DATABASE

第1章

コンテナからデータベースへ

組み込みシステムにおける大容量データの管理技法

街谷 君次

プログラミングにおいて，データの扱いは重要である．適切なアルゴリズムを選択することによって性能が向上し，フットプリントも小さくなる．ここではデータを扱う基本的な手法として，初めにコンテナについて解説する．続いてコンテナよりも強力であるデータベースについて述べる．（編集部）

1 アルゴリズム＋データ構造＝プログラム

● データ構造が重要

「アルゴリズム＋データ構造＝プログラム」ということばがあります．これはN. Wirthの古典的名著“Algorithms + Data Structures = Programs”のタイトルからきたことばですが，まさにプログラムの本質を的確に突いているといえるでしょう[注1]．

プログラムを書いていて感じるのは，「結局，プログラムというのは，入力されたデータを演算して出力する，という単純な機械」ということです．それゆえ，データ構造をどのように定義し，それをどのようなアルゴリズムで処理するのか，といった面に注力することになります．

● どのようにデータをもつか

プログラミングにおいてデータを保持する手法にはいろいろなものがあります．本誌の読者になじみの深いC言語を前提に考えてみると，データは変数にもたせ，それをまとめた構造体として保持するのが一般的でしょう（**リスト1**）．

リスト1の構造体を一つ確保しただけでは，一つの情報しか保持することができません．そこで複数の構造体を管理する必要性が生じます．そのような場合，どのような方法を使えば良いのか――これが本特集のテーマです．

リスト1　構造体

```
/* ディレクトリ情報 */
typedef struct {
    FILELIST *filelist_ptr;
    FILELIST *filelist_top;
    FILELIST *filelist_end;
    FILELIST *disp_top_ptr;
    FILELIST *cursor_ptr;
    char dirpath[256];
    char hostname[48];
    char local;
    char parent;
    int count;
} DIRINFO;
```

2 C言語レベルのコンテナを使ったデータ管理

●「コンテナ＝データの入れもの」という概念

コンテナということばは，輸送用の箱からきたことばです．プログラミングにおいては，たとえば配列やリスト，二分木，スタックなど，実例を挙げれば「ああ，そのあたりのことね」と納得していただけるかと思います．

復習のために，ここではコンテナについて説明します．

● 配列

図1（**a**）の配列は，固定長のデータを複数もつためのデータ構造です．それぞれのデータには，インデックス（たとえばa[i]のi）を用いてアクセスすることができます．このアクセスはつねに同じ速度で行える$O(1)$のアルゴリズムです．最悪値（この場合，データ・アクセスにかかる時間のうち，いちばん遅い場合の時間）の保証も可能なので，リアルタイム処理にはうってつけです．

配列の欠点は，「配列の数を動的に変更できない（ことが多い）」点です．最大N個の構造体を管理したい場合，最初にN個ぶんのデータを確保する必要があります．これは，実際には1個しか使っていない場合でも，最大量のデータを確保する必要があることを意味し，たいへんむだです．そこで現在では，「動的配列」という，最大数を可変にできるものがあります．動的配列はプログラミング言語のPerlなどにも実装されています．

動的配列の中身は（実装にもよるが）おそらくリストで，それを配列のように見せかけています．そのため最悪値を保証できるかどうかは，その実装で使われているアルゴリズムを精査し

注1：筆者としては，これに「＋インターフェース」ということばを付け加えたいところだが，原著自体が1976年発行ということもあり，さすがにインターフェースの設計が重要というところまではいっていなかったのだろう．

