

# 計算機の歴史と 働く仕組み

アトリエ未来【著】

暮らして生きるコンピュータとして

計算機の歩みとその仕組みを

情報社会を築いた技術者や

アルゴリズム/プログラミングとともにご紹介します

# 計算機の歴史と働く仕組み

— 暮らしに生きるコンピュータ —

[著] アトリエ未来

「技術書典 14 新刊」  
令和五年五月五日

### **■免責**

本書は情報の提供のみを目的としています。

本書の内容を実行・適用・運用したことで何が起きようとも、それは実行・適用・運用した人自身の責任であり、著者や関係者はいかなる責任も負いません。

### **■商標**

本書に登場するシステム名や製品名は、関係各社の商標または登録商標です。

また本書では、<sup>TM</sup>、<sup>®</sup>、<sup>©</sup>などのマークは省略しています。

# 始めに

楽しいプログラミングの世界へようこそ。

情報技術「IT」に囲まれた生活を送るわたくしたち。多くの先人が築いた歴史の上に今日があります。<sup>こんにち</sup>

本書では、「計算機の歴史と働く仕組み - 暮らしに生きるコンピュータ」として、算盤から iPhone に至る計算機の歴史をご案内し、そして今日の情報社会を拓いた數学者・科学者・技術者 24 人の略伝や、コンピュータが動作する為の仕組みとして、二進数やデジタルデータの表現法、ハードウェア、アルゴリズムや学校教育におけるプログラミングについても簡単にご紹介しています。

また巻末には、様々な興味関心から知的好奇心を満たせるよう、お勧め書籍もご紹介いたしております。

現代の魔法、それがプログラミングです。自由自在にコンピュータを操って、幸せな未来へと大きく羽ばたいていってください。

## ♣ 対象読者

コンピュータやプログラミング、歴史や偉人伝に興味関心をお持ちの方を想定しています。中高生から社会人まで、多くの方にご愛読いただければ幸いです。

## ♣ 謝辞

Re:VIEW Starter<sup>a</sup>を用いて、快適に執筆することができました。作者の kauplan さんに厚く御礼申し上げます。

---

<sup>a</sup> <https://kauplan.org/reviewstarter/>

また、表紙絵の女の子は、千葉県松戸市在住のフリーランス SD イラストレー  
ター 早瀬ひろむ<sup>a</sup>さんの作品です。素敵なイラストを描いてください、感謝です。

---

<sup>a</sup> <https://hiromu-hayase.tumblr.com>



早瀬ひろむ さん

背景の桜と青空の絵は、くらうど職人<sup>a</sup>さんの作品です。桜咲く青空で心  
も明るくなります。

---

<sup>a</sup> <https://www.photo-ac.com/profile/590911>



くらうど職人 さん

また、信頼できる文献に触れて欲しいとの思いから、[ウィキペディア<sup>\\*1</sup>](#)等、多くの文献より引用させていただいております。貢献に感謝するとともに、厚く御礼申し上げます。

## ♣ 著者紹介



卓越した技能を有する者として認められる国家資格「応用情報技術者」を保持。平成30年より「アトリエ未来<sup>a</sup>」を創業。HTML講座やRuby講座などプログラミングの個人指導や、ITパスポート講座等の資格講座の開催、ウェブサイト作成等を受注している。

趣味の将棋は、日本将棋連盟より三段の免状を允許。日本の美しい自然や豊かな精神性を宿す熊野古道を歩くことや、たくさんの花に囲まれた日々を愛している。

---

<sup>a</sup> <https://atelier-mirai.net/>

## 【コラム】金の延棒クイズ

二進数の不思議を感じながら、豊かになれるクイズです。(正解はこの本のどこかに)



七日の給料の支払いとして金の延べ棒が一本あります。これを使って、毎日の日当を支払いたいと思います。

六回鉢を入れて七等分すれば日払いできますが、金の延べ棒を六回も切り取るのは大変です。

二回切り取るだけで、日当を支払えるのですが、どことどこを切れば良いでしょうか。

---

<sup>\*1</sup> <https://ja.wikipedia.org/>

# 目次

<b>始めに</b>	i
<b>第1章 暮らしの中のコンピュータ</b>	1
<b>第2章 コンピュータの歴史</b>	3
2.1 さまざまな計算 - 曆・税・工事 - . . . . .	3
2.2 算盤から現代のコンピュータまでの歩み . . . . .	8
<b>第3章 コンピュータを創った人々</b>	15
<b>第4章 コンピュータの仕組み</b>	23
4.1 コンピュータ / 電子計算機 / 電算機 . . . . .	23
4.2 ハードウェア . . . . .	25
4.3 ソフトウェア . . . . .	27
<b>第5章 二進数の話</b>	29
5.1 二進数と十進数、十六進数 . . . . .	29
5.2 コンピュータでのデータ表現 . . . . .	30
5.3 単位の話 . . . . .	36
5.4 計算機理論入門 . . . . .	37
<b>第6章 アルゴリズム とは</b>	41
6.1 アルゴリズムとは . . . . .	41
6.2 データ構造 . . . . .	43
6.3 アルゴリズムと効率性 . . . . .	46
6.4 素数を求めるプログラム . . . . .	47
<b>第7章 プログラムとは</b>	55
7.1 「プログラム」 = 「コンピュータへの指示書」 . . . . .	55
7.2 プログラム・プログラミングの定義 . . . . .	56
7.3 プログラミング言語の種類 . . . . .	57
7.4 学校教育でのプログラミング . . . . .	58
7.5 大学入試試験 情報 . . . . .	61
<b>第8章 ウェブの歴史と技術</b>	63
8.1 ウェブサイトの発祥 . . . . .	63
8.2 HTML とは . . . . .	64
8.3 CSS とは . . . . .	65
<small>ジャバスクリプト</small>	
8.4 JavaScript とは . . . . .	66

<b>付録 A 珠玉の名著のご紹介</b>	<b>69</b>
A.1 コンピュータを作った人々を訪ねて . . . . .	69
A.2 計算機科学に関する一般教養を身に付けたい方の為に . . . . .	71
A.3 プログラミングを始めたい方に . . . . .	74
<b>終わりに</b>	<b>75</b>

# 第 1 章

## 暮らしの中のコンピュータ

コンピュータが歩んできた歴史を巡る旅の始まりは、身の回りで使われているコンピュータを発見するところから始まります。様々なところで使われているコンピュータ。皆さんも一緒に探求していきましょう。



▲ 図 1.1: 暮らしの中に生きるコンピュータ

### コンピュータ (Mac / iPad / iPhone)

なんといってもコンピュータの代表です。いろいろなウェブサイトを見たり、書類作成などお仕事に活用したり、そして「プログラミング」など。iPad でお絵描きや読書、iPhone で連絡を取り合ったり、写真や音楽、ゲームなどを楽しむことも出来ます。ロケットを打ち上げ、宇宙観測や天気予報に活かしたり、都市計画から住宅設計、工場で車や船、飛行機など様々なものを清算したり、音楽や映画、アニメーションなど芸術の分野に至るまで、様々な分野でコンピュータは使われています。

### 衣服

「天衣無縫」 - 「天人や天女の着物には縫い目がないという意から、詩文などが、よけいな修飾がなく、自然でわざとらしくなく完成されていること。また、人柄が純真で素直で、まったく嫌みがないさま。物事が完全無欠であることの形容 (goo 辞書)」

衣食住は、人が生きる上で生活の基盤となる要素です。大麻、木綿、絹糸、<sup>いにしえ</sup>古の昔から日本人の身を纏う衣服に用いられてきました。縦糸と横糸を編んで一枚の布にして、布から切り取つて縫いあわせて一着の衣服を仕立てていきます。コンピュータの活用により編み機から直接衣服を生み出すことが出来るようになりました。縫い目がないから着心地も良く、布の無駄もないと特徴を持ちます。夢であった「天衣無縫」が顕現した瞬間です。

### 炊飯器

美味しい御飯を炊き上げてくれる炊飯器。「初めちょろちょろ、中ぱっぽ、赤子泣いてもふた取るな」と、朝早くから起きて竈で御飯を炊くのはなかなか難しいものでした。キャンプで飯盒炊飯した経験をお持ちの方もいらっしゃると思いますが、火加減が難しく焦げになってしまったり芯が残ってしまったこともあるかと思います。火力の調整をしたり、毎朝御飯が炊き上がるためのタイマー機能など、小さなコンピュータ（マイコン）を組み込むことで、美味しい御飯を頂けるようになりました。

### エアコン

部屋にあるエアコン。暑い夏には涼風を、寒い冬には暖風を送り、快適に過ごせるよう室温を調整しています。部屋の温度を感じるセンサー機能、設定温度を保つように計算する演算機能、実際に風を送り出す送風機能から成り立っています。

### 給湯器

台所や浴室の給湯器もコンピュータの産物です。昔の人のお風呂として思い浮かぶのは、五右衛門風呂でしょうか。川や井戸から水を汲んできて、薪でお湯を沸かして、湯加減を確かめてと、お風呂は贅沢品でしたので、庶民は銭湯へ通いました。今ではとても簡単に給湯器を設定すると、浴槽一杯に給湯してくれ、温かいシャワーも使うことが出来ます。

### 信号機

会社や学校へ行く途中に見かける信号機。これにもコンピュータが使われています。赤信号、青信号と、色を変えるのはもちろん、道路の状況に応じて、近くの信号機と連携、青信号の長さを調整することで渋滞緩和を図るなどしています。

### バス

通勤通学に電車やバスを利用する方も多いでしょう。例えば「さくら高校 行」と電光掲示板で行先表示したり、現金の他、ICカードを繋ぐことで乗車代金を払うことが出来ます。車両自体のガソリンや電気自動車の出力を制御する際や、バス停に「停留所を発車しました」と運行状況表示するなどの用途にも使われています。写真は、茨城県境町の「自動運転バス」<sup>\*1</sup>で、地域の足として活躍しています。

こうして見てただけでも身の回りのいろいろなところにコンピュータが使われていますことが分かります。他にはどこに使われているでしょうか。家の中で、お店で、学校や会社でなど、探してみましょう。

---

<sup>\*1</sup> 自治体初！ 境町で自動運転バスを定常運行しています (<https://www.town.ibaraki-sakai.lg.jp/page/page002440.html>)

# 第2章

## コンピュータの歴史

昔から人は暦や税、土木工事など、様々な計算を行ってきました。この章では、太古から現代に至るまでの計算機の歴史を振り返っていきます。

「コンピュータ」とは何でしょうか。語源を訪ねてみましょう。<sup>\*1</sup>

計算手（けいさんしゅ、英：computer, human computer）とは、電子計算機が実用化される以前の時代において、研究機関や企業などで数学的な計算を担当していた人間のことである。現在では「コンピュータ」と言えば電子計算機を指すが、当時は "computer" という語の成り立ちが表す通り「計算する人間」のことであった。

昔から人は暦や税、土木工事など、様々な計算を行ってきました。そうした計算をより容易にするために、暗算や筆算をしたり、算盤や計算尺を発明しました。そしてたくさんの歯車を組み合わせた機械式計算機が生まれ、電気に関する理解が深まり、「電子計算機」＝「コンピュータ」が現れました。

現代のコンピュータの歩みについては、[コンピュータ博物館<sup>\\*2</sup>](#)にも豊富な資料がございますので、ご覧ください。[ウィキペディア<sup>\\*3</sup>](#)からの引用を中心のご紹介していきます。

### 2.1 さまざまな計算 - 暦・税・工事 -

#### ♣ 暦について

「こよみ」の語源は、江戸時代の谷川士清の『和訓栞』では「日読み」（かよみ）が定説となっており、一日・二日…と正しく数えることを意味する。ほかに、本居宣長の「一日一日とつぎつぎと来歴（きふ）るを数へゆく由（よし）の名」、新井白石は「古語にコといひしには、詳細の義あり、ヨミとは数をかぞふる事をいひけり」などの定義がある。

古代エジプトにおいて、ナイル川の氾濫の時期に周期性があることに気づいたのが暦の始まりといわれている（シリウス暦）。人類が農耕を行うようになると、適切な農作業の時期を知るために暦は重要なものとなっていました。まず昼夜の周期（地球の自転）が日となり、月の満ち欠けの周期（月の公転）が月に、季節の周期（地球の公転）が年となった。このように暦法は天体運動の周期性に基づいていることから、その観測と周期性の研究が重要であり、これが天文学の基礎となった。

\*1 出典：[ウィキペディア](#)

\*2 <https://museum.ipsj.or.jp>

\*3 <https://ja.wikipedia.org/wiki/>

一方で、石器時代の35000年前に暦を創ったらしいとの意見もある。紀元前3000年頃のシュメール文明では、季節が冬と夏の2つで、1か月29日か30日の12か月の比較的簡単な暦を作り上げたといわれている。

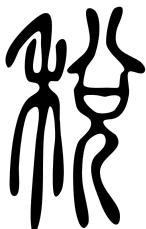
何を基準として一年を定めるか、閏（閏日・閏月）をどのようにして決めるかなどにより、太陽を基準とした太陽暦、月の運行を基準とした太陰暦、両者を折衷した太陰太陽暦など、さまざまな暦法が作られた。

太陰暦は月の運行を意識した暦で、「何日」と月のみかけの形が一致する。したがって月が出てさえいれば、その日が何日であるか暦がなくてもわかる。深夜に月の明かりを頼りとして活動をする場合には、月のみかけの形がわかると都合がよい。また、潮の満ち引きは月の位置と密接な関係があるため、漁業などにも役に立つ。

ただし、完全な太陰暦においては一年が約354日であり、太陽暦に比べ11日短くなるため、3年間で33日（約一ヶ月）ずれてしまい、実際の季節と大きく食い違ってしまう。このため、これを調整する方法として太陽暦を補助的に使用し、閏月の挿入により実際の季節と暦とのズレを修正する方法がとられるようになった。これが太陰太陽暦である。

それに対して、太陽暦は月の形とは関係なく暦が作られている。従って暦だけではその日の月の形は分からぬ。しかし、太陽の運行と暦の月日が一致している為、草花の開花、鳥の渡り等、同じ月日に同じ季節現象を期待でき、その早遅を観察することで、その年の寒暖の傾向を知ることができる。このことは農業や漁業にとって極めて大切である。<sup>\*4</sup>

### ♣ 税について



古代漢字研究の第一人者として知られる白川静さんは、字書三部作『字統』『字訓』『字通』を出版されました。左は、その中から「税」の篆文體です。神に稻穂を捧げる、そういう意味でしょうか。

最近は税の計算をする際にも、コンピュータが用いられ、手で計算することはほとんど無くなりましたが、いかほど納めるべきか、計算するのは大変であったことが想像できます。

国税庁の税の学習コーナー<sup>\*5</sup>より、その税の歴史をご紹介いたします。

### 飛鳥時代

飛鳥時代に行われた大化の革新では、公地公民など、新しい政治の方針が示されました。大宝律令では、租・庸・調という税や労役をかける税のしくみができました。租は男女の農民に課税され、税率は収穫の約3%でした。庸は都での労働、又は布を納める税、調は布や絹などの諸国の特産物を納める税だったようです。

### 奈良・平安・鎌倉・室町時代

奈良時代には、墾田永年私財法が制定され、土地の私有化へと展開していきました。また、平安時代には大きな寺社や貴族の莊園が各地にでき、農民は莊園領主に年貢や公事、夫役などを納めました。鎌倉時代は守護、地頭や莊園領主のもとで経済が発達します。室町時代には、税の中心は年貢でしたが、商業活動の発達により商工業者に対しても税が課せられ、街道に設けられた関所では、関銭などが課せられました。

---

<sup>\*4</sup> 出典: ウィキペディア

<sup>\*5</sup> <https://www.nta.go.jp/taxes/kids/hatten/page16.htm>

## 安土桃山・江戸時代

全国統一を行った豊臣秀吉は、太閤検地を行い、農地の面積や収穫高などを調べて年貢を納めるようにしました。当時の税率は、二公一民といい、収穫の三分の二を年貢として納める厳しいものでした。

\*6江戸時代には、田畠に課税される年貢が中心で米などを納めたそうです。また、商工業者に対する税も、運上金・冥加金として納められました。

## 明治時代

明治政府は歳入の安定を図るため、地租改正を実施しました。土地の地価の3%を地租として貨幣で納めさせたそうです。また所得税や法人税が導入されたのもこの頃です。ちなみに所得税は、所得金額300円以上の所得者に課税されるものでした。

## 大正・昭和時代

大正時代から昭和初期にかけては、戦費調達のため、増税が続きました。一方で、現在ある税のしくみができ始めたのもこの頃です。昭和15年に源泉徴収制度が採用されました。昭和21年には日本国憲法が公布され、教育、勤労にならぶ三大義務の一つとして「納税の義務」が定められました。また翌年には、納税者が自主的に自分の所得や税額を計算して申告・納税する申告納税制度が導入され、昭和25年にはシャウプ勧告に基づき税制改革が行われ、今日においても税制度の基盤となっています。

## 平成時代

平成元年に、商品の販売やサービスの提供に対して3%の税金を納める消費税が導入されました。消費税は平成9年には5%、平成26年から8%、令和元年からは10%に増税され、税収の三分の一を占めるまでになりました。<sup>\*7</sup>

## 仁徳天皇陵

古来より様々な土木工事が行われてきました。日本で一番の大土木工事と言えば、まず「仁徳天皇陵」をあげることができます。



正式名称は百舌鳥耳原中陵<sup>もずのみみはらのなかのみささぎ</sup>と云い、堺市にある世界最大のお墓で、全長は486メートル、三重の堀を含めると全長850メートルに及びます。

仁徳天皇の治績は「古事記」「日本書紀」など広く知られています。ここでは南北朝時代に編纂された「神皇正統記」<sup>a</sup>より原文を、現代語訳を伊勢雅臣さんの書かれた「日本人として知っておきたい 皇位継承問題の真実」よりご紹介いたします。

<sup>a</sup> <https://ja.wikisource.org/wiki/神皇正統記>

\*6 太閤検地こそ二公一民でしたが、江戸中期以降、四公六民から三公七民が一般的でした。昭和55年に25%であった国民負担率(租税・社会保障負担率)は、今日では約50%となり、苛政・酷税となっています。

\*7 大型間接税は導入しないとの公約の下、竹下内閣により導入された消費税は日本の衰退を齎しました。君たちはまだ長いトンネルの中 - 高校生たちが日本の未来を問う社会派青春ストーリー。(http://www.kimiton.com)

## 民のかまど 烟立つたみ

第十七代、仁徳天皇は応神第一の御子。御母仲姫の命、五百城入彦皇子女也。大鷦鷯の尊と申。応神の御時、菟道稚皇子と申は最末の御子にてましノヽしをうつくしみ給て、太子に立たてむとおぼしめしけり。兄の御子達うけがひ給はざりしを、此天皇ひとりうけがひ給しによりて、応神悦まして、菟道稚を太子とし、此尊を輔佐になん定め給ける。応神かくれましノヽしかば、御兄達太子を失はんとせられしを、此尊さとりて太子と心を一にして彼を誅せられき。爰太子天位を尊に譲給。尊堅いなみ給、三年になるまで互に譲て位を空す。太子は山城の宇治にます。尊は攝津の難波にましけり。国々の御つき物もあなたかなたにうけとらずして、民の愁となりしかば、太子みづから失給ぬ。尊おどろき歎給ことかぎりなし。されどのがれますべきみちならねば、癸酉の年即位。攝津國難波高津の宮にまします。日嗣をうけ給ひより国をしづめ民をあはれみ給たまふこと、ためしもまれなりし御事にや。民間の貧きことをおぼして、三年の御調を止どめられき。高殿にのぼりてみ給へば、にぎはゝしくみえけるによりて、

### 高屋にのぼりてみれば煙立つたみのかまどはにぎはひにけり

とぞよませ給ける。さて猶三年を許されければ、宮の中破雨露もたまらず。宮人の衣壊て其よそほひ全からず。御門は是をたのしみとなむおぼしける。かくて六年と云に、国々の民各まわり集あつまりて大宮造づくりし、色いろノヽの御調を備そなへけるとぞ。ありがたかりし御政まつりことなるべし。天下を治給こと八十七年。百十歳おましノヽき。

仁徳天皇が高台から遠くをご覧になられて「民のかまどから煙が立ちのぼっていない。思うに、貧しくて炊事もままならないのではないか。不作で民が窮乏しているのだろう」と仰せられ、「向こう三年、税を免じ、百姓の苦を安んじよ」と詔された。

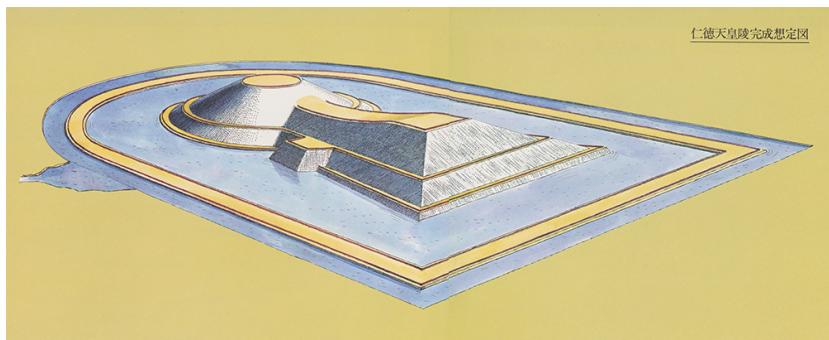
それからは、天皇は衣服や靴も破れるまで使い、宮垣が崩れ、茅葺屋根が破れても修理されず、そのため風雨が衣を濡らし、星の光が破れた隙間から見えるという有様だった。しかし、やがて天候も安定して、豊作となった。三年が経って、天皇が高台から遠くを望むと、炊煙が盛んに立っていた。そして皇后に「朕はすでに富んだ」と言われた。皇后は「宮垣が崩れ、屋根が破れて、衣服も濡れるのに、どうして富んだと言われるのですか」と尋ねた。

「天が君を立てるのは、百姓(民)の為だ。君は民を本とする。だから古の聖の君は、一人でも餓え凍える時は、省みて自分を責めた。民が貧しければ君も貧しい。民が富めば君が富んだことになる」

そのころ、諸国より「三年も課税を許されて、宮殿は朽ち破れているのに、民は富んでいます。もしこの時に、税を献じ、宮殿を修理させていただかないと、かえって天罰を蒙ります」との申し出が盛んに寄せられた。それでも、天皇はさらに三年間、税を献ずることをお聞き届けにならなかった。

六年の歳月がすぎ、天皇はようやく宮殿の修理をお許しになった。民は督促もされないので、老人を助け幼児を連れて、材木を運び、土を入れた籠を背負い、日夜をいとわず力を尽くして作業をした。これにより瞬く間に宮殿が完成した。それ故に聖帝と褒め称えられてきた。

## 現代技術と古代技術による仁徳天皇陵の建設



**仁徳天皇陵完成想定図**

仁徳天皇の「民の竈」<sup>かまど</sup>の逸話などで知られる遺徳を偲んで、日本最大の古墳が築かれました。大林組プロジェクトチームにより、当時の土木技術で復元を試みた例がございますので、引用してご紹介いたします。<sup>\*8</sup>

