話題・解説 (IV)

C++プログラミング言語を用いた理論計算コードの開発

日本原子力研究所
核データセンター
岩本　修
iwamoto@ndc.tokai.jaeri.go.jp

1. はじめに

現在、私は C++プログラミング言語を用いて、核データ評価のための理論モデル計算コードを開発しています。C++はオブジェクト指向プログラミングをサポートしているプログラミング言語の一つで、C 言語を拡張して作られたものです。特徴として、C から受け継いだ低レベルな記述から、オブジェクト指向を用いた高レベルな記述まで可能であることが挙げられ、オブジェクト指向プログラミングを行いながら、実行効率を上げることが可能となっています。核データでは要求される反応が非常に多く、核データ評価のための理論計算では、色々なモデルを組み合わせる必要があり、データ構造も複雑になってきます。そこで扱うデータや理論モデルをオブジェクトとして扱うことにより、複雑さを見やすい形で分割できれば柔軟性・拡張性を持ったコードが開発可能であると考えて C++言語を用いて計算コードを開発することにしました。

核データ関係のコードでは C++等のオブジェクト指向言語を用いたものがほとんど無く、珍しいということで、この文章の依頼が来ました。オブジェクト指向言語を理論計算に用いる利点についての解説という事でしたが、私にこれらの包括的な説明を行うことは不可能ですので、ここではオブジェクト指向と私の開発しているコードの一部についての説明を簡単に書きたいと思います（だだし、オブジェクト指向の説明の部分は、眉につばを多めにつけて読んでください)。

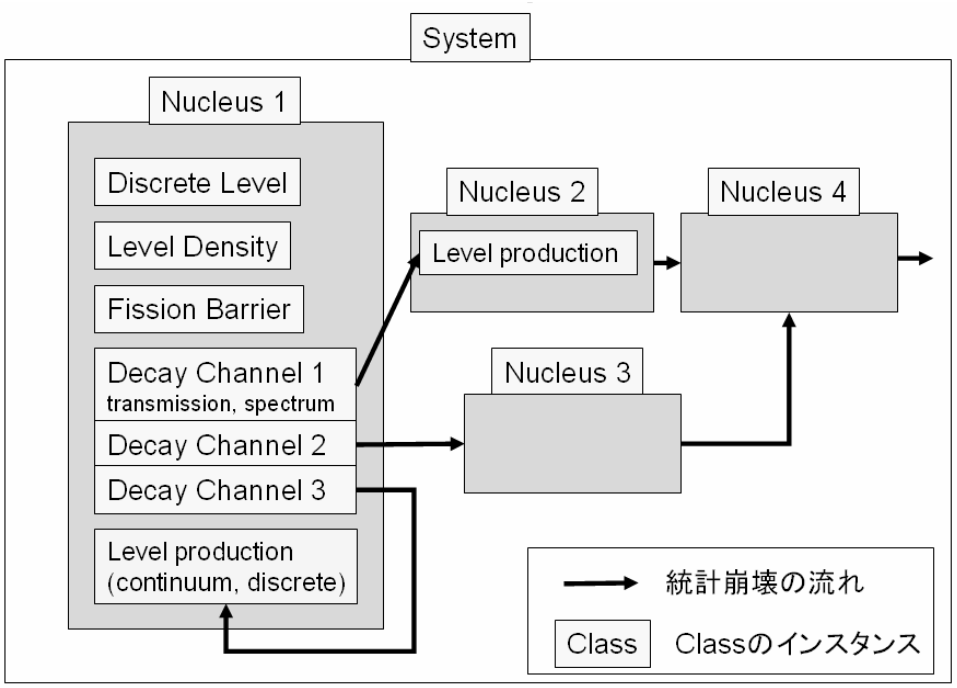
2. オブジェクト指向

オブジェクト指向プログラミングでは、データとその操作が一体となったオブジェクトを基本的な単位としてコードが構成されます。従ってオブジェクトをどういう風に設計するかが重要となります。C++ではオブジェクトの設計図として class というキーワー

ドが使われ、そのクラス内に定義されたメンバー関数を通してオブジェクトの操作を行います。ここで大切なのは、クラスの内部データ構造を外から隠し、メンバー関数を通してオブジェクトの操作をすることです。それにより、特定の操作を持ったオブジェクトを個々のオブジェクトの詳細に関わらず処理できるようになります。これがオブジェクト指向の大切なポイントです（しかし現実には、クラスの設計にばかり時間をかけてもいられないので、私のコードでは情報の隠蔽が十分で無い部分が多々あります。徐々に良いものにしていきたいとは考えています)。内部状態を変更することで、多くのオブジェクトを表現できますが、クラスの継承を行うと更に多様な同種のオブジェクトが作成することが可能です。クラスの継承とは親クラスの性質を引き継ぎ、親クラスとの違う部分のみ定義しなおす事です。この他にも C++にはテンプレートを使ったジェネリックプログラミングやそれを使った便利なテンプレートライブラリがありますが、ここでは説明を割愛します。

3. 統計モデルコード CCSM

私が現在開発しているコードは、光学モデル、DWBA、チャンネル結合モデル、励起子モデル、Hauser-Feshbach 統計モデル等で構成されています。それぞれの部分は、独立性を高めたものとしています。これらの中で、核データ評価計算の主要な部分である統計モデルコード（CCSM）について、クラスの概略を説明します。次の図は CCSM でのクラスの構造を表しています。



全体を System オブジェクトとし、その中に統計崩壊で生成される原子核を Nucleus クラスのオブジェクト（オブジェクトはインスタンスとも呼ばれる）Nucleus1, Nucleus2,…として作ります。Nucleus クラスは原子核の構造の情報をもつ Discrete_Level、Level_Density 等のクラスのオブジェクトのほか、統計崩壊を記述する複数の Decay_Channel クラスのオブジェクトを持ちます。Decay_Channel は粒子放出やガンマ線放出、核分裂などの種類があり、これらは Decay_Channel クラスを継承してそれぞれ、Decay_Channel_Particle、Decay_Channel_Gamma、Decay_Channel_Fission クラスとして設計しています。個々の Nucleus オブジェクトが持つ Decay_Channel オブジェクトの統計崩壊メンバー関数を呼び出すことを、繰り返し行うことで最終的に原子核が安定状態に至るまで統計崩壊を行わせます。統計崩壊で最終的に生成される原子核及び放出粒子の生成量から、核反応の断面積を得ることができます。

（注：ここで用いたクラスの名前等は分かりやすくするため実際のコード中の名前から変えています。）

4. おわりに

理論モデル計算では、実行速度も重要なファクターになりますが、実行速度と構造的なプログラムの美しさとは、相容れないことが多いです。実行速度を上げる有効な方法として、同じ計算は一度だけ行い、計算結果を保存しておいて複数の部分で結果を共有する事がありますが、これを行うと独立していたコード間に相関がでてきて、保守しにくくなります。きれいなコードを書くことは大変ですが、コードが大きくなってくるとバグを防いだり、改良し易くしたりする為に、非常に大切なことです。オブジェクト指向言語を使うと、プログラム構造について意識せざるを得ず、言語仕様の助けを借りて整ったコードを作成しやすくなっていると思います。