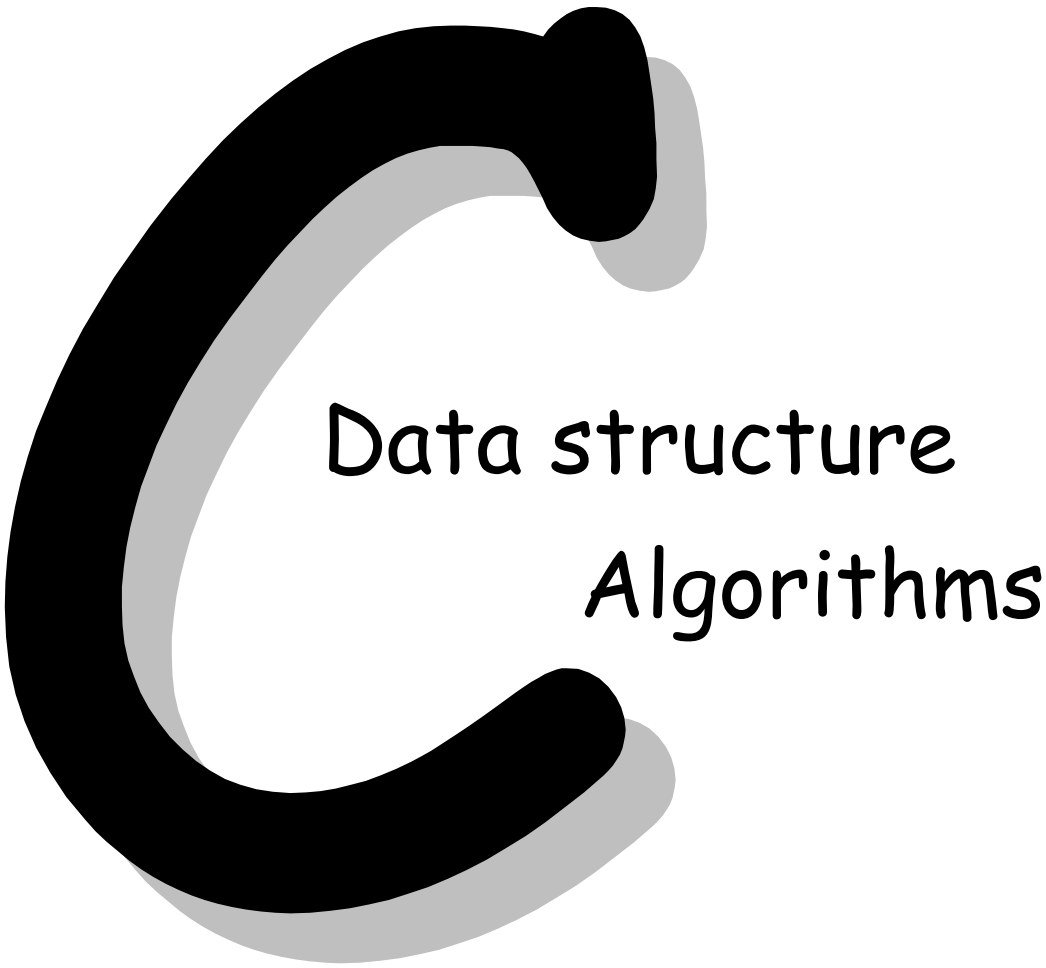
C言語プログラミング
データ構造とアルゴリズム編



株式会社デンソークリエイト

はじめに

外国語の文法を知っているということは、外国語を使ったコミュニケーションの基礎能力があるということです。しかし、ただ、それだけに過ぎません。それ以上の知識がなければ、有意義な会話はできません。例えば、「WABI や SABI はどういう意味ですか」と問いかけられても、日本文化を知らなければ答えられません。文法を知っていることと、有意義な会話ができることは、違った問題です。加えて、聞く人に心地良さを与えるように話すことは、より一層深い知識を必要とします。