### 着工

現場は、まず敷地全体が整然と打ち込まれた多数の木杭と、引かれた縄や水糸が縦横に張り巡らされた姿となる。その上で、内堀・外堀の周間に沿って縁取るように溝を巡らす。大規模な土木工事には周到な排水計画が必要だ。地下水位は高く、降雨でも排水が不完全だと、現場は泥田のようになる。

### 主要工事の設計数量

項目	数量
敷地面積	478,000 m <sup>2</sup>
主要部の面積	外濠 44,580 m <sup>3</sup> 内濠 131,690 m <sup>3</sup> 中堤 65,800 m <sup>3</sup> 墳丘 103,410 m <sup>3</sup>
陵の規模	主軸長 475m 前方端幅 300m 前方丘高約 27m 後円丘径 245m 後円丘高約 30m 墳丘の土量 1405866 m <sup>3</sup>
濠の掘削量	内濠 599,000 m <sup>3</sup> 外濠 139,000 m <sup>3</sup>
客土量	742,000 m <sup>3</sup>
運搬土量	1,998,000 m <sup>3</sup>
葺石数量	5,365,000 個 (14,000t)
埴輪数量	約 15,000 個

古代の土木工事で、使用する道具は先端に鉄製の刃を付けた鋤に鍬、土砂運搬のための畚<sup>すき</sup>くわ<sup>くわ</sup>くらいで<sup>もつこ</sup>あった。そこで掘削の作業効率も現在の半分（一人一日 2 m<sup>3</sup>）として、仕事量を試算した。二重の濠で

\*8 現代技術と古代技術による仁徳天皇陵の建設 ([https://www.obayashi.co.jp/kikan\\_obayashi/detail/kikan\\_20\\_idea.html](https://www.obayashi.co.jp/kikan_obayashi/detail/kikan_20_idea.html))

陵全体に要する 140 万 m<sup>3</sup> もの土を得るには、濠全域にわたって深さ 10 m を掘る必要があるが、10 m も掘り下げる多量の地下水が溜り、人間は首まで泥水に浸かって作業することになることから、5 m 堀り下げるのが現実的だと考えた。これなら地下水を排除しつつ掘削作業もできそうである。濠部から得られる土量は 70 万 m<sup>3</sup> ほどで、必要量の約半分となることから、残りは近隣から客土することとした。

掘削した土は **畚**<sup>もつこ</sup> をつかって二人で一度に 60kg を運ぶものとする。現場で降ろされた土砂は、その上を足で踏み固めていく。この方法は、原始的ではあるが、きめ細かい効果があがるものだ。

### 葺石と埴輪

古墳全体に敷かれた葺石は、斜面保護のために有効である。斜面を風雨による崩壊から防ぎ、植物の生育を妨げる。石津川で採取した小石を、運搬専用の水路を開削して、いかだを曳いて運んだと考えた。小石を集める要員や水路開削に携わる人びともふくめ延べ 17 万人の作業員が必要だが、地上を運ぶ場合に比べ、半数の要員で済む。

古墳にはよく知られるように、埴輪が設置されるが、ここでは墳丘や中堤の上に垣根のように並べられる円筒埴輪とした。当初は紅殻<sup>ベンガラ</sup>などで表面が赤く彩色されていたという。

この埴輪の設置が完了すると、後円部の中央に、石棺を納めるための深さ 2m ほどに堅穴式の石室を構築する。これで仁徳陵の土木工事が完了したことになる。この膨大な土を積み上げた、きわめて人工的な巨大な形に、びっしり表面を小石で葺かれ、陽があたると白く輝き、赤く塗られた埴輪が横列に並び、その姿は相当の見ものであったことだろう。

### 施工期間、工事費

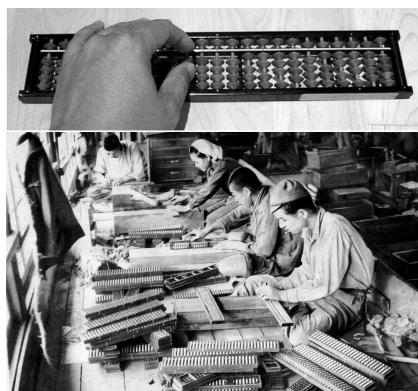
この工事のピーク時には 1 日に 2,000 人も作業していた。さらに現場で働く人びとのために、膨大な数の鋤や鍬などを作る人員、さらに管理や再生産のためには、集団によるシステムも必要であり、専門技術の指導者なども含めると、ここには総勢 3,000 人の人びとが常駐したと想定される。

そればかりではない。この大集団に食事なども支給されなければならない。3,000 人に毎日食事を用意する「後備え」には、陵を造る直接の労働力とは別個にほぼ同数の要員が必要である。すなわち、この場所に一時に 6,000 人もが集中したことになる。

古代工法での全工期は十七年、総工費は約八百億円に及ぶ。この工事を現代工法で実施した場合には工期二年半、工費約二十億円となる。

## 2.2

### そろばん 算盤から現代のコンピュータまでの歩み



#### 算盤(そろばん)

算盤とは、物体に状態で数を記憶させるため、串で刺した珠の位置などで数を表現し、計算の助けとする道具である。ひとつ串（ひと筋の串）が数の「ひと桁」に対応しており、珠を指で上下に移動させることで各数字の表現や変更を行う。加・減・乗・除などの計算が行える。

珠算は整数や小数を扱う場合には桁数が多くても敏速かつ正確に計算できる長所があり、四則計算などは簡易な加減法九九の適用によって計算できる。

起源についてはアステカ起源説、アラブ起源説、バビロニア起源説、中国起源説など諸説ある。メソ

ポタミアなどでは砂の絵に線を引き、そこに石を置いて計算を行っていた「砂算盤」の痕跡がある。同様のものはギリシャなどにも残るが、ギリシャ時代には砂だけでなくテーブルの上などにも置いていた。このテーブルを「アバクス (abacus)」と言う。ローマ時代に持ち運びができるように小さな板に溝を作りその溝に珠を置く溝算盤が発明され、中東を経て中国に伝わり現在の原型となった。現存する最古の算盤はギリシアのサラミス島で発見された「サラミスの算盤」で、紀元前300年頃のものである。

江戸時代には「読み書き算盤」といわれ寺子屋や私塾などで実用的な算術が教えられており、昭和中期までは、事務職や経理職に就くには珠算が必須条件だった。なお、この時代、手動式アナログ計算器としては計算尺があり、理系の人間はそちらも使いこなした。

日本国内では兵庫県小野市と島根県奥出雲町が二大産地であり、小野市の算盤は播州算盤、奥出雲町の算盤は雲州算盤として知られ、正月の「はじき初め」や、八月八日はパチパチと算盤の珠をはじく音に通じるため「算盤の日」となっている。



### アンティキティラ島の機械

アンティキティラ島近海の沈没船から発見された古代ギリシア時代の遺物で、製作時期は紀元前3世紀～紀元前1世紀中頃と推定されている。天体運行を計算するため作られた歯車式機械であると推定されている。

非常に精巧な構造から、古代の著名な科学者が作成にかかわった可能性が取り立たされる。例えば、天文学と数学の中心として知られたロドス島の天文学者ヒッパルコスやストア哲学者ポセイドニオスなどである。

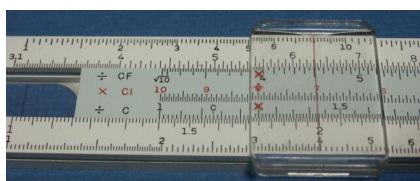
機械の概念は古代コリントスの植民地に起源をたどることができるとし、アルキメデスとの関係も示唆されている。多くの歯車が組み合わさっていることから、最古のアナログ計算機と呼ぶ人もいる。

アンティキティラ島の機械は最古の複雑な科学計算機と考えられており、縮小化と部品の複雑さは、18世紀の時計と比較しても遜色ない程である。30以上の歯車を持ち、クランクを回転させると機構が太陽、月やその他の天体の位置を計算する。

装置には主な表示盤が前面に一つ、背面に二つあり、前面の表示盤には2つの同心円状の目盛が刻まれている。外側のリングはソティス周期に基づく365日のエジプト式カレンダーまたはソティス年を表示する。内側の目盛りにはギリシャの黄道十二星座の記号が刻まれていて角度により区切られている。この暦ダイヤルを4年に1回1日分戻すことにより実際の1太陽年(約365.2422日)との誤差を補正することができる。注目すべきは、最古のうるう年を含んだ暦であるユリウス暦の成立は、この機械が作られた100年後の紀元前46年だということである。

前面の表示盤は少なくとも3つの針を持ち、1つは日付、残りは太陽と月の位置を示していた。月の表示針を動かして月軌道の真近点角が求められる。太陽についても同様の機能があると想像されるが、該当する歯車は発見されておらず定かではない。前面の表示盤には第二の機能として球体模型を使った月相表示機能がある。

機械に刻まれた文字には火星と水星に関する記述があり、製作者には確かにそれらの惑星の位置を示す歯車を盛り込める十分な技術があったように思われる。この機械は当時のギリシャ人が知り得た5惑星全ての位置を表せたとも推測されている。



### 計算尺

対数の原理を利用したアナログ式計算用具で、乗除算および三角関数、対数、平方根、立方根などの計算に用いる。計算尺は様々な関数の値の対数を計算し、その比率を目盛として固定尺や滑尺に配置したものである。

対数は1614年にスコットランドのジョン・ネイピアが発表した。その6年後にイギリスのエド蒙ト・ガンターが対数尺を考案した。これは数の対数や三角関数  $\sin$ ,  $\tan$  の対数などを幾何的に配置したものであり、コンパスを利用して2つの目盛の長さの加減をしていた。現在の形式の計算尺、つまり複数の尺をずらして計算をするという形の計算尺を発明したのはオートレッドであり1632年のことである。様々な計算尺が考案され、電卓（電子式卓上計算機）が普及する1980年代まで広く使われた。

高性能の関数電卓が普及するまで計算尺は数理系の研究者にとって必須のアイテムであり、マンハッタン計画やアポロ計画の記録映像などにおいても科学者が現場で用いていた。映画の『アポロ13』でも軌道計算を検算する場面に、『風立ちぬ』でも航空機の設計の場面で登場している。

日本製の計算尺は竹製で狂いが少ないと評価され、戦前は世界シェアの80%を占めた頃もあった。



### シッカートの計算機

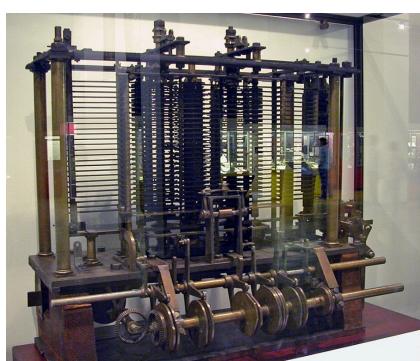
テュービンゲン大学のヘブライ語教授であったヴィルヘルム・シッカートが1623年に発明した機械式計算機である。パスカル、ライプニッツの計算機よりも機能は少ないが、20年先行している。

この計算機は、6桁の加減算およびオーバーフローの検出、複数のネイピアの骨を使った乗算が可能であった。シッカートのヨハネス・ケプラーへの手紙には、天体計算への利用方法が記されている。

「ネイピアの骨」は、対数を利用して「掛け算を足し算だけの計算にする」道具で、ネイピアが科学で扱われる計算の簡略化に尽力した成果として得られた道具である。

ネイピアの死後、ネイピアの骨は様々なに改良されるが、特に1623年のヴィルヘルム・シッカートによる改良が重要である。シッカートは、ネイピアの骨を歯車などを用いて自動化した。シッカートの計算機は、足し算機能も組み込まれており、ダイヤルを回すことにより6桁と1桁の掛け算ができる。

このシッカートの計算機は、世界初の歯車式計算機としても知られ、その後のコンピュータの歴史へ繋がる一步であった。



### バベッジの解析機関 試作品 (1837年)

イギリス人數学者チャールズ・バベッジが設計した、蒸気機関で動くはずだった機械式汎用コンピュータであり、対数や三角関数の数表を作ることに特化した計算機である。資金などの問題があり、この機械は実際には製作されなかったが、1871年の死去直前まで設計が続けられた。論理的に解析機関に匹敵する機能を持つ汎用コンピュータは、時代を下ること百年、1940年代に現実のものとなつた。コンピュータの歴史上、重要なステップを刻んだ一台である。

当時、フランス政府はいくつかの数表を新しい手法で製作していた。数人の数学者が数表の計算方法を決定し、6人ほどでそれを単純な工程に分解して、個々の工程は加算か減算をすればよいだけにする。そして加減算だけを教え込まれた80人の計算手に計算させるのである。これが計算における大量生産的手法の最初の適用例であった。1812年、バベッジは熟練していない計算手を完全に機械に置き換えれば、より素早く正確に数表を作れるというアイデアを思いついた。

解析機関は、制御情報にしたがってオルゴールのようにピンを配置してあって回転、停止、逆回転するドラム群が中心となっている。そして、多くの歯車や力の伝達機構、位置や回転角などで情報を記憶・表示する仕組みなどから構成される、複雑で大きな機械である。蒸気機関を動力として、完成すれば長さ30m、幅10mという、いまの電車1.5両分もの巨大さとなっていたはずである。いわば蒸気機関車ならぬ蒸気機関計算機である。

プログラムとデータの入力は、当時既にジャカード織機のような機械式織機で使われていたパンチカードで供給される予定だった。出力としては印刷原版作成機、曲線プロッターおよびベルを準備していた。演算方式は十進数の固定小数点演算であり、1,000個の50桁の数値を格納できる。演算装置は四則演算が可能で、さらに比較と、オプションで平方根の演算が可能であった。

現代コンピュータのCPUのように命令を持ち、演算装置内部の手続きはパレルと呼ぶ回転するドラムにペグ(釘)を刺することで格納され、それにより複雑な命令を実現した。

バベッジはさらに汎用的な解析機関を構想、1871年に亡くなる直前までその設計を改良し続け、パンチカードでプログラミング可能とした。プログラムをカードで用意することで、最初にプログラムを組めば、それを機械に入れるだけで実行可能である。解析機関はジャカード織機のパンチカードのループで計算機構を制御し、前の計算結果に基づいて次の計算を行うことができる。

プログラミングは機械語であるが、現在のアセンブリ言語の原型のような記述法が考案されており、逐次制御、条件分岐、繰り返しといった現代のコンピュータのような特徴すら備えていた。チューリング完全を達成していたのではないかと考える者もいる。

エイダ・ラブレスはバベッジのアイデアを完全に理解していた数少ない人物の1人で、単なる計算機に留まらない解析機関の可能性を見出していた。その能力を示すために、ベルヌイ数の数列を計算するプログラムなどを作成したことから、世界初のプログラマと呼ばれている。1979年には、彼女に因んだプログラミング言語Adaが誕生した。



### タイガー計算機(大正13年)

歯車などの機械要素により計算を行う機械式計算器としての代表的存在であり、大正時代に発明されたタイガー計算機は昭和45年まで発売された。

明治45年9月大阪府西成郡豊崎町南浜で営業していた大本鉄鋼所は、好景気の大波に乗って注文に忙殺されていた。見積の為には設計図に沿って原材料費、労務費、その他諸経費等の原価を算定する為「簡単に計算する機械」

を作ろうと、大阪府西成郡鷺洲村海老江に工場を新設、移転すると共に試作品の研究に着手。4年5ヶ月と多額の経費を投じ、大正12年漸く完成した第一号計算器は、発明者大本寅治郎の「寅」をとつて「虎印計算器」と命名された。

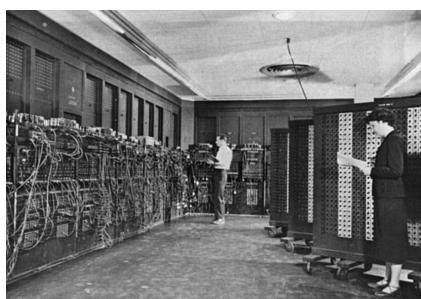
関東大震災後、東京復興の機運がみなぎり大建造物、大工場の建設が始まられた。鉄筋・鉄骨造の建築物や大工事には強度その他の計算を必要とし、しかも算盤や筆算では間に合わず大量の計算器が必要とされることから、広く普及し、電卓の登場と共に、昭和45年、舞台を降りることになった。<sup>\*9</sup>

<sup>\*9</sup> タイガー手廻計算器資料館 <https://www.tiger-inc.co.jp/temawashi/temawashi.html> より引用改変

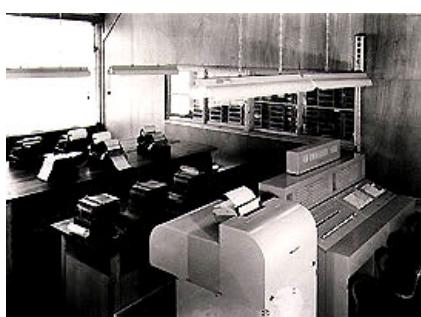
**東京帝国大学 九元連立方程式求解機(昭和 19 年)**

昭和 13 年頃、東京帝国大学航空研究所の佐々木達治郎を中心として、機械式計算機の開発が進んでいた。MIT の Wilbur の作った連立方程式の求解機の情報に基づき、志賀亮と三井田純一が製作を担当した。昭和 19 年頃に完成、9 個の変数を持つ連立方程式を解く、我が国初のアナログ計算機となった。

九元連立一次方程式は、一般に 9 個の未知数と 1 個の定数を含んだ 9 個の一次方程式からなっている。本機は鉄のフレームに角度を変えられる真鍮のバーが取り付けられており、その角度が各未知数に対応し、バー上のブーリーにかけられた鉄のテープの長さが各方程式を表している。バーを動かし、テープの長さを読み取ることにより方程式の解を求めることができた。<sup>\*10</sup>

**米国陸軍 ENIAC(昭和 21 年)**

米国ペンシルベニア大学で開発された黎明期の電子計算機。パッチパネルによるプログラミングは煩雑ではあるが、米陸軍による砲撃射表の計算、マンハッタン計画、円周率計算など、汎用的な計算問題を求解できた。17468 本の真空管で作られていて、幅 24m、高さ 2.5m、奥行き 0.9m、総重量 30 トンと大掛かりな装置であった。消費電力 150kW、開発費 49 万ドルであった。

**FACOM100(昭和 29 年)**

昭和 29 年に完成したわが国初の実用リレー式自動計算機。昭和 26 年に東京大学の山下英男教授の指導により、リレーを主体とした統計分類集計機が完成し、昭和 28 年にリレー式の株式取引高精算用計算機を試作した。その後、その開発経験を活かし、自由にプログラミングができる本格的なリレー式自動計算機 FACOM100 が完成した。完成後間もなく、日本で初めてノーベル賞を受賞した湯川秀樹博士から極めて複雑な多重積分の計算を依頼された。

FACOM100 は人間の手でやったら 2 年はかかる計算を見事に 3 日間で答えをだし、湯川秀樹博士は、「これで研究のスピードが飛躍的に進む」と喜んだそうである。計算機の名称は、Fuji Automatic Computer の頭文字をとって「FACOM」と命名されている。<sup>\*11</sup>

**Z80 (昭和 51 年)**

米国ザイログによって製造された 8 ビット・マイクロプロセッサ。昭和 60 年頃までは、パソコンコンピュータの CPU としてなど、幅広い用途に使用された。現在でも組み込み用途など、目に見えないところで多用されている。

\*10 コンピュータ博物館, <http://museum.ipjs.or.jp/heritage/kyugen.html> より引用改変

\*11 富士通ミュージアム, <https://www.fujitsu.com/jp/about/plus/museum/> より引用改変



### 日本電気 PC-9801(昭和 57 年)

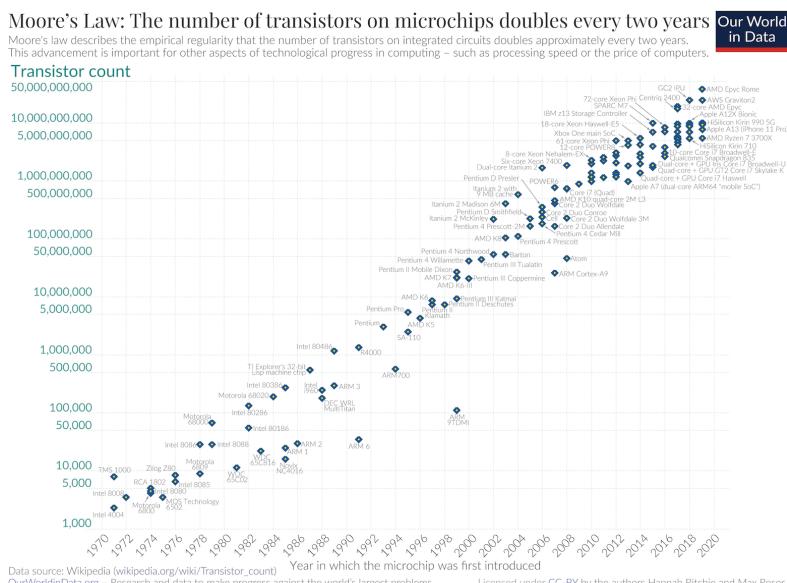
日本電気が発売したパーソナルコンピュータ。キーボードと本体が一体化したデザインで、ハード・ソフトとともに高い機能と完成度を有しており、数多くのソフトウェアや周辺機器が販売された日本のパソコンの代表機種。

PC-9801 は、徐々に広がり始めたビジネス市場に対応するために PC-8001、PC-8801 に代表される NEC の 8 ビットパソコンのソフト・ハード両方の資産を継承しながら、一層の高速処理を実現するために、CPU に NEC 製 16 ビット

マイクロプロセッサ mPD8086 (i8086 互換) 5MHz、画像処理に NEC 開発の LSI であるグラフィック・ディスプレイ・コントローラ mPD7220 を搭載し、日本語処理とカラーグラフィクス表示機能を備え、主記憶容量は最大 640 キロバイト、外部記憶装置のフロッピーディスクは外付けであった。

PC-9801 は、独立系ソフト会社に開発マシンや技術資料を提供することで販売開始初期に各種アプリケーションが揃ったことなどにより、ビジネス市場を中心に広く受け入れられ、ソフトウェア会社、周辺機器会社、出版社、ソフト流通業、販売店、システムハウスなどパソコン産業というべき産業構造の形成に大きく寄与し、98 文化の基礎を築いた。<sup>\*12</sup>

### 【コラム】ムーアの法則



▲ 図 2.1: 集積回路のトランジスタ数の増大 \*13

\*12 コンピュータ博物館, <http://museum.ipsj.or.jp/heritage/pc9801.html> より引用改変

ムーアの法則<sup>\*14</sup>とは、大規模集積回路 (LSI) の製造・生産における長期傾向について論じた指標であり、経験則による将来予測。米インテル社の創業者であるゴードン・ムーアが、集積回路あたりの部品数が毎年 2 倍になると予測した。

対数を使うと簡単に計算できますが、毎日 0.2% ずつ (=月々 6%) の複利で成長すると一年で二倍になります ( $1.06 \times 1.06 \times 1.06 \times \dots$  と、12 回繰り返して下さい)。

『失われた三十年』となった平成はともかく、戦後の『高度成長期』には 18 年に渡り年率約 9% の成長を続けました。<sup>\*15</sup>

コンピュータの成長がどれくらいなのかより実感できるよう、速さに置き換えて感じてみましょう。赤ちゃんのハイハイは、時速 1km と言われています。マラソン選手は、時速 20km で走ります。ジェット機は、時速 900km で空を飛びます。ロケットは、時速 40,000km で、惑星探査に向かいます。数万倍に速くなりました。

