Фролов Никита 7 группа 7 вариант

1.

2. Поток - объект ядра, которому операционная система выделяет процессорное время для исполнения

3. Мьютекс - объект ядра, находящийся в одном из сигнальных состояний, который используется для решения проблем взаимного исключения между потоками

4. Событие - мех. синх. который уведомляет для согласования действий между потоками или процессами, способное находиться в двух состояниях

5. Сравнительный анализ C++98
   1. Наличие STL
   2. Поддержка Throws
   3. RTTI
   4. новые слова: "bool", "explicit", "typedef"
   5. Облегчают разработку программ
   6. Улучшение производительности

Вторая группа:

ООП - программирование - процесс разрешения

5. ~~C++98~~

~~Без Boost~~ ~~С Boost~~

Вторая группа:

1. ООП-декомпозиция — это процесс разделения сложной системы на более простые и понятные компоненты, которые могут быть легче управляемыми

Аспекты ООП-дизайна:

1. Идент. объектов
2. Определение классов
3. Инкапсуляция
4. Определение отношений
5. Абстракция
6. Полиморфизм
7. Упр. зависимостями

2. Статический полиморфизм относится к использованию перегрузки функций шаблонов и других механизмов языка на этапе компиляции для обеспечения множественности форм

3. Инкапсуляция — принцип ООП, который представляет собой механизм ограничения доступа к состоянию объекта и предоставление доступа к его функциональности через интерфейс

Инкапсуляция позволяет скрывать внутренние детали реализации объекта и обеспечивает его незав. от вн. изменений

Factory Method - паттерн проектирования который предоставляет интерфейс некоторого класса для создания экземпляра некоторого подкласса. Он позволяет изменять тип создаваемого

Структура:

Product

Concrete Product

Creator

concrete Creator

Пример

class Logger{

}...

class ConsoleFileLogger : public FileLogger{

...

}

class FileLoggerFactory : public FileLogger{

...

}

В примере "Logger" представляет собой интерфейс "Logger Factory" - инт. соз. логгеров, а "FileLogger" - реализацию "FileLogger Factory" - соз.

Template Method - ~~фактор~~ дизайн
поведения, который опр. скелет алгоритма в суперклассе, но позволяет подклассам переопределять определенные шаги алгоритма без изм. его структуры. Этот шаблон используется для создания алгоритма, позволяющего опр. вариации данного реализуем с помощью подклассов

```
class AbstractClass {
    templateMethod() {I,II,III}
}

class ConcreteClassA: public AbstractClass{
    void I {}
    void II {}
    void III {}
}

class ConcreteClassB: public AbstractClass{
    void I {}
    void IV {}
    void III {}
}
```

github.com/Trinper