プログラムの文法を知っていることは、このこととよく似ています。C言語を知っていることは、コーディングする基礎能力があるということです。しかし、ただ、それだけに過ぎません。それ以上の知識がなければ、有意義なコーディングはできません。例えば、「この部分は線形探索でサーチするコーディングをしてください」と指示されても、線形探索がなんであるかを知らなければコーディングできません。プログラミングの文法を知っていることと、有意義なコーディングができることとは、違った問題です。加えて、効率よくソフトウェアを開発するには、より一層深い知識を必要とします。

優れたプログラムを作るには、データ構造とアルゴリズムの知識が必要です。この知識により、高速な動作・少ないバグ・高い保守性などといった、高品質なプログラム開発を行う基礎体力がつきます。その上で、問題の分析能力や設計技術の知識を身に付け、実践を積むことで優れたソフトウェア技術者となります。

本書では、基礎的なデータ構造とアルゴリズムの基礎知識をC言語でのサンプルを示しながら解説します。これらの基礎知識に親しみ自らのプログラミングに応用して、高品質なプログラム開発を目指してください。
また、皆さんが設計技術を勉強する時には、このデータ構造とアルゴリズムの知識が基礎知識として役立ちます。学習を続け、プログラマとしてだけではなく設計者として、優れたソフトウェア技術者を目指してください。

本書では、理解をしやすくするため、難しいと思われるデータ構造やアルゴリズムの解説は、次の手順で行います。

- まず、例題を用いて説明します。
- 次に、HCP チャートを示します。
- 最後に、C 言語プログラムを掲載します。

また、基礎問題や応用問題を載せています。
基礎問題の解答例は、巻末にあります。自習の時にお役立て下さい。
応用問題に関しては、解答例を載せていません。各自で、お考え下さい。データ構造とアルゴリズムの知識を元に、このような発展があることを知って下さい。

本書で述べたデータ構造とアルゴリズムは、全体の一部です。巻末の参考文献を参考にして、より多くの種類のデータ構造とアルゴリズムに親しんで下さい。

（中略）

第1節 バブルソート

バブルソートは、隣り合う２つのデータを比較して昇順（もしくは降順）に入れ替える操作を繰り返してソーティングする方法です。並べ替えの様子が、まるで気泡（バブル）が水の中から水面に向かって浮かびあがる様子に似ているので、この名前がついています。次に例を挙げて説明します。

最初に、次のデータ列があるものとします。このデータ列を、バブルソートで昇順にソーティングします。

３,１,４,６,５,２

まず、左側２つのデータを比較して昇順に入れ替えます。つまり、下記で四角で囲まれた２つのデータを比較して入れ替えます。

[3],[1],４,６,５,２

昇順に並びかえると、次のようになります。

１,３,４,６,５,２

この、「隣り合う 2 つずつのデータを比較して交換する」操作を、順に右にずらしながら右端に至るまで行います。

１,[3],[4],６,５,２
１,３,[4],[6],５,２
１,３,４,[6],[5],２
１,３,４,５,[6],[2]
１,３,４,５,２,6

この時点で、最大値が最も右側になります。つまり、この場合、「６」は昇順にソーティングした場合の定位置に来ます。
次に「６」以外の「１,３,４,５,２」について、同様に左側から順に２つのデータを比較して昇順に入れ替えます。すると、次のようになります。

[1],[3],４,５,２,6
１,[3],[4],５,２,6
１,３,[4],[5],２,6
１,３,４,[5],[2],6
１,３,４,２,5,6

この時点で、「６」以外の「１,３,４,５,２」の中での最大値がもっとも右側になります。つまり、この場合「５」が右側に来ます。こうして、「５」と「６」は昇順にソーティングした場合の定位置に来ます。

このように、「隣り合う 2 つずつのデータを比較して交換する」操作を繰り返します。すると、下記のように最後のデータの組み合わせに対して入れ替え処理を行った段階で、全てのデータをソーティングし終わります。

1,2,3,4,5,6
1,2,3,4,5,6

以上の動作を見ますと、未ソートのデータ列の中で最大値が右側に一つずつその位置を変えて行きます。この動きは、まるで、気泡（バブル）が水面に向かって徐々に浮きあがってくる様子に似ています。そこで、このソーティング方法をバブルソートと呼びます。
バブルソートのＨＣＰチャートによるアルゴリズム表記は、次のようになります。

