7．データ構造とアルゴリズム

7.1　データ構造

(1) 配列

：複数の同一形式のデータをまとめて取り扱う場合に使用されるデータ構造。一つひとつのデータ（要素）を，番号（[　　]又は指標）で区別する。

1)［　　］配列：同一形式のデータを一列に並べて扱うデータ構造。

2)［　　］配列：配列要素を複数の関連で取り扱うデータ構造。

3)［　　］配列：配列要素が異なる形式のデータの集合になっているデータ構造。レコード型配列ともいう。

(2) リスト

：データの並びをポインタによって決定するデータ構造。

1)［　　］リスト：データを片方向にしかたどれないリスト。線形リスト，単リスト，片方向リストともいう。

2)［　　］リスト：次の要素の記録位置を示す後ポインタと前の要素の記録位置を示す前ポインタをもつことで，データを両方向にたどれるリスト。双リスト，両方向リストともいう。

3)［　　］リスト：全要素が環状につながっているリスト。このリストを線形リストで実現する場合は，最後尾要素のポインタに先頭要素へのポインタを記録する。

(3) スタックとキュー

①スタック

：後から記録したデータを先に取り出す［　　］（後入れ先出し）方式のデータ構造。データの入口と出口が同じである。スタックにデータを格納することを［　　］，スタックからデータを取り出すことを［　　］という。

・［　　］(SP)

：現時点でのスタックの最上位要素の位置（配列構造なら添字）を表す。

②キュー

：先に記録したデータを先に取り出す［　　］（先入れ先出し）方式のデータ構造。データの入口と出口が異なる。キューにデータを格納することを［　　］，キューからデータを取り出すことを［　　］という。

(4) 木構造（ツリー構造）

：親に対して複数の子が存在する１対多の階層構造を表すためのデータ構造。

【木構造の構成要素】

・[　　]　　：各要素（データ）。
・[　　]（ルート）　：最上位のノード。木構造に一つだけ存在する。
・[　　]（リーフ）　：最下位のノード。
・[　　]（ブランチ）：各ノード（ルート，リーフを含む）を関係付ける線。
・[　　]　　：木構造の階層の深さ。

・[　　]：子ノードの数を２個以下に限定した木構造。

・[　　]

：最終レベルの葉が左詰めになっていて，根からすべての葉までのレベルが等しい，又は高々１しか違わない木構造。

・[　　]：子ノードの数が２個より多いｎ個の木構造。ｎ進木ともいう。

【木の巡回法】

・[　　]探索

：根から順に，同一レベルの左から右にノード値を取り出していく。

・深さ優先探索

：根から順に，左部分木から右部分木まで外周をたどりながら，ノード値を順に取り出していく。

・[　　]

：節点→左部分木→右部分木の順に取り出す（行きがけのなぞり）。

・[　　]

：左部分木→節点→右部分木の順に取り出す（通りがけのなぞり）。

・[　　]

：左部分木→右部分木→節点の順に取り出す（帰りがけのなぞり）。

①２分木の活用

1) ２分木による式の表記

・[　　]記法

：計算式を２分木で表現したとき，深さ優先探索の先行順でノード値を取り出す。

・[　　]記法

：計算式を２分木で表現したとき，深さ優先探索の後行順でノード値を取り出す。

2) [　　]

：すべてのノードにおいて“左部分木 ＜ 節点 ＜ 右部分木”の関係が成立する２分木のこと。その名が示すとおり，データを探索するのに適した木構造である。

3)［　　］

：親ノードと子ノードの間に一定の大小関係が成立する完全２分木。親子の大小関係だけが一定であり，兄弟間では一定の大小関係は成立しない（２分探索木やヒープのように，ノードに一定の順序関係がある木構造を［　　］という）。完全２分木の構成を保つために［　　］が行われる。

