# ２００６年度夏学期
# シケプリ
# de
# 情報

# 第１章　情報の学び方

ここは授業では扱われませんでした。と、いうわけで、悪いんですがここは各自教科書を読んで勉強してください。自分自身まだちゃんと読んでないんで何とも言えないんですが、まぁ授業でやってないんだからそんなに突っ込んで出題されないんじゃないですか。教科書読んで疑問点などあればビシバシメールください。答えられたら答えます。私は植田なにがしとかいう情報機器オタクではないので、「わかりません」って返事が返ってきても殺意とか抱かないでください。とりあえず全力でお答えしようとはしますから。

もちろん第１章以外のところでも疑問点あればビシバシメールください。それでは、次のページから本格的な情報のお勉強に入りましょう。２章あたりはなんだか堅苦しくダラダラといっぱい書いてあってわかりにくいかもしれませんが（この文章は２章、３章を作り終えてから書きました)、だんだんくだけてきます。途中から著作権も肖像権もどうでもよくなってきます。だから極力このシケプリの外部流出は避けたいところです。まぁ、こんなことはどうでもいいのでさっさと勉強始めましょう。はい、次のページへどうぞ。

# 第２章　情報の表現

【情報表現の様々な側面】

◆道案内における情報表現の多様性

相手が日本人 かつ 目的地が近い … 「２つ目の角を左」など自然言語による道案内

相手が外国人 かつ 目的地が近い … 身振り手振りのジェスチャーを駆使

目的地がやや遠い … 地図を書いての説明

さまざまな表現手段を状況に応じて使い分ける必要がある

◆言語による情報表現の違い

**自然言語** → 人間が日常的に使用している言語

**人工言語** → プログラミング言語のように人工的に作られた言語

◆情報の説明の仕方の違い

**手続き的表現** … 時間を追った手順を説明（第５，６章のアルゴリズムに関連）

Ex.駒場図書館の位置の説明

「正門を入って右、左、右と進めば、工事用の白い壁に突き当たります。

右へ曲がり壁沿いに進んで行けば図書館の『開館中』の看板が見えるはずです。」

**宣言的表現** … 対象間の関係や対象の属性を説明（第４章のデータモデルに関連）

Ex.駒場図書館の位置の説明

「コミュニティプラザの南隣にある、工事用の白い壁に囲まれた建物です。」

◆情報の表現のされ方の違い

**記号表現** … 与えられた記号の集合と、それらを解釈するための規則体系

Ex.数学の記号や数式　日本語　英語　楽譜

**パターン表現** … 構成要素間の時空間パターン

Ex.地図（地図記号は記号表現）　ジェスチャー　発声の長短・強弱

たとえば日本語という記号表現の規則体系を理解していない外国人でも、

ジェスチャーなどのパターン表現は理解可能。

◆他の側面

ディジタル／アナログによる表現の違い（後述）

情報量から見る側面（後述）

【情報表現とモデル】

◆**モデル** … 単純化／抽象化 された 事物／事象／概念

目的によって異なるモデル化が必要

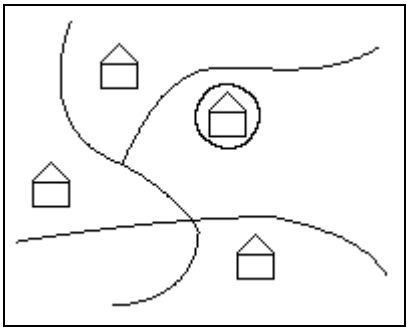


図 2-1

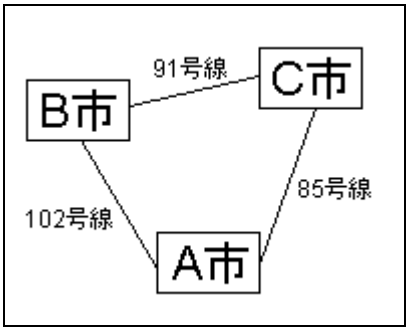



図 2-2

徒歩で移動する場合 → 図 2-1 の方が便利

車で移動する場合 → 図 2-2 の方が便利

◇ジェット旅客機のモデルの場合

コスト削減のためモデルでの実験

風洞実験を行う場合 → 実物と同じ材料でモデルを作るべき

外観を見たい場合 → 加工・修正のしやすい材料を採用すべき

◆モデル表現形式の例

①**表（table）**

こみいった事柄を整理 … モデル化の重要な役割の１つ

歴史年表／貸借対照表／成績表　等

パソコンなどでの表計算ソフト（Excel 等）の利用も一般的

②**図**

何らかの目的で描いた２次元図形（絵画・スケッチも含む）

人間の思考・推論を支援／拡張する … ベン図　等

設計図／地図　等

③**グラフ**（[　　] 内は図 2-2 との対応を示す）

**ノード（node）**［市名の書かれた長方形］と

**エッジ（edge）**［道路を表す線分］から構成される

**ラベル**［～号線］付きのエッジもある → **ラベル付きグラフ**

方向を持つエッジは有向エッジ、弧と呼ばれる

道路ネットワーク／組織図／意味ネットワーク／工場の作業工程　等

【情報表現で考慮すべき事】

◆情報表現の受け手側

日常生活に存在する多くの表現を適切に理解・解釈しなければならない

◆情報表現のデザイン側

多くの表現手段の中から目的・状況に応じて適切なものを選択しなければならない

◇３つの考慮すべき点

①表現の**対象** … 物理的実体・抽象的アイディア・思考の方法　等

②表現の**目的** … 他者への伝達・自身のアイディアの整理・効率的な問題解決　等

③表現の**方法** … 表現に関わるコストや目的に照らして、よりよい方法を選択

【ピクトグラム―記号と意味】

◆記号が表す２側面

意味されるもの―**シニフィエ** ／ 意味するもの―**シニフィアン**

◆サービスエリアのピクトグラム（図 2-3）

抽象化された図形を用いてデザイン

→ 視覚的に認知されやすいように工夫

ドライバーが瞬時に表示内容を認識できるように…

図 2-3


◇記号表現とパターン表現（前述）の混在

記号表現 … "Parking"の頭文字「P」の利用、距離の尺度に数字を利用

パターン表現 … 「ナイフとフォーク」をモチーフにして表現、矢印で方向を示す

◇３つの考慮すべき点（前述）を検証

①表現の対象 … サービスエリア

②表現の目的 … 高速道路を走るドライバーにサービスエリアの所在を知らせる

③表現の方法 … ピクトグラムとして図で表現

◇記号が表す２側面（前述）を検証

意味されるもの … サービスエリア

意味するもの … 「P」や、「ナイフとフォーク」などの図記号

◇図記号の修辞法

**提喩** … ある事物を表現する時、それと意味的包含関係のある事物を代用

Ex.「小町」で「美人」を表す、「花」で「桜」を表す

→ 「サービスエリア」を表現するのに「ナイフとフォーク」の図を使用

**隠喩** … ある事物を、「～のようだ」などの形を用いず、直接他のもので表現

Ex.「時は金なり」、「俺の胃袋は宇宙だ。」

→ "ゴミ箱"アイコン … 「ゴミを捨てる」という行為の暗喩

◆解釈の枠組み

「シニフィアン」から「シニフィエ」へ対応づけるための枠組み（規則）

文脈や状況、文化的背景によって大きく変化

Ex.車両通行禁止を表す標識―日本と欧州で差がある（図 2-4）

→ **記号の恣意性** … 日本では○＝肯定、×＝否定と「対応づけられた」にすぎない

図 2-4
「すべての車両通行禁止」

日本　欧州

◆日本語文字コード

「文字」と「計算機上の符号（数値）」を対応づけるための枠組み

JIS、シフト JIS、EUC など、異なった日本語文字コードの混在

| 文字コード | 4547 | 9366 | 376C | 8C8C |
|---|---|---|---|---|
| JIS | 吐 | 劦 | 血 | ・ |
| シフトJIS | (なし) | 吐 | (なし) | 血 |

