать их дефолтными: = default;   Вообще в связи с тем, что данный класс выглядит тривиальным, если судить по полям. То можно воспользоваться правилом нуля и убрать эти перемещающие конструктор и оператор присваивания, т.к. все поля типа std::string, а у них внутри соблюдено правило 5.  В дальнейшем, если же наоборот будет что-то нетривиальное, что нужно будет что-то делать в их телах, то нужно будет соблюсти правило 5: дописать еще реализацию виртуального деструктора и реализацию(или запретить их) копирования(конструктор и оператор присваивания). | 3 |
| 87, 72, 99 | Мне кажется, что безопасней будет всё таки поменять тип поля encryptionKey c указателя на массив char, на std::array<char, 16>  Так мы сможем закрепить магическое число 16, как размер этого массива. Избавимся от проблемы со слежкой за выделенной памятью. и к последнему элементу будет легко обратиться используя итератор: \*prev(end(encryptionKey)) = 0;  ===  или можно использовать все таки std::string с фиксированным размером. для 16 байтов там по идеи сработает sso(small string optimization), которая не будет создавать объект в динамической памяти, а создаст на стеке.  Тогда сделав для поля тип строки, в списке инициализации надо будет просто вызвать конструктор от указанного размера, чтобы создать строку размером 16.  ===  Если все таки и использовать массив чаров, то я бы настоял на использовании умных указателей, вместо использования сырых. Например поменять тип поля на unique\_ptr<char []> (лучше создавать не используя make\_unique если с++11, если выше то можно. причина: вызов правильного delete[]) | 3 |
| 115-128 | Предлагаю все же отойти от использования сырых указателей в сторону использования умных, например std::unique\_ptr.  Тогда нужно будет поменять тип возвращаемого значения и в ретернах использовать std::make\_unique вместо new  Плюс можно будет убрать ручное освобождение памяти на 154-156, 160-162 | 3 |
|  |  |  |
| 23, 44, 77 | Здесь для подчеркивания факта переопределения метода processFile наследниками используется ключевое слово virtual  Лучше использовать подход с обозначение переопределения с помощью ключевого слова override  Оно не только будет указывать на то, что наследник переопределяет родительский виртуальный метод, но также убережет от случая, если в родительском методе изменится сигнатура – компилятор ругнется, что в наследниках переопределяется метод, которого нет в родителе. | 2 |
| 39, 54 | В методе compressFiles происходит вывод параметров, однако из инициализации видно, что эти две строки будут склеены в одну без разделителей.  У меня есть предположение, что это не то, что первоначально задумывалось и при их выводе ожидается увидеть их разделенными пробелами. Поэтому предлагаю использовать в выводе поля param1, param2.  Возможно поле compressFiles можно впринципе убрать из класса, тк больше нигде оно не будет использоваться. | 2 |
|  |  |  |
| 10, 18, 23, 31, 44, 52, 77, 93, 115 | В данных методах принимается один параметр типа std::string – по значению. Везде параметр используется только для чтения и не меняется.  Предлагаю принимать данную строку по конст ссылке или если это с++17, то поменять тип на std::string\_view. Так мы избавимся от лишнего копирования. | 1 |
| 10, 23, 31, 44, 52, 77, 93, | noexept(false) кажется излишним. Конечно, если так принято по стайл-гайду, чтобы явно обозначить что данный метод не кидает исключений, то ок. Но по идеи можно считать по умолчанию, что все методы могут кидать исключения, если не указано иное с помощью noexcept | 1 |
| 58-60 | Мне не очень нравится такое наименование полей для параметров. Кажется, что можно легко ошибиться ибо имена отличаются только цифрой. Я бы предложил переименовать в: firstParam, secondParam | 1 |
| 38 | В зависимости от задумки можно тут сделать две вещи: - поменять тип параметров констурктора на std::string\_view(или конст ссылки) и конструировать по ним поля param1, param2 - либо оставить сигнатуру как есть и конструировать param1, param2 конструктором перемещения. Для этого надо добавить std::move к p1, p2  И то, и то направлено на то, чтобы убрать лишнее копирование.  В первом случае это уберет копирование строк при передаче в конструктор.  Во втором случае уберет одно копирование при конструировании полей. | 1 |
| 150 | Предлагаю принимать по конст ссылке элементы вектора, чтобы избежать лишнего копирования | 1 |
| 67-72 | Предлагаю выделить магическое число 16 в локальную константу с говорящим названием. | 1 |
| 66 | Лучше использовать nullptr вместо NULL | 1 |
|  |  |  |
| 70 | Можно добавить статик каст к char выражения справа от «равно», чтобы избежать неявных конвертаций | 0 |
| 10 | Судя по реализации наследников, можно сделать данный метод константным, т.к. исполнение данного метода не изменяет внутренности наследников.  Так как код тестовый, то я не настаиваю на этом, тк не очень понимаю действительно ли этот метод не будет изменять экземпляр этого класса(наследника). | 0 |
| 147 | Можно заменить на emplace\_back | 0 |
| 39, 58-60 | При текущей реализации, кажется излишним создавать поле compressionParams. Можно обойтись и без него в методе compressFiles используя только param1, param2  Возможно я не прав, ибо не знаком со спецификой. | 0 |
| 1 - 4 | Я так понял это стандартные либы, поэтому лучше их в угловые скобки записать. | 0 |
| 6 | Лучше заменить на использование либы <ctime>, т.к. обычно использование сишных либ задепрекейчено в плюсовом коде | 0 |
| 103 | Опечатка: MoveConstructible -> MoveConstructable | 0 |
| 163 | Опечатка: Sometihing -> Something | 0 |