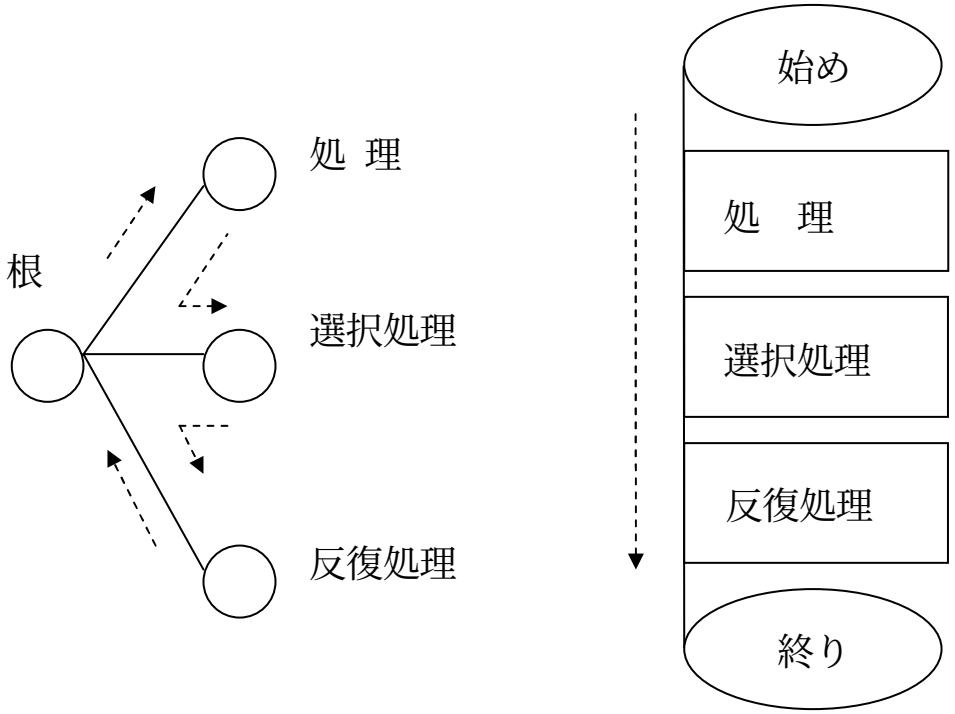
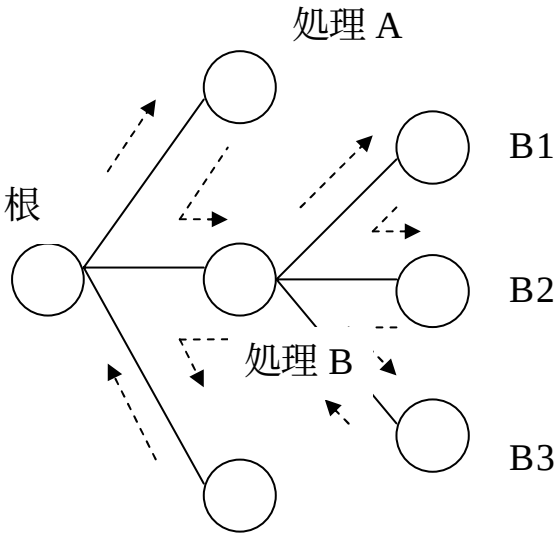
部品を再利用する効率的ソフトウェア開発