図 2-5

**文字化け** … パソコン上のプログラムの想定と違う文字コードが使用されると起こる

Ex.「吐血」という文字データの入ったファイルをシフト JIS に基づいて作成

→ パソコンには「9366　8C8C」と記録される（本当はさらに 0 と 1 に直される）

→ そのファイルを読み込むプログラムが JIS に基づいた処理を行った

→ 文字化けの発生 … 「劦・」と表示される

文字コードの標準化・統一化が望ましいが、歴史的経緯、利害関係から困難も多い

【アナログとディジタル】

◆コンピュータでの情報の表現

「0」と「1」の 2 種類の記号（ビット）の組み合わせ（ビット列）で表現

＝ 連続量（アナログ）としてではなく、有限個の離散的な値（ディジタル）として表現

◇表現できる数値はコンピュータに依る

システムが 16 ビットでは 0～65,535、32 ビットでは 0～4,294,967,295 まで表現可能

◆アナログ表現とディジタル表現

**◇アナログ表現**

無限の精度を必要とするため、データの複製は元のデータの近似にすぎない

レコード／オーディオテープ／ネガフィルム／ビデオテープ　等

**◇ディジタル表現**

ある情報に対して一定の間隔の尺度を導入し、その尺度の値に近似して表現

Ex.気温を表現するのに 0.1℃間隔で尺度を設定

→実際には 19.870427...℃の場合も 19.9℃と表現される

精度が有限で近接する量と明確に区別できる → 複製時にデータが劣化しにくい

情報コンテンツの著作権保護への問題をもたらす

◆アナログ→ディジタルの変換

情報を離散化する間隔を選択し、表現する必要がある

◇**量子化**

もともと連続量であった情報をディジタルの形で表現する作業（標本化の次の作業）

用途によって詳細度を決める … 一般に人間の知覚でその差が判別不可であれば十分

▽コンピュータディスプレイ装置

赤（R)、緑（G)、青（B）を混色した RGB 形式を用いている

各々256 種類の異なる色で表現（[黒］暗 0 ←‥‥→ 256 明［赤、緑、青]）

256×256×256＝16,777,216 色を表示できる

▽音楽 CD

量子化のために 16 ビット（16 文字）用いている

音の振幅を 65536 の段階に分類している

◇**標本化**

連続量の情報をある間隔ごとに抽出する作業

**標本空間** … 対象の情報が定義される時間や領域

Ex. 1 日の気温変化の情報の標本空間 → 時刻 t

標本化の間隔はデータの利用目的により変化

Ex.

1 時間毎に気温を記録すればその 1 日の気温の推移をとらえられる

1 日毎に気温を記録すればその月の気温の推移をとらえられる

1 ヶ月毎に気温を記録すれば年間の気温の推移をとらえられる

◇**標本化定理**（ちょっとややこしい。説明下手なんで、わかんなかったらメールを。）

フーリエ解析をご存じですか？

ほとんどあらゆる関数が周期関数の和として表せる、ってことなんですけど。

周期関数っていったら、sin とか cos とか、

ある一定の周期で同じ変化を繰り返す関数ですよね。

あの、波みたいになってるやつです。

で、今から標本化しようとするアナログ量 F が、

時間 t の関数として F = f(t)と定義されることにします。

そうすると、この F も、いろんな周期関数の和で表すことが出来ます。

A っていう周期関数、B っていう周期関数、C っていう…

これらの周期関数の波を重ね合わせると F の形になるってわけです。

で、A とか B とかの周期はそれぞれ違うんですが、そのうち最大の周期を T とします。

このときその周期関数の周波数 W は W = 1/T (Hz) で表されます。

周波数は、1 秒間に何回繰り返すか、という値です。

そして、次に掲げるのが**シャノン**によって導かれた標本化定理と呼ばれるものです。

標本化を行う際に、
1 / 2W（= T/2）未満の間隔で対象となる情報を標本化すれば、
元のアナログ関数 F を完全に復元できることが保証される。

たとえば、周期が 0.0025 秒の周期関数があるとします。
この周期関数の周波数は 400Hz です。
この関数を完全に復元するためには 1/800（= 0.00125）秒未満の間隔で標本化、
すなわち、１秒間に 800 回より多くの標本化を行えばいいわけです。

今度は逆に、標本化を行う間隔を $t_0$ と定めるとします。
このとき、$1 / 2t_0$ を「**ナイキスト周波数**」と呼びます。
上で扱った W の値がこのナイキスト周波数以上であると、
完全な復元を行うことは出来ません。
（内容としては上の標本化定理と同じようなことを言っています。）

では、1 / 2W 以上の間隔で標本化する
（＝元の関数に、ナイキスト周波数以上の周波数をもつ周期関数が含まれている）
と、どうなるのか。「**エイリアシング**」と呼ばれる現象が生じます。

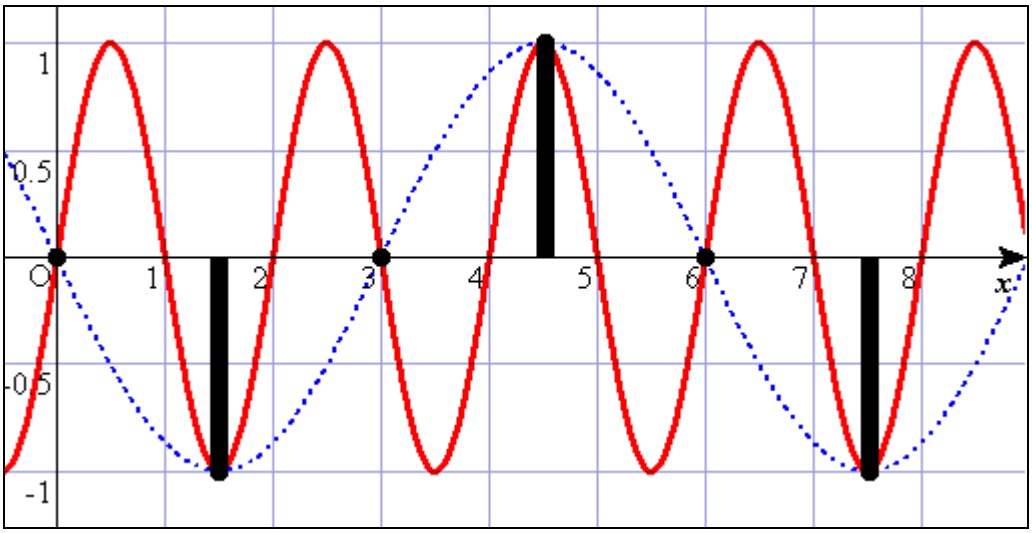


図 2-6

赤の実線が元のデータです。周期は2なので、1未満の間隔で標本化の必要があります。
にもかかわらず、1.5 という間隔で標本化してしまった場合について考えます。
黒で表されるデータが得られます。これを再びアナログ量として再現したいのですが、
元の赤のデータ以外にも、青の破線で表されるデータも考えられます。
そこで、コンピュータが、青い、誤ったデータを表示してしまう、
これが「エイリアシング」です。
同じことですが、
http://mars.elcom.nitech.ac.jp/java-cai/signal/sampling2.html
http://www.teu.ac.jp/hiha/SignalProcessing/aliasing/aliasing.html
ここでもエイリアシングについて説明がされていますので、参考までに。
画像においてもエイリアシングは起こり得ます。
そんなに重要じゃないと思いますけど、教科書 p.25 参照です。