黎明期のコンピュータとして有名な ENIAC(エニアック) は、17468 本の真空管を使い、10 枠の整数の足し算を毎秒 5000 回実行することができました。設置面積は  $167 \text{ m}^2$ (約 100 頃)、消費電力は 150kW(電気ポット約 150 台分) でした。最新の iPhone は、160 億個のトランジスタを搭載し、毎秒 17 兆回の計算ができます。

計算を速く行う為に積み重ねられた人の営み。それが今日の豊かな社会を形作りました。

\*11 トランジスタ数のグラフであり、計算速度のグラフではありませんが、傾向を掴むためのものとして掲載しています。

\*12 ムーアの法則: Wikipedia より引用改変

\*13 昭和 30 年の GDP 約 48 兆円、昭和 48 年の GDP 約 229 兆円と 4.8 倍、年率換算 9.1% 成長で、10 年で約 2.4 倍になります。出典: 経済産業省

---

## 第3章

# コンピュータを創った人々

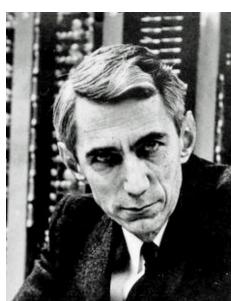
---

数多くの科学者や技術者の貢献により、今日の情報社会が築かれました。ウィキペディアより引用・要約する形で、24人の方々の業績をご紹介いたします。



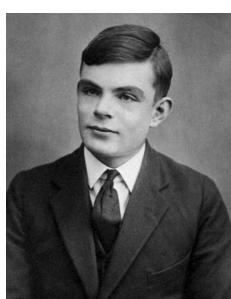
### ジョン・冯・ノイマン

ハンガリー出身の数学者。ほとんどのコンピュータの動作原理であるプログラム内蔵方式を考案した。原子爆弾（マンハッタン計画）や黎明期の電子計算機 ENIAC(エニアック) の開発でも有名である。ゲーム理論（複数人の意思決定を数学的に研究する学問）の成立に貢献した。企業経営や軍事戦略理論や、将棋やチェスなどの零和ゲームの戦略など、社会に大きな影響を与えた。



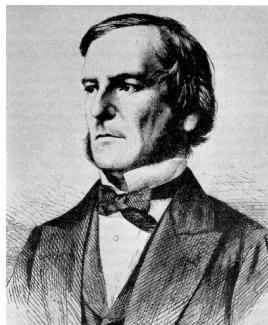
### クロード・シャノン

米国の電気工学者、数学者。情報、通信、暗号、データ圧縮、符号化など今日の情報社会に欠かせない分野を研究し、「情報理論の父」と呼ばれている。マサチューセッツ工科大学で論文を書き、電気回路・電子回路が論理演算に対応することを示した。これにより、デジタル回路・論理回路の概念が確立され、コンピュータの実現に向け、とても大きな一歩となった。情報量の単位「ビット」もシャノンの貢献である。



### アラン・チューリング

イギリスの数学者、論理学者、暗号解読者、計算機科学者。「チャーチ=チューリングのテーゼ（提唱）」と計算可能性理論への貢献で広く知られている。アルゴリズム（算法）を実行する機械を形式的に記述した「チューリングマシン」にその名を残す。また「停止性問題の決定不能性」（無限の計算能力を持つコンピュータでも、解けない問題がある）を示した。また、エニグマの暗号解読への貢献や、チューリングテスト、コンピュータチェス、さらに実際面でもコンピュータの誕生に重要な役割を果たし、計算機科学、人工知能の父とも言われている。



### ジョージ・ブール

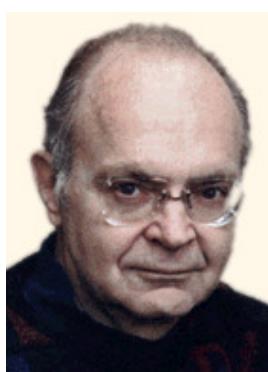
イギリスの数学者・哲学者。今日のコンピュータ科学の分野の基礎的な理論であるブール代数を確立した。組み合わせ回路（論理回路）はブール代数で表現できる。0と1を電圧の高低に対応させると、デジタル回路の入力と出力をブール論理の式で表現することができる。これにより、ANDゲート、ORゲート、NOTゲートのような基本論理回路や、NANDゲート、NORゲート、XORゲートなどを組み合わせてデジタル回路を構成することができる。



#663399

### ニクラウス・ヴィルト

スイスの計算機科学者。プログラミング言語 Pascal、Modula-2などの開発や、ソフトウェア工学分野の開拓的研究で知られる。ヴィルトは、プログラミング言語 ALGOL W、Pascal、Modula、Modula-2 やオペレーティングシステム Oberon の開発などの功績により、ヴィルトは1984年にチューリング賞を受賞した。プログラミングの教育法について書いた記事 Program Development by Stepwise Refinement は、ソフトウェア工学の分野における古典である。1975年の著作『アルゴリズム+データ構造=プログラム』は広く知られ、今なお価値を失っていない。同書では、コンパイラ設計の説明のために、単純なプログラミング言語 PL/0 を設計。様々な大学のコンパイラ設計の授業で利用された。



### ドナルド・クヌース

アメリカ合衆国の数学者、計算機科学者。スタンフォード大学名誉教授。クヌースによるアルゴリズムに関する著作 The Art of Computer Programming は有名である。アルゴリズム解析と呼ばれる分野を開拓し、計算理論の発展に多大な貢献をしている。その過程で漸近記法（ランダウ記法、O記法）で計算量を表すことを一般化させた。

計算機科学への貢献に加え、コンピュータによる組版システム TeX とフォント設計システム METAFONT の開発者でもあり、Computer Modern という書体ファミリも開発した。



### ブライアン・カーニハン

ブライアン・カーニハンは、ベル研究所に在籍していたカナダ出身の計算機科学者である。C言語やUNIXの開発者であるデニス・リッチャー、ケン・トンプソンと共に、C言語およびUNIXに対する多くの研究開発結果による貢献で知られている。

デニス・リッチャーと共に著の『プログラミング言語 C』(通称: K&R) は、事実上の規格書として扱われ、現在でも古典的な教科書の一つである。

多くのプログラミング言語入門書で、最初のプログラムとして書かれる Hello world は、彼がベル研究所で書いた B 言語のチュートリアルで初めて使われた。



### デニス・リッチャー

アメリカ合衆国の計算機科学者。同僚のケン・トンプソンと共に、ベル研究所で独自のオペレーティングシステム UNIX を作り始める。この UNIX 上で動作するアプリケーション作成の為に、トンプソンによって B 言語が開発され、リッチャーがこれにデータ型と新しい文法等を追加し C 言語が出来、アセンブリ言語で書かれていた UNIX を C 言語で書き換えることに成功した。C 言語の開発は、リッチャーの UNIX への最大の貢献である。

UNIX 開発の功績により、ケン・トンプソンと共にチューリング賞を受賞している。今日、C 言語は組込システムからスーパーコンピュータまであらゆるプラットフォームで用いられ、彼の業績は偉大である。



### ケン・トンプソン

ケン・トンプソンは、アメリカ合衆国の計算機科学者。長年ベル研究所に勤め、オリジナルの Unix を開発した。また C 言語の前身である B 言語を開発した。2006 年から Google で勤務しており、Go を共同開発した。他の主な業績として、正規表現、テキストエディタ QED と ed、UTF-8 コードの定義に加え、チェスの終盤定跡データベースやチェスマシン Belle の開発などコンピュータチェスへの貢献がある。1983 年に彼の長年の同僚であるデニス・リッチャーと共にチューリング賞を受賞した。「信用を信頼することについての考察」は、トンプソンハックとして知られる、セキュリティに関する重要な研究成果である。



### ジェイコブ・ジヴ

ジェイコブ・ジヴは、イスラエルの電気工学者である。エイブラハム・レンペルと共に可逆データ圧縮アルゴリズムである LZ77・LZ78 の開発者として知られる。

ジヴの研究分野はデータ圧縮・情報理論・統計的通信理論であり、イスラエル最高の栄誉である「イスラエル賞」受賞の他、「情報理論への貢献とデータ圧縮の理論と実践」について IEEE リチャード・ハミングメダルを、IEEE 情報理論ソサイエティよりクロード・E・シャノン賞などを受賞した。



### アーサー・サミュエル

アーサー・リー・サミュエルはアメリカの計算機科学者で、コンピュータゲームと人工知能の分野で主に知られている。

コンピュータによる一般問題への適切な戦術を開発するにはゲームを学習させるのが非常に有益だと考え、単純だが奥が深いチェックャーを選択、世界初の学習型コンピュータ・チェックャーの開発を行った。現在状態から到達可能な盤面の探索木を構成し、アルファ・ベータ法と呼ばれる技法での枝刈りや任意の盤面を評価する関数を開発、研究後期には腕の立つアマチュアと互角に戦えるレベルとなった。

人工知能 (AI) の基本的概念をいち早く世界に示し、機械学習も彼の造語である。



### アラン・ケイ

アラン・カーティス・ケイは、アメリカ合衆国の計算機科学者。主にオブジェクト指向プログラミングとユーザインターフェース設計に関する初期の功績で知られている。

マイクロコンピュータ以前の時代に、個人の活動を支援する「パーソナルコンピュータ」という概念を提唱した。高価で大きく複数人で「共有」するのが当たり前だったコンピュータに「個人向け」という利用状況を想定し、それに相応しいコンピュータ環境がどうあるべきかを考えた人物である。自らがそう名付けた「ダイナブック構想」の提唱者であり、「コンピュータ・リテラシー」という言葉の発明者でもある。



### デイヴィッド・ドイッチュ

デイヴィッド・ドイッチュは、イギリスの物理学者である。量子チュー  
リングマシンの記述や、量子コンピュータ上で動作するように設計されたアルゴリズムを規定し、量子計算の分野を開拓した。

「量子計算理論の基礎を築き、その後、最初の量子アルゴリズムの発見、量子論理ゲートと量子計算ネットワークの理論、最初の量子エラー訂正スキーム、いくつかの基本的な量子普遍性の結果など、この分野で最も重要な進歩の多くを作り、またはそれに参加してきた。」貢献から、王立協会 (FRS) のフェローに選ばれている。



### リーナス・トーバルズ

リーナス・ベネディクト・トーバルズは、フィンランド出身の米国プログラマ。リナックス Linux カーネルを開発し、公開した。

アンドリュー・タネンbaumが開発したカーネルとオペレーティングシステム (OS) である MINIX に刺激を受け、自宅のパーソナルコンピュータ上で動作可能な UNIX OS の必要性を感じ、自分の趣味の時間と自宅の設備で Linux カーネルの初期の開発を行った。

また、Linux カーネルのソースコードなどの変更履歴を記録・追跡するための分散型バージョン管理システム ギット Git も開発、ほかの多くのプロジェクトでも事実上の標準として広く用いられている。



### ティム・バーナーズ=リー

ティム・バーナーズ=リーは、イギリスの計算機科学者。ロバート・カリューとともに World Wide Web (WWW) を考案し、ハイパーテキストシステムを実装・開発した人物である。また URL、HTTP、HTML の最初の設計は彼によるものである。

スイス・ジュネーヴの欧州原子核研究機構 (CERN) にて、世界最初のウェブサイト <http://info.cern.ch/> を公開する。

マサチューセッツ工科大学に着任した後、World Wide Web Consortium (W3C) を設立。WWW の仕様や指針、標準技術を策定・開発に関わっている。



### ホーコン・ウィウム・リー

ホーコン・ウィウム・リーは、ノルウェーのオペラ・ソフトウェアの最高技術責任者である。

WWW 生誕の地である CERN にて、Cascading Style Sheets (CSS) の概念を初めて提唱した。W3C にも参画、ウェブブラウザのレンダリングエンジンが W3C の勧告を適切に準拠しているかを検証する Acid2 テストを提唱した。

その後も、ウェブフォントやビデオ要素の提言や、Opera でのモバイルブラウザの開発などに貢献した。



### エリック・メイヤー

エリック・メイヤーは、アメリカのウェブデザインコンサルタントであり、作家である。ウェブ標準を支持する活動で最もよく知られており、特に HTML の表示方法を管理する技術である CSS (Cascading Style Sheets) について多くの本や記事を書き、その使用を促進するために多くのプレゼンテーションを行っている。

次女のレベッカが 6 歳で亡くなった際には、彼女の思い出として、#663399 は「rebeccapurple」と名付けられ CSS Colors リストに加えられた。



### ブレンダン・アイク

ブレンダン・アイクはアメリカ合衆国のプログラマであり、プログラミング言語 JavaScript の生みの親である。

主に Netscape と Mozilla での業績で知られ、ネットスケープコミュニケーションズで、ウェブブラウザ Netscape Navigator 向けの JavaScript 開発に携わった。

Mozilla Corporation で最高技術責任者に就任、十年以上に渡り活動した後、Brave Software を設立、新ブラウザ Brave の開発を行っている。



### デイヴィッド・ハイネマイヤー・ハンソン

デイヴィッド・ハイネマイヤー・ハンソンは、デンマーク出身のプログラマ。ウェブアプリケーションフレームワーク「Ruby on Rails」の作者であり、Basecamp の創設者、最高技術責任者でもある。通称として「DHH」と表記されることが多い。

カーレーサーとしてもル・マン 24 時間レースでは、アマクラスで優勝を果たし、国債自動車連盟世界耐久選手権 (WEC) でもドライバーズチャンピオンに輝くなど実績は一級である。



### 和田 英一

和田 英一は、日本のコンピュータ科学者。東京大学理学部物理学科・高橋秀俊教授の研究室で開発されていたパラメトロンコンピュータにおいて、プログラム本体の一部と変換テーブルを兼用するなどしてサイズ圧縮したことは、「日本初のハッカー作品」と称される。(ASCII.jp<sup>a</sup>に特集。写真出典も同じ。)

マサチューセッツ工科大学の准教授を務め、コンピュータ科学の専門過程への入門的教科書「計算機プログラムの構造と解釈」を翻訳する。

日本におけるインターネットの発展をその創世記から見守り、電子メールで日本語を取り扱う為の JUNET 漢字コードの解説文書を書き、普及させた。また、大学院生の田中哲郎らにより開発されたいわゆる「和田研フォント」や「鍵盤配列にも大いなる関心を」がきっかけとなり開発された「Happy Hacking Keyboard」シリーズでも知られる。

<sup>a</sup> <https://onl.bz/7RcpYa7>



### 坂村 健

坂村 健は、日本のコンピュータ科学者。自ら提唱した TRON プロジェクトにてリーダーとして多種多様な仕様を策定した。

マイクロコントローラによる組込みシステムにより身の回りのあらゆるもののがインテリジェントになり、またネットワーク化されるヴィジョンから TRON プロジェクト (The Realtime Operating system Nucleus) を開始。住宅、乗用車から家電に至るまで、組み込み機器用の標準 OS として広く利用されている。

平成十五年には、紫綬褒章を受章。平成二十五年には国際電気通信連合 150 周年賞を受賞している。

東京大学名誉教授、東洋大学情報連携学部学部長、日本電信電話会社社外取締役 (顔写真は同社ウェブサイトより) でもある。



### 奥村 晴彦

奥村 晴彦は、日本の計算機科学者。学生時代から「月刊マイコン」に投稿するなど、若いころから既に知られた存在であった。日本における TeX の第一人者であり、その普及に尽力している。また、圧縮アルゴリズム「LZARI 法 (後の LZA)」を開発したことでも知られている。

プラズマ・核融合学会の電子化グループに参画、学会のソフトウェア構造設計、事務処理作業や英文電子ジャーナル編集作業のオンライン化に取り組んだ業績から、同学会貢献賞を受賞した。

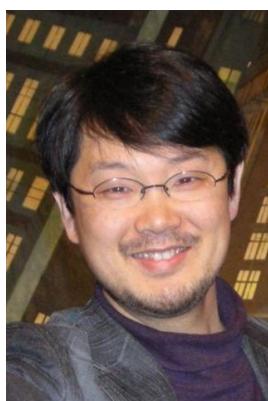


### 村井 純

村井 純は、日本の計算機科学者で、専門は情報工学（コンピュータネットワーク）。

日本におけるインターネット黎明期からインターネットの技術基盤作り、運用、啓蒙活動等に関わり続けている。「日本のインターネットの父」とされ、広域ネットワーク上で日本語を使えるようにすることにも尽力し、UNIX や C 言語を国際化する動きと連携をとりながら、英語を中心だった初期のインターネットを多言語対応へと導いた。

東南アジアの研究教育ネットワークの開発、発展にも尽力し、東南アジア各国のインターネットを牽引する研究運用人材を多数輩出している。



### まつもと ゆきひろ

まつもと ゆきひろは、日本のソフトウェア技術者。株式会社ネットワーク応用通信研究所フェロー、Ruby アソシエーション理事長、松江市名誉市民。通称は Matz。

プログラミング言語「Ruby」の開発者。効率的に記述できるプログラム言語の実現を目指し、平成 5 年から開発を始め、平成 7 年にオブジェクト指向スクリプト言語 Ruby を公開した。Ruby を用いて「Hello, world!」という文字列を出力するために半年を要して苦労したが、「Ruby 言語の開発で飽きたり、辛く感じたりすることはなかった」と語っている。

平成 9 年から松江市に在住し、同市のネットワーク応用通信研究所 (NaCl) にフェローとして勤務している。

Ruby の普及を目的として設立された一般財団法人「Ruby アソシエーション」の理事長も務める。平成 24 年、内閣府から「世界で活躍し『日本』を発信する日本人」の一人に選ばれた。<sup>\*1</sup>

## 【コラム】名前重要

著者: Matz

ネイティブ・アメリカンの信仰に「すべての人物・事物には真の名前があり、その名前を知るものはそれを支配することができる」というものがあるのだそうです。ですから、彼らは自分の真の名前を秘密にして、家族など本当に信頼できる人にしか打ち明けないのだそうです。そして、対外的にはあだ名を用意してそちらを使うということです。そういえばアニメ化もされた U・K・ルーグウィンの「ゲド戦記」でも同じ設定が用いられていましたね。「ゲド」というのは主人公の真の名前なので物語中にほとんど登場せず、物語の中では彼は一貫して「ハイタカ」と呼ばれていました。

さて、プログラミングの世界において、この信仰はある程度真実ではないかと感じることがたびたびあります。つまり、事物の名前には、理屈では説明しきれない不思議なパワーがあるような気がするのです。

\*1 小さな目標を立て続けたからこそ Ruby はできた まつもと ゆきひろ氏が語る、「言語を作りたい」気持ちからの道程 (<https://logmi.jp/tech/articles/325269>)

たとえば、私が開発している Ruby も、名前のパワーを体現しているように思えます。1993年にRubyの開発を始めた時、Perlにあやかって宝石の名前を選んでRubyと命名しました。あまり深刻に考えず、宝石の名前のなかから、短く、覚えやすく、美しい名前としてRubyを選んだだけでしたが、後にRubyが、6月の誕生石である真珠(パール)に続く、7月の誕生石であることに気がついた時、まさに適切な名前であると感じました。また、活字もそれぞれの大きさに応じて宝石の名前が付けられているのですが、パールは5ポイント、ルビーは5.5ポイントで並んでいます。このルビーがふりがなの「ルビ」の語源になったのはまた別の話。

今、振り返って思うのは、もし私がRubyという名前を選ばなかつたらきっと、現在のRubyの普及を見ることはなかつただろうということです。このRubyという名前にパワーがあったからこそ、Rubyの魅力が増加したのではないかと感じます。ただ単にRubyがプログラミング言語として優れているだけでなく、この名前の持つパワーによって、愛される存在となっているのではないかと感じます。この名前があればこそ、これまでの長い間Rubyを開発し続けるモチベーションが維持できたり、また多くのユーザがRubyという言語に関心をもってくださったのではないかと感じています。

そんなこと也有って、私の設計上の座右の銘は「名前重要」です。あらゆる機能をデザインする時に、私はその名前にもっともこだわります。プログラマとしてのキャリアの中で、適切な名前をつけることができた機能は成功し、そうでない機能については後悔することが多かったように思うからです。

実際、Rubyに対する機能追加の要求に対しても、しばしば「要求は分かった。あれば便利なのも理解できる。でも、名前が気に入らない。良い名前が決まつたら採用する」として拒否したものも数限りなくあります。しかし、名前が気に入らなかったもので、取り入れなかつことを後で後悔したことほんんどありません。

これはつまりこういうことなのではないかと思います。適切な名前をつけられると言うことは、その機能が正しく理解されて、設計されているということで、逆にふさわしい名前がつけられないということは、その機能が果たすべき役割を設計者自身も十分理解できていないことなのでないでしょうか。個人的には適切な名前をつけることができた機能については、その設計の8割が完成したと考えても言い過ぎでないことが多いように思います。

ソフトウェアの設計のアプローチとして、「まず名前から入る」というのは、あまり語られていない秘訣としてもっと広く知られてもよいように思います。<sup>\*2</sup>

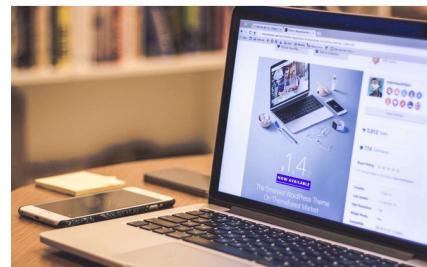
<sup>\*2</sup> <https://プログラマが知るべき97のこと.com> より引用

# 第4章

## コンピュータの仕組み

机の上に置かれたコンピュータ。その中身を覗いてみると、演算・制御装置、記憶装置、入力装置、出力装置など、たくさんの中身からできています。IT用語辞典<sup>a</sup>から抜粋しつつ、「コンピュータ」の定義を紹介するとともに、その構成要素の詳細を見ていきましょう。

<sup>a</sup> IT用語辞典 <https://e-words.jp/>



### 4.1 コンピュータ / 電子計算機 / 電算機

#### ♣ コンピュータとは

コンピュータとは、与えられた手順に従って複雑な計算を自動的に行う機械。特に、電子回路などを用いてデジタルデータの入出力、演算、変換などを連続的に行うことができ、詳細な処理手順を人間などが記述して与えることで、様々な用途に用いることができる電気機械のこと。

歴史的には手回しで歯車などを駆動する機械式の自動計算機なども存在したが、現代でコンピュータと呼ばれる機械は一般に、マイクロプロセッサ（CPU）や半導体メモリなどの半導体集積回路（ICチップ）を中心に構成され、記憶装置に記録されたオペレーティングシステム（OS）やアプリケーションソフトなどのコンピュータプログラム（ソフトウェア）を実行するものを指す。

#### ♣ コンピュータの分類

一般的にコンピュータとみなされる機器には、個人向けの汎用コンピュータである「パソコンコンピュータ」（PC／パソコン）や、企業や官公庁などの情報システムで用いられる大規模・高性能コンピュータである「サーバ」や「メインフレーム」、科学技術計算などに用いる超高性能コンピュータである「スーパーコンピュータ」<sup>\*1</sup>などがある。