川端　信賢

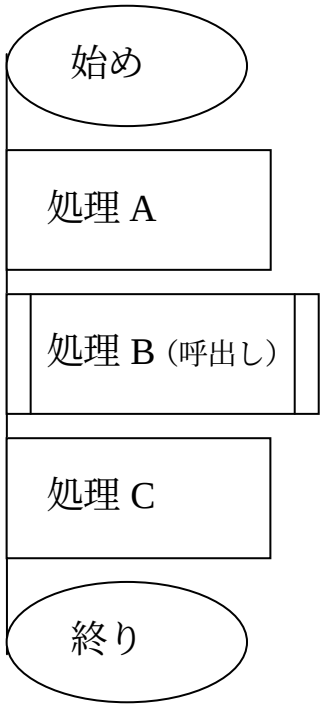
最近はソフトウェアの製造と信頼性について､かつてほどの大きな心配は聞かれなくなりました。ひとむかし前には「ソフトウェアの危機」がさけばれました。大型コンピュータの基本ソフトウェア（OS）の巨大化によって、ソフトウェアの諸問題が浮かび上がったのです。当時、あるメーカのOS の部品数は、大型タンカーの部品数よりも多いといわれました。なるほどと思いました。スペースシャトルや巨大ソフトウェアのような部品数が大変多いシステムを、無事に完成させる。それは至難のわざです。各部品の信頼性を 0．99(99 パーセント)まで高めても、10 万個の部品からなるシステムの信頼性はほぼ 0 です。これではまったくだめですね。部品の信頼性を 0．999999（6 けた）まで上げてはじめて、このシステムの信頼性は 0．90 になります。ソフトウェア危機の背景には、プログラムの作成技術と信頼性に関して、次のような問題があったと思います。すなわち、たとえ大きなプログラムであっても「見通しのよい形」に書く技法、そういう書き方が可能なプログラミング言語、うっかりしたプログラムエラーが入り込みにくい仕組み、部品の修正に伴う全体の修正の最小化、部品の再利用、能率的ソフトウェア開発、（ソフトウェア開発を成功に導く）プロジェクト管理、さらに、ソフトウェア技術者の養成などです。前述のように、期待したとおりに動作する巨大ソフトウェアの作成は容易ではありません。しかし、地道な努力のおかげにより、プログラミングの技術は洗練され、問題はだんだんと解決されてきました。