さて、標本化定理の説明はこんなもんですかね。
もとの書式に戻りたいと思います。

◇音楽 CD における標本化
高い周波数の音も記録するのであれば、かなりの高周波での標本化が必要
→ 人間に聞き取れる範囲の周波数でＯＫ
→ 44.1kHz、つまり 1/44100 = 0.0000227 秒の間隔で標本化している

◇標本化定理のまとめ

| |
|---|
| 標本化定理は、<br>成分となる周期関数の最大周波数に応じた適切な細かい標本間隔を用いれば、<br>元のアナログ量を完全にディジタル量で扱えることを数理的に保証している。 |

# 第３章　情報の伝達と通信

【情報の伝達】

◆ＡさんがＢさんに伝える情報

Ａさんが持っている情報をＢさんに伝える場合をいくつかの状況にわけて考える

①手紙で送る

事前：Ａの手元に手紙があり、Ｂの手元にはない。

事後：Ａの手元に手紙はなく、Ｂの手元に手紙がある。

②手紙をコピーし、そのコピーを送る

事前：Ａの手元に手紙があり、Ｂの手元にはない。

事後：Ａの手元に手紙があり、Ｂの手元にそのコピーがある。

③電子メールで送る

事前：Ａの手元に電子メールが作成されており、Ｂの手元にはない。

事後：Ａの手元にはコピーが残る場合と残らない場合があるが、Ｂの手元には届いた。

①、②から、情報の伝達は「**受け取り側の状態の変化**」に関係していることがわかる

③を見ても「送信側の状態の変化」は本質でないことが確かめられる

また、①②と、③を比べると、伝達の手段（メディア）とも独立であることがわかる

【情報を受け取った効果】

◆「情報を受け取った」とは

・自分に影響のある、これまで知らなかった事実を知った

・何らかの判断の材料にできる事実を知った

※全く関心のない手紙をもらっても、情報を受け取ったとは言えない（迷惑メール等）

◆歴史の試験範囲の例

事前に伝えられた出題範囲は日本史、東洋史、西洋史、アメリカ史、のうちから１つ

◇「今回は日本史から出題する」というメッセージが与えられた場合

事前：日本史からアメリカ史の４種類すべての試験勉強が必要

事後：日本史の勉強だけでよい

出題に関するあいまいさが４から１に減った → かなり有用な情報

◇「今回は世界史（日本史以外）から出題する」というメッセージが与えられた場合

事前：日本史からアメリカ史の４種類すべての試験勉強が必要

事後：東洋史、西洋史、アメリカ史の３種類の勉強だけでよい

出題に関するあいまいさが４から３に減った → それほど有用でない

【情報量】

◆情報量の表し方Ⅰ（場合の数に基づいた場合）

◇**商の対数（底は２）で表す情報量**

定義：**情報量 ＝ $\log_2$(事前の場合の数 ÷ 事後の場合の数)**

単位：ビット

性質：場合の数が大きく減る程、情報量も大きくなる

二者択一（場合の数２→１）の時に情報量が 1.000 になる

前頁のメッセージの情報量：日本史→$\log_2(4/1) = 2.000$／世界史→$\log_2(4/3) = 0.415$

※情報量を差（事前の場合の数 － 事後の場合の数）で表そうとすると

場合の数が 100→97 の場合と 4→1 の場合で等価になってしまう

明らかに後者の方が有用で、情報量は大きくあるべき

※情報量を商（事前の場合の数 ÷ 事後の場合の数）で表そうとすると

情報の加法性を満たさない

◇**情報の加法性**

ある情報Ｘを一度に受け取った場合の情報量と、

ある情報Ｘを何段階かに分けて受け取った場合の情報量は、等しい

Ex.歴史の試験範囲の例（図 3-1）

「今回はアメリカ史から出題」というメッセージの情報量

| |

「今回は世界史から出題」というメッセージが与えられた上で

「東洋史、西洋史は出題しません」というメッセージが与えられた時の情報量

対数で表された情報量はこの条件（情報の加法性）を満たす

①「今回は世界史から出題」の情報量：$\log_2(4/3) = 0.415$

②「(①の後で）東洋史、西洋史以外から出題」の情報量：$\log_2(3/1) = 1.585$

③「今回はアメリカ史から出題」の情報量：$\log_2(4/1) = 2.000$

①0.415 + ②1.585 = ③2.000