また、現代の電気機器の多くは内部の装置の制御などのために機器内部に小型のコンピュータシステムを内蔵しており「組み込みシステム」と呼ばれる。

<sup>\*1</sup> 普通のパソコンの処理能力は 80GFLOPS です。つまり毎秒 800 億回=0.08 兆回の浮動小数点数の計算が出来ます。理化学研究所の「富岳」は 440PLOPS(毎秒 44 京回=440000 兆回) の演算が可能です。つまり普通のパソコンの 550 万倍の性能があります。

このような組み込み型のコンピュータを備えた機器には携帯電話・スマートフォンやタブレット端末、ビデオゲーム機、通信装置やネットワーク機器、テレビ受像機、ビデオレコーダー、デジタルカメラ、電子制御の家電製品や産業機械、輸送機械などがある。

### ♣ コンピュータの構成

一般的なコンピュータは、プログラムの実行状況や各装置の状態を制御する「制御装置」、データの計算や加工を行う「演算装置」、データを記録する「記憶装置」、人間や他の機器など外界との情報のやり取りを行う「入力装置」および「出力装置」などで構成される。この五つの要素を「コンピュータの五大装置」と呼ぶこともある。

このうち、制御装置と演算装置は現代では一つの装置や半導体チップとして統合されていることが多く、コンピュータシステム全体の制御を司る中心的な処理装置のことを「中央処理装置」(CPU) という。

記憶装置は当座の動作に必要なプログラムやデータの一時的な記憶に用いる「主記憶装置」(メインメモリ) と、永続的な記録に用いる「外部記憶装置」(ストレージ) に分かれていることが多い。

計算手順は CPU に対する命令の列を記憶装置にデータとして記録し、順に読み出して実行していく方式（プログラム内蔵方式）になっており、これを「コンピュータプログラム」あるいは単にプログラムという。

### ♣ プログラム内蔵方式 / ノイマン型コンピュータ

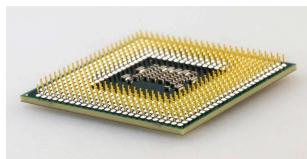
ノイマン型コンピュータとは、プログラムをデータとして記憶装置に格納し、これを順番に読み込んで実行するコンピュータ。現在のコンピュータのほとんどがこの方式を採用している。「コンピュータの父」とも呼ばれるアメリカの数学者、ジョン・冯・ノイマンの名に由来する。

第二次大戦前後に生まれた最初期の電子式コンピュータは真空管の配列や配線が計算内容をそのまま反映したものになっており、別の計算を行うためには配線をすべてやり直さなければならず、汎用性が著しく乏しかった。

ノイマンら新型コンピュータ「EDVAC」開発チームは、計算手順や入力値をハードウェアから独立させてデータとして外部から与え、汎用の回路群でこれを処理する方式を構想した。ソフトウェア（コンピュータプログラム）という概念もこのとき誕生した。

この方式が「ノイマン型」と呼ばれるのは、最初に発表された "First Draft of a Report on the EDVAC" という文書の著者がノイマンだったからだが、ノイマン自身は EDVAC 開発プロジェクトには後から参加しており、プログラム内蔵方式というアイデアの確立には当初から関わっていたジョン・エッカートとジョン・モーカリーの功績が大きかったとされる。

## 4.2 ハードウェア



### CPU

中央処理装置 (Central Processing Unit) とは、コンピュータの主要な構成要素の一つ。他の装置・回路の制御やデータの演算などを行う装置で、演算装置と制御装置が統合されている。CPU はメインメモリ (RAM) に格納された機械語のプログラムを、バスを通じて

一命令ずつ順番に読み出し (フェッチ)、その内容を解釈して行うべき動作を決定 (デコード) し、内部の回路を駆動して実際に処理を実行する。

一度に 4 ビットのデータを処理できる CPU を 4 ビット CPU というように呼び、当初の 4 ビットから、8, 16, 32 ビットと拡張され、現代では 64 ビット CPU が普及している。

高い周波数信号で動作する CPU ほど、多くの処理を行え性能が高い。2GHz(ギガヘルツ：毎秒 10 億回) で動作する CPU と 1GHz の CPU ならば、約 2 倍の速度差がある。



### メモリ

メモリとは、記憶、記憶力、回想、追憶、記念などの意味を持つ英単語。IT 分野ではコンピュータに内蔵される半導体集積回路 (IC) を利用したデータの記憶装置を指す。コンピュータを構成する装置の一つで、CPU から直接読み書きすることができる記憶装置のことを「主記憶装置」という。

一般に主記憶装置は外部記憶装置よりはるかに高速に動作する装置が用いられるが、単価や装置構成上の制約から少ない搭載容量となっている。このため、コンピュータは起動すると外部記憶から主記憶に必要なプログラムやデータを読み込んで実行し、不要となったデータは主記憶から消去して新たに必要になったものに入れ替える。永続的な保管が必要なデータは外部記憶へ書き込み保存する。



### ストレージ

ストレージとは、コンピュータの主要な構成要素の一つで、コンピュータが利用するプログラムやデータなどを永続的に記憶する装置。磁気ディスク (ハードディスク HDD) や光学ディスク (CD/DVD/Blu-ray)、フラッシュメモリ (USB メモリ/メモリカード/SSD)、磁気テープなどがある。

同じコンピュータに搭載される装置同士で比較すると、ストレージはメモリに比べて記憶容量が数桁 (数十～数千倍) 大きく、容量あたりのコストが数桁小さいが、読み書きに要する時間が数桁大きい。

写真は、磁性体を利用した HDD (ハードディスクドライブ) のものであるが、半導体を利用した SSD (ソリッドステートドライブ) がその高速性から普及している。



### マザーボード

マザーボードとは、コンピュータの主要な構成部品の一つで、マイクロプロセッサやメモリなど他の部品を装着し、通電したり相互に通信できるようにする基板のこと。プラスチックなどでできた板状の装置で、表面や内部に各装置を結ぶ配線や制御用の半導体チップ、

電子部品などが高密度に実装されている。プロセッサやメモリモジュール、拡張カードなどを装着するためのスロットやソケットなどの接続部品、電源ユニットからのコードを差し込む電源コネクタ、ストレージ(外部記憶装置)など周辺機器接続用のケーブルを差し込むコネクタなども配置されている。



### 入力装置(キーボード)

正方形や横長の小さなボタンが縦横に整然と並び、文字や記号、コンピュータへの指示などを送信するための入力装置。100前後のキーが4~5段に並び、各キーの上部に入力される文字や機能が記されて

おり、キーを押すと、そのキーが押されたという信号がコンピュータへ送信される。



### 出力装置(ディスプレイ)

ディスプレイとは、表示、展示、陳列、掲示、飾り付け、示すなどの意味を持つ英単語。IT分野では、コンピュータの出力装置の一つで、画面を発光させて像を映し出す表示装置(display device、ディスプレイ装置)のこと。「モニター」(monitor)とも呼ぶ。

コンピュータの操作画面を映像として電気的に映し出し、処理状況の変化や利用者の操作に即時に反応して表示内容を変化させることができる。

データとして記録された動画像を再生・表示することもできる。ディスプレイ以前に主要な出力装置として利用されていたのは印字装置(プリンタ)であり、状況や操作を表示内容にリアルタイムに反映する特徴は画期的で便利な特性だった。

ディスプレイの画面は格子状に規則正しく並んだ微細な画素(ドット/ピクセル)から成り、その発光状態を電気的に制御してコンピュータから受信した映像信号を表示する。一つの画素を光の三原色に対応する微細な素子で構成し、カラー表示を行う。



**USB ユニバーサルシリアルバス** USBとは、主にコンピュータと周辺機器を繋ぐ為に用いられるデータ伝送路の標準規格。キーボードやマウス、プリンタ等の接続方式として広く普及している。

初期の規格(USB 1.1)では 12Mbps(メガビット毎秒)、最新の規格(USB4)では 40Gbps(ギガビット毎秒)までの通信速度に対応する。当初はキーボードなどの入出力装置から普及が始まったが、

通信速度の向上に伴いネットワークアダプタ(EthernetアダプタやWi-Fiアダプタ)や、外部接続の光学ドライブ、ハードディスクなどに利用が広がっていった。手軽なデータの受け渡し手段としてフラッシュメモリを内蔵したUSBメモリもよく使われる。

## 4.3 ソフトウェア

コンピュータを構成する電子回路や装置など、物理的実体がある「ハードウェア」に対し、それ自体は形を持たないプログラムや付随するデータなどをソフトウェアという。

コンピュータを動作させる命令の集まりであるコンピュータプログラムを組み合わせ、何らかの機能や目的を果たすようまとめたもので、その役割により、ハードウェアの制御や他のソフトウェアへの基盤的な機能の提供、利用者への基本的な操作手段の提供などを行なう「オペレーティングシステム」(OS : Operating System / 基本ソフト) と、特定の個別的な機能や目的のために作られた「アプリケーションソフト」に大別される。

### ♣ OS (Operating System / 基本ソフト)

機器の基本的な管理や制御のための機能や、多くのソフトウェアが共通して利用する基本的な機能などを実装した、システム全体を管理するソフトウェア。

入出力装置や主記憶装置、外部記憶装置の管理、外部の別の装置やネットワークとのデータ通信の制御などが主な役割で、コンピュータに電源が投入されると最初に起動し、電源が落とされるまで動作し続ける。利用者からの指示に基いて記憶装置内に格納されたソフトウェアを起動したり終了させたりすることができる。

パソコン向けの OS としては、Microsoft 社の Windows や Apple 社の mac OS や、Linux がある。スマートフォンやタブレットでは、Apple 社の iOS や、Google 社の Android OS が普及している。

東京大学名誉教授 坂村健 氏が提唱した TRON<sup>トロン</sup>も、組込機器向けの μ ITRON<sup>マイクロアイトロン</sup>として活用されており、任天堂のゲーム機「Nintendo Switch」などに採用されている。<sup>\*2</sup>

### ♣ アプリケーションソフト

OS の機能を利用し、OS の上で動作するソフトウェアである。アプリケーションの開発者は、OS の提供する機能を利用するによって、開発の手間を省くことができ、操作性を統一することができる。また、ハードウェアの仕様の細かな違いは OS が吸収してくれるため、ある OS 向けに開発されたソフトウェアは、基本的にはその OS が動作するどんなコンピュータでも利用できる。

用途や目的に応じて多種多様なアプリケーションソフトがある。一例を挙げると、ワープロや表計算、画像閲覧・編集、動画・音楽再生、ゲーム、Web ブラウザ、電子メール、カレンダー・スケジュール管理、電卓、カメラ撮影、地図閲覧、プレゼンテーションやデータベース、財務会計、人事管理、在庫管理、プロジェクト管理、文書管理などなどである。

利用者が配布・販売パッケージ入手・購入して OS に組み込む作業「インストール」を行うことで使用可能となる。

<sup>\*2</sup> 「Nintendo Switch」が μ ITRON4.0 仕様準拠リアルタイム OS を採用 (<https://monoist.itmedia.co.jp/mn/articles/1707/07/news029.html>)

## 【コラム】単位の話

コンピュータでは大きな数、小さな数を良く使います。

- $1,000 \text{ m} = 1 \text{ km}$
- $100 \text{ a} = 1 \text{ ha}$
- $1 \text{ dL} = 1 / 10 \text{ L}$
- $1 \text{ cm} = 1 / 100 \text{ m}$
- $1 \text{ mm} = 1 / 1,000 \text{ m}$

などがお馴染みですが、他にも様々な記号がございますので、ご紹介いたします。

国際単位系(SI単位系)での接頭辞一覧

接頭辞	接頭辞	記号	乗数
ヨタ	yota	Y	10 の 24 乗=1,000,000,000,000,000,000,000,000 倍
ゼタ	zeta	Z	10 の 21 乗=1,000,000,000,000,000,000,000 倍
エクサ	exa	E	10 の 18 乗=1,000,000,000,000,000,000 倍
ペタ	peta	P	10 の 15 乗=1,000,000,000,000,000 倍
テラ	tera	T	10 の 12 乗=1,000,000,000,000 倍
ギガ	giga	G	10 の 9 乗=1,000,000,000 倍
メガ	mega	M	10 の 6 乗=1,000,000 倍
キロ	kilo	k	10 の 3 乗=1,000 倍
ヘクト	hecto	h	10 の 2 乗=100 倍
デカ	deca	da	10 の 1 乗=10 倍
			10 の 0 乗=1 倍
デシ	deci	d	10 の -1 乗=10 分の 1
センチ	centi	c	10 の -2 乗=100 分の 1
ミリ	milli	m	10 の -3 乗=1,000 分の 1
マイクロ	micro	μ	10 の -6 乗=1,000,000 分の 1
ナノ	nano	n	10 の -9 乗=1,000,000,000 分の 1
ピコ	pico	p	10 の -12 乗=1,000,000,000,000 分の 1
フェムト	femto	f	10 の -15 乗=1,000,000,000,000,000 分の 1
アト	atto	a	10 の -18 乗=1,000,000,000,000,000,000 分の 1
ゼプト	zepto	z	10 の -21 乗=1,000,000,000,000,000,000,000 分の 1
ヨクト	yocto	y	10 の -24 乗=1,000,000,000,000,000,000,000,000 分の 1

# 第 5 章

## 二進数の話

コンピュータは、「0」と「1」の二つの値を用いる二進数で動いています。黎明期には十進数を用いた計算機もありましたが、実行速度や作成費用などから二進法を用いるものが主流となりました。

今日のコンピュータの基礎となる二進数と、関連して様々な情報表現について紹介します。

また、より深く計算機理論を学びたい方へ向けて、デジタル回路のご紹介と、富山大学幸山教授が書かれた「計算機理論入門」への参照を挙げました。

### 5.1 二進数と十進数、十六進数

十進数	二進数	十六進数
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

十進数は、わたくしたちが日常使っている位取り記法で、とても馴染みがあります。

0、1、2、…、8、9と大きくなり、「一の位」が「十」集まる上位に「進」みます。

10、11、…、99と大きくなり、「十の位」が「十」集まる上位に「進」みます。

100、101、…、999と大きくなり、「百の位」が「十」集まる上位に「進」みます。

「十」集まって一つ上の位に「進」むたびに、その桁の重みが十倍になるのが、「十進数」の特徴です。例えば、「9 4 3」という数なら、百を9つ十を4つ一を3つ併せた数のことです。

二進数でも同様ですので、見ていきましょう。

0、1と大きくなり、「一の位」が「二」集まる上位に「進」みます。

10、11と大きくなり、「二の位」が「二」集まる上位に「進」みます。

100、101～111と大きくなり、「四の位」が「二」集まる上位に「進」みます。

「二」集まって一つ上の位に「進」むたびに、その桁の重みが二倍になるのが、「二進数」の特徴です。

例えば、「1011」という数なら、八を1つ二を1つ1を1つ併せた数、つまり じゅういちのことです。

二進数では、すぐに桁数が増えていくので、四桁をひとまとめにした、十六進数もよく使われます。十六進数では 10~15 までを一桁で表せるような記号を準備する必要があります。黎明期には様々な記号が提案されましたが、今日では A, B, C, D, E, F のアルファベットの使用が定着しました。(a, b, c, d, e, f と小文字を使うことも出来ます。) <sup>\*1</sup>

## 5.2 コンピュータでのデータ表現

### ♣ 文字

コンピュータで文字を表現するにはどのようにすれば良いでしょうか？

文 字 進 進	10 16	文 字 進 進	10 16	文 字 進 進	10 16	文 字 進 進	10 16	文 字 進 進	10 16	文 字 進 進	10 16	文 字 進 進
NUL 0 00	DLE 16 10	SP 32 20	0 48 30	@ 64 40	P 80 50	` 96 60	p 112 70					
SOH 1 01	DC1 17 11	! 33 21	1 49 31	A 65 41	Q 81 51	a 97 61	q 113 71					
STX 2 02	DC2 18 12	" 34 22	2 50 32	B 66 42	R 82 52	b 98 62	r 114 72					
ETX 3 03	DC3 19 13	# 35 23	3 51 33	C 67 43	S 83 53	c 99 63	s 115 73					
EOT 4 04	DC4 20 14	\$ 36 24	4 52 34	D 68 44	T 84 54	d 100 64	t 116 74					
ENQ 5 05	NAK 21 15	% 37 25	5 53 35	E 69 45	U 85 55	e 101 65	u 117 75					
ACK 6 06	SYN 22 16	& 38 26	6 54 36	F 70 46	V 86 56	f 102 66	v 118 76					
BEL 7 07	ETB 23 17	' 39 27	7 55 37	G 71 47	W 87 57	g 103 67	w 119 77					
BS 8 08	CAN 24 18	( 40 28	8 56 38	H 72 48	X 88 58	h 104 68	x 120 78					
HT 9 09	EM 25 19	) 41 29	9 57 39	I 73 49	Y 89 59	i 105 69	y 121 79					
LF* 10 0a	SUB 26 1a	* 42 2a	: 58 3a	J 74 4a	Z 90 5a	j 106 6a	z 122 7a					
VT 11 0b	ESC 27 1b	+ 43 2b	; 59 3b	K 75 4b	[ 91 5b	k 107 6b	{ 123 7b					
FF* 12 0c	FS 28 1c	, 44 2c	< 60 3c	L 76 4c	¥ 92 5c	l 108 6c	124 7c					
CR 13 0d	GS 29 1d	- 45 2d	= 61 3d	M 77 4d	] 93 5d	m 109 6d	} 125 7d					
SO 14 0e	RS 30 1e	. 46 2e	> 62 3e	N 78 4e	^ 94 5e	n 110 6e	- 126 7e					
SI 15 0f	US 31 1f	/ 47 2f	? 63 3f	O 79 4f	_ 95 5f	o 111 6f	DEL 127 7f					

▲図 5.1: ASCII コード表

ASCII コードと呼ばれる表で、アルファベット「A」には十進数の「65」十六進数の「41」が、「B」には十進数の「66」十六進数の「42」が対応しています。

コンピュータは西洋で生まれましたので、アルファベット ( A - Z, a - z )、数字 ( 0 - 9 ) や、記号 (@, #, ! など) など約 100 種類といくつかの制御用のコードを加えて、番号付けした、今日でも現役で使われている表です。

\*1 卷末の参考文献「プログラマの数学」にも位取りを始めとして様々な話題がございます。是非ご一読下さい。

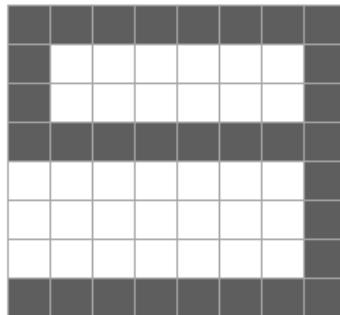
## ♣ 画像

コンピュータで画像を表現するにはどのようにしたら良いのでしょうか？

今日、デジタルカメラで写真を撮影することはとても一般的になりました。その嚆矢となったのが、未来技術遺産にも登録された、カシオ QV-10 \*2です。

QV-10 では、横 320 画素 x 縦 240 画素 (=76,800 画素) の写真を撮ることができました。つまり横 320 箇、縦 240 箇の升目に分割、各升目の色を表現することにより、一枚の画像を表しています。

$8 \times 8 = 64$  個の画素を使って作成した 数字の「9」の画像です。



画素が光っていない黒を 0、光っている白を 1 とすると、次のようにになります。

数値表現の例

0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0

合計 64 bit (= 8 Byte) で表現できました。

それでは、色の情報はどのように表せば良いでしょうか。

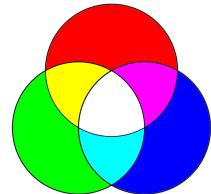
20 年ほど前に、愛知県総合教育センター \*3 で公開された記事よりご紹介します。(歴史を感じていただけると幸いです。)

\*2 10 年先の写真を見据えて——カシオ「QV-10」(<https://www.itmedia.co.jp/dc/articles/1101/11/news028.htm>) に、愛情溢れる記事が掲載されています。

\*3 出典: なぜ、フルカラーで表示できる色は、1677万色なの? 愛知県総合教育センター (<https://apec.aichi-c.ed.jp/kyouka/jouho/contents/2018/jissyuu/043/gazou.files/iro.htm>)

## 光の三原色

テレビやコンピュータのブラウン管に表示される色は、光の三原色の赤(R)、緑(G)、青(B)を混ぜて、作り出されています(加法混合)。テレビのブラウン管を虫眼鏡などで拡大してみると、赤、緑、青の蛍光体が光っている様子がわかると思います。

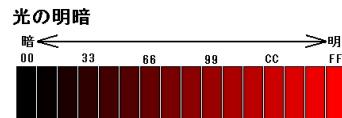


初期のコンピュータ<sup>\*4</sup>は、現在のように多彩な色表示ではなく、白や黒を含めて8色表示が可能でした。それは、赤、緑、青の各色についてON/OFF(光らせる／消す)を制御し、その組み合わせで、 $2 \times 2 \times 2 = 8$ 通りの色を表現しました。

つまり、赤、青、緑が単独で光ることで、(1)赤、(2)青、(3)緑の三色が表現できます。赤と緑、緑と青、青と赤の各二色が光ると、(4)黄色、(5)シアン、(6)マゼンタの三色が表現できます。そして三色が全て光ることで、(7)白が表現でき、全て消えることで(8)黒が表現できるので、合計8色を表現できました。

## 光の明暗

その後、ON / OFFの状態だけではなく、明るさを何段階かに調節した表示が可能になりました。下の図は、赤色を暗い方から明るい方へと16段階で表示したものです。



明暗を変化させることにより、赤色で16通りの表現ができます。同じように、緑色、青色も16段階で表示すると、その各色16通りを組み合わせて、 $16 \times 16 \times 16 = 4,096$ 通りの色が表現できます。

現在では、256段階( $=8\text{ bit}$ )で明るさが調節できるようになりました。それで、 $256 \times 256 \times 256 = 16,777,216$ 通りの色が表現できます。人の目で識別できる色数は、数百万～一千万色と言われるので、この約1677万色のことを「フルカラー」と呼ぶこともあります。

## 画像の容量

QV-10では、横320画素×縦240画素( $=76,800$ 画素)の写真を撮ることができました。色を各色256段階で表現することになると、一つの画素につき、赤8ビット( $=1\text{ Byte}$ )、青8ビット( $=1\text{ Byte}$ )、緑8ビット( $=1\text{ Byte}$ )と、合計3Byte必要ですから、 $230,400\text{ Byte} (=225\text{ kB})$ の容量になります。QV-10には2MB<sup>\*5</sup>のフラッシュメモリが内蔵されており、96枚までの写真が保存できました。<sup>\*5</sup>

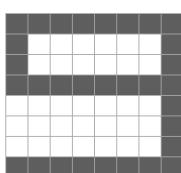
それから四半世紀経った今日のiPhoneでは、 $6,048 \times 8,064 = 48,771,072$ もの画素数で写真を撮ることができ、さらに赤1,024段階( $=10\text{ bit}$ )、緑1,024段階( $=10\text{ bit}$ )、青1,024段階( $=10\text{ bit}$ )の1,073,741,824色が表現できます。なので、写真一枚を保存するために、 $6,048 \times 8,064 \times 10 \times 10 \times 10 = 48,771,072,000\text{ bit} (=6,096,384,000\text{ Byte} = 5,953,500\text{ kB} = 5,814\text{ MB} = 5.678\text{ GB})$ もの容量が必要となります。数十枚写真を撮るだけで、容量一杯となってしまいます。

