目　次

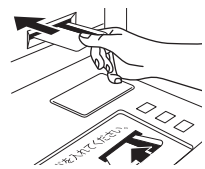
1. アルゴリズムとは？

アルゴリズム

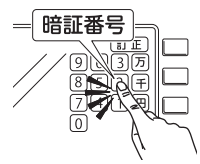
アルゴリズム（algorithm）をひと言でいえば，「問題を解決するための手順」です。例えば，銀行の ATM でお金を引き出すときの行動を考えてみましょう。このときの問題は，「お金を受け取る」ことです。それを解決するためには，「キャッシュカードを入れる」，「暗証番号を押す」などの行動が必要です。

このときの行動を順番に表すと，次のようになります。

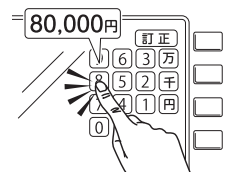
① ATM からお金を引き出すため，ATM メニューから「引き出し」を選択する。



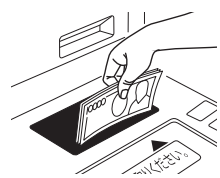
② キャッシュカードを入れる。



③ 暗証番号を押す。



④ 引き出す金額を入力する。



⑤ お金が出てくる。(受け取る)

図表 1－1　銀行の ATM でお金を引き出すときのアルゴリズム

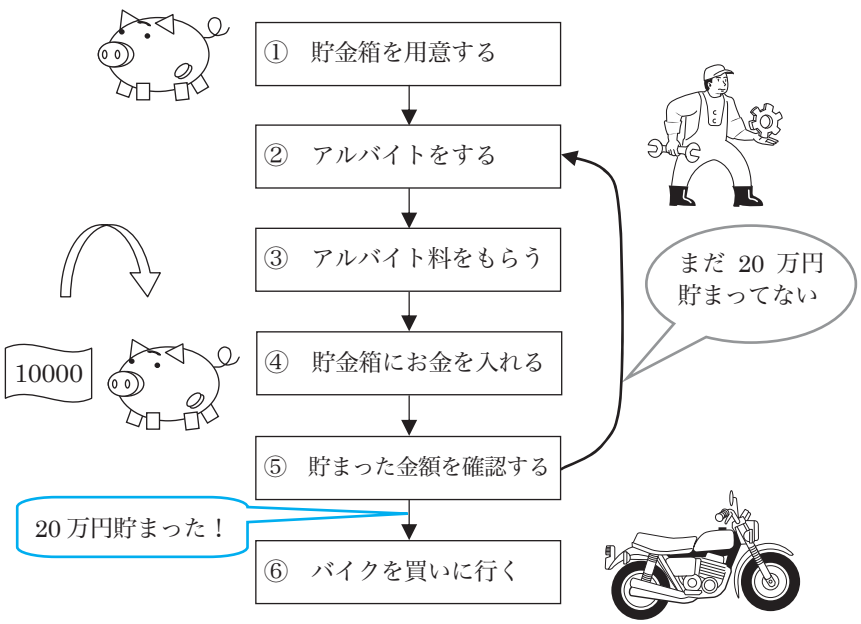
この順序のとおりに行動すれば，お金を引き出すことができます。しかし，例えば，①と②の順序が入れ替わったら，キャッシュカードが挿入口から入らない機械もありますし，②と③が入れ替わったら，暗証番号を押しても ATM は反応しません。①，②，③，④，⑤と決められた順番に操作することで，はじめてお金を受け取ることができます。

このように，「しなければいけないことをその順番に並べたもの」，これがアルゴリズムです。ただし，順番に並べればそれでよいというわけではなく，次の二つのポイントを押さえておく必要があります。