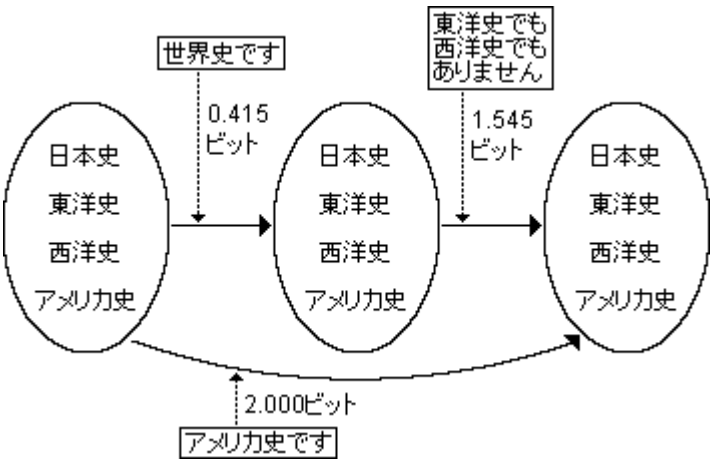



図 3-1

◆情報量の表し方Ⅱ（確率に基づいた場合）

◇**確率の逆数の対数（底は２）で表す情報量**

定義：**情報量 ＝ $-\log_2$(確率)**

単位：ビット

性質：低確率の（＝珍しい）ことを伝えるメッセージの方が情報量は大きい

→「犬が人間を噛んだ」に比べて「人間が犬を噛んだ」の情報量の方が大きい

→ニュースになる

すべてが等確率で起こる場合は、場合の数で表しても同じ結果になる

▽白い小球と赤い小球

Ａの袋には５つ、Ｂの袋には２つ、Ｃの袋には３つ、小球が入っている

10 個の小球は９つが白く、１つだけが赤い

中を見ないでいずれかの袋からひとつだけ小球を取り出す

取り出した小球が赤い小球であれば大当たり、ハワイ旅行１年分がもらえる

事前の場合の数は「赤い小球がＡ［Ｂ，Ｃ］に入っている」の３通り

では、仮に下の①②③いずれかの情報が入ったとき、どれが最も有用といえるか

①「赤い小球はＡの袋に入っている」 → 確率は 50%

②「赤い小球はＢの袋に入っている」 → 確率は 20%

③「赤い小球はＣの袋には入っていない」 → 確率は 70%

場合の数に基づけば③は３→２で、情報量は 0.585 ビット

①②は３→１で、情報量は 1.585 ビットで等しくなる

これでは情報量を正しく表しているとはいえない

そこで、確率を考慮に入れる

①の情報量：$-\log_2(50\%) = -\log_2(1/2) = \log_2(2/1) = 1.000$

②の情報量：$-\log_2(20\%) = -\log_2(1/5) = \log_2(5/1) = 2.322$

③の情報量：$-\log_2(70\%) = -\log_2(7/10) = \log_2(10/7) = 0.515$

情報量の最も大きい②が、最も有用といえる

【情報通信】

◆**プロトコル**

プロトコルとは、お互いの意図を理解するために定める約束事

「トランシーバで話す時は最後に『どうぞ』という」

「クーショとの会話が仏語でも日本語でも上手くいかなかったら英語で話す」

なども一種のプロトコル

人間なら少々約束違反があっても理解できることがあるが、コンピュータにはできない

→コンピュータ同士の通信ではより厳密にプロトコルが規定

◆暗号化

メッセージを送信するとき、平文のまま送信すると第三者に不法に読み取られる危険

→「鍵」に基づいて意味を持たない文字列に変換する … **暗号化**

**復号化** … 暗号化された文字列を、「鍵」に基づいて平文に変換し直すこと

**◇共通鍵暗号**

暗号化と復号化に同じ鍵を用いる

鍵の受け渡しの時点で他者に鍵を知られないように注意

▽植田の陰謀 with 加藤さん（図 3-2）

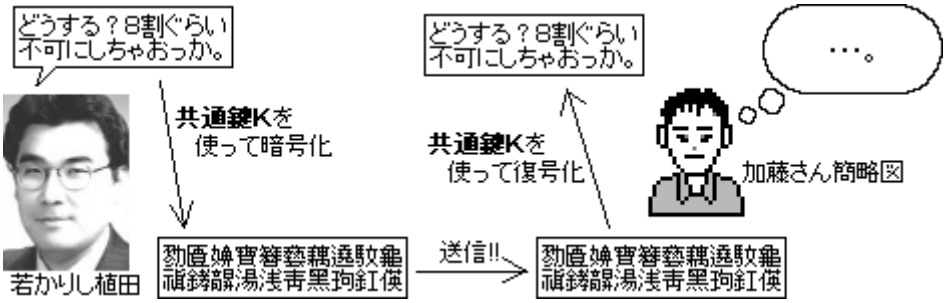



図 3-2

**◇公開鍵暗号**

暗号化の鍵は誰でも入手可能にしておく … 公開鍵

復号化の鍵は自分だけが持っておく … 秘密鍵

▽「兄、妹が心配」の巻（図 3-3）

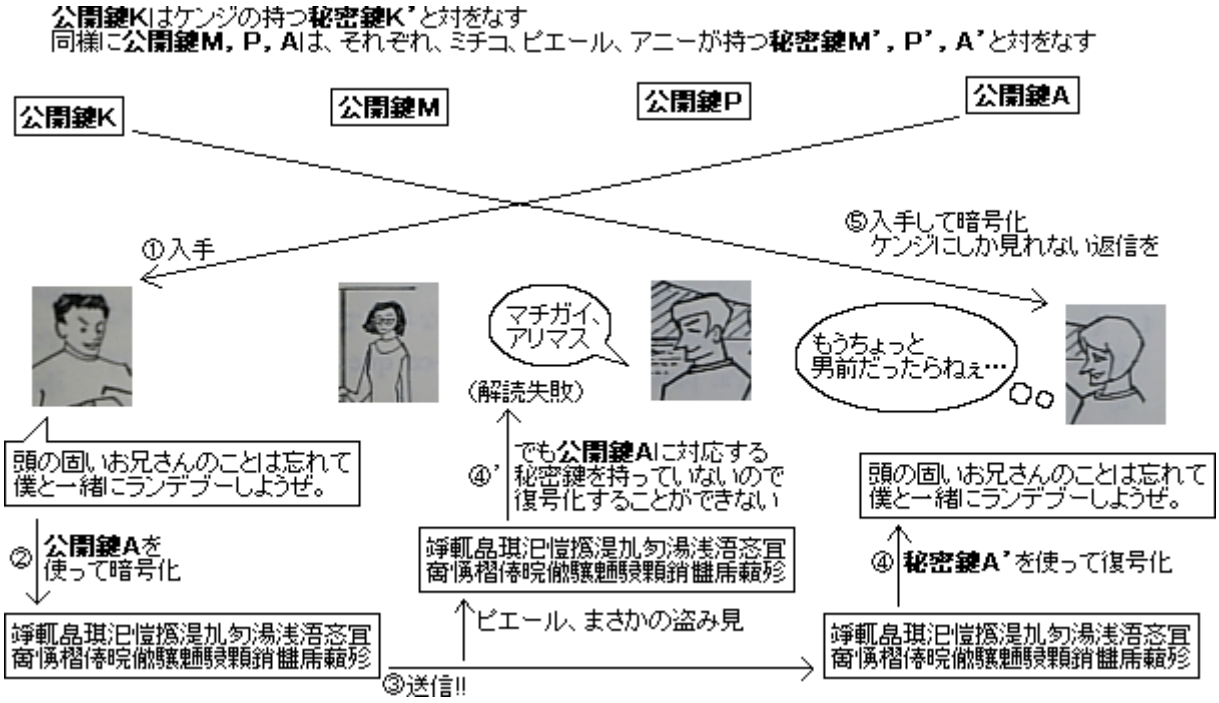



図 3-3

【情報ネットワークの枠組】

◆交換の方式

**交換機** … 通信される情報を経路に振り分ける装置

**◇回線交換**

通信する両者の間に仮想的な通信路を確保

つながった両者の間では安定した通信が可能

回線を確保できなければまったく通信できない

特に通信するデータがなくても回線は確保されたまま → 回線の利用効率が低い

▽ケンジのジレンマ（図 3-4）

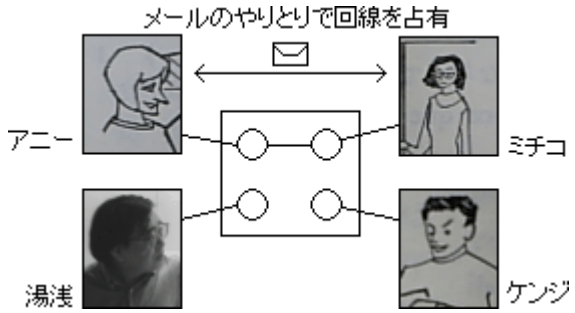


図 3-4

①ケンジはアニーとメールのやりとりをしたい

②アニーはミチコとメール中

③回線が使われているのでアニーとの通信はできない

④じゃぁミチコでいいや、と思ったが、同様にミチコとも通信不可

⑤ケンジの通信できる相手は湯浅しかいない

**◇パケット交換**

データをパケットに分割 → 次々と交換機に届け、交換機はそれを順次配送

１パケット送るたびに回線の占有は解かれるので、どの両者も通信可能

インターネットではこの方式が主流