そこで、写真の情報量をそのまま保存するのではなく、「圧縮」して保存することにします。

<sup>\*4</sup> 昭和54年に発売されたPC-8001では、テキスト表示80桁×25行、グラフィック表示160×100ドットデジタル8色であった。

<sup>\*5</sup> 当時は24枚撮りの写真フィルムが発売されていましたが、4倍もの写真を撮って、その場で確認することができる、とても画期的な製品でした。

## ◆ 画像データの圧縮



コンピュータでは左の絵のように、ある一定の区画ごとに画像を区切り、それぞれの明暗に応じ数値化することで、画像を表現します。

$8 \times 8 = 64$  個の画素を使って作成した 数字の「9」の画像を、画素が光っていない黒を 0、光っている白を 1 とすると、**00000000 01111110 01111110  
00000000 11111110 11111110 11111110 00000000** の 64 bit (= 8 Byte) ビット バイト で、表現できました。

これを圧縮することを考えます。

黒	黒	黒	黒	黒	黒	黒	黒
黒	白	白	白	白	白	白	黒
黒	白	白	白	白	白	白	黒
黒	黒	黒	黒	黒	黒	黒	黒
白	白	白	白	白	白	白	黒
白	白	白	白	白	白	白	黒
白	白	白	白	白	白	白	黒
黒	黒	黒	黒	黒	黒	黒	黒

そのまま表現する代わりに、同じ色が続くところは色の数を記すことにします。すると、

黒9白6黒2白6黒9白7黒1白7黒1白7黒9
------------------------

と 22 文字で表現できます。

さらに黒の次は白と決まっているので、色の表記も省略できます。すると、

9 6 2 6 9 7 1 7 1 7 9
-----------------------

と 11 文字で表現できます。

そのまま表現していた際には 64 文字必要でしたが、 $11 \div 64 = 17\%$  と 約  $1 / 6$  で済みました。これが圧縮の原理です。(ランレンジス圧縮というアルゴリズムで、FAX などで用いられています。)

FAX などで用いられているランレンジス圧縮ですが、繰り返しが少ないと効率が悪化するという弱点を抱えています。そこで、より優れた様々な圧縮アルゴリズムが考案されています。

その嚆矢となったのがイスラエルの計算機科学者ジェイコブ・ジヴとエイブラハム・レンペルによって開発されたデータ圧縮アルゴリズム LZ77 です。今日まで続くファイル圧縮フォーマット ZIP や、電子文書フォーマット PDF、音声ファイルフォーマット MP3 の基礎となりました。<sup>\*6</sup>

\*6 ZIP 圧縮や PNG・PDF などファイルフォーマットの基礎を作ったジェイコブ・ジヴ博士が死去 (<https://gigazine.net/news/20230327-jacob-ziv-passed-away/>)

その後もより良い圧縮アルゴリズムの研究開発が続けられ、例えば、iPhone では HEVC(High Efficiency Video Coding) 方式を採用することにより、一枚の写真を保存するために 5.678 GB<sup>ギガバイト</sup> 必要であったものを、数 MB<sup>メガバイト</sup> と、数百分の一に圧縮しています。

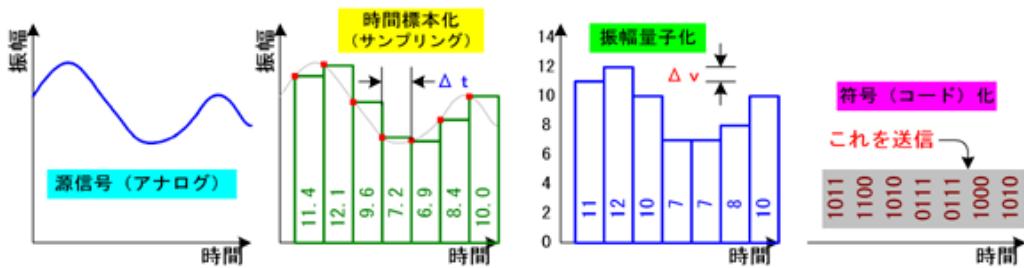
## ♣ 音声

音声をコンピュータで扱うために、様々な方式が考案されています。その中からパルス符号変調(PCM Pulse Code Modulation)を紹介します。

**PCM** とは、音声などのアナログ信号をデジタルデータに変換する方式の一つで、信号の強度を一定周期で標本化（サンプリング）したものです。そのまま保存すれば無圧縮データとなります。

アナログ信号の強度をサンプリング周波数に従って一定間隔で測定し、定められたビット数の範囲で整数値として量子化します。例えば、CD の音声はサンプリング周波数 44.1kHz (キロヘルツ)、量子化 16 ビットで記録されています。これは毎秒 44,100 回信号を測定し、その強度を 65,536 (2 の 16 乗) 段階の値で表していることを意味しています。

**標本化定理**により、サンプリング周波数の半分の周波数までの信号は再現可能です。人間の可聴音の上限は 20kHz 程度であることが知られているので、40kHz 超のサンプリング周波数を用いれば録音データから概ね自然な音が再生できると言われています。<sup>\*7 \*8</sup>



▲図 5.2: パルス符号変調(PCM)方式によるアナログ-デジタル変換

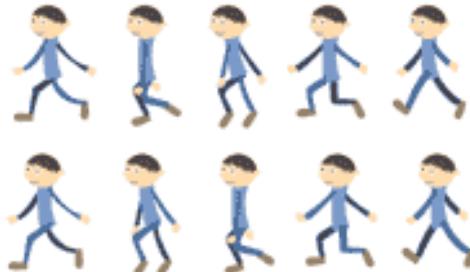
\*7 出典: IT 用語辞典

\*8 図版出典: パルス変調方式に用いるアナログ信号のデジタル化方式の説明 ([http://www.gxk.jp/elec/musen/lama/H19/html/H1912A09\\_.html](http://www.gxk.jp/elec/musen/lama/H19/html/H1912A09_.html))

## ♣ 動画

パラパラ漫画を読んだことはありますか。<sup>\*9</sup>

一枚一枚は人が写っている普通の画像ですが、それらの画像を短い間に切り替えます。すると、「残像効果」<sup>\*10</sup>「仮現運動」<sup>\*11</sup>と呼ばれる人間の特性により、あたかも動いているように見えます。



▲図 5.3: パラパラ漫画

それでは、一秒間に何枚の画像を切り替えると、滑らかな動画に見えるのでしょうか。

一秒間に表示される画像の枚数が少なく、一枚の画像が表示される時間が長いと、動きのカクカクとした不自然で低品質な動画となります。短時間で画像が書き換わると、滑らかで高品質な動画となります。<sup>\*12</sup>映画では一秒間に 24 コマ、テレビ放送では一秒間に 30 コマの表示が行われています。

一秒間に 24 コマ、あるいは、30 コマの絵を表示すれば、「動画」になることが分かりました。二時間の映画ではたくさんの画像 ( $172,800 \text{ 枚} = 24 \times 2 \times 60 \times 60$ ) が必要になります。動画の容量が気になります。

映画は 24 コマ/秒で、1 枚の標準的な画像は (480 画素 × 720 画素) です。画素一つ一つにつき、色を表現するために 24 ビット (=3 バイト) 必要ですから、一秒間の動画を保存するために約 24MB(メガバイト) (=480 × 720 × 3 × 24 / 1024 / 1024) が必要になります。DVD 一枚は 4.7GB(ギガバイト)<sup>\*13</sup>の容量がありますので、約 184 秒 = 約 3 分<sup>\*14</sup> (=4.25 × 1024 / 23.73 / 60) の動画が記録できます。

### 情報圧縮

DVD 一枚には、約 184 秒 = 約 3 分の動画が保存できることが分かりました。二時間 120 分の映画の映画を保存するためには、40 枚の DVD が必要となります。しかし、実際には一枚の DVD の中に、

<sup>\*9</sup> 画像出典: IT 用語辞典

<sup>\*10</sup> 出典: Wikipedia 人の視覚で光を見たとき、その光が消えた後も、それまで見ていた光や映像が残って見えるような現象のこと。

<sup>\*11</sup> 出典: goo 国語辞典 実際には運動がないのに、次々と類似の刺激を与えられると、運動があるように感じる現象。映画はこの現象を応用したもの。

<sup>\*12</sup> 出典: IT 用語辞典

<sup>\*13</sup> DVD の容量は 4.7GB と表示されて販売されています。このときは 1GB=10 億バイトとして計算しています。コンピュータの内部では、二進数で綺麗に計算できる 2 の 30 乗 =  $1,024 \times 1,024 \times 1,024 = 1,073,741,824$  バイトを、1GB として扱います (1GiB と書くこともあります)。

<sup>\*14</sup> DVD は 4.7GB(=4.37GiB) の容量がありますが、ファイル管理用の領域があるため、利用者が使えるデータ領域は 4.25GiB となります

二時間 120 分の映画が収録されています。これを支えているのが、「情報圧縮」技術です。

### 画像・音声圧縮の基本原理と要素技術 <sup>\*15</sup>

#### 1. 画像の性質を利用

二値画像: 白黒画素が連續しやすいので、「ランレングス符号化」<sup>\*16</sup>する。

静止画: 近くの画素は似ているので、「離散コサイン変換 (DCT)」<sup>\*17</sup>する。

動画像: 現画面は前画面に似ているので、「フレーム間予測」<sup>\*18</sup>する。

#### 2. 人間の視聴覚特性を利用

画像: 色信号の劣化には鈍感なので、色情報を「サブ・サンプリング処理」<sup>\*19</sup>により間引く。

音声: 大きな音と同時に存在する小さい音は聞こえにくいという「マスキング効果」<sup>\*20</sup>を利用する。

#### 3. 符号の発生確率の偏りを利用

符号発生確率に差があるので、「可変長符号化」や「算術符号化」<sup>\*21</sup>を行う。

今日では、**Moving Picture Experts Group** が規格した、**MPEG** が広く動画圧縮方式として用いられています。

## 5.3

## 単位の話

コンピュータでよく使われる単位の話です。既出も含め紹介します。

- 1 ビット (bit) コンピュータで扱うことができる最小の情報量です。スイッチが入っている (1) か、切れている (0) かの、二通りの状態を表せます。二進数 1 桁の情報量です。
- 2 桁 (2 ビット) では、00, 01, 10, 11 の 4 通りを表せます。
- 3 桁 (3 ビット) では、000, 001, 010, 011, 100, 101, 110, 111 の 8 通りを表せます。
- 1 バイト (Byte) 半角文字 1 文字分 (8 ビット 二進法の 8 桁分) の情報量です。256 通りの文字を区別できるため、数字や記号、アルファベットが表せます。
- 2 バイト 全角文字 1 文字分の情報量です。日本語は「ひらがな」や「カタカナ」、「漢字」などから成り立ちます。2 バイトあれば、 $256 \times 256 = 65536$  通りの文字を区別することができます。どの文字にどの番号を割り振るのか、様々な文字体系があります。以前は Shift-JIS(シフトジス) と呼ばれる文字体系が使われていましたが、現在では世界中の文字を 3 バイトで表現する UTF-8(ユーティーエフエイト) が広く使われています。
- 1 キロバイト (kB) = 1,024 バイト。2 進数 10 桁 ( $=2$  の 10 乗) で 1024 通りを表現できます。10 進法で、1,000 倍のことを k (キロ) といいます。コンピュータの世界では、 $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$  を十回掛けると 1024 になるので、1024 でひとまとめにして、

\*16 ランレングス圧縮 (<https://e-words.jp/w/ランレングス圧縮.html>)

\*17 離散コサイン変換 (DCT) (<https://e-words.jp/w/離散コサイン変換.html>)

\*18 フレーム間予測 (<https://e-words.jp/w/フレーム間予測.html>)

\*19 サブ・サンプリング (<http://www.hdmi-navi.com/subsampling/>)

\*20 マスキング効果 (<https://ja.wikipedia.org/wiki/音響心理学#マスキング効果>)

\*21 符号割り当て ([https://www.ieice-hbkb.org/files/02/02gun\\_05hen\\_07.pdf](https://www.ieice-hbkb.org/files/02/02gun_05hen_07.pdf))

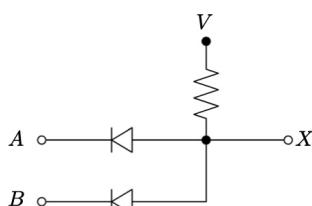
\*15 出典: 日本規格協会グループ 画像・映像圧縮 (JPEG/MPEG) マルチメディアの基盤である 画像や映像に関する技術と関連国際規格 ([https://www.jsa.or.jp/datas/media/10000/md\\_2471.pdf](https://www.jsa.or.jp/datas/media/10000/md_2471.pdf))

1024 バイトのことを 1kB といい、およそ A4 用紙一枚程度の情報量です。

- 1 メガバイト (MB) = 1,024 キロバイト = 1,048,576 バイトです。  
小さめの写真約 1 枚分、A4 約 1000 枚分の情報量です。昔懐かしいフロッピーディスク 1 枚は 1.44MB でした。
- 1 ギガバイト (GB) = 1,024 メガバイト = 1,048,576 キロバイト = 1,073,741,824 バイト。  
CD 1 枚 74 分で 640MB です。DVD 1 枚 2 時間で、4.7GB ~ 8.5GB です。高画質の映画などブルーレイディスクは、25GB ~ 100GB です。
- 1 テラバイト (TB) = 1,024 ギガバイト = 1,048,576 メガバイト = 1,073,741,824 キロバイト  
 $= 1,099,511,627,776$  バイト。ハードディスクはとても大容量です。1TB のハードディスクで毎日 2 時間ずつ動画を見るなら、おおよそ 100 日分の容量になります。

## 5.4

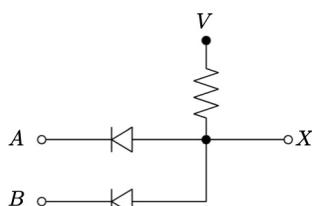
## 計算機理論入門



**AND 回路**

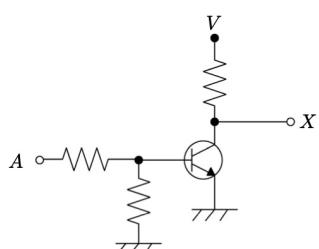
コンピュータは、0 と 1 の二進数で動いていること、その二進数を組み合わせて文字や画像など様々な情報表現が可能なことをご紹介いたしました。

黎明期には、真空管を用いて作られていたコンピュータですが、トランジスタ、IC、LSI と集積化が進み、小さなチップの中に、数百億ものトランジスタが実装されるようになりました。



**OR 回路**

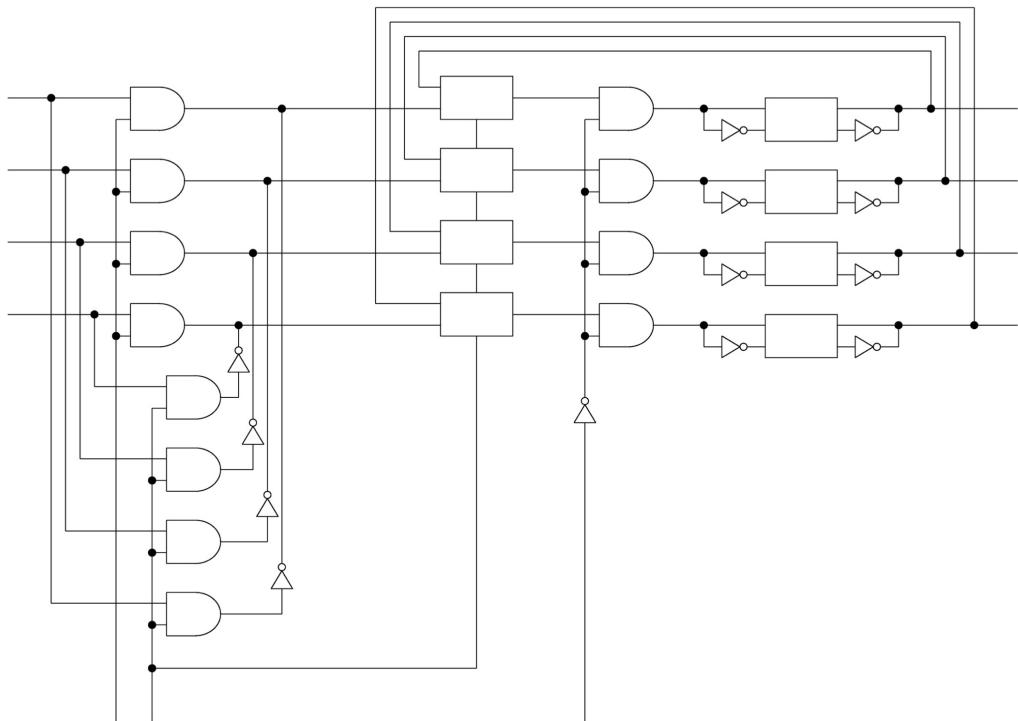
多くのトランジスタが高速に動作することで今日のコンピュータの性能がありますが、デジタル回路を支える三つの論理素子、それは AND, OR, NOT です。この三つの論理素子を組み合わせることで、演算、制御、記憶など、コンピュータを形作ることができます。



**NOT 回路**

より本格的に計算機理論を学びたい方のために、富山大学の幸山教授が書かれた「計算機理論入門～コンピュータを設計しよう～」がございます。

二進数の基礎から始まり、論理演算や論理回路を学び、最終的には 4 ビットの加減算ができるコンピュータシステムを作り上げていく内容です。半導体やトランジスタの仕組み、電子回路の基礎も学べる内容となっています。



▲図 5.4: 4 ビット計算機

上の図は、[計算機理論入門～コンピュータを設計しよう～](#)<sup>\*22</sup>で、紹介されている「4 ビット計算機」の回路図です。

とても良く纏まっているテキストで、少し高度な内容を扱った高校生向けの内容となります。是非ご一読なさってみてください。

\*22 <https://kouyama.sci.u-toyama.ac.jp/main/etc/2003/ssh/sshi.pdf>

## 【コラム】コンピュータ豆知識

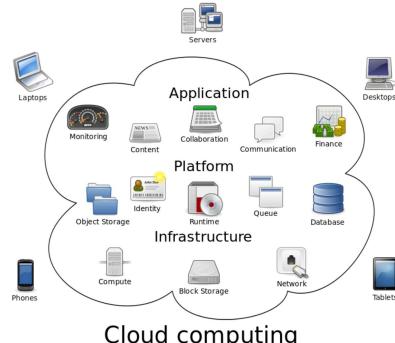
### インターネット

世界中のコンピュータ同士が繋がった巨大なネットワーク(網)。前身は ARPANET(アーパネット、高等研究計画局ネットワーク)。ジョゼフ・カール・ロブネット・リックライダー等がアイディアを創り上げた。日本では、村井純氏等が普及に多大な貢献をしている。

### クラウド \*23

英語で「雲」の意味。従来、自分のコンピュータで行った処理を、インターネット上のコンピュータ上で行う。演算の他、情報伝達、データの保等、様々な役務が提供されている。

例えば、政府の行政システムのクラウド化には、米国アマゾンとグーグル社が使われることとなった。



### 電子署名(デジタル署名)

電子署名とは、文書やメッセージなどのデータの真正性を証明するために付加される、短い暗号データ。作成者を証明し、改竄やすり替えが行われていないことを保証する。欧米で紙の文書に記されるサインに似た働きをするためこのように呼ばれる。 \*24

### 量子コンピュータ

$$\Delta x \Delta p \geq \frac{\hbar}{2}$$

量子コンピュータは、量子力学的な重ね合わせを用いて並列性を実現するコンピュータである。従来のコンピュータ(以下「古典コンピュータ」)の基本素子は、情報量が0か1のいずれの値しか持つ得ない1ビットを扱うものであるのに対して、

量子コンピュータでは量子ビット(qubit; quantum bit、キュービット)により、1キュービットにつき0と1の値を任意の割合で重ね合わせて保持する。n量子ビットあれば、2のn乗の状態を同時に計算できる。もし、数千qubitのハードウェアが実現した場合、この量子ビットを複数利用して、量子コンピュータは古典コンピュータでは実現し得ない規模の並列コンピューティングが実現する。理論上、現在の最速スーパーコンピュータで数千年かかるかも解けないような計算でも、例えば数十秒といった短い時間でこなすことができる、とされている。

\*22 出典: IT用語辞典

\*23 画像出典: Wikipedia



# 第6章

## アルゴリズムとは

アルゴリズムとは、「計算や作業を遂行するための手順」のことです。良いプログラムを作る上で、アルゴリズムの理解は欠かせません。有名なアルゴリズムの紹介とともに、その表現に良く用いられる「流れ図」や「効率性」の考察も行います。

### 6.1 アルゴリズムとは

語源は、九世紀前半を生きたアラビアの数学者アル・フワーリズミーに因みます。日本語では「算法」と訳されます。

IT用語辞典<sup>\*1</sup>には、次のように説明されています。

アルゴリズムとは、「ある特定の問題を解く手順を、単純な計算や操作の組み合わせとして明確に定義したもの」。数学の解法や計算手順なども含まれるが、ITの分野ではコンピュータにプログラムの形で与えて実行させができるよう定式化された、処理手順の集合のことを指すことが多い。

曖昧さのない単純で明確な手順の組み合わせとして記述された一連の手続きで、必ず有限回の操作で終了し、解を求めるか、解が得られないことが示される。コンピュータで実行する場合は、基礎的な演算、値の比較、条件分岐、手順の繰り返しなどを指示する命令を組み合わせたプログラムとして実装される。

数値などの列を大きい順または小さい順に並べ替える「整列アルゴリズム」、たくさんのデータの中から目的のものを探し出す「探索アルゴリズム」、データが表す情報を損なわずに短いデータに変換する「圧縮アルゴリズム」といった基本的なものから、画像の中に含まれる人間の顔を検出する、といった複雑なものまで様々な種類のアルゴリズムがある。

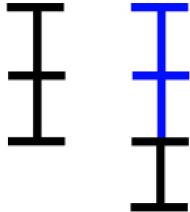
同じ問題を解くアルゴリズムが複数存在することもあり、必要な計算回数や記憶領域の大きさ、手順のシンプルさ、解の精度などがそれぞれに異なり、目的に応じて使い分けられる。例えば、ある同じ問題に対して、原理が単純で簡単にプログラムを記述できるが性能は低いアルゴリズム、計算手順が少なく高速に実行できるが膨大な記憶領域を必要とするアルゴリズム、厳密な解を求めるものより何桁も高速に近似解を求めることができるアルゴリズムなどがある。

巻末の参考文献「アルゴリズム図鑑」には、並び替えや探索など基本的なものから、暗号化やデータ

\*1 <https://e-words.jp/w/アルゴリズム.html>

圧縮まで様々なアルゴリズムが分かりやすく図解されていますので、是非、ご一読下さい。

### ♣ ユークリッドの互除法



紀元前300年頃に記されたユークリッドの『原論』に記されている世界最古のアルゴリズムである。2つの自然数(1071と1029)の最大公約数を求める例を挙げる。1071を1029で割った余りは42。1029を42で割った余りは21。42を21で割った余りは0。よって、最大公約数は21である。そして簡単に求めることができる。それぞれの自然数を素因数分解しても最大公約数を求めることができるが、ユークリッドの互除法を用いると格段に速く計算可能である。

(「ラメの定理」により、割って余りを取る操作を、最悪でも小さい十進数の桁数の約5倍繰り返せば、最大公約数に達する。)