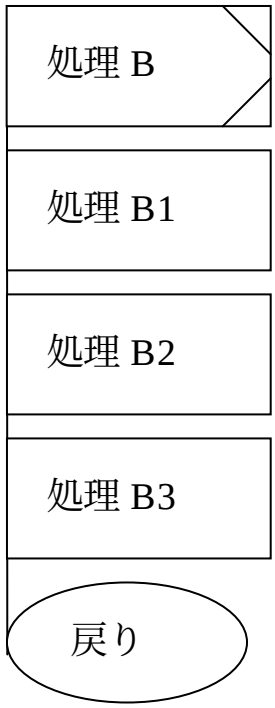


構 造 化 プ ロ グ ラ ム の 構 造 例



プログラムの階層木構造例



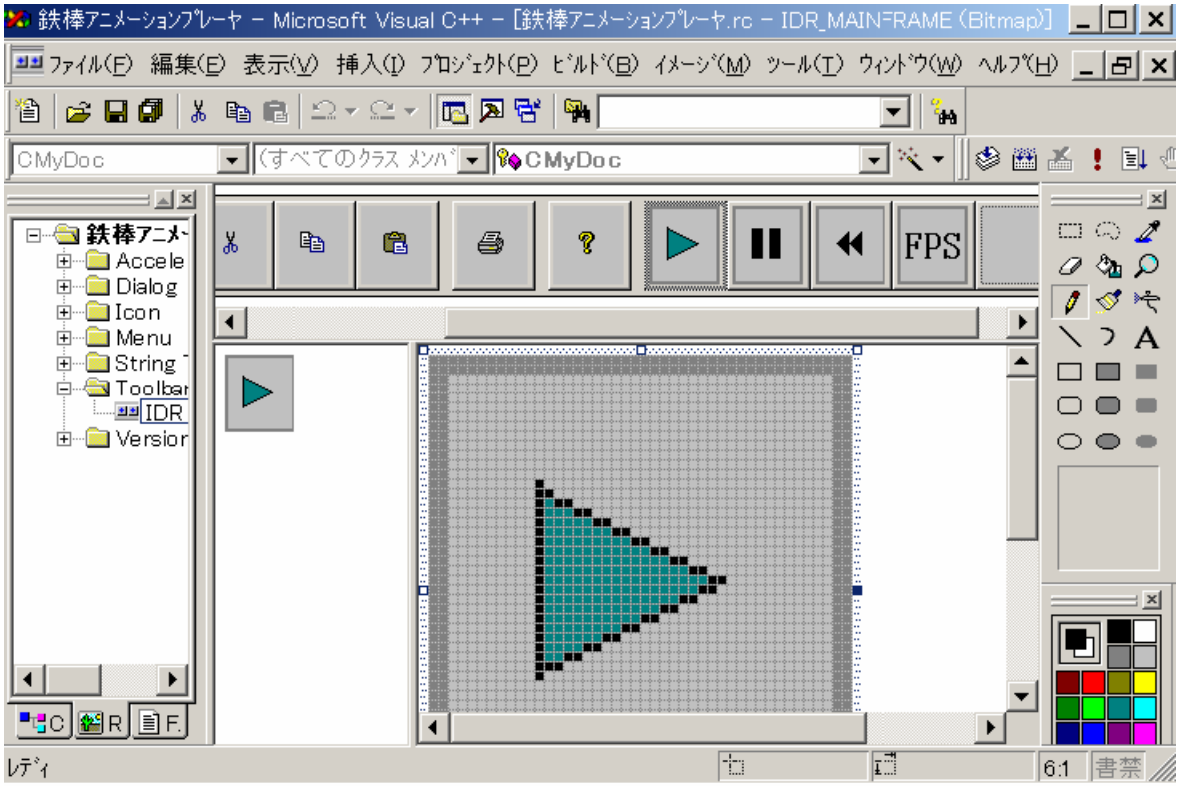


階層木構造を
各段に分けて書く

それでは洗練されてきた、現在のプログラミング技術の象徴は何でしょうか。それはいわゆる「構造化プログラミング」と「（オブジェクト指向の）統合開発環境」だと思います。構造化プログラミングはプログラムをジャンプ命令、すなわち、「goto 文」なしで書こうというものです。goto 文はプログラムをわかりにくくするので、goto 文なしにするわけです。この代わりに「反復」または「選択」という処理が使えます。この結果、構造化プログラムは「（通常）処理」、「反復処理」、「選択処理」の三種類の処理単位をたて一列に並べた形になります。すなわち、構造化プログラム全体の実行順序はプログラムの上方から下方へだけ流れます。このため、わかりやすいです。これが構造化プログラムの長所です。今では構造化プログラミングは標準の技法です。ある問題のコンピュータによる処理手順を明確に書いたものを「アルゴリズム」といいます。構造的に書かれたアルゴリズムからは、容易に構造化プログラムが書けます。