通信速度の保証は難しい … パケット管理のための遅延、混雑時の速度低下

▽それぞれの思い（図 3-5）

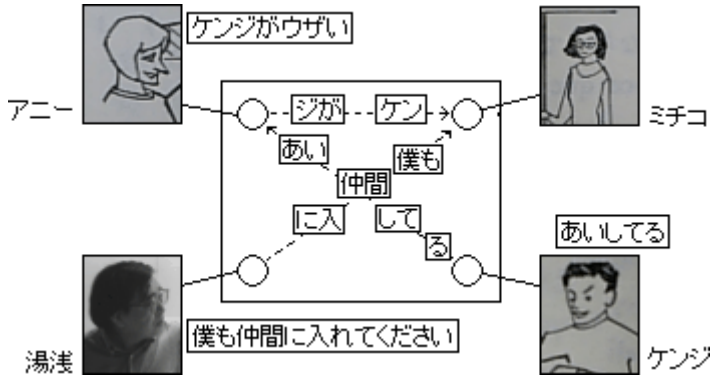


図 3-5

同時に複数方向へのメールを送ることが出来る

【インターネット】

◆プロトコル

現在のインターネットでは **TCP/IP** と呼ばれるプロトコル群が使われる

◇TCP/IP

４階層から成る**階層プロトコル**　図 3-6

| TCP/IPモデル | 主なプロトコル | 主な役割 |
|---|---|---|
| アプリケーション層 | **HTTP**,SMTP | アプリケーション間の通信 |
| トランスポート層 | **TCP**,UDP | 信頼性のある1:1の通信 |
| インターネット層 | **IP** | ネットワーク間通信 |
| ネットワークインタフェース層 | （イーサネット） | ネットワーク内通信 |

**TCP** … データのパケット分割・合成を担当

**IP** … 分割されたパケットを宛先に届ける、また、そのために適切なルートを選ぶ

◇**カプセル化**

上位のプロトコル（図 3-6 において上にある方が上位）に則って作成されたデータ

… 下位のプロトコルのための制御データが付加される

先頭に付加される物 … **ヘッダ**

末尾に付加される物 … **トレーラ**

さらに下位のプロトコルによる通信を行うには、

ヘッダを含めた全データに対して再びカプセル化が行われる（図 3-7）

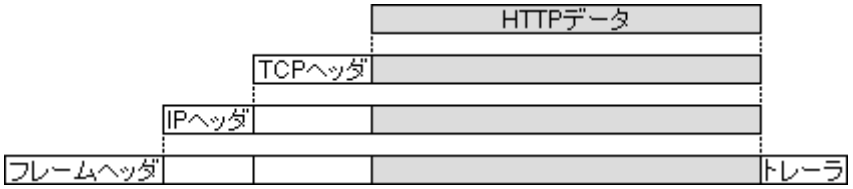


図 3-7

◆ネットワークの集合体と通信

**インターネット** … 小規模なネットワークが互いに接続された集合体（図 3-8）

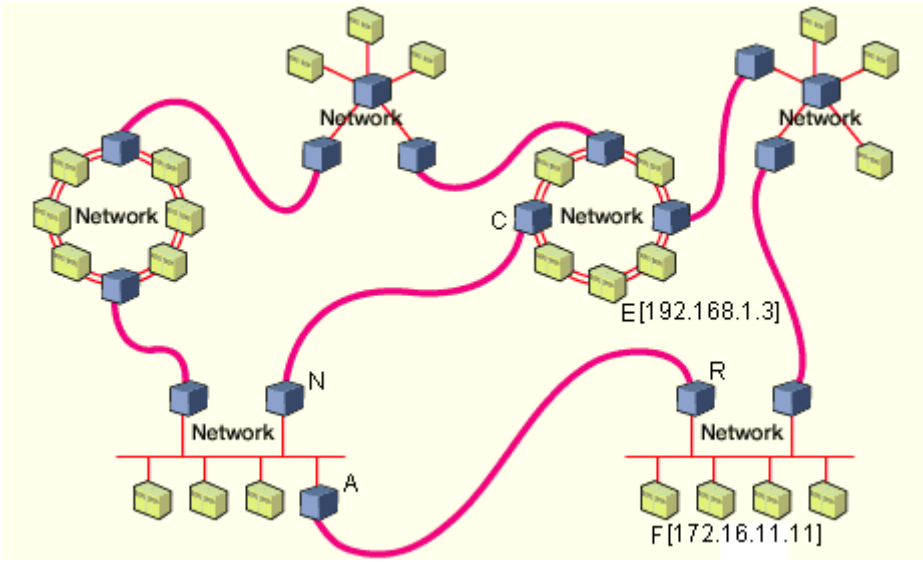


図 3-8

**ルータ** … ネットワーク同士を接続する中継器機（図 3-8 でいうと紫のやつ）

◇マシン F のウェブブラウザで、マシン E のサーバにあるウェブサイトを閲覧する例

①ブラウザがウェブサイトの URL（アドレス）から **IP アドレス**を調べる

IP アドレスはコンピュータの住所のようなもので、32 ビットの数字列で表される

それを人間にわかるように置き換えたのが **URL**

②ブラウザが、宛先の IP アドレスのウェブサーバに対して HTTP のメッセージを送る

(a)送信元マシン F がメッセージをパケットに分割（TCP プロトコル）

(b)分割されたパケットを宛先の IP アドレスに送信する

(c)各パケットはまず、宛先の IP アドレス向けの適切なルータである R に届けられる

(d)F→R→A→N→C→E の適切な順に転送される（IP プロトコル）

(e)受信したマシン E はパケットを元の順番に並べ、HTTP メッセージを読みとる

③(e)のメッセージを受け取ったウェブサーバがブラウザにメッセージを返す

（②の行程を逆に辿る）

※情報の加法性についての補足

Ex.歴史の試験範囲の例（図 3-1）

「今回はアメリカ史から出題」というメッセージの情報量

||

「今回は世界史から出題」というメッセージが与えられた上で

「東洋史、西洋史は出題しません」というメッセージが与えられた時の情報量

と、書きましたが、実際我々がこれらを等価と考えられるかというと、必ずしもそうではないことがわかると思います。「今回は世界史から出題」というメッセージを得た後、アメリカ史以外の勉強をしてしまった人にとっては「東洋史、西洋史は出題しません」というメッセージを聞いたら損した気分になります。この損した気分は、最初から「今回はアメリカ史から出題」というメッセージを得ていたら味合わずにすんだものですから、やはり２つの情報が等価とは言い難いです。時間の経過が進めば進むほど情報量は失われていくというのが、我々の感じるところです。しかし、ここでは、時間という概念や、勉強を進めてしまうという偶然の事象は考えずに、情報量を扱っています。教科書にも「この加法性は絶対的なものでなく、情報量を扱いやすくするための要請である。」と書かれています。まぁ、細かいことは気にするな、ってことです。