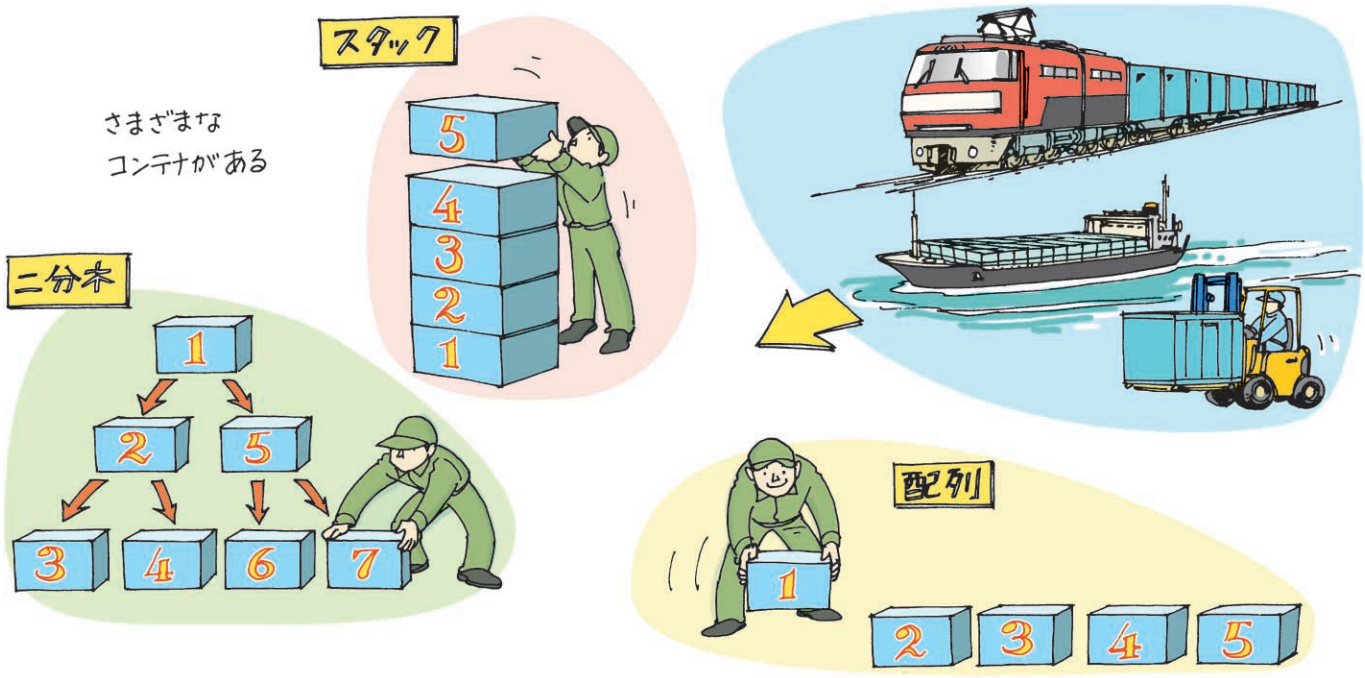

イラスト1　さまざまなコンテナ

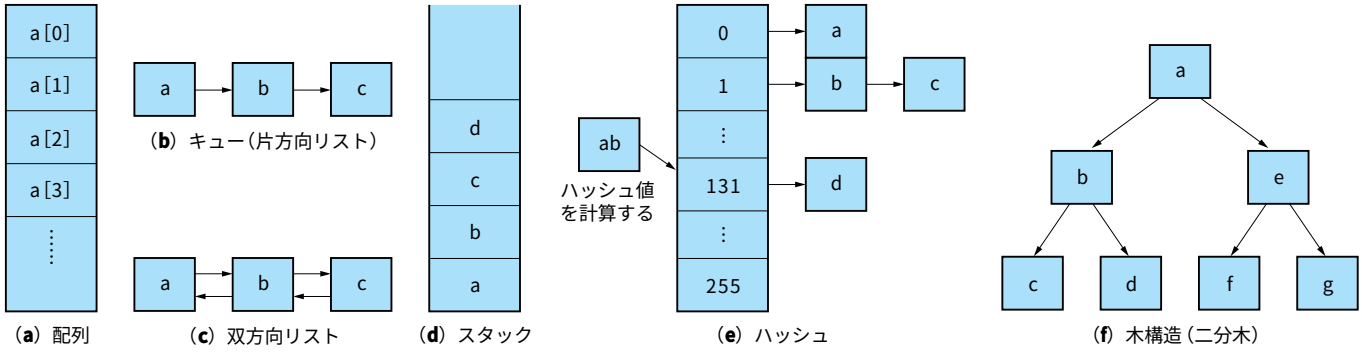

図1　コンテナの数々

なければなりません．まあ，最悪値が気にならない用途であれば，動的配列は便利に使えるコンテナでしょう．

● キュー

キューは，**図1**(**b**)のようないわゆる片方向リストです．または，単にリストとも呼びます．データを末尾にどんどん追加し，いちばん先頭から取り出していくという形式です．組み込み技術者には「バッファ」と言ったほうがすぐにわかるでしょう．FIFO(First In First Out)ともいいます．

これの利点は，つねに同じ速度でデータの追加・取り出しが可能であることです．また，欠点は基本的にデータの追加は末尾からしか行えず，データの取り出しも先頭からしか行えないということです．C言語においては，データの結び付き(**図1**(**b**)の矢印)はポインタによって行います．実装するうえでは，動的なメモリ確保/解放〔いわゆるmalloc()/free()〕が必要になります．

● 双方向リスト

双方向リストは，**図1**(**c**)のように矢印が両方向についたリストです．任意の位置に対してデータの追加・削除が可能で，データの最大数にも制限はありません(メモリ容量の許す限り管理できる)．主流といえるコンテナでしょう．

欠点は，任意のデータを検索して取り出すために，リストの先頭から一つずつ検索する必要があるということです．そのため，データ数が多くなればなるほど，検索速度は低下します．みなさんの近くにも，たくさんのデータを入れると処理が遅くなる機器はないでしょうか．このようなときプログラマは，「はは〜ん，データをリストでもっているな」と推測するものです．

● スタック

スタックは，**図1**(**d**)のように先頭にデータを入れ，先頭からデータを取り出す方式です．LIFO(Last In First Out)ともいいます．

余談ですが，物だらけの汚い部屋を指してFINO(First In