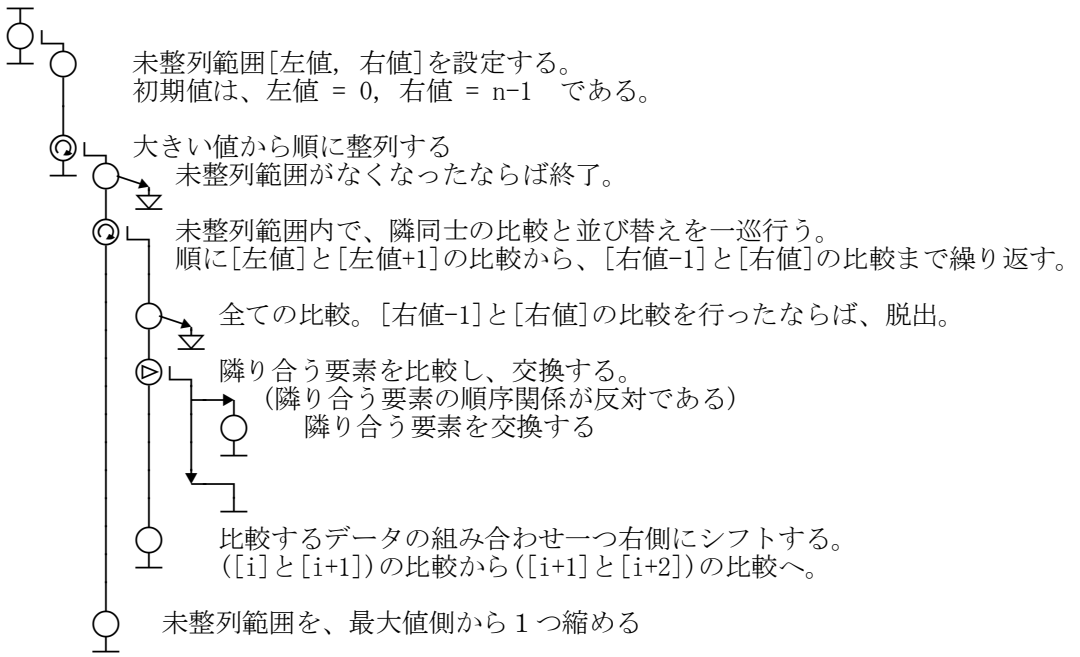


図１　バブルソートのＨＣＰチャート

このように、バブルソートは繰り返しが二重構造となっています。次に、このオーダーを求めます。
外側の繰り返しは、N-1 回行われます。内側の繰り返しは、外側の繰り返しに応じて繰り返し回数が異なりますが、平均すると約 N/2 回行われます。そこで、このアルゴリズムのオーダーは、$O(n^2)$となります。

次に、バブルソートプログラムのコーディング例を示します。

```
void bubsort(
    int iNum,       /* データ数 */
    int aiData[])   /* データ */
{
    int iLp1, iLp2;
    int iWork;

    for(iLp1 = iNum - 1; iLp1 > 0; --iLp1){
                            /* 未整列範囲を１つずつ減らす */
        for(iLp2 = 0; iLp2 < iLp1; ++iLp2){
                            /* 未整列範囲内の並び替え */
            if(aiData[iLp2] > aiData[iLp2 + 1]){
                            /* 隣り合う２つの比較 */
                iWork = aiData[iLp2];  /* 交換 */
                aiData[iLp2] = aiData[iLp2 + 1];
                aiData[iLp2 + 1] = iWork;
            }
        }
    }
}
```

バブルソートのアルゴリズムは簡単で、プログラムサイズも小さくできます。しかし、オーダーが $O(n^2)$なので、データ個数が多くなると急速に実行速度が低下します。ですから、バブルソートを実際のアプリケーションで用いることは避けるべきです。

（以下省略）