# 第４章　データの扱い

【データモデル】

◆データとデータモデル

**データ** … コンピュータの処理対象となる符号化された情報

**データモデル** … データを体系的に扱うためのモデル

【代表的なデータモデルと演算】

**◆集合モデル**

グループは集合で表すことが多い … 集合Ａと集合Ｂを定義

**共通部分（積集合）**[＝Ａ∩Ｂ]、**和集合**[＝Ａ∪Ｂ]、**差集合**[＝Ａ－Ｂ]の演算をもつ

対象「ＡかＢのメンバーでＣのメンバーでない人たち」→ モデル「（Ａ∪Ｂ）－Ｃ」

Ex.検索エンジンの AND 検索、OR 検索、NOT 検索

◇ベン図とオイラー図

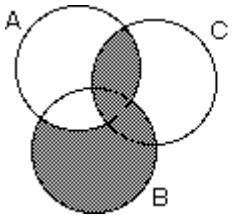




図 4-1

**ベン図** … 図 4-1 左図、円の重なった部分で共通部分を表す、網掛け部には要素がない

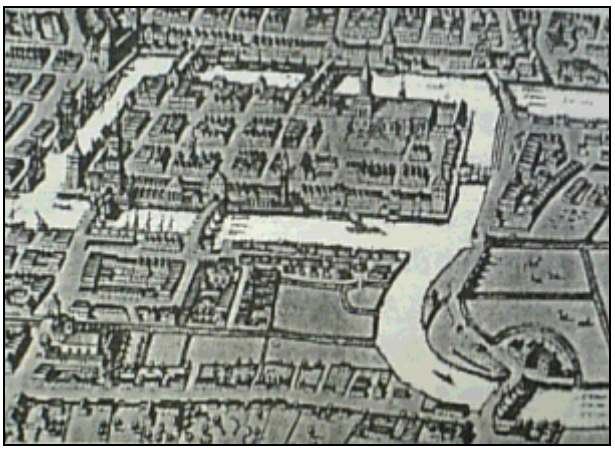
**オイラー図** … 図 4-1 右図、共通部分がなければ別々の円で表す

完全な包含関係の場合は円の中に円を入れる

**◆ネットワークモデル**

◇ケーニヒスベルグの橋（図 4-2、図 4-3）

図 4-2



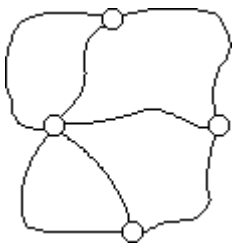


図 4-3

「同じ橋を二度渡らずに、すべての橋を渡ることは可能か」

→この問題を考えるために図をモデル化（図 4-2 → 図 4-3） … **グラフ**で表す

→「それぞれの線（エッジ）を一度だけ辿って、すべてのエッジを辿れるか」

オイラー経路 … すべての線を重複なく辿る経路

◇ウェブ

ウェブのそれぞれのページをノード（図 4-3 でいうと○）と考える

→ それぞれのリンクはノードとノードをつなぐ矢印(エッジ)と見ることが出来る

**◆階層モデル／木構造**

**◇階層的ファイルシステム**

コンピュータのファイルシステムも木構造になっていることが多い（図 4-4）

▽ファイルシステムを木構造にすることによる利点

①フォルダの中にサブフォルダを作って細かい仕分けができる

②フォルダを分けておけば、それぞれのサブフォルダが混ざることはない

③フォルダをたどることで、すべてのフォルダとファイルを重複なく処理できる

④サブフォルダ以下の内容を、そのサブフォルダで代表させることができる

たとえば、階層構造の中でサブフォルダを移動すれば、

サブフォルダ以下をまとめて移動させることが出来る

**◇住所の階層性**

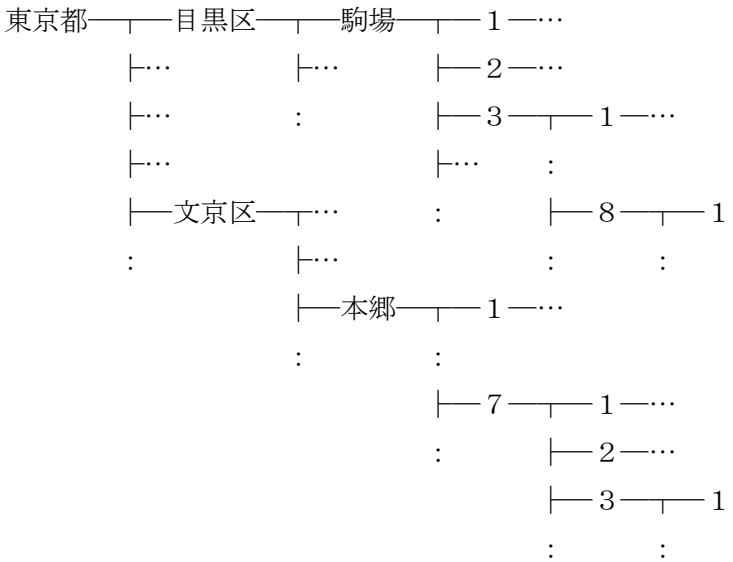
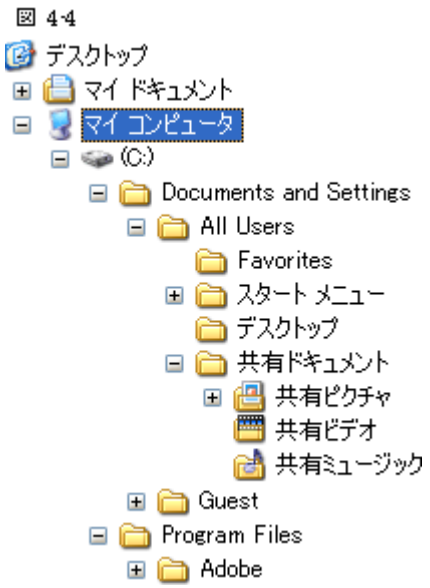