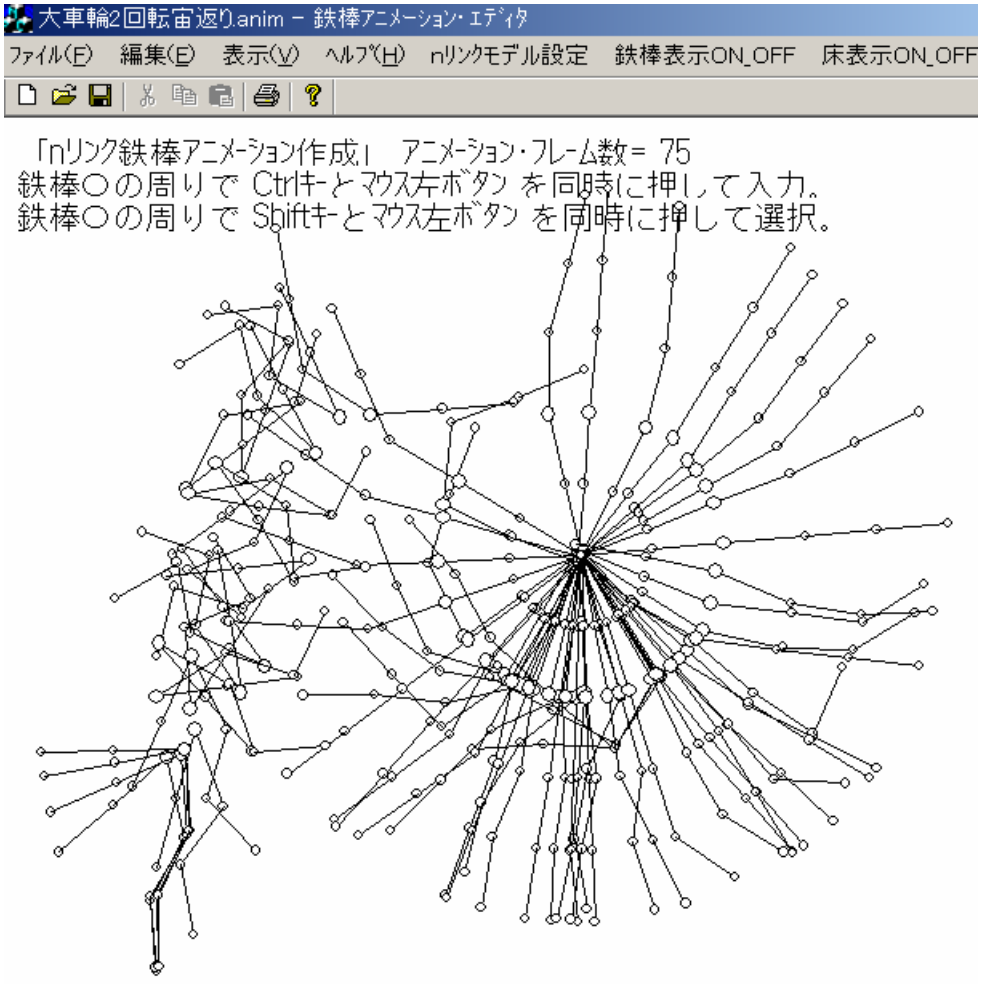
ところで、実際のプログラムは大きいために、これを見通しのよい構造に書かなければなりません。率直に言えば、たとえ大きいプログラムでも、それをしかるべき順序で見れば、抵抗感なく読み進める構造、といってもいいでしょう。そのためにはプログラムの構造を、会社の組織構造や本の章節構成のように、「階層木構造」にします。この名称は植物の木の形からきています。まず、プログラムの処理全体を、覚えやすい少数の処理グループに分けます。同じグループには関連の強い処理をまとめます。これが一段目です。次に、一段目のグループをまた少数の処理グループに分けます。二段目です。このように各グループがそれ以上分ける必要がなくなるまでグループ分けを行います。プログラムの実行順序は木の根っこから始まり、根から出ている一番上側の枝に進みます。次に、その先の一番上の枝へ、先の枝がなければ、元の枝を戻って下隣の枝へと進みます。こうして木全体を一筆書きするように枝をたどり、基本的に木の根っこに戻って実行を終わります。

実際には、プログラムの記述は画面やノート幅に収まるようにしなければなりません。そこで、階層木はそのままの形ではなく、一段目、二段目と各段に分解して書きます。こうして、見通しのよいプログラムができます。なお、プログラムやデータを磁気記憶装置に保存する際にも、階層木構造に整理して保存します。



統合環境によるプログラムの作成過程

次に「統合開発環境」はビジュアルで能率的なプログラミング環境です。ただし、ここではオブジェクト指向の統合環境です。なぜかというと、オブジェクト指向には、ソフトウェアの諸問題の解決を図る仕組みがあるからです。「オブジェクト」はプログラムの構成要素になる賢い部品です。オブジェクトは固有のデータとデータに対する処理とを併せ持ちます。あるオブジェクトをプログラムの部品に使うには、まず、オブジェクト自身の内部構成を表す「クラス」を用意しなければなりません。たとえば、ある統合環境のクラスライブラリ（参照クラス集）では、合計 230 種類のクラスになります。



統合環境で作ったプログラムの画面例

それでは統合環境のクラスライブラリは、どんな役に立つのでしょうか。統合環境の信頼性の高いクラスライブラリは、ソフトウェア開発に何回でも使われます。統合環境では、まず、プログラムの骨格部分（骨格プログラム）を自動的に生成します。そして、この骨格プログラムにオブジェクト、データ変数、処理などをマウス選択によって組み込みます。また、オブジェクト内部の修正

に伴う、プログラム全体の修正を少なくできます。手入力を大幅に軽減します。メニュー、ダイアログボックスのような、分かりやすくて、使いやすい、グラフィカル・ユーザ・インタフェース（GUI）のプログラムになります。このように、統合開発環境は、ソフトウェア開発の効率化と信頼化の土台です。本学部では、これを実習形式で学ぶことができます。