```cpp
// (1) Ridefinire il distruttore di DataBlock
~DataBlock() {
    delete[] values;
}

// (2) Ridefinire il costruttore di copia di DataBlock
DataBlock(const DataBlock& other) : size(other.size), values(nullptr) {
    if (size > 0 && other.values != nullptr) {
        values = new double[size];
        for (int i = 0; i < size; i++) {
            values[i] = other.values[i];
        }
    }
}

// (3) Ridefinire l'operatore di assegnazione di DataBlock
DataBlock& operator=(const DataBlock& other) {
    if (this != &other) {
        delete[] values;
        size = other.size;
        values = nullptr;

        if (size > 0 && other.values != nullptr) {
            values = new double[size];
            for (int i = 0; i < size; i++) {
                values[i] = other.values[i];
            }
        }
    }
    return *this;
}

// (4) Implementare il distruttore di AdvancedProcessor
~AdvancedProcessor() {
    delete extraData;
}

// (5) Implementare l'operatore di assegnazione di AdvancedProcessor
AdvancedProcessor& operator=(const AdvancedProcessor& other) {
    if (this != &other) {
        // Chiamare l'operatore di assegnazione delle classi base
        StatisticalProcessor::operator=(other);
        RegressionProcessor::operator=(other);

        // Copiare i membri propri
        normalized = other.normalized;
```

```
        // Gestire la memoria allocata dinamicamente
        delete extraData;
        extraData = nullptr;

        if (other.extraData != nullptr) {
            extraData = new DataBlock(*other.extraData);
        }
    }
    return *this;
}

// (6) Implementare il metodo process() di AdvancedProcessor
void process() override {
    // Chiamare entrambe le implementazioni di preprocess()
    StatisticalProcessor::preprocess();
    RegressionProcessor::preprocess();

    // Calcolare slope e intercept basati sui dati
    // (questa è una implementazione fittizia, il calcolo reale dipenderebbe
dai dati)
    slope = mean * 0.5;  // Esempio: slope dipende dalla media
    intercept = variance * 0.2;  // Esempio: intercept dipende dalla
varianza

    // Impostare normalized in base alla varianza
    normalized = (variance < 1.0);
}

// (7) Implementare il metodo di clonazione di AdvancedProcessor
AdvancedProcessor* clone() const {
    return new AdvancedProcessor(*this);
}
```

Spiegazione

1. Distruttore di DataBlock

Il distruttore di DataBlock si occupa di deallocare la memoria allocata dinamicamente per l'array `values`. Questo previene memory leak.

2. Costruttore di copia di DataBlock

Il costruttore di copia implementa una copia profonda. Inizializza `size` con il valore di `other.size` e, se necessario, alloca un nuovo array per `values` copiando tutti gli elementi dall'oggetto originale.

3. Operatore di assegnazione di DataBlock

L'operatore di assegnazione implementa anche una copia profonda, ma deve prima deallocare la memoria esistente. Include una verifica per l'auto-assegnazione (`this != &other`). Imposta `values` a `nullptr` prima della potenziale allocazione, garantendo che se l'allocazione fallisce l'oggetto rimanga in uno stato consistente.

4. Distruttore di AdvancedProcessor

Il distruttore di AdvancedProcessor si occupa di deallocare la memoria allocata per `extraData` . I distruttori delle classi base verranno chiamati automaticamente.

5. Operatore di assegnazione di AdvancedProcessor

L'operatore di assegnazione chiama prima gli operatori di assegnazione delle classi base, poi copia il membro `normalized` e gestisce la memoria allocata per `extraData` con una copia profonda. Include una verifica per l'auto-assegnazione.

6. Metodo process() di AdvancedProcessor

Il metodo `process()` chiama entrambe le implementazioni di `preprocess()` delle classi base, calcola `slope` e `intercept` (qui con una implementazione fittizia) e imposta `normalized` a `true` se la varianza è minore di 1.0.

7. Metodo di clonazione di AdvancedProcessor

Il metodo `clone()` crea un nuovo oggetto `AdvancedProcessor` usando il costruttore di copia, implementando così la semantica di clonazione polimorfa.