図 4-4




▽**ドメイン名**も階層構造になっている

u-tokyo.ac.jp … jp（日本）→ac（大学等）→u-tokyo（東京大学）→…

◆**関係モデル**

「関係」によるモデル化

**関係** … 実際に関係のある組を集めたデータ

Ex.「山口君は東京に住んでおり、電話番号は 03-1987-0427」という事実

→山口、東京、03-1987-0427 の関係

**正規化** … 同じデータが何回も出てくるような冗長な関係を、分割して簡潔にすること

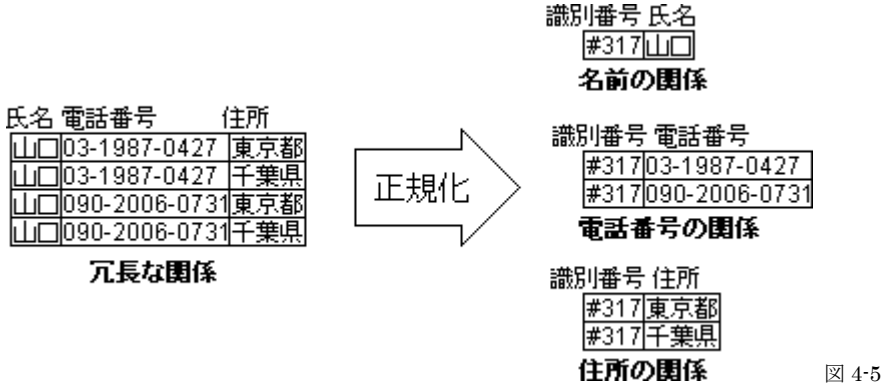



図 4-5

◇**関係代数**

▽５つの演算から成る

①**和** … ２つの関係の組を集める

②**差** … 関係の組の中で他の関係にないものを集める

③**射影** … 関係の組から不要な項目を除いた組を集める

④**選択** … 関係の組のなかで指定された条件を満たすものだけ集める

⑤**直積** … ２つの関係の組を総当たりでつないだ組を集める

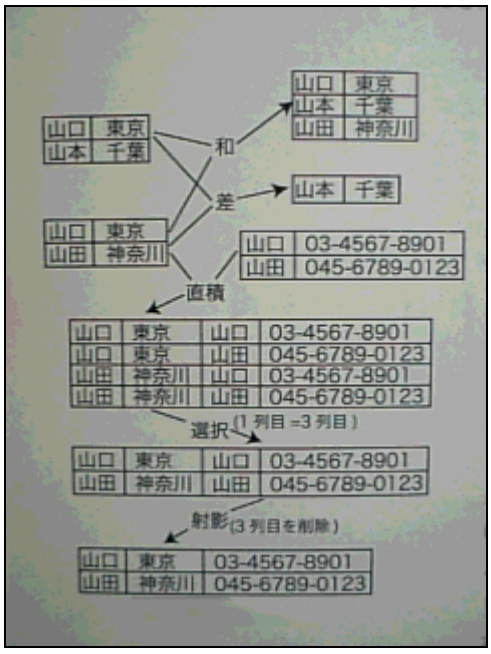



図 4-6

# 第５章　計算の方法

【計算の方法】

◆計数（counting）の方法

ある集合Ａに含まれる要素数を求める

◇**取り出し型**（図 5-1）

↓<答>（変数）を 0 にする

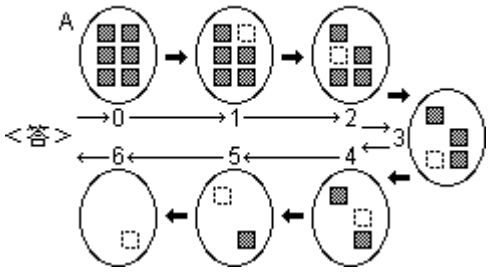
↓Ａが空でない間、以下の処理を繰り返す

↓要素を１つ取り出す

↓<答>の値を１増やす

↓<答>の出力

図 5-1




◇**分割型**（図 5-2）

↓Ａが空なら<答>は 0、要素数が１なら<答>は１

↓そうでなければ

↓ＡをＡ$_B$とＡ$_C$に分割する（Ａ$_B$もＡ$_C$も空集合ではない）

↓Ａ$_B$、Ａ$_C$に関してもＡと同じ処理を行う

↓<答>＝Ａ$_B$の要素数＋Ａ$_C$の要素数

↓<答>の出力

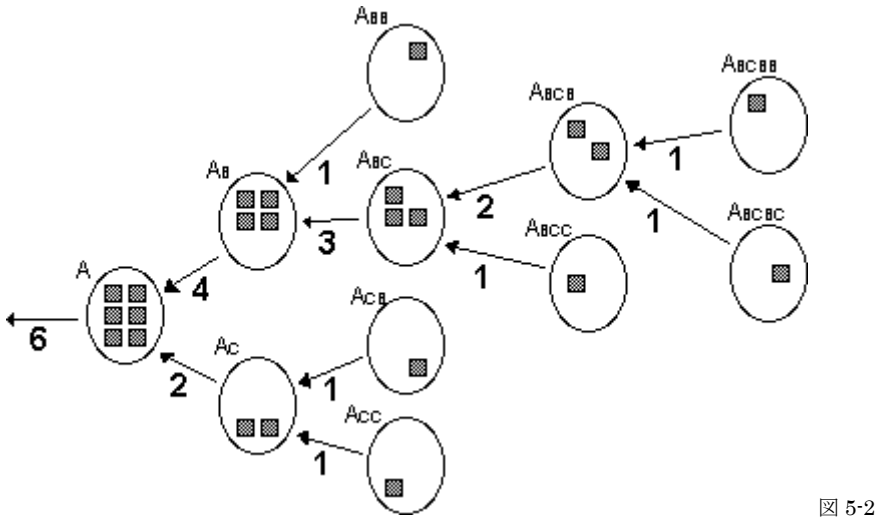



図 5-2

【計算の記述】

**◆八十八夜問題**

| 今年の八十八夜は何月何日か。ただし、今年の立春は２月４日で、今年は平年である。 |
|---|

八十八夜 … 立春から数えて 88 日目の日（＝立春の 87 日後）

◇解法の手順

２月４日の 87 日後を求めたい

→２月 91（＝4＋87）日という仮想日付になる

<残り日数>を 91 にする

91＞28（２月の日数）なので

<残り日数>から２月の日数を引き、３月に進む

→３月 63（＝91 - 28）日という仮想日付になる

63＞31（３月の日数）なので

<残り日数>から３月の日数を引き、４月に進む

→４月 32（＝63 - 31）日という仮想日付になる

32＞30（４月の日数）なので

<残り日数>から４月の日数を引き、５月に進む

→５月 2（＝32 - 30）日という仮想日付になる

2＜31（５月の日数）なので、計算終了

◇よりプログラミングっぽい記述

<残り日数> … はじめは 91 に設定 → 次第に小さい値に変化 … 「**変数**」と呼ぶ

このとき、「残り日数」という文字列自体を「**変数名**」と呼ぶ

**代入** … 変数に値を設定する操作

**「変数名」 ← 「値 or 式」** という形式で表すことにする

**逐次処理** … 計算処理の基本の１つ

途中を抜かしたり、複数の処理を同時にやったりせず、

書かれている順番に順序よく処理

**条件付き処理** … ～ならば…という処理を行う、という操作

```
if 条件
  then 条件が成立した場合に行う処理
  else そうでない場合に行う処理
endif
```

という形式で表すことにする（else は省略が可能）

計算の記述で使われる「変数」 → 値を書き換えられる

数学で使われる「変数」 → 値と結合する … 「**パラメタ**」と呼ばれる

◇よりプログラミングっぽい記述を用いて書き直し

<残り日数> ← 4＋87
if <残り日数> ＞ 28（２月の日数）
then <残り日数> ← <残り日数>－２月の日数
*if <残り日数> ＞ 31（３月の日数）*
*then <残り日数> ← <残り日数>－３月の日数*
if <残り日数> ＞ 30（４月の日数）
then <残り日数> ← <残り日数>－４月の日数
if <残り日数> ＞ 31（５月の日数）
then <残り日数> ← <残り日数>－５月の日数
:
（６月以降の処理）
:
else “５月”<残り日数>”日” と表示
endif
else “４月”<残り日数>”日” と表示
endif
*else “３月”<残り日数>”日” と表示*
*endif*
else “２月”<残り日数>”日” と表示
endif

▽**字下げ（インデント）**

この例では then と else が、対応する if と endif よりも右にずらされている
→ まとまりの構造をわかりやすくするため
たとえば、緑の斜体文字で表された部分は、３月の処理を表すひとまとまり

