1D－3

グラフィカルエディタ作成用ツールキット GIST (3)
— ユーティリティー —

Paul C. Brown　Terry M. Topka
General Electric Company, Corporate Research and Development
井上 健　○土田 崇　上原 みな　村田 なつめ　渡邊 多恵子
横河電機株式会社 オープンシステム研究所

1 序

GISTは，GIST自身やそのアプリケーションを構築するのに有用な汎用のC＋＋ユーティリティライブラリを提供している．GISTの用意している基本クラスGtObjectから導出される任意のクラスは，次のような機能を簡単に利用することができる．

- オブジェクトの参照数やメモリの自動管理を行なうポインタや，さまざまなコンテナクラスを定義することができる．
- オブジェクトのネットワークの任意の部分をコピーするための機能があり，ポインタの付け換えなどを自動的に行なうことができる．
- 汎用のアンドゥー機能があり，アプリケーション中のオブジェクトの状態を記録・復元することができる．

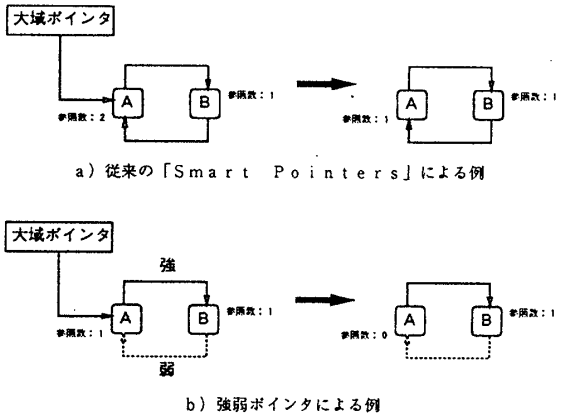
以下ではこれらの機能について，詳しく説明する．

2 ポインタとコンテナクラス

2.1 強ポインタと弱ポインタ

C＋＋アプリケーションにおけるメモリ管理の煩雑さ軽減のための「Smart Pointers」は既に提案されている．これらは通常のポインタと同じように振舞い，更にポインタによるオブジェクトの参照数を管理することにより，メモリを解放する手順の簡略化を可能にしている．しかしオブジェクトどうしが相互に参照するような循環参照が存在する場合，これらのオブジェクトは永久に破壊されないため，メモリ・リークが発生してしまうことがある．

GIST — A Toolkit for Graphical Editor Development
Paul C. Brown, Terry M. Topka
Corporate Research & Development, General Electric Company
P.O.Box 8, Schenectady, NY 12301, USA
Takeshi Inoue, Takamu Tsuchida, Mina Uehara,
Natsume Murata, Taeko Watanabe
Open Systems Laboratory, Yokogawa Electric Corporation
2-9-32 Nakacho, Musashino-shi, Tokyo, 180 Japan



a）従来の「Smart Pointers」による例

b）強弱ポインタによる例

図1．循環参照

GISTでは，参照数を管理するポインタを強ポインタおよび弱ポインタの2種類に分けている．強ポインタが指しているオブジェクトは，そのポインタが別のオブジェクトを指すように（またはヌルに）書き換えられることにより，参照数が減じられる．強ポインタによる参照数が0になると，そのオブジェクトは自動的に破壊される．一方弱ポインタは，参照しているオブジェクトが破壊されるときにその値が自動的にヌルにセットされる．したがって，破壊されたはずのオブジェクトを指すようなポインタは存在し得ない．

循環参照の問題を回避するためには，その循環のうちの一つのポインタを弱ポインタとし，残りを強ポインタを使用するように構成するだけでよい．強／弱ポインタを使用することにより，明示的にdelete演算子を使用せずにメモリ・リークを起こさないプログラムを作成することができる．

GISTには，任意のGtObjectのサブクラスの強ポインタや弱ポインタを定義するためのマクロが用意されている．

2.2 コンテナクラス

上記の強／弱ポインタを要素とするコンテナ（配列，順序付集合，集合，連想配列，連想表）クラスを定義

することもできるようになっている．これらのコンテナクラスでは，要素の追加／挿入／削除など（そのためのメモリ管理は自動的になされる）を行なったり，要素の探索などの問い合わせをしたりするためのさまざまなメンバ関数が用意されている．