②［　　］

：データの追加や削除によって，ある葉のレベルだけが深くなってノードへのアクセス効率が低下することを防止するために，再編成を行う木構造。

1)［　　］

：各ノードにおいて左右の葉までの深さの差が１以内の２分木。

2)［　　］

：２分木の考え方を，多分木に発展させたバランス木。

・すべての葉（リーフ）が［　　］レベルにある。

・根（ルート）以外の各ノードは，ｎ個以上2n個以下の要素をもつ。

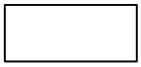
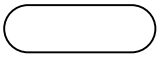
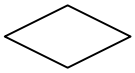
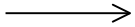
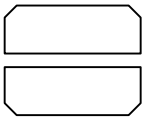
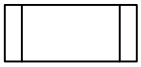
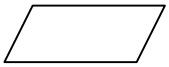
・各ノード（根を含む）は，［　　］個の子ノードをもつ。

・ノードに含まれる各要素は，整列されている。

7.2　基本アルゴリズム

(1) フローチャート

①代表的な記号

記号	記号名	意味
	［　　］	さまざまな処理（代入，計算など）を表す。
	［　　］	フローチャートの開始／終了を表す。開始の端子記号には名称を付ける場合もある。
	［　　］	条件により処理を分岐させる。
	線	各記号を結び，実行順序を表す。
	［　　］	繰返し処理（ループ）の開始／終了を表す。“変数名：初期値，増分，終値”の形式で記述することもできる。
	定義済み処理	定義済み処理を呼び出す（同名の端子記号から始まるフローチャートを実行する）。
	データ	データの入力，出力を表す（処理記号で代用されることもある）。

②三つの基本構造

：わかりやすく処理効率の良いアルゴリズムを作成するための［　　］で利用される構造単位。

1)［　　］　：処理を上から下へ順に実行する構造。

2)［　　］(判定)：条件により処理を分岐させる構造。

・［　　］：条件によって処理を二つに分岐させる。

・［　　］：処理を複数（通常は三つ以上）に分岐させる。

3)［　　］(反復)：条件が成立している間（又は条件が成立するまでの間）処理を繰り返す構造。

・［　　］：繰返し処理の前に条件判定する。

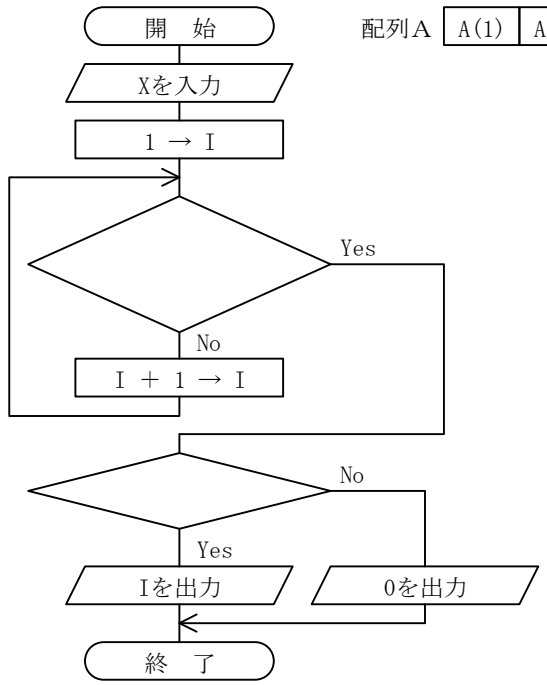
・［　　］：繰返し処理の後に条件判定する。

(2) データ探索処理

①［　　］(順次探索法，逐次探索法)

：先頭要素から順に目的値を探索していくアルゴリズム。

【例題１】　要素数N個の配列Aから入力された値Xを探索し，格納されている位置（添字）を出力するフローチャートである。判断記号中に条件を記入せよ。なお，要素中にXが存在しない場合は“０”を出力する。



配列A

A(1)	A(2)	A(3)	･･･	A(N)

・［　　］：条件比較（判断処理）を減らして，処理効率を向上させる方法。探索の前に全要素の末尾に目的値を格納しておく。