2H−3

永続プログラミング言語INADAの機能を実現するための C++言語プリミティブの設計

天野浩文　有次正義　寺本圭一　牧之内顕文

九州大学工学部情報工学科

1．まえがき

ＣＡＤやマルチメディア処理など複雑なデータ構造を使用する必要のある応用分野では，再利用されるデータをプログラムの終了時にファイルのような永続的な媒体に保存しなければならない．しかし，そのような複雑なデータ構造を表現するのに通常のプログラミング言語が使用しているポインタは個々のデータが記憶空間内のどこに取られているかという番地の情報であるため，それらを記憶空間内の異なる位置に取られてもよいような情報（例えばファイル内でのオフセット）に変換したのち保存しなければならない．

通常のプログラミング言語では，このような揮発データ・永続データ間の変換はプログラムの責任で行わなければならないが，これを言語の機能として提供しようとする永続プログラミング言語[AB87]の考え方が注目されている．

現在開発中の言語ＩＮＡＤＡ[AA91]は、並列分散マルチメディア応用のための４つの永続プログラミング言語群「出世魚」の第３層を構成するオブジェクト指向永続プログラミング言語である。ＩＮＡＤＡはオブジェクト指向プログラミング言語Ｃ＋＋を拡張し、プログラムの起動・終了を越えて存続する永続データを通常の揮発データと同様に扱う機能、および、永続データに対する問い合わせの機能を提供する。これらの機能はＣ＋＋の言語仕様には含まれていないが、これを実現するＣ＋＋のライブラリを構築することができる。これらのプリミティブは，ＩＮＡＤＡの処理系が生成する中間言語（Ｃ＋＋）コードの基本となるほか，Ｃ＋＋による一般の応用プログラムでも利用可能なライブラリを構成する．

本稿では、ＩＮＡＤＡを構成するために開発中のＣ＋＋プリミティブクラスとその使用法の概要について述べる。

2．ＩＮＡＤＡの概要

クラス定義などに関するＩＮＡＤＡの構文はＣ＋＋に準拠する．ただし，揮発オブジェクトがＣ＋＋と同様の扱いを受けるのに対し，永続オブジェクトは実行時記憶空間の中でディスク上のファイルと１対１に対応付けられた**永続ヒープ**（Persistent Heap, PH）領域[AA91]に取られる．

変数を宣言する際には，それが永続オブジェクトであることを明示するキーワード“persistent”を前に置く．

persistent 《クラス名》＊ 《変数名》；

“persistent”と宣言されたオブジェクトは，プログラムの終了後もファイルに保存される．以前作成された永続オブジェクトを再利用する場合には，そのＰＨファイルを作成時と同じように実行時記憶空間にマップする．

これらのオブジェクトのうち与えられた条件を満たすものを一括して操作を加えるために，以下のような構文を用意する．

for all 《変域》 ［**such that** 《検索条件》］
　do 《文》；

また，同一のクラスの永続オブジェクトの集合を一時的に保持するような集合変数の使用も許す．

multiple 《クラス名》 《集合変数名》；
《集合変数》 ＝ 《変域》 **where** 《検索条件》；

また，“multiple”と宣言された変数は前述の一括操作文の《変域》としても使用できる．

3．ＰＨ領域の実現

本節では，ＩＮＡＤＡにおける永続性の実現方式であるＰＨ領域の管理，および，永続オブジェクトに対する擬似ポインタを実現するためのプリミティブクラスについてその設計の概要を述べる．

3.1 ＰＨ領域とクラスBTPHeap

ユーザプログラムは，ＰＨ領域の使用を開始する際に，クラスBTPHeapのインスタンスを生成する．このインスタンスは，ファイルのオープン，１つのＰＨ領域と１つのファイルとのマッピング（新規のファイルの場合にはその生成と初期化も含む），および，同領域内の記憶管理からファイルのクローズまでを担当する．記憶域管理にここでは境界タグ方式を用いている．

あるクラスのインスタンスのうち“persistent”と宣言されたものについては，すべて同一のＰＨファイルに格納される．

ユーザプログラム中に“persistent”宣言がある場合，ＩＮＡＤＡの処理系は，当該クラスの「永続オブジェクト版」をユーザプログラムのクラス定義に付加する．

この永続クラスのコンストラクタでは，該当するＰＨ領域を管理するBTPHeapのインスタンスに領域確保を行わせ，続いて，確保させた領域に必要な値を書き込む．同時に，オブジェクト参照テーブルにこのオブジェクトのエントリを登録する．

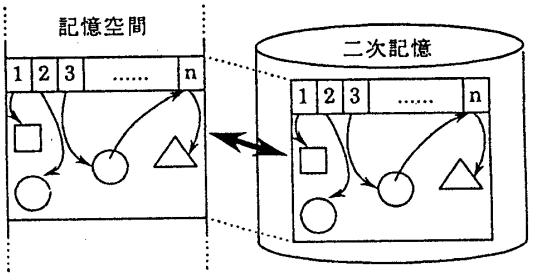

図１：ＰＨ領域と永続オブジェクト

Designing C++ Primitives for the Persistent Programming Language INADA
H. Amano, M. Aritsugi, K. Teramoto, and A. Makinouchi
Department of Computer Science and Communication Engineering, Kyushu University

3.2 永続オブジェクトへの擬似ポインタ

ＰＨ領域は実行時記憶空間のどの位置に取られても同じ情報が再生できなければならない．このため，通常のポインタで表現される情報を，リロケータブルな情報として記録しなければならない．しかし，ユーザプログラムの動作中は，これらがあたかも通常のポインタであるかのように扱えることが望ましい．