1. することがもれなく決められていて，機械的にできること
2. 必ず終わること

例として，「お金を 20 万円貯めて，バイクを買う」という問題を考えてみましょう。このとき，「思いつくままに何でも行い，お金を手に入れる」では，アルゴリズムになりません。まず，しなければいけないことをきちんと定義します。ここでは，お金が貯まるまで，日給 1 万円のアルバイトを行うことにします。

アルバイト料は新しく買ってきた貯金箱に入れ，ほかでは使わないことにします。そして，何度かアルバイトをして，20 万円貯まったら，アルバイトをやめて，即バイクを買いに行きます。これが終了条件で，この条件があることで，必ず終わることになります。



図表 1−2　バイクを買うためのアルゴリズム（その 1）

アドバイス

問題を解決するということをあまり難しく考える必要はありません。アルゴリズムとは「目的を実現するために必要な行動を順番に書いたもの」と考えるとよいかもしれません。

1. 整列処理の概要

第 1 章では，アルゴリズムとは何かということから始まって，その表記方法，そして，変数や配列といったコンピュータのプログラム特有の考え方などについて学習してきました。第 2 章以降では，基本的なアルゴリズムを題材として，具体的にアルゴリズムを作成するときに基本となる考え方を学習していきます。

みなさんが，これからアルゴリズムの学習をして，その知識を活かしてプログラムなどを作成するときに，本書で学習するような基本アルゴリズムを，そのまま利用することは少ないかもしれません。しかし，どんなに複雑な機能をもつプログラムでも，その内容は基本的な機能の組合せで実現されています。基本アルゴリズムを通してみなさんがこれから学習する内容は，そのときに利用できるアイデア（アルゴリズムを作成するときに利用できる考え方）なのです。アルゴリズムの基本となるアイデアを理解し，活用できるようにしましょう。

(1) 整列とは

整列
ソート
昇順
降順

整列（sort；**ソート**）とは，値を**昇順**（小→大）もしくは**降順**（大→小）に並び替えることです。例えば，小学生の身長を昇順に整列すると，次のようになります。

	(整列前)		(整列後)
1	150		120
2	120		125
3	160		130
4	130		140
5	155		150
6	125		155
7	140		160

図表 2－1　身長の整列

1. 逐次探索

探索
サーチ

探索（search）は，複数のデータの中から目的のデータを探すことで，**サーチ**ともいいます。検索という用語が使われる場合もあります。

ここでは探索の基本的なアルゴリズムとして，逐次探索と 2 分探索について学びます。

逐次探索
順次探索
線形探索

逐次探索とは，目的のデータを単純に先頭から一つずつ順番に探していく方法です。**順次探索**，**線形探索**ともいいます。このアルゴリズムは，すでに学習した挿入法（整列）で，挿入位置の決定をするために同じような方法を使うので，理解しやすいと思います。ただし，挿入法では挿入位置を決定するときに，目的のデータ（挿入するデータ）よりも，大きな値をもつデータを探しました。また，探索の対象となるデータも整列済みであることが前提でした。しかし，逐次探索では，同じ値のデータを探すということが目的ですし，探索対象が整列されている必要もありません。

(1) 逐次探索の考え方

例えば，数字列 {4，8，3，5，7} の中から，“5” を探索していく例を考えてみます。

逐次探索では，まず，1 番目のデータと探索したいデータの値を比較します。1 番目のデータは 4 ですので，探索したいデータとは異なります。異なる場合は，次のデータと比較します。つまり，2 番目のデータと探索したいデータを比較します。2 番目のデータは 8 ですので，今度も探索したいデータとは異なります。同じように，次のデータと探索したいデータを比較すると，3 番目のデータは 3 で，探索したいデータとは異なりますが，4 番目のデータが 5 で，探索したいデータと一致します。

1. 文字列の探索

文字列

文字列とは，複数の文字が集まったものです。これまで学習してきた整列と探索では，数値などを中心に，配列の要素を単位として考えてきました。文字列は，文字という単位が複数集まったものです。複数のものをまとめて扱う方法について学習していきましょう。