### ♣ エラトステネスの篩

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

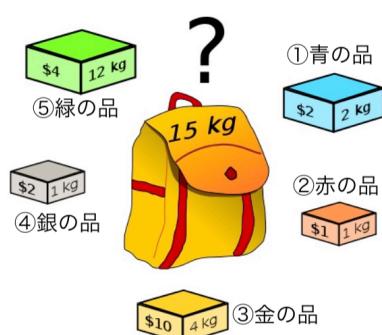
古代ギリシアの科学者エラトステネスが考案した、指定された整数以下の全ての素数を見つけるための単純なアルゴリズム。100までの素数を見つける場合、図のように2から100までの数を篩に入れておく。次に篩の中で最小の数2は素数の為、その倍数を篩い落とす、残った数の中で最小の数3は素数の為、その倍数を篩い落とす。これを順に繰り返していく、 $\sqrt{100}$ より大きな最初の素数に達した時点で、篩に残っていた全ての数は素数である。また、地球の大きさを最初に測定したことでも知られている。

### ♣ ハノイの塔



パズルの一種。三本の杭と、大きさの異なる複数の円盤があり、左端の杭に小さいものが上になるように順に積み重ねられている。円盤を一回に一枚ずつどれかの杭に移動させることができが、小さな円盤の上に大きな円盤を乗せることはできない。右端の杭に全ての円盤を移動させよ。

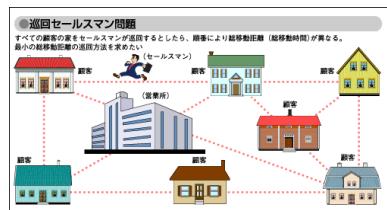
### ♣ ナップサック問題



「袋には15kgまで入れられる。重量や価値の異なる様々な品があるが、袋の価値を最大化するためには、どの品を詰めたら良いか?」という問題で、計算複雑性理論に於て、NP困難と呼ばれる問題のクラスに属する。動的計画法を用いた擬多項式時間アルゴリズムにより、実用的にはほぼ最適な解が得られる。

貪欲法と呼ばれるアルゴリズムでの解答例を示す。「貪欲」という名前の通り、各荷物の評価値(=価格÷重量)を算出し、評価値の高いものからナップサックに入していくのが骨子である。①の品は価値が2ドル重量が2kgであるから評価値は $2/2=1$ 、②の品は・・・と順に算出していくと、①～④の品を選んだときに、価値が15ドル重量が8kgであると(最適解ではないにしろ一応の)解が得られる。

### ♣ 巡回セールスマン問題 \*2



ある地域の営業担当のセールスマンが、その地域に住んでいる顧客全員の家を巡回して訪ねることになったとする。巡回の順番によって移動距離が大きく変わるので、移動距離が少なくなるような巡回の方法を求める。

## 6.2 データ構造

プログラミング言語 Pascal の開発者 ニクラウス・ヴィルト氏による名言、「プログラミング」 = 「データ構造」 + 「アルゴリズム」は、広く知られています。アルゴリズムに密接な関係があるデータ構造についても、IT用語辞典<sup>\*3</sup>より、ご紹介いたします。

### ♣ データ構造とは

データ構造とは、データの集まりをコンピュータプログラムで扱いやすいように、一定の形式で格納したもの。特定の問題を解く手順(アルゴリズム)には、それぞれに適したデータ構造がある。

複数のデータの配置や関係性、データの参照や出し入れなどの操作のルールを定義したもので、様々な種類がある。それぞれに特徴や適した処理があるため、同じ処理の記述でも、目的に対して適切なデータ構造を選択できるかどうかでプログラムの複雑さや処理性能に大きな差がつくことがある。

### ♣ データ構造の種類

最も基本的なデータ構造には、要素を一列に並べた「配列」(array)、要素を格納した順に取り出すことができる「キュー」(queue)、格納したのとは逆順に取り出すことができる「スタック」(stack)などがある。

格納された要素への参照データを含むデータ構造もあり、任意の標識と要素を一对一に関連付けて格納する「連想配列」(辞書、ハッシュ、マップとも呼ばれる)、要素が前後の要素への参照を持つ「連結リスト」(linked list)、要素が任意個の他の要素への参照を持つ「グラフ」(graph)、一つの頂点から樹状に枝分かれしたグラフである「木構造」(ツリー構造)などがある。

### ♣ 言語上の扱い

プログラミング言語では、基本的なデータ構造のいくつかが言語仕様や標準ライブラリなどにあらかじめ組み込まれて提供されていることが多い。用意されていない場合でも、既存のデータ構造や複合データ型、クラスなどの仕様を利用して開発者がデータ構造の定義や挙動の実装を行うことがある。

\*1 巡回セールスマン問題は imidas より、他はウィキペディアより引用・改変

\*3 <https://e-words.jp/w/%E3%82%B7%E3%82%A7%E3%82%A4%E3%82%A8%E3%82%A6%E3%82%A9.html>

実際のプログラム上では基本的なデータ構造を単体で用いることが多いが、あるデータ構造の要素として別のデータ構造を格納するなど、アルゴリズムに合わせて組み合わせて用いる場合もある。

## ♣ 各データ構造のイメージ

データ構造のイメージをご紹介します。図の出典は巻末の参考文献「アルゴリズム図鑑」(グラフについてはウィキペディア)です。

### 配列

もっとも基本的なデータ構造です。いくつかの要素をまとめたもので、個々の要素にアクセスする際には、 $a[0]$ 、 $a[1]$ 、 $a[2]$ と添字と呼ばれる番号を用いてアクセスできます。どの要素にも添字ですぐにアクセスできる長所の反面、途中への要素の追加や削除は苦手です。

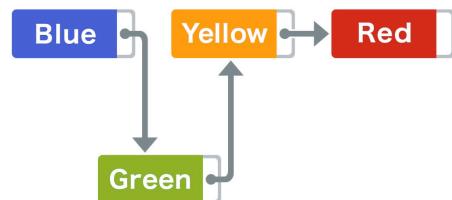
とある学級で出席番号で呼ぶような雰囲気です。



### リスト

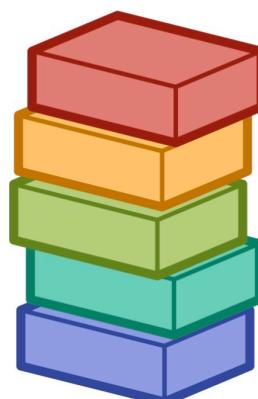
連結リストとも呼ばれます。個々の要素が、次の要素への参照を保持している点が特徴です。リストの途中に新しい要素「緑」を追加したい際には、「青」の参照先を「緑」に変更するのみで良く、要素の追加や削除が容易であるとの利点があります。反面、どの要素があるかは、先頭から順に辿ることとなるため、時間がかかることとなります。

とある学級で一列に生徒が並んでいるとき、生徒は自分の名前とその次の生徒の名前を知っているような状態です。



### スタック (積ん読)

いつか読みたいと思っている本を机の上に積み上げておく事はないでしょうか。「積ん読」と呼ばれる行為です。スタックもこれと同様で、後からデータが来るたびに上に積み上げ、データを取り出す際も上にあるもの(後から来たもの)から順に取り出す構造です。後入れ先出し、LIFOです。



### キュー (待ち行列)

買い物などの待ち行列のイメージで捉えられるデータ構造です。先に来たデータを先に処理していきます。先入れ先出し、FIFO です。



### 連想配列

辞書を引くと「Joe」の項目の説明文には「M」と書かれているように見えるところから「辞書」、「Joe」というデータを適宜「切り刻んで(ハッシュ)」「M」にするところから「ハッシュ」、「Joe」を「M」に「対応付ける(マッピングする)」ところから「マップ」とも呼ばれるデータ構造です。

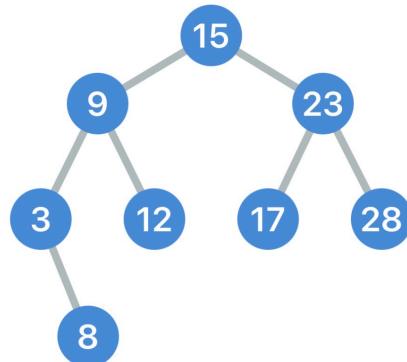
Key	Value
Joe	M
Sue	F
Dan	M
Nell	F
Ally	F
Bob	M

### 木構造

図の上下をひっくり返すと、丁度、木が空に伸びて行くように見えるので、「木」、「木構造」と呼ばれるデータ構造です。

「木」に準えて、根元にある「15」は「根」、「9」や「23」などは「節」、先端の「8」や「12」などは「葉」と呼ばれます。また「3」から「8」などのつながりは「枝」と呼ばれます。「根」の「15」から「葉」の「8」までは、三本の「枝」があるので、木の「高さ」(または深さ)は「3」です。

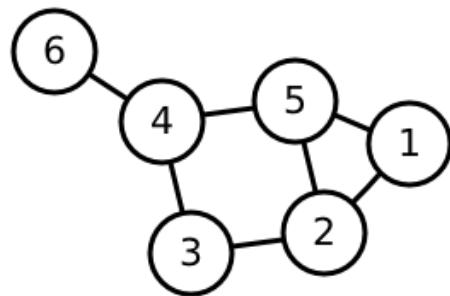
図は、「二分木」と呼ばれる「木」で、データの探索や並び替えなどに良く用いられます。



**グラフ**

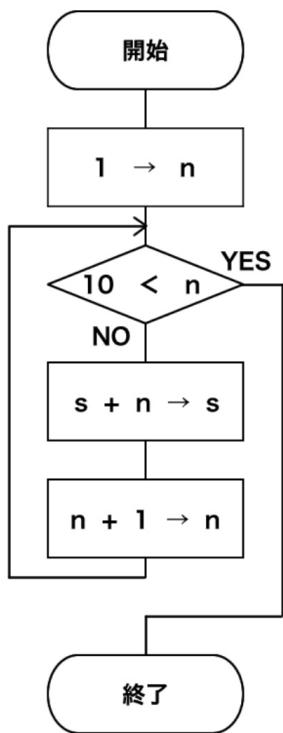
「木」と似ていますが、「枝」が繋がって「閉路」を為す点が異なります。

「1」番の駅から「6」番の駅に行くには、どの経路を辿ったら良いかなど、様々に利用されています。



## 6.3 アルゴリズムと効率性

ここでは、アルゴリズムの例として、1から10までの合計を求めます。左図は「流れ図(フローチャート)」と呼ばれ、手順を示す際に良く使われます。



1. 「n」という名前の箱を用意し、1を入れます。
2. 10よりnが大きいか、条件判断します。
3. NOなら、「s」という名前の箱に、 $s + n$ を計算した答えを入れて、「n」という名前の箱に、 $n + 1$ を計算した答えを入れます。そして、2に戻ります。
4. YESなら、「終了」です。1から10までの合計が、「s」という名前の箱に入っています。

計算するための手順が「きちん」と書かれているので、確実に答えを求めることができます。

それではこのアルゴリズムの効率をみていきましょう。1から10までの合計を求める時の計算量(足し算の回数)は10回になります。1から100までの合計を求める時の計算量(足し算の回数)は100回になります。1から1000までの合計を求める時の計算量(足し算の回数)は1000回になります。

このように、nに比例して計算量が増えることを、 $O(n)$ と書き、オーダーnと読みます。

より良いアルゴリズムとして、次のように求めることも出来ます。

$$(1 + n) * (n / 2) \rightarrow s$$

計算量は $O(1)$ と、nによらず、一定の時間で計算することができ、最高に効率的です。

解きたい課題に対して複数のアルゴリズムがあります。計算速度や実装の容易さなどを思案し、良いプログラムを書いていきましょう。

## 6.4 素数を求めるプログラム

### ♣ 手計算で行う

アルゴリズムのご紹介で、素数を求める為のアルゴリズム「エラトステネスの篩」をご紹介いたしました。

少しアルゴリズムは異なりますが、素数を求めて見ましょう。素数は、1と自分自身以外では割り切れない数のことです。最初の偶数2は素数ですが、次の偶数4以降は2で割り切れますので、奇数を対象に調べて行くことにしましょう。

例えば、21が素数かどうかは、21以下の素数(3, 5, 7, 11, 13, 17, 19)で割り切れるかどうか、計算することによって調べることが出来ます。

$21 \div 3 = 7$  余り 0 なので 割り切れる  
なので、21は素数ではない。

あるいは、23が素数かどうかは、23以下の素数(3, 5, 7, 11, 13, 17, 19)で割り切れるかどうか、計算することによって調べることが出来ます。

$23 \div 3 = 7$  余り 2 なので 割り切れない  
 $23 \div 5 = 4$  余り 3 なので 割り切れない  
 $23 \div 7 = 3$  余り 2 なので 割り切れない  
 $23 \div 11 = 2$  余り 1 なので 割り切れない  
 $23 \div 13 = 1$  余り 10 なので 割り切れない  
 $23 \div 17 = 1$  余り 6 なので 割り切れない  
 $23 \div 19 = 1$  余り 4 なので 割り切れない  
 なので、23は素数である。  
 新しい素数が見つかったので、素数達に追加する。

25が素数かどうかは、25以下の素数(3, 5, 7, 11, 13, 17, 19, 23)で割り切れるかどうか、計算することによって調べることが出来ます。

$25 \div 3 = 8$  余り 1 なので 割り切れない  
 $25 \div 5 = 5$  余り 0 なので 割り切れる  
 なので、25は素数ではない。

と順に調べていくことで、100までの素数は、2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97と求められます。

この手順を順に繰り返せば、1,000までの素数、10,000までの素数とどんどん素数を見つけて行くことができます。

## ♣ プログラムを書いて見る

手作業で行うには少し労力がかかりますので、プログラムを書いて見ましょう。様々なプログラミング言語がありますが、ここでは優れた書き味で人気の プログラミング言語 Ruby で、書いてみましょう。

次のようなコードになります。

### ▼ 素数を求める Ruby のプログラム

```

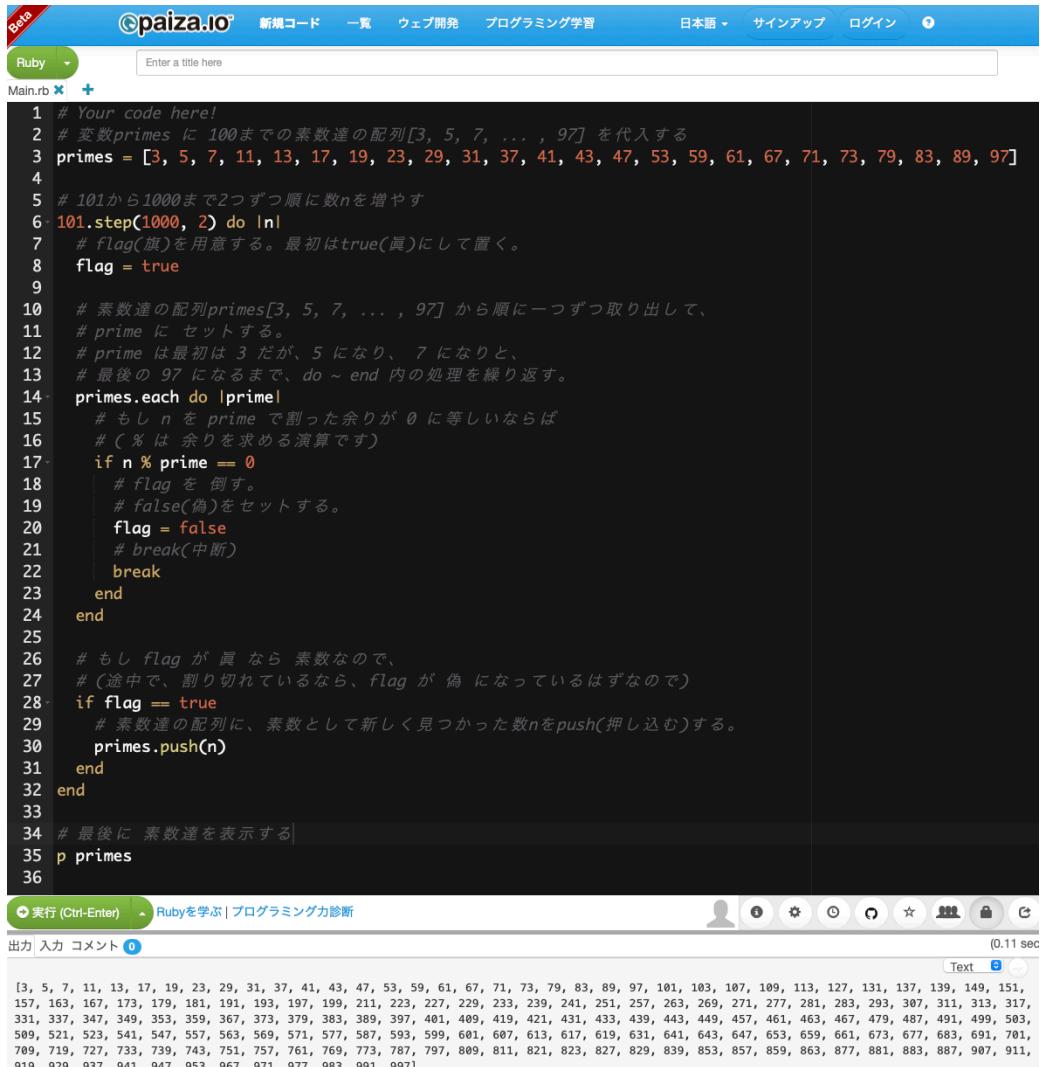
1 # 変数primes に 100までの素数達の配列[3, 5, 7, ... , 97] を代入する
2 primes = [3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73,
3   , 79, 83, 89, 97]
4
5 # 101から1000まで2つずつ順に数nを増やす
6 101.step(1000, 2) do |n|
7   # flag(旗)を用意する。最初はtrue(真)にして置く。
8   flag = true
9
10  # 素数達の配列primes[3, 5, 7, ... , 97] から順に一つずつ取り出して、
11  # prime にセットする。
12  # prime は最初は 3 だが、5 になり、7 になると、
13  # 最後の 97 になるまで、do ~ end 内の処理を繰り返す。
14  primes.each do |prime|
15    # もし n を prime で割った余りが 0 に等しいならば
16    # ( % は 余りを求める演算です)
17    if n % prime == 0
18      # flag を 倒す。
19      # false(偽)をセットする。
20      flag = false
21      # break(中断)
22      break
23    end
24  end
25
26  # もし flag が 真 なら 素数なので、
27  # (途中で、割り切れているなら、flag が 偽 になっているはずなので)
28  if flag == true
29    # 素数達の配列に、素数として新しく見つかった数nをpush(押し込む)する。
30    primes.push(n)
31  end
32
33 # 最後に 素数達を表示する
34 p primes

```

Ruby では、# の後ろに、コメント（説明書き）を書くことが出来ます。豊富に説明書きを入れておきましたので、英語に近いものとして自然に読みといて行けるかと思いますが、いかがでしょうか。

お手元のコンピュータに Ruby をインストールして、動かすことも出来ますし、よりお手軽に、[paiza.io](https://paiza.io/)\*4 より、動作を確かめることも出来ます。

\*4 <https://paiza.io/projects/UqC6dww3tK5wEl94SKYPeg>



The screenshot shows the paiza.io web interface for Ruby. The code editor contains the following Ruby script:

```

1 # Your code here!
2 # 変数primesに100までの素数達の配列[3, 5, 7, ..., 97]を代入する
3 primes = [3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
4
5 # 101から1000まで2つずつ順に数nを増やす
6 101.step(1000, 2) do |n|
7   # flag(旗)を用意する。最初はtrue(真)にして置く。
8   flag = true
9
10  # 素数達の配列[3, 5, 7, ..., 97]から順に一つずつ取り出して、
11  # primeにセットする。
12  # primeは最初は3だが、5になり、7になると、
13  # 最後の97になるまで、do ~ end内の処理を繰り返す。
14  primes.each do |prime|
15    #もし n を prime で割った余りが 0 に等しいならば
16    # (% は余りを求める演算です)
17    if n % prime == 0
18      # flag を倒す。
19      # false(偽)をセットする。
20      flag = false
21      # break(中断)
22      break
23    end
24  end
25
26  #もし flag が 真なら 素数なので、
27  # (途中で、割り切れているなら、flag が 傷 なっているはずなので)
28  if flag == true
29    # 素数達の配列に、素数として新しく見つかった数nをpush(押し込む)する。
30    primes.push(n)
31  end
32 end
33
34 # 最後に 素数達を表示する
35 p primes
36

```

The output window below the code editor shows the generated prime numbers from 3 to 997.