そこで，ＰＨ領域内でのオフセット値を保持しながらポインタのように振る舞う**擬似ポインタ**のクラスを考える．

ユーザプログラム中に，

persistent 《クラス名》＊《ポインタ変数名》；

のようなポインタの宣言がある場合，ＩＮＡＤＡ処理系はこれに対応する擬似ポインタクラスの定義を付加するとともに，上記の宣言を

《擬似ポインタクラス名》 《ポインタ変数名》；

のように置換する．

このクラスは，ポインタ演算子“->”，“*”，“&”を多重定義することによって，実際にはオフセット値である擬似ポインタ値をＣ＋＋における通常のポインタと同様の

《擬似ポインタ》→《メンバ関数》；

といった構文で使用することを可能にする．オフセット値から実効アドレスへの変換は，BTPHeapの保持するＰＨ領域開始番地を用いて擬似ポインタクラスで自動的に実行される．したがって，上記の宣言以降のユーザプログラム中で「永続オブジェクトを指すポインタ変数」を用いていた部分にＩＮＡＤＡ処理系で変換を加える必要がない．

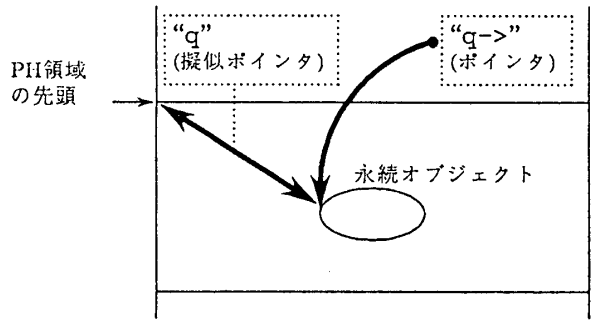

図２　擬似ポインタと実効アドレス

4．永続オブジェクト再利用のための機能について

本節では，マップされたＰＨファイルの中に保存されている永続オブジェクトを再利用するための機能について，これらを実現するためのＣ＋＋中間コードの設計方針を述べる．

4.1 問い合わせ機能

２節で述べた“for all”文は既存の永続オブジェクトに対するごく単純な問い合わせ機能を提供する．

《変域》がクラス名である場合には，ＩＮＡＤＡ処理系はＰＨファイル内のオブジェクト参照テーブルのエントリを順次探索しながら《文》を実行するようなＣ＋＋のループを生成し，この“for all”文を置換する．

一方，《変域》に集合変数を用いる場合には，このクラスのインスタンスを複数個保持できるデータ構造（例えばリスト構造）を持つクラスを導入する必要がある．

このクラスに，条件を満たす要素の検索，要素の挿入・削除，要素数のカウント，集合演算，集合対集合の比較比較演算などのメンバ関数を用意することによって，条件検索の高度化を行える．

この集合オブジェクトに対する一括操作文に対して，処理系は，対応する集合オブジェクトのクラスをユーザプログラムに付加するとともに，この集合オブジェクトの要素を探索して《文》を実行するようなループを生成する．

集合オブジェクトを用いる場合には，検索条件の性質に応じて，索引などの副次的データ構造を利用して処理の効率化を図れることがある．このため，ライブラリには集合の表現のための種々のデータ構造とその上での操作を具現するクラスを用意しておき，ＩＮＡＤＡプログラム中では処理系に対する指令をおくことによってこれらの中から選択できるようにすことを計画している．

4.2 パースペクティブ／アスペクト

永続オブジェクトに対して複数の「役割」や当初の定義とは異なる「見方」を付与しなければならないこともありうる[SB86,RS91]．

そこで，ＩＮＡＤＡでは，プログラムＡが作成した永続データオブジェクトを別のプログラムＢが再利用する場合に，Ａで用いられたクラス定義にＢで新たな変数やメンバ関数を付加して「拡張」するための機構を導入する．

ここで，「拡張」されたオブジェクトは，新規の変数の他に「古い」永続オブジェクトへのリンクを備えるように変換される．「古い」定義にあるメンバ関数の起動は，このリンクの先のオブジェクトのクラスのメンバ関数の起動へと置き換えられなければならない．

5．むすび

本稿では，永続プログラミング言語ＩＮＡＤＡの機能を実現するためのＣ＋＋プリミティブクラスの概要について述べた．

現在，境界タグ方式によるＰＨ領域の管理を行うクラスBTPHeapを中心に，本稿で述べた種々のプリミティブクラスの設計と実装を進めている．

参考文献

[AA91] 天野，有次，牧之内：“永続プログラミング言語ＩＮＡＤＡとその問合せ機能”，情報処理学会第43回(平成3年後期)全国大会，5N-2，平成3年10月．

[AB87] Atkinson, M.P. and Buneman, O.P.: “Types and Persistence in Database Programming Languages,” ACM Comput. Surv., Vol.19, No.2, pp.105-190, June 1987.

[AG89] Agrawal, R. and Gehani, N. H.: “ODE(Object Database Environment): The Language and the Data Model,” Proc. the 1989 ACM SIGMOD Int. Conf. on Management of Data, pp. 36-45, May 1989.

[NW91] Nguyen, T.A., Wagner, M. et al.: “PC++: An Obejct-Oriented Database System for C++ Applications,” Proc. the 2nd Int. Workshop on Database Systems for Advanced Applications, pp.109-115, April 1991.

[RS91] Richardson, J. and Schwarz, P.: “Aspects: Extending Objects to Support Multiple, Independent Roles,” Proc. ACM SIGMOD Int. Conf. on Management of Data, pp.298-307, May 1991.

[SB86] Stefik, M. and Bobrow, D.G.: “Object-Oriented Programming: Themes and Variations,” The AI Magazine, Vol.6, No.4, 1986.