文字列の探索
テキスト
パターン

文字列の探索とは，ある文字列（一般に**テキスト**と呼びます）から，指定された文字列（こちらは**パターン**と呼ばれます）と同じ文字列を探す処理のことです。テキスト中から，パターンと一致するすべての文字列を探すこともありますが，まずは，テキストの先頭から探索していき，最初に出現したもの一つだけを探すということを考えましょう。

ポイント！

文字列を単位とした扱い方もできますが，ここでは，各文字がそれぞれ配列の要素であると考えて文字列処理の基本を学習していきます。

(1) 文字列探索と逐次探索

これまでに学習したアルゴリズムの中で，これと同じようなことをするものがありましたが，お分かりですか？　そうです，逐次探索です。逐次探索とは，たくさんのデータの中から，指定された値と同じものを探すためのアルゴリズムで，次のように，先頭のデータから順に調べていく方法でした。文字列探索も，基本的にはこの方法によって行うことができます。

```
(逐次探索のアルゴリズム)
・i ← 1
■ A[i]≠x　かつ　i≠N
  ・i ← i+1
■
▲ A[i]＝ x
  ・[見つかったときの処理内容]
  ・[見つからなかったときの処理内容]
▼
```

1. 応用データ構造の概要

配列に格納されたデータを対象にして，繰返し処理の基本的な考え方から始めて，整列・探索処理と文字列処理のアルゴリズムを学習しました。どのアルゴリズムも，「同じ形式のデータが連続した領域に記録されている」という配列の性質を利用して，繰返し処理を行いました。このように，「データをどのように記録してそれをどのように処理するかをまとめて考えたもの」を**データ構造**といいます。言い換えると，プログラムで使用する「データの表現方法とその操作に関して共通した性質」ともいえます。

データ構造

アドバイス

処理の方法に合わせたデータの記録方法がデータ構造です。適切なデータ構造を利用するとアルゴリズムが考えやすくなります。

この章では，配列以外のデータ構造の特徴について，最も基本的なところから学習していきます。処理の内容に合ったデータ構造を用いてデータを記録させることによって，データの処理を効率的に行うことができます。

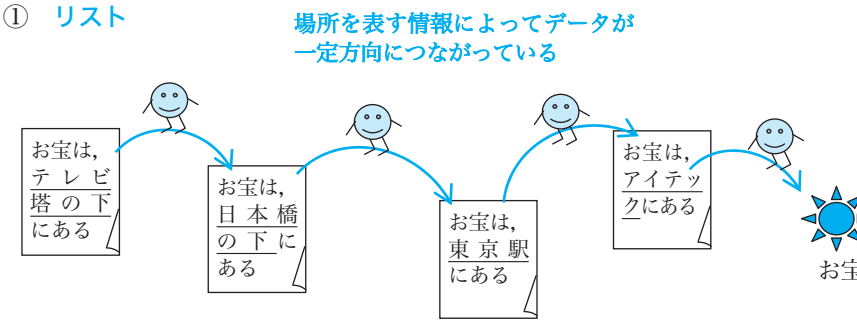
代表的なデータ構造としては，リスト，スタック，キュー，木があります。それぞれのイメージを最も大まかに表現すると次のようになります。

リスト

アドバイス

まず，代表的なデータ構造を大まかにながめましょう。

① リスト



図表 5−1　リストのイメージ

1. 階乗計算

再帰
リカーシブ

この章では，これまで紹介してきたものとは少し違ったアルゴリズムの考え方を紹介します。それは**再帰**（recursive；**リカーシブ**）と呼ばれる考え方です。再帰とは，「自分の中に自分がいて，またその中にも自分が存在する……」ということがいくつも続くことをいいます。なんだか奇妙な話ですね。ちょうど鏡の中に鏡を写し出すようなイメージを想像してみてください。鏡の中に鏡が存在し，その中にもさらに鏡が存在している……，そんな世界です。