[3, 5, 7, ..., 977, 983, 991, 997]と求めた素数達も画面下に表示されています。手計算で1000までの素数を求めるのは少し手間ですし、間違いも起きやすいのですが、わずか14行のプログラムで求めることができました。

### ♣ 素数定理 - いくつの素数が存在するか推測する

最初の `101.step(1000, 2) do |n|` と書かれているところで、1000を10000に変えると、10000までの素数も求めることができますので、やって見ましょう。すぐに結果が出るかと思います。

もう一桁増やして100000までの素数はいかがでしょうか。Rubyでは大きな数を入力するときに、`10_0000`と、`_`(アンダースコア・下線)を使って見やすくすることもできます。お手元のコンピュータで動かしている方はだいぶ待ってから結果が表示されたことだと思いますし、またpaiza.ioで動かしている方は結果が表示されなかったかと思います。

数が大きくなると、素数の数も増えるだろうと想像はつきますが、一体どれくらいの割合で素数は現

れるのでしょうか。素数定理<sup>\*5</sup>という、素数の出現率に関する定理があります。 $10^n$ までの素数の数を $f(10^n)$ とすると、(若干の誤差はあります)次の公式によって求めることができます。

$$f(10^n) = \frac{10^n}{n} \log_{10} e \doteq \frac{10^n}{n} \times 0.4343$$

$n = 3$ として、計算してみます。

$$f(10^3) = \frac{10^3}{3} \times 0.4343 = 144.6219$$

となりました。1000までの実際の素数の数は167個ですから、144.6219個という推測結果は、少し誤差はありますが、おおよその数を求めることが出来ています。素数の分布はとても興味深い話題で、素数定理を紐解く<sup>\*6</sup>により詳しく説明がございます。読みやすい文章ですので、是非ごらんになってください。

### ♣ より効率的なアルゴリズムに改良する

そこで少しプログラムの素数を求めるアルゴリズム（計算手法）を見直して見ましょう。

#### 1000までの素数

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997

最後に発見した997が素数かどうかを調べる手順を振り返って見ましょう。

997 ÷ 3 = 332 余り 1 ので 割り切れない  
 997 ÷ 5 = 249 余り 2 ので 割り切れない  
 997 ÷ 7 = 142 余り 3 ので 割り切れない  
 (略)  
 997 ÷ 977 = 1 余り 20 ので 割り切れない  
 997 ÷ 983 = 1 余り 14 ので 割り切れない  
 997 ÷ 991 = 1 余り 6 ので 割り切れない

と、166回の余りを求める演算（剩余演算）を行いましたが、もう少し演算回数を減らすことは出来ないものでしょうか。

\*5 後の大数学者ガウス少年は15歳の時に発見したそうです。最初の証明は死後40年を経て与えられました。

\*6 <https://www.shokabo.co.jp/column-math/column-math0004.html>

最初の 3 や 5、7 で割り切れるか確認するのは仕方ないとしても、最後の 983 や 991 で割り切れるか調べるのは無駄な気がします。

例えば  $969 = 57 \times 17$  ですので、969 が素数か合成数かを調べる為には、57 で割る以前に 17 で割ることで検出されているはずです。

$969 \div 17 = 57$  余り 0 なので 割り切れるので合成数  
 $969 \div 57 = 17$  余り 0 なので 割り切れるので合成数

つまり、逆に言えば、17 以下の素数で割り切れるか否かを調べることで、 $17^2 = 289$  以下の素数を、57 以下の素数で割り切れるか否かを調べることで、 $57^2 = 3249$  以下の素数を調べることができます。

$37^2 = 1369$  なので、1000 以下の素数を求める為には、37 以下の素数で割り切れるか調べれば良いので、166 回必要だった剰余演算の回数を 11 回まで、劇的に減らすことが出来ます。

#### ▼ 改良版のプログラム

```

1 # 変数primes に 100までの素数達の配列[3, 5, 7, ..., 97] を代入する
2 primes = [3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73,
3   , 79, 83, 89, 97]
4
5 # 101から1000まで2つずつ順に数nを増やす
6 101.step(1000, 2) do |n|
7   # flag(旗)を用意する。最初はtrue(眞)にして置く。
8   flag = true
9
10  # 素数達の配列primes[3, 5, 7, ..., 97] から順に一つずつ取り出して、
11  # prime にセットする。
12  # prime は最初は 3 だが、5 になり、7 になりと、
13  # 最後の 97 になるまで、do ~ end 内の処理を繰り返す。
14  primes.each do |prime|
15    # divmod は、割り算の商と余りを同時に求める演算(メソッド)です。
16    # n を prime で割った商をqに、余りをrに代入します。
17    q, r = n.divmod(prime)
18    # もし 余り r が 0 に等しいならば
19    if r == 0
20      # flag を 倒す。
21      # false(偽)をセットする。
22      flag = false
23      # break(中断)
24      break
25    end
26
27    # もし q < prime となっており、
28    # その時にまだ flag が眞のままならば
29    #  $971 \div 37 = 26$  余り 9 なので 割り切れず、flag は 真のままである。
30    # つまり素数なので、
31    if q < prime && flag == true
32      # 素数達の配列に、素数として新しく見つかった数nをpush(押し込む)する。
33      primes.push(n)
34      # 素数として発見できたので、
35      # もう他の素数で割って検証する必要もない。
36      # なので、break(中断)して、
37      # 次の素数候補の検証に移ることとする。
38      break

```

```

39 end
40 end
41
42 # 最後に 素数達を表示する
43 p primes

```

お手元のコンピュータで実行しても良いですし、ブラウザから [paiza.io](#)<sup>\*7</sup> で確認することも出来ます。

効率的に素数判定できるようになったので、1\_0000, 10\_0000 と増やしてみましょう。速く結果が返ってくるようになりました。

### ♣ 実行速度の表

各人、さまざまな計算機環境かとは思いますが、ご参考までに筆者の手元で計算した結果を掲載いたします。

素数を求める実行時間

いくつまでの 素数を求めるか	初期版の処理時間 単位: 秒	改良版の処理時間 単位: 秒	性能比 単位: 倍
1000	0.0007889999978	0.0007849999964	1.0051
1_0000	0.0311270000092	0.0057969999907	5.3695
10_0000	1.7517559999978	0.0777740000048	22.5237
100_0000	112.8859519999969	1.2790039999963	88.2608
1000_0000	(計測せず)	25.1513839999970	
1_0000_0000	(計測せず)	546.1275220000098	

アルゴリズムを工夫することで、多くの素数を知ることが出来るようになりました。

127 や 8191 は素数であり、これらの積 1040257 も容易に計算することができます。逆に素因数分解することは順に素数で割って見るより他無く、大変に時間がかかります。この性質を利用したのが RSA 暗号で、公開鍵暗号方式として、安全にインターネット通信が行われる基盤として、広く使われている技術です。

より良いアルゴリズムを求めて今日でも様々な研究開発が行われています。ご自身の解決したい課題に適切なアルゴリズムを工夫されるのも良いですし、多くの先人達が優れた技法を考案されておりまして、その叡智を拝借するのも良いでしょう。また、もしかすると、ご自身の研究開発されたアルゴリズムが、未来の人類への貢献になるやもしれません。

アルゴリズムの世界を楽しみ、探求する最初の一歩になれば幸いです。

\*7 [https://paiza.io/projects/jhzHstHwnmnx3vNgAa-\\_IQ](https://paiza.io/projects/jhzHstHwnmnx3vNgAa-_IQ)

## 【コラム】カール・フリードリヒ・ガウス

後に数学王と呼ばれたガウス少年の逸話をウィキペディアより紹介します。

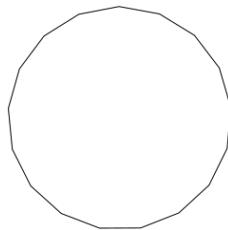


ガウスが7歳の時、数学の授業で教師が「1から100までの数字すべてを足せ」という問題を出した。教師は生徒たちが問題を解くには相当な時間がかかるだろうと考えていたが、ガウスはわずか数秒で「5050」という解答を出し、教師を驚かせた。

1から100までの数字を足すと、 $1 + 100 = 101$ 、 $2 + 99 = 101$ 、…、 $50 + 51 = 101$ で、101の集まりが50個できるため、 $101 \times 50 = 5050$ になるとガウスは計算しました。

19歳の時にコンパスと定規のみで正十七角形を作図できることを証明したガウスは、自身の墓碑に正十七角形を刻むよう遺言した逸話も知られています。<sup>\*8</sup>

その後もガウスは、素数定理や最小自乗法など赫々たる業績を挙げ、数学界に巨大な足跡を記しました。



正十七角形

1777年 - ドイツ、ブラウンシュヴァイクに生まれる

1792年 - 素数定理の成立を予想

1795年 - 最小二乗法発見

1796年 - 平方剩余の相互法則の証明。

コンパスと定規のみで正十七角形を作図できることを証明

1799年 - 代数学の基本定理の証明

1801年 - 『整数論の研究』出版 複素数表記、現代整数の表記導入

1801年 - 円周等分多項式の研究

1807年 - ゲッティンゲンの天文台長になり、以後40年同職につく

1809年 - 『天体運行論』出版 最小二乗法を用いたデータ補正、正規分布

1811年 - 複素積分、ガウス平面（複素数平面）ベッセルへの手紙

1827年 - 『曲面の研究』出版、微分幾何学を創始

1855年 - ゲッティンゲンで死去

<sup>\*3</sup> 作図可能な正多角形 (<http://hooktail.sub.jp/algebra/ConstructablePolygons/>)



---

# 第 7 章

## プログラムとは

---

### 7.1 「プログラム」 = 「コンピュータへの指示書」

今日の「コンピュータ」 = 「電子計算機」が登場する以前から「プログラム」という言葉は使われてきました。語源は「前に書かれたもの」で、街頭で大勢に示す文書「公文書」の意味です。

アルゴリズムとは、ある特定の課題への解法、解き方の手順を書き表したものでしたが、このアルゴリズムを、コンピュータが計算できるよう、コンピュータが理解できる言語で書き表したもの、それが「プログラム」です。「プログラム」 = 「コンピュータへの指示書」です。

その昔のコンピュータは、特定の用途の計算のみを行うことができました。（専用計算機）黎明期のコンピュータは、回路と回路の間の配線を繋ぎ代えることで、異なる種類の計算も行うことができるようになりましたが、たくさんの配線を繋ぎ代えることはとても大変です。そのため生まれたアイディアが、「計算機への指令」 = 「プログラム」そのものを計算機に内蔵するというアイディアです。これにより、様々な手順を必要なときに読み込んで書かれた通りの計算を行うことができるようになりました。これが、現在ほとんどのコンピュータで広く採用されている「プログラム内蔵方式」 = 「ノイマン・アーキテクチャ」です。

また、シャノン等によりスイッチのオンとオフの回路（スイッチング回路）から、論理回路・デジタル回路への道が拓かれたことや、チューリングによるチューリングマシンの研究、19世紀を生きた数学者であるジョージ・ブールの仕事、ブール代数も、今日のコンピュータの存在に大きく貢献しています。

コンピュータのプログラムは、3つの要素で成り立ちます。

- 順次処理：上から下へ順番に進みます。
- 条件判断：YES か NO で答えられる質問です。
- 繰り返し：何回でも反復します。

これらを組み合わせて、自らの意図する機能を創り上げていきます。

## 7.2 プログラム・プログラミングの定義

より正確な定義を **IT用語辞典<sup>\*1</sup>** で確認してみましょう。

### プログラム

プログラムとは、予定（表）、計画（表）、課程、式次第などの意味を持つ英単語。ITの分野では、コンピュータに行わせる処理を記述したコンピュータプログラムのことを略して単にプログラムといふことが多い。

### コンピュータプログラム

コンピュータが行うべき処理を順序立てて記述したもの。広義の「ソフトウェア」の一部であるが、実用上はプログラムとソフトウェアはほとんど同義のように扱われることが多い。

現代のコンピュータではプログラムは一定の形式に従ってデータとして表現され、記憶装置（メインメモリ）に格納される。実行時にはCPU（中央処理装置）がプログラムに記述された命令を順番に読み出して解釈・実行していく。

プログラムを作成する作業や工程を「プログラミング」、これを行う人や職種のことを「プログラマ」という。人間がプログラムを記述する際には、人間が理解しやすい人工言語である「プログラミング言語」を使うことが多い。プログラミング言語で記述されたプログラムを「ソースコード」という。

ソースコードはコンピュータが解釈・実行することができないため、コンパイラなどの変換ソフトによってコンピュータが解釈・実行できる機械語などで構成された「オブジェクトコード」に変換されてから実行される。スクリプト言語のように、この変換処理を開発時には行わず、実行時にインタプリタなどのソフトウェアによって動的に行う場合もある。

### プログラミング

プログラミングとは、コンピュータに意図した動作を行わせるために、まとまった処理手順を作成し、与えること。作成された手順のことをコンピュータプログラムあるいは単にプログラムという。プログラミングを行う人や職種のことをプログラマという。狭義には、プログラミング言語やそれに相当する仕組みや道具を用いて、人間が読み書きしやすい形式のプログラム（ソースコード）を記述していくコーディング作業を指す。広義には、その前後に行われる、設計や試験（テスト）、修正（デバッグ）、実行可能形式への変換（コンパイルやビルドなど）といった一連の作業を含む。

### プログラムの作成

プログラミングを行うには、まず何をするプログラムを作るのかを明確に定義し、仕様や要件を自然言語で記述したり、大まかな処理の流れを箇条書きやフローチャートなどの図表を用いて設計する。集団でソフトウェア開発を行う場合はプログラムの記述者とは別の設計者が専門に作業を行い、仕様書や設計書などの形でまとめる場合もあるが、個人が小規模のプログラムを作成する場合はこの工程を頭の中で行い、作業や手順としては省略する場合もある。

どんなプログラムを作りたいか決まったら、これをコンピュータが解釈できるプログラミング言語を用いてソースコードとして記述していく。言語やプログラムの記述法には様々な種類があるが、手続き型の言語（手続き型プログラミング）の場合、実行すべき命令を先頭から順に書き下していく。必要に応じて、複数の命令をひとまとめにして名前をつけて呼び出せるようにしたり（関

<sup>\*1</sup> <https://e-words.jp/w/プログラム.html>

数やサブルーチンなど)、条件分岐や反復(繰り返し)などで命令の流れの制御を行う。

### プログラムの実行

ソースコードそのものはコンピュータ(の処理装置)が解釈・実行できる形式ではないため、これを機械語(マシン語)のプログラムなど実行可能な形式に変換する必要がある。ソースコードを機械語などのコード(オブジェクトコード)に変換する工程をコンパイルと呼び、プログラムの起動処理やライブラリなど実行に必要なコードを連結する工程をリンクという。これら一連の工程を行って実行可能ファイルやパッケージを作ることをビルドという。

スクリプト言語の場合はこうした明示的な変換工程は不要で、ソースコードを機械語に変換しながら同時に実行するインタプリタなどの処理系で直に実行することができる。記述したコードをすぐ実行でき手軽だが、変換しながら実行するため実行可能ファイルを生成する場合より実行速度やメモリ効率では劣る。

### プログラムの修正(デバッグ)

作成したプログラムが一度で完全に思い描いたとおりに動作する場合もあるが、大抵は何らかの誤りや不具合を抱えているものである。このため、ビルドしたプログラムを実行してみてテスト(動作試験)を行い、仕様通りに動くか調べる。

誤り(バグ)が発見されると原因や解決策を考え、正しく動作するようにプログラムを書き換える(デバッグ)。バグには単純な記述ミスのようなものから、そもそも解くべき問題に対して選択した計算手順(アルゴリズム)が合っていないといった根本的なレベルのものまで様々な種類がある。

デバッグ作業が完了したら再びビルドとテストを行い、誤りが正されていることを確認する。このビルド→テスト→デバッグの繰り返しによって次第にプログラムの完成度や品質が上がっていき、実際に実用可能なプログラムに仕上げることができる。実際のプログラミングにおいては作業時間の多くがこの繰り返しの工程に費やされる。

## 7.3 プログラミング言語の種類

人間がコンピュータに指示するために作られた言語が、プログラミング言語です。用途に応じ様々なプログラミング言語が考案されてきました。そのいくつかをご紹介します。

### ♣ 機械語(左)とアセンブリ言語(右)

18	CLC
F8	SED
A9 34 12	LDA #\$1234
69 21 43	ADC #\$4321
8F 03 7F 01	STA \$017F03
D8	CLD
E2 30	SEP #\$30
00	BRK

機械語は、「0」と「1」の二進数や「0」-「F」までの十六進数を書き連ねたものです。コンピュータはこれを読み取り直接解釈、実行可能な命令データの集まりです。

アセンブリ言語は、コンピュータが直接解釈実行可能な機械語を、人間にわかりやすいよう英略語を用いて書き表した言語です。

左側の機械語は分かりにくいですが、右側のアセンブリ言語は、CLC: 比較せよ、LDA: 読み込み、ADC: 足せ、STA: 書き込み、BRK: 中断など、人に理解しやすくなっています。

## ♣ C 言語

```
int n;
int answer = 0;
for(n = 1; n <= 10; n++){
    answer += n;
}
printf("%d\n", answer);
```

### 1から10までの合計を求める

略してC(シー)ともいいます。比較的人に読みやすい文法を持ちながら、機械に近いところまで書き表すことができるので、小さなコンピュータ(マイコン)が組み込まれた電子機器のプログラミングや、UNIX(ユニックス)などOS(オペレーティングシステム)の開発などに広く用いられています。

## ♣ Ruby(ルビー)

```
answer = 0
(1..10).each do |n|
    answer += n
end
print answer
```

### 1から10までの合計を求める

まつもとゆきひろさん(通称 Matz)により開発されたオブジェクト指向スクリプト言語です。日本で開発されたプログラミング言語として初めて国際電気標準会議で国際規格に認証されました。

開発者のまつもとゆきひろさんは、「Rubyの言語仕様策定において最も重視しているのはストレスなくプログラミングを楽しむことである(enjoy programming)」と述べています。

## ♣ Scratch(スクラッチ)



### 1から10までの合計を求める

MITメディアラボが開発した視覚的なプログラミング言語で、初めてプログラミングをする小学生が、遊び心ある実験やアニメ、ゲームなどを作ったりすることができます。

小学生でもRubyを使ってプログラムやロボットを作れるようにした、Smalrubyもお薦めです。<sup>a</sup>

<sup>a</sup> NPO 法人 Ruby プログラミング少年団 (<https://smalruby.jp>)

## 7.4 学校教育でのプログラミング

小学校からプログラミング教育が始まりました。概要を文部科学省より引用します。

## ♣ なぜプログラミング教育を導入するのか

今日、コンピュータは人々の生活の様々な場面で活用されています。家電や自動車をはじめ身近なものの中にもコンピュータが内蔵され、人々の生活を便利で豊かにしています。誰にとっても、職業生活をはじめ、学校での学習や生涯学習、家庭生活や余暇生活など、コンピュータなどの情報機器やサービスとそれによって齎される情報を適切に選択・活用し問題を解決していくことが不可欠な社会が到来しつつあります。

コンピュータをより適切、効果的に活用していくためには、その仕組みを知ることが重要です。コンピュータは人が命令を与えることによって動作します。端的に言えば、この命令が「プログラム」であり、命令を与えることが「プログラミング」です。プログラミングによって、コンピュータ自分が求め

る動作をさせることができるとともに、コンピュータの仕組みの一端をうかがい知ることできるので、コンピュータが「魔法の箱ではなくなり、より主体的に活用することにつながります。

プログラミング教育は子供たちの可能性を広げることにも繋ります。プログラミング能力を開花させ、創造力を發揮して、起業する若者や特許を取得する子供も現れています。秘めた可能性を発掘し、社会で活躍できるきっかけとなることも期待できるのです。

このように、コンピュータを理解し上手に活用していく力を身に付けることは、あらゆる活動においてコンピュータ等の活用が求められるこれからの中学生を生きていく子供たちにとって、将来どのような職業に就くとしても、極めて重要なこととなっています。<sup>\*2</sup>

### ♣ 小学校でのプログラミング教育のねらい

1. 「プログラミング的思考」を育むこと。
2. プログラムの働きや良さ、情報社会がコンピュータ等の情報技術によって支えられていることに気付き、コンピュータを活用して身近な問題の解決や、よりよい社会を築く態度を育むこと。
3. 各教科内容を指導する際に行う場合は、各教科で学びをより確実なものとすること。

小学校においては、文字入力など基本的な操作を習得、プログラミングを体験しながらコンピュータに意図した処理を行わせるために必要な論理的思考力を身に付ける。

### ♣ 中学校でのプログラミング教育のねらい

中学校においては、技術・家庭科においてプログラミング、情報セキュリティに関する内容を充実「計測・制御のプログラミング」に加え、「ネットワークを利用した双方向性のあるコンテンツのプログラミング」等について学ぶ。

### ♣ 高等学校でのプログラミング教育のねらい

高等学校においては、情報科において共通必履修科目「情報 I」を新設し、全ての生徒がプログラミングのほか、ネットワーク（情報セキュリティを含む）やデータベースの基礎等について学習「情報 I」に加え、選択科目「情報 II」を開設。「情報 I」において培った基礎の上に、情報システムや多様なデータを適切かつ効果的に活用し、あるいはコンテンツを創造する力を育成。<sup>\*3</sup>

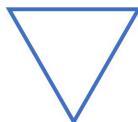
### ♣ プログラミング的思考とは

「自分が意図する一連の活動を実現するために、どのような動きの組合せが必要であり、一つ一つの動きに対応した記号を、どのように組み合わせたらいいか、記号の組合せをどのように改善していくべき、より意図した活動に近づくのか」を論理的に考えていく力。

<sup>\*2</sup> 出典：文部科学省：小学校プログラミング教育の手引（第三版）

<sup>\*3</sup> 出典：文部科学省 新学習指導要領のポイント（情報活用能力の育成・ICT活用）

### ♣ 「正多角形を描く」場合について考える



コンピュータで正三角形を描く場合を見てみます。「正三角形を描く」という命令は通常は用意されていないので、そのままでは実行できません。そこで、コンピュータが理解できる(用意されている)命令を組み合わせて命令することを考えます。紙の上に作図する場合、正多角形がもっている「辺の長さが全て等しい」、「角の大きさが全て等しい」、「円に内接する」、「中心角の大きさが全て等しい」のような正多角形の意味や性質を使って作図します。

コンピュータで作図する場合にも同様に考えます。ここでは「辺の長さと、角の大きさが全て等しい」という正多角形の意味を使い作図する例を考えます。

この場合、「長さ 100 進む(線を引く)」、「左に 120 度曲がる」といったコンピュータが理解できる(用意されている)命令を組み合わせることで「正三角形を描け」ます。

より大きな正三角形を描きたければ、「長さ 200 進む(線を引く)」というように修正します。曲がる角度を変えることで、正六角形や正八角形も描くことができます。

紙の上に鉛筆と定規、分度器やコンパス等を用いて正三角形を描く際も、用いる性質や手順は異なるとしても、児童は同じように手順を考えた上で作図しているはずです。

### ♣ 未来の学びコンソーシアムより 実践例の紹介



## 7.5 大学入試試験 情報

大学入試センター<sup>\*4</sup>には、令和7年度からの大学入試試験で出題される予定となる「情報」に関する試験問題の例が掲載されています。

**問3** 次の文章の空欄 [ク] ~ [コ]に入れるのに最も適当なものを、それぞれの解答群のうちから一つずつ選べ。

次の図1は、モノクロの画像を16画素モノクロ8階調のデジタルデータに変換する手順を図したものである。このとき、手順2では[ク]、このことを[ケ]化といふ。手順1から3のような方法でデジタル化された画像データは、[コ]などのメリットがある。

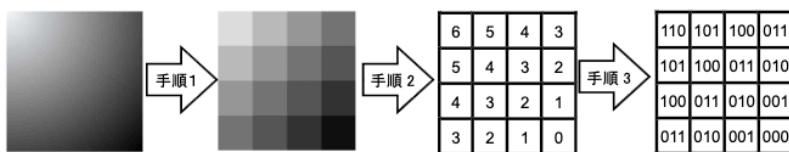


図1 画像をデジタルデータに変換する手順

[ク] の解答群

- ① 区画の濃淡を一定の規則に従って整数値に置き換えており
- ② 画像を等間隔の格子状の区画に分割しており
- ③ 整数値を二進法で表現しており
- ④ しきい値を基準に白と黒の2階調に変換しており

[ケ] の解答群

- ① 符号
- ② 量子
- ③ 標本
- ④ 二値

[コ] の解答群

- ① コピーを繰り返したり、伝送したりしても画質が劣化しない
- ② ディスプレイ上で拡大してもギザギザが現れない
- ③ データを圧縮した際、圧縮方式に関係なく完全に元の画像に戻すことができる
- ④ 著作権を気にすることなくコピーして多くの人に配布することができる

デジタル化についての理解度を問う出題で、解答群も用意されていますので、解答は容易で、正解は次のようになります。

\*4 <https://www.dnc.ac.jp/>

## 正解

次の図は、モノクロの画像を16画素モノクロ8階調のデジタルデータに変換する手順を図したものである。この時、手順2では「ク」区画の濃淡を一定の規則に従って整数値に置き換えており、このことを「ケ）量子化」という。手順1から3のような方法でデジタル化された画像データは、「コ）コピーを繰り返したり、伝送したりしても画質が劣化しない」などのメリットがある。