コンテナクラスを使用する場合にも，その定義のためのマクロを呼び出せばよい．

3 コピー機能

エディタのような対話型アプリケーションでは，編集対象を表現するオブジェクトをコピーする機能は必須であるが，これらのオブジェクトは通常ネットワーク状に結び付いている．一般にコピー機能として必要なのは，オブジェクトのネットワークの任意の部分の複製を作り，オブジェクトの持っているポインタを適当に書き換えるという処理である．GIST ではこのようなコピー処理を実現するためのメソッドを GtCopyObject クラスに用意している．

オブジェクトのネットワークのコピーは以下のような手順で行なわれる．

(1) ネットワーク中のオブジェクトのうち，複製すべきオブジェクトを決定し，複製後のオブジェクトが持つポインタが何を指すようにするかを決定する．
(2) 複製を作成する．
(3) 複製オブジェクトのポインタを更新する．
(4) コピー後に必要な後処理を実行する．

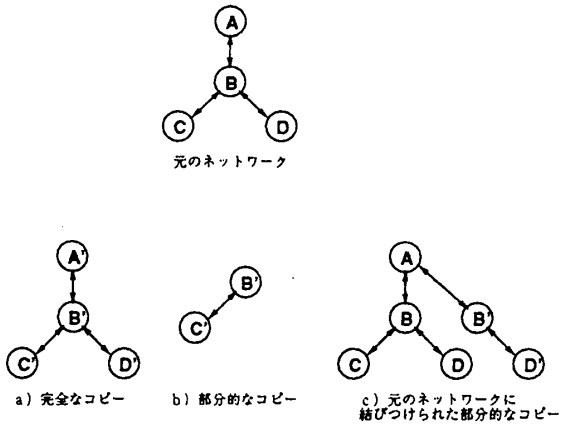


図2．さまざまなコピー方法の例

コピーを実行する場合には，ネットワーク中のオブジェクトのうち，i) 複製を作りたいオブジェクト，ii) 複製を作らないが，複製前のオブジェクトが参照しており，かつ複製のオブジェクトでは参照先を書き換えるもの，およびその更新後の参照先，iii) 複製を作らず，複製前のオブジェクトが参照していたポインタが複製のオブジェクトではヌルに書き換えられるオブジェクト，の3種類をそれぞれ指定した後，コピー実行メソッド（`GtCopyObject::execute()`）を呼び出す．コピー後に必要な後処理があるオブジェクトのクラスでは，その処理を行なうメソッドを定められた名前で（`postCopyFixup()`）を用意しておく．

GIST ではそのアーキテクチャに基づいて上記必要な3種類のオブジェクトを計算するメソッドや，提供しているクラスで必要な後処理を行なうメソッドを用意することにより，GIST アプリケーションでコピー処理をより容易に実現できるようにしている．

4 アンドゥー機能

アンドゥー／リドゥー機能も，対話型アプリケーションでは欠かせない機能である．GIST ではアンドゥーとはアプリケーション中のオブジェクトのある時点における内部状態（属性の値）を予め保存しておいて，それを復帰すること，リドゥーとはアンドゥー操作を行なう前の状態に復帰することであるととらえる．

GIST にはオブジェクトの任意の時点の状態を保存／復帰するためのメカニズムが用意されている．アプリケーションでは，オブジェクトの持つメソッドが自分自身の属性の値を変更する場合，変更する直前に状態を記録する処理を呼び出すようにしておくだけで，任意の回数のアンドゥー／リドゥーを実現することができる．

5 おわりに

GIST の一部である汎用のユーティリティライブラリには，以上に説明した機能の他，オブジェクトの内容のファイルへの入出力機能を現在開発中である．これらの機能を用いることにより，GIST を使用した対話型アプリケーションをより容易に，かつ高い信頼性をもって開発することが可能になる．

[参考文献]

[1] M. A. Ellise and B. Stroustrup, *The Annoataed C++ Reference Manual*, Addison-Wesley, 1990.