再帰呼出し

それが，どのようにアルゴリズムと関係するのか疑問に感じる人もいるかもしれませんが，コンピュータの世界には，この再帰的な処理というものが，実は至るところに存在しているのです。そのような処理は**再帰呼出し**（recursive call）と呼ばれる手法を用いることによって，とても単純な構造のプログラムにすることができます。この章では，この再帰について学んでいきます。

階乗計算

まず手始めに，再帰の例として次の**階乗計算**を取り上げてみましょう。

$5！=5\times4\times3\times2\times1$

5 の階乗は，この式に示したように，5 から 1 までの数をすべて掛け合わせることによって計算することができます。まずは普通に階乗計算をするためのプログラムを示します。

ポイント！

N×（N−1）×…×2×1 のことをNの階乗と呼びます。

```
(階乗を計算する副プログラム)
○プログラム名：Factorial(N)
○整数型：i，R
・i←1
・R←1
■ i≦N
  ・R←R×i
  ・i←i＋1
■
・R の値を戻り値として返す
```

1. 事務処理のアルゴリズムの概要

ここまでの内容で，配列や繰返し処理の応用など，アルゴリズムに関する基本的な知識と考え方について学習してきました。

事務処理のアルゴリズム

ここでは今までの考え方を応用した**事務処理のアルゴリズム**をいくつか紹介します。身近な例からそうでない例まで，いろいろなアルゴリズムがあります。ここでも，全体の大まかな処理の流れを理解してから，細かい部分の処理内容を考えていくという基本的な方法で学習していきましょう。

買い物をしてお金を払った後にもらうレシートは，みなさん見慣れていますよね。このレシートには買い物をした日付，買った物の品名，値段，個数，小計，消費税，税込みの合計などが印刷されています。

また，銀行の通帳を見てみると，お金を預けた日付，預けた金額，お金を引き出した日付，引き出した金額のほか，電気や水道などの自動引落しをしている人なら，その内容や金額が印刷されています。

1. 方程式の解法

技術計算

電子計算機，すなわちコンピュータが誕生するきっかけは，1940 年代にミサイルの弾道計算に伴う複雑な技術計算処理を速く行う必要性からだったことを，みなさん，ご存知ですか？　数学を直接・間接的に応用した計算を**技術計算**といいます。大学や企業の研究所で必要とされる技術計算や，天気予報，ロケットの打上げなどに必要となる膨大な量の技術計算は，今ではコンピュータ抜きでは考えられなくなっています。

といっても，日常の生活の中で，技術計算をコンピュータで行うことはあまりありませんね。しかし，電卓なら，みなさんもよく使っているでしょう。電卓も立派なコンピュータで，例えば，[2] [√] と押すと，$\sqrt{2}$ の値である 1.41421356…という結果が出てきますね。この結果（1.4142…という値）は電卓の中で記憶されている情報が出されたのではなく，押された 2 という値を使い，[√] が押されてから計算しているのです。

これから学習する技術計算のアルゴリズムについて，数学的な内容を理解する必要はありません。どのようにして結果を求めているのかという，大まかな流れが理解できれば十分です。

ニュートン法

方程式には，1 次方程式，2 次方程式，連立方程式などいくつかの種類がありますが，技術計算のアルゴリズムにも方程式の答え（解）を求めるいくつかの方法があります。ここでは平方根を計算する**ニュートン法**を紹介します。