◇反復処理と配列

**反復処理** … ある条件が成立している間、指定された処理を繰り返す操作

**while 条件 do**
**繰り返し実行する処理**
**done**

という形式で表すことにする

**配列** … 要素の集合から**添字**によって値を取り出し、変数として扱うことができる
この要素の集合が、配列
たとえば、daymonth という配列には１月～12 月の日数が要素として格納
$daymonth_1$ ＝ 31、$daymonth_2$ ＝ 28 … 1、2 が添字、31、28 が要素
配列全体をまとめて扱うこともできる

◇反復処理と配列を用いて書き直し（daymonth には既に要素が格納済み）

<残り日数> ← 4＋87
m ← 2
while <残り日数> ＞ $daymonth_m$ do
　<残り日数> ← <残り日数>－$daymonth_m$
　m ← m＋1
done
m“月”<残り日数>”日”と表示

これがわからない人は、今からコンピュータになったつもりで順に記述を辿ってみましょう。<残り日数>って書くと文字数食うので代わりに「凹」って書きますね。なんか箱っぽいから（変数は値を格納する箱と喩えられます)。それではスタート。

よし、まず1行目。凹に 91 を入れて、と、2行目、m に 2 を入れる、と。3行目から6行目までは反復処理だな。じゃぁ while の条件を見てみよう。「凹 ＞ $daymonth_m$」か。これが真のとき、do と done に挟まれてる処理を行えばいいんだな。今凹は 91、m は 2 で、$daymonth_2$ っていったら 28 だから、「91 ＞ 28」だ。これは真だから、中の処理をやっちゃおう。凹に凹－$daymonth_m$ を代入。と、いうことは、凹は 91（凹）－28（$daymonth_2$）に書き換わるわけだから、新しい凹の値は 63 だ。で、m の新しい値は 2＋1 で 3 だな。よし、中の処理は一通り終わったな。反復処理は条件が真である限りまた中の処理を行うんだから、ここでもう一度条件の確認だ。今凹は 63、m は 3 で、$daymonth_3$ っていったら 31 だから、「凹 ＞ $daymonth_m$」は「63 ＞ 31」、真だな。じゃあもう一回中の処理を行おう。凹－$daymonth_m$ は、63－31 で 32、つまり新しい凹の値は 32。m は 4。done 以上の処理が終わったからもう一度条件の確認。「凹 ＞ $daymonth_m$」、「32 ＞ 30」、まだ真だ。凹－$daymonth_m$ は、32－30 だから凹は 2、m は 4 の次だから 5 だ。そろそろどうだろう、条件をもう一度確認してみよう。「凹 ＞ $daymonth_m$」に、凹＝2、m＝5 を当てはめてみると、「2 ＞ 31」、お、これは偽だ、条件が成立していない。じゃあ、これで while～done の処理はおしまいだ。次へ行こう。「m“月”凹”日”と表示」すればいいんだな。今 m は 5、凹は 2 だから、表示する文字列は、「5 月 2 日」。これでオッケー。

はい、こんな感じです。わかんねえよ、ばか。って怒っちゃった人はビシバシメールをください。

【プログラム言語処理系】

◆機械語とソフトウェア（用語集的な感じで。これで十分だと思いますが、教科書は p115）

**機械語** … コンピュータ（ハードウェア）が理解できる言語（すべて２進数）

**バイナリプログラム** … 機械語で書かれたプログラムの別名

**ハードウェア** … コンピュータの、バイナリプログラムを解釈し実行するための部分

**ソフトウェア** … バイナリプログラムを含む各種のプログラム

**アセンブリ言語** … 機械語の機能部分やデータの場所に、人が読める名前をつける言語

**アセンブラ** … アセンブリ言語でつけた名前を、
数値や”0 と 1 の集まり”（機械語）に変換するもの（図 5-3）

**言語処理系** … “プログラム言語で書かれたプログラム”を、
“機械語で書かれたプログラム”に変換（翻訳）するシステム（図 5-4）

**高水準言語** … 人間が理解しやすい形でプログラムを読み書きできる言語
そのままではコンピュータは実行できない
C 言語、C++、Java など

**コンパイラ** … プログラム言語で書かれたプログラムを機械語に変換するソフトウェア
言語処理系の１つ
ソースプログラムに誤りがあれば指摘
→プログラムが誤りを含んだまま実行されてしまう危険が減る

**ソースプログラム** … コンパイラで変換される前のプログラム

**オブジェクトプログラム** … コンパイラで変換されたプログラム
ライブラリと結合して最終的なバイナリプログラムとなる

**ライブラリ** … システムに予め用意されている標準機能を提供するプログラム群
たとえば、データの入出力、コンピュータ内でのデータ管理など

**インタプリタ** … コンパイラを通さずに実行可能な言語処理系
いちいちコンパイル（機械語に変換）しなくていいので、
プログラムの変更は容易、ただし解釈に時間がかかり実行は遅い

**中間言語方式** … ソースプログラムを”ある程度機械語に近い”段階にまで変換
実際の実行は中間言語で記述されたプログラムを、
また別のプログラムで解釈実行することで実現

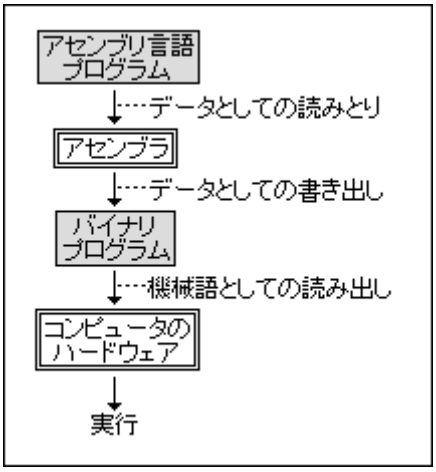

アセンブリ言語
プログラム
…データとしての読みとり
アセンブラ
…データとしての書き出し
バイナリ
プログラム
…機械語としての読み出し
コンピュータの
ハードウェア
実行

図 5-3

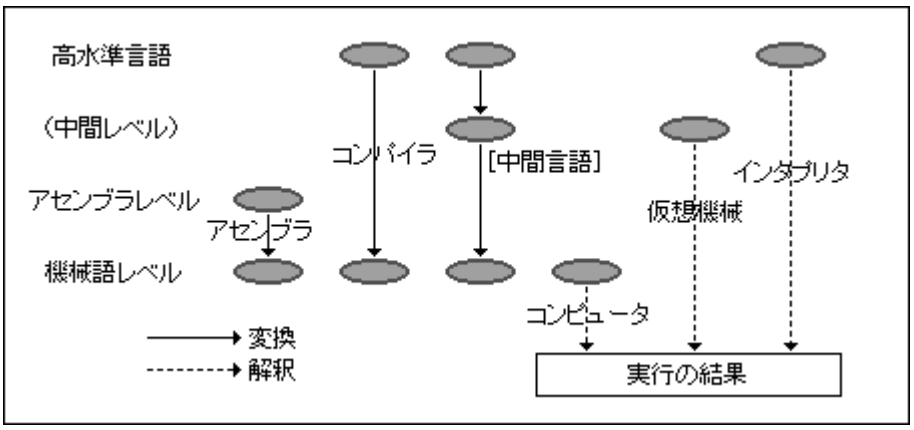

高水準言語
（中間レベル）
コンパイラ
[中間言語]
インタプリタ
仮想機械
アセンブラレベル
アセンブラ
機械語レベル
コンピュータ
変換
解釈
実行の結果

図 5-4