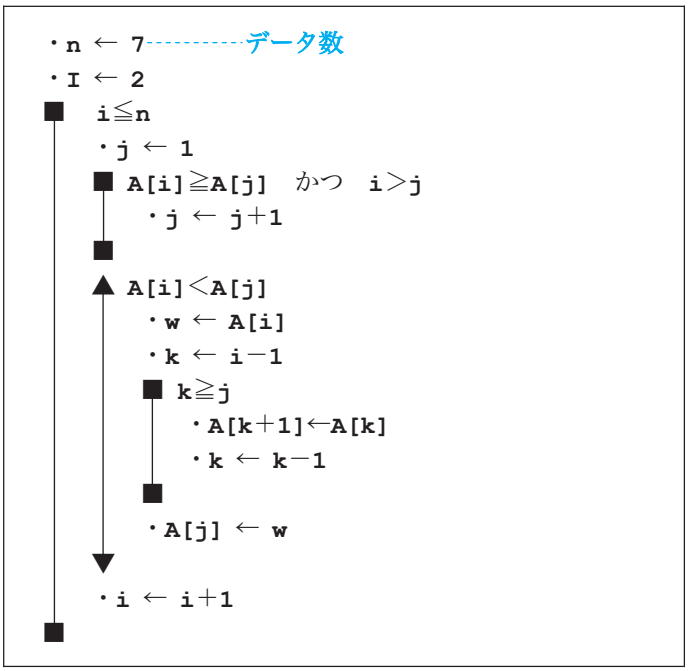
(1) ニュートン法とは

ニュートン・ラフソン法

ニュートン法は，**ニュートン・ラフソン法**ともいわれ，方程式の解を適当な近似値から始めて，同じ計算を繰り返し行って，徐々に真の値に近い解を求めていきます。例えば，方程式 $x^2-2=0$ の解で正の数のほうは $x=\sqrt{2}$ になりますが，これは $y=x^2-2$ のグラフと x 軸との交点（$x>0$）が $\sqrt{2}$ になることを示しています。

研究 1 挿入法のアルゴリズムについて

2 章の「4. 挿入法」で学習したアルゴリズムについて見直しをしてみましょう。



2 章「4. 挿入法」のアルゴリズム

このアルゴリズムが複雑に見えるのは，挿入位置以降の値をずらすための繰返しが，ずいぶん右のほうにあるためです。繰返しや選択の構造の中に，さらに同じような構造が現れることをネスト（入れ子）と呼びます。慣れてくれば三重のネストくらいは，単純なほうですが，最初は少し気になります。この部分がもう少し単純にならないかどうかを考えてみます。

第1章

問1-1　整数の総和を求める流れ図

【解答】　ウ

1からN（N≧1）までの整数の総和（1＋2＋…＋N）を求め，結果を変数xに入れるアルゴリズムです。まず，変数xに初期値の0を，変数iに初期値の1をそれぞれ設定します。その後，xにiの値を加え，iをカウントアップ（＋1）するという処理を繰り返しています。問題の流れ図ではループ端を使わずに，矢印付きの線を使って上に戻ることを表現しています。情報処理技術者試験の問題では，このような表現がありますので，慣れておきましょう。

空欄aは，この繰返しの継続条件です。求めるのはNまでの総和ですが，xへの加算直後にiを1カウントアップしているので，最後のNを加算して繰返しを終了するかどうかの判定を行う時点では，iの値はN＋1になっていることに注意してください。したがって，繰返しから抜け出るのは，i≦NがNoのときです。なお，選択肢にはありませんがi≠N＋1やi＜N＋1でも正しく終了できます。

ア：i≠N……Nを加算せずに終了してしまいます。この結果，変数xに求められるのは1からN－1までの総和です。

イ：i≧N……もしN＝1ならば永遠にループしてしまいます。また，N＞1ならば最初の判定で終了条件を満たし，終了してしまいます。

エ：x≦N……xは総和を求める変数なので，Nと比較してもNまで加算されたかどうかの判定はできません。

問1-2　合計を求めるアルゴリズム

【解答】

```
・合計 ← 0
・足す数 ← 2
■ 足す数 ≦ 20
  ・合計 ← 合計＋足す数
  ・足す数 ← 足す数＋2
■
・合計を表示
```

1から20までの偶数を足していくので，初期値は2になります。合計に足し込むごとに＋2していきます。