ちなみに、手順1は「標本化（サンプリング）」、手順2は「量子化」、手順3は「符号化」です。  
他の例題は、[例題\\*5](#)をご覧ください。

これらの例題の狙いとして、次のように案内されています。

高等学校学習指導要領「情報I」で、

1. 情報社会の問題解決
2. コミュニケーションと情報デザイン
3. コンピュータとプログラミング
4. 情報通信ネットワークとデータの活用

を学習することを踏まえて、

- 情報技術の仕組みとその利点、情報社会と人の関わりやその課題に関連する理解を問う。
- 発表の場において伝えたい情報を分かりやすく表現する情報デザインの考え方や方法を理解し表現する力を問う。
- 画像のデジタル化に関する一連の流れと、デジタル化のメリットについての理解を問う。
- IPv4におけるネットワーク部を表すビット数を題材に、生徒が主体的に学習し探究する場面を設定して、IPアドレスの理解と基數変換の考え方を基に考察する力を問う。
- 比例代表選挙の議席配分の考え方をプログラムで処理するなど、情報社会の問題解決の過程を題材に、生徒が主体的に学習し探究する場面を設定し、配列、最大値探索、繰り返し処理を用いたアルゴリズムを理解し、そのアルゴリズムをプログラムで表現し、さらに具体的な状況設定に応じてプログラムを修正することを通して問題解決に向けて考察する力を問う。(なお、問題の中で使用するプログラム言語は、大学入試センター独自の日本語表記の疑似言語としている。これは、高等学校の授業で何らかのプログラム言語を用いて実習した生徒であれば容易に理解できるものである。)
- オープンデータを用いて、基本統計量などから全体の傾向を読み取ったり、予測したりする問題解決の活動の中で、データの活用に関する考察する力を問う。
- 基本統計量を読み取り、データに含まれる傾向を見いだし、さらに、データの散らばりから傾向を読み取るなど、実践的なデータの活用及び分析に関する基本的な理解と考察する力を問う。

概ねITパスポート以上基本情報技術者未満の出題です。より詳しくは、[狙い\\*6](#)をご覧ください。

\*5 [https://www.dnc.ac.jp/albums/abm.php?d=33&f=abm00000307.pdf&n=%E3%82%A4%E3%82%BF%E3%82%BF%E3%83%88%E3%83%BC%E3%83%89%EF%BC%88%E6%8A%A5%E6%9C%8B%EF%BC%89\\_%E9%96%93%E9%96%93.pdf](https://www.dnc.ac.jp/albums/abm.php?d=33&f=abm00000307.pdf&n=%E3%82%A4%E3%82%BF%E3%82%BF%E3%83%88%E3%83%BC%E3%83%89%EF%BC%88%E6%8A%A5%E6%9C%8B%EF%BC%89_%E9%96%93%E9%96%93.pdf)

\*6 [https://www.dnc.ac.jp/albums/abm.php?d=33&f=abm00000300.pdf&n=%E3%82%A4%E3%82%BF%E3%82%BF%E3%83%88%E3%83%BC%E3%83%89%EF%BC%88%E6%8A%A5%E6%9C%8B%EF%BC%89\\_%E6%9D%A1%E3%82%8B.pdf](https://www.dnc.ac.jp/albums/abm.php?d=33&f=abm00000300.pdf&n=%E3%82%A4%E3%82%BF%E3%82%BF%E3%83%88%E3%83%BC%E3%83%89%EF%BC%88%E6%8A%A5%E6%9C%8B%EF%BC%89_%E6%9D%A1%E3%82%8B.pdf)

# 第8章

## ウェブの歴史と技術

今日では不可欠となったウェブサイト。その歴史をみていきます。そしてサーバとブラウザとの関係、サイト作成に用いる HTML / CSS / JavaScript の概要を紹介します。

### 8.1 ウェブサイトの発祥

HTML でウェブページを作成し、情報を発信するシステムは、平成 2 年に欧洲原子核研究機構(CERN) のティム・バーナーズ・リー氏により提案されました。「どのようなコンピュータからでも情報を見覽し、共用できる」ことが目的で、WWW(World Wide Web) と呼ばれます。

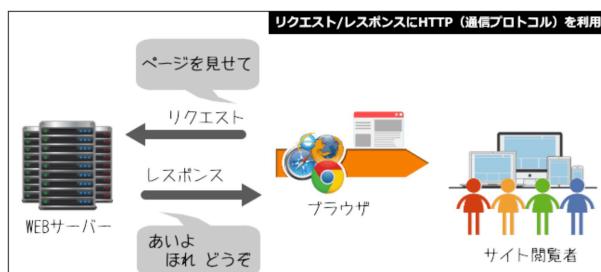
WWW は、研究機関や大学を中心に開発が進み、平成 6 年にオープンな形となりました。同時に米国立スーパーコンピュータ応用研究所(NCSA) により提供された Mosaic ブラウザにより、多くの研究者が使い始めます。やがて Netscape(Firefox の前身) ブラウザの誕生と Windows 95 により、誰もが使える環境が整い、爆発的に普及しました。<sup>\*1</sup>

#### ♣ サーバとブラウザ

サーバとは、利用者側のコンピュータに対し、ネットワークを通じて情報や機能を提供するコンピュータおよびソフトウェアのことです。

ウェブサーバは、利用者のコンピュータから操作されるブラウザからの要求に応え、自身の管理するデータを送信します。より具体的には HTML ファイルや画像ファイルなどウェブページを構成するファイルを送信します。

このやり取りには HTTP (HyperText Transfer Protocol) と呼ばれる通信規約(プロトコル)が用いられるので、わたくしたちがウェブサイトを閲覧する際には、「<https://>」から始まる URL を、ブラウザのアドレスバーに入力します。<sup>\*2</sup>



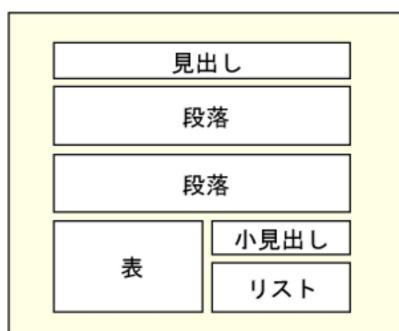
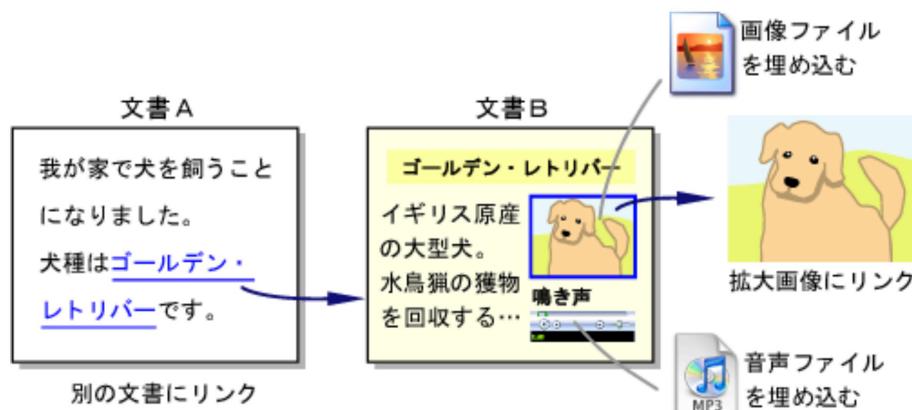
\*1 出典: CSS グリッドで作る HTML5&CSS3 レッスンブック

\*2 画像出典: <https://www.nkshopping.biz/index.php?レスポンスとは。説明文: IT 用語辞典より改変>

## 8.2 HTML とは

HTML(HyperText Markup Language) は、ウェブページを作成するために開発された言語で、HyperText Markup Language を日本語で表すなら、「ハイパーテキストに目印をつける言語」くらいの意味になります。ハイパーテキスト (HyperText) とは、ハイパーリンクを埋め込むことができる高機能な文書です。ハイパーリンクというのは、ウェブページで下線の付いたテキストなどをクリックすると別ページへ移動する、あのリンクのことです。

ハイパーテキストでは、ウェブページから別のウェブページにリンクを貼ったり、ウェブページ内に画像・動画・音声などのデータファイルをリンクで埋め込むことができます。HTMLには、このハイパーリンク機能で関連する情報同士を結びつけて、情報を整理するという特徴があります。



また、目印をつける (Markup) というのは、文書の各部分が、どのような役割を持っているのかを示すということです。例えば、見出し・段落・表・リストなど、文書の中で各部分が果たしている役割が分かるように目印をつけます。こうした見出し・段落・表・リストなどの文書内の各部分を要素と呼びます。文書内の各部分に目印をつけ、その部分がどんな要素かを明確にすることで、コンピュータがその文書の構造を理解できるようになります。

具体的には、検索エンジンがウェブページの構造を把握して解析したり、ブラウザがウェブページ内の各要素の意味を理解して閲覧しやすいように表示することなどが可能になります。

このようにコンピュータに理解できるように文書の構造を定義することこそが、HTML の最も重要な役割と言えるでしょう。この際、目印をつけるための記号として使用されるのが HTML タグです。

\*3

\*3 出典：HTML クイックリファレンス

## 8.3 CSS とは

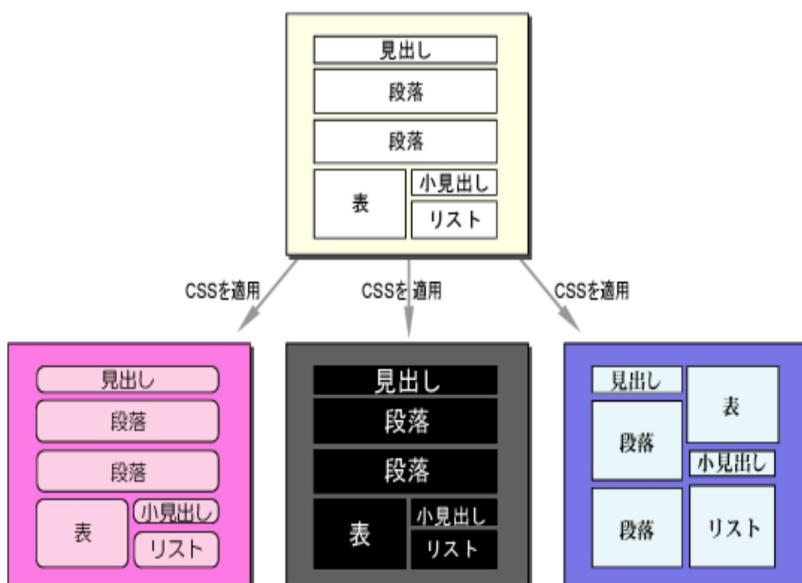
CSS(Cascading Style Sheet) とは、Web ページの要素の配置や見栄えなどを記述するための言語で、指定できる項目は、要素の大きさや配置、要素間の位置関係や空白、要素の境界線や余白、要素間の空白や周囲の余白、文字の大きさや文字や行の間隔、書体（フォント）の種類や変形（太字や斜体など）、箇条書きの表示書式、背景色や背景画像など多岐に渡ります。

HTML タグが親子関係（包含関係）にある場合、多くの設定値は親要素に指定されたものが子要素、孫要素に引き継がれ、子要素で指定されたものが追加されていきます。このように設定値が上から下へ伝播していく様子を、階段状の瀧を意味する **cascade**（カスケード）になぞらえて、CSS(Cascading Style Sheet) という名称になりました。<sup>\*4</sup>

### ◆ 文書構造とスタイル指定とを分離する

HTML に文章構造を記述し、CSS に見栄えに関する記述を行うと、対象機器（パソコンや携帯電話など）に応じた CSS を適用することで、それぞれに適した表示が行えます。

また、共通する見栄えの部分を他のページにも適用することで、効率的に統一感のあるウェブサイトを作成できるようになります。<sup>\*5</sup>



▲ 図 8.1: CSS で見た目を切り替える

\*4 出典：IT 用語辞典

\*5 出典：HTML クイックリファレンス

## 8.4

# JavaScriptとは

JavaScriptは主にブラウザで動くプログラミング言語です。

### ♣ 主な特徴

C言語やJavaに似た記法や文法を採用した手続き型の言語で、オブジェクト指向にも対応しています。関数を変数のように(第一級オブジェクトとして)扱ったり、関数を引数に取る高階関数を定義できるなど、関数型プログラミング言語の仕様も取り込んでいます。

### ♣ ブラウザでの使われ方

JavaScriptは、HTMLファイルからJavaScriptが書かれたファイルを読み込む形で良く使われます。ブラウザによってページが表示される際に、ページ内の写真などの要素に動きや効果を加えたり、閲覧者の操作に反応して何らかの処理を行ったりする為に用います。

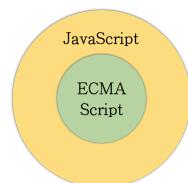
例えば、ウェブサイトで何か操作をしたら表示が書き換わったり、ウェブサイトのサーバと通信してデータを取得するなど、現在のウェブサイトには欠かせないプログラミング言語となっています。

### ♣ 他の実行環境

当初は、ブラウザ上で実行されるプログラミング言語でしたが、Node.jsというサーバ側のアプリケーションを作る仕組みでも利用されています。また、デスクトップアプリやスマートフォンアプリ、IoT(Internet of Things)デバイスでもJavaScriptを使って動かせるものが現れるなど、幅広い環境で動いているプログラミング言語となっています。

### ♣ 標準規格

JavaScriptという言語はECMAScriptという仕様によって動作が決められています。ECMAScript仕様には、基本的な記法や文法、式、宣言、関数、変数、データ型、コンテナ、組み込みオブジェクト、例外処理などの仕様を定めています。近年の拡張ではクラスやモジュール、非同期処理についても盛り込まれました。実行環境に左右されない汎用的な言語のコア部分についての規格です。



これに、WebブラウザにおけるDOM操作や、特定のプロトコルによるネットワーク通信、ローカル環境でのファイル入出力と言った個別具体的な機能のAPIなどを追加したものが、JavaScriptです。

### ♣ 歴史

JavaScriptは1995年にネットスケープ・コミュニケーションズ(Netscape Communications)社のブレンダン・アイク(Brendan Eich)氏により開発されました。当時最も人気の高いウェブブラウザだったNetscape Navigator 2.0に実装されたのが、その始まりです。当初は「LiveScript」という名称でしたが、同社がJava言語の開発元のサン・マイクロシステムズ(Sun Microsystems)社と提携していたことから、JavaScriptに改称されました。

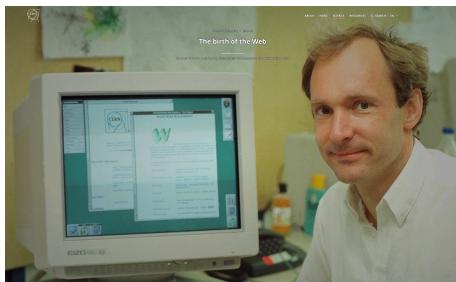
名称にJavaと付いてはいますが、記法や予約語などの一部が共通するのみで、Java言語との直接的な繋がりや互換性はありません。<sup>\*6</sup>

<sup>\*6</sup> 出典：IT用語辞典、JavaScript Primer 迷わないための入門書

## 【コラム】ウェブの誕生

欧州原子核研究機構 (CERN) は、スイスのジュネーヴ郊外に位置する世界最大規模の素粒子物理学の研究所である。地下には 全周 27km の円形加速器・大型ハドロン衝突型加速器が、設置されており、加速器を用いた素粒子物理学および原子核物理学の研究のほか、研究に必要な有用な技術の開発などを行っている。ウェブ発祥の地としても有名である。

### ティム・バーナーズ=リー



ティム・バーナーズ=リーは、同僚のロバート・カイリューとともに World Wide Web (WWW) を考案した。そして、文献の検索および連携のために考案された言語である HTML、インターネット通信のために定めた規則 (HyperText Transfer Protocol 通称 : HTTP)、ハイパーテキストシステムを実装・開発した。<sup>\*8</sup>

イギリスの科学者 ティム・バーナーズ=リーは、CERN 在籍していた時に World Wide Web (WWW) を発明した。世界中の大学や研究機関の科学者間で、情報を自動で共用したいという要望に応えるために考案され、開発された。<sup>\*7</sup>

### World Wide Web

The WorldWideWeb (W3) is a wide-area [hypermedia](#) information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an [executive summary](#) of the project, [Mailing lists](#) , [Policy](#) , November's [W3 news](#) , [Frequently Asked Questions](#) .

[What's out there?](#)  
Pointers to the world's online information, [subjects](#) , [W3 servers](#) , etc.  
[Help](#)  
on the browser you are using  
[Software Products](#)  
A list of W3 project components and their current state. (e.g. [Line Mode](#) , [X11 Viola](#) , [NeXTStep](#) , [Servers](#) , [Tools](#) , [Mail robot](#) , [Library](#))  
[Technicals](#)  
Details of protocols, formats, program internals etc  
[Bibliography](#)  
Paper documentation on W3 and references.  
[People](#)  
A list of some people involved in the project.  
[History](#)  
A summary of the history of the project.  
[How can I help ?](#)  
If you would like to support the web..  
[Getting code](#)  
Getting the code by [anonymous FTP](#) , etc.



右上は、初めて公開されたウェブサイト。文字のみから成るものであったが、今日と同じく、URLとハイパーリンクの仕組みにより情報共有が出来た。

右は、公開に用いられたウェブサーバである NeXTcube。平成二年のクリスマスに世界初のウェブが誕生した。

\*7 <https://home.web.cern.ch/science/computing/birth-web>

\*8 画像と文章はウィキペディアより引用・改変

## 【コラム】人工知能 AI

「人工知能 AI」という言葉を良く聞きます。Artificial Intelligence の略で、人にしかできなかった知的な行為（認識、推論、言語運用、創造など）を、どのような手順（アルゴリズム）とどのようなデータ（事前情報や知識）を準備すれば、それを機械的に実行できるか」を研究する学問です。（情報工学者・通信工学者 佐藤理史 氏）

人の知的能力を計算機上で実現する、様々な技術・ソフトウェア・コンピュータシステムであり、応用例として、自然言語処理（機械翻訳・かな漢字変換・構文解析・文章要約等）、専門家の推論・判断を模倣するエキスパートシステム、画像データを解析し特定のパターンを検出・抽出する画像認識等があります。昭和31年にダートマス会議でジョン・マッカーシーにより命名されました。現在では、記号処理を用いた知能の記述を主体とする情報処理や研究でのアプローチという意味合いでも使われ、また家庭用電気器具の制御システムやゲームの思考ルーチンもこう呼ばれます。

画像処理における深層学習（ディープラーニング）の有用性が競技会で世界的に認知された平成24年頃から急速に研究が活発となり、第三次人工知能ブームが到来しました。平成28年には、深層学習を導入したAIが完全情報ゲームである囲碁や将棋などのトップ棋士を破るなど、最先端技術として注目されています。

音楽分野においては、既存の曲を学習することで、特定の作曲家の作風を真似て作曲する自動作曲ソフトが登場しています。絵画分野においては、コンセプトアート用背景やアニメーションの中割の自動生成、モノクロ漫画の自動彩色など、人間の作業を補助するAIが実現しています。

また、指示に応じて、あたかも人間のような文章を生成できるChatGPTや、それに合わせた画像を生成するStable Diffusion, DALL-E2, Midjourneyなど、幾多の課題を孕みつつも実用に供されています。

プログラミングの領域においても、数十億行のコードから学習されたGitHub Copilotや、Pulsarのプラグインとして紹介したtabnineもコード補完機能にAIを利用しています。

一方、「深層学習の父」と呼ばれているヨシュア・ベンジオは、中国共産党が市民の監視や政治目的で人工知能を利用していることに警鐘を鳴らしています。ヘルメットや帽子に埋め込んだセンサから、国民の脳波と感情を人工知能で監視するプロジェクトが推し進められ、ネット検閲や、官僚や囚人・歩行者に至るまで、監視カメラと警察のサングラス型スマートグラスやロボットに顔認識システム（天網）を搭載するなど、人工知能による監視社会・管理社会化が行われ、新疆ウイグル自治区では、人工知能により、約1万5千人の人々がテロリストや犯罪者の可能性があるとして、再教育キャンプに収容・拘禁されています。<sup>\*9</sup>

古くからの格言に「大いなる力には、大いなる責任が伴う」があります。技術を人の仕合せの為に用いる叡智が求められています。<sup>\*10</sup>

\*9 出典：ウィキペディア

\*10 社会への影響について 人工知能とバイアス (<http://www.is.nagoya-u.ac.jp/dep-ss/phil/kukita/works/AI-and-bias-20190717.pdf>) が示唆に富む考察をしています。

# 付録 A

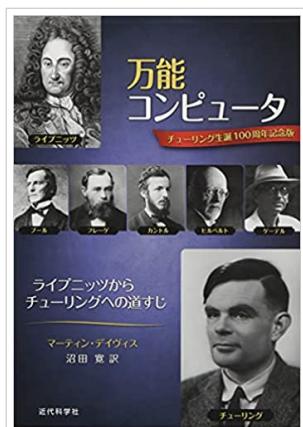
## 珠玉の名著のご紹介

様々な名著が出版されております。その中からもう少しコンピューターの歴史や仕組み、プログラミングに関して知りたいと思われた方へおすすめの書籍をご紹介いたします。手に取っていただきお読みいただければ幸いです。<sup>\*1</sup>

### A.1 コンピュータを作った人々を訪ねて

今日の情報社会を形作った偉人たちの足跡をたどります。知的好奇心を満たす読み物として、お楽しみください。

#### ♣ 万能コンピュータ：ライプニッツからチューリングへの道すじ

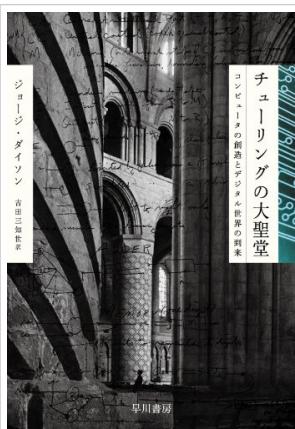


ライプニッツからチューリングに至る数理論理学の系譜は、コンピュータの理論的背景を形成し人工知能（AI）の登場までも予見している。代数の記号表現を通じて人間の思考の範囲すべてを包括するような記号体系の構築に献身した数理論理学者たちの苦闘を、時代背景を取り込みながら解説する。さらに本書を構成する7人の数理論理学者（ライプニッツ、布尔、フレーゲ、カントル、ヒルベルト、ゲーデル、チューリング）を、豊富なエピソードをもとにその人となりを描写する。

比較的平易に書かれているので、コンピュータロジックの成り立ちに関心のある高校生以上の読者や、人工知能（AI）の成り立ちに関心のある読者にも必携の書である。

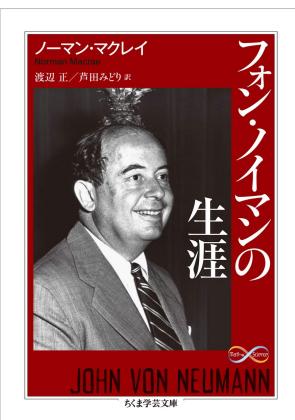
<sup>\*1</sup> 書籍紹介文から、引用・改変。

## ♣ チューリングの大聖堂



数学学者チューリングの構想した「チューリングマシン」。そしてそれを現実の装置として創りあげたのが万能の科学者フォン・ノイマンである。彼の実現した「プログラム内蔵型」コンピュータが数に関する概念を変え、デジタル宇宙を創生した。自由闊達なプリンストン高等研究所という舞台で、チューリングは何を考え、ノイマンはどう立ち回り、AINシュタインやゲーデルを擁した高等研究所はいかにしてその自由性を得るにいたったのか。そして彼らとともにコンピュータ開発を支えた科学者・技術者はいかにして関わりを持つようになり、現代に直結するどんな偉業を成し遂げたのか。高等研究所などに収められた詳細な文献や写真資料、豊富なインタビュー取材をもとに、歴史事情や知られざる人々の肖像から綴る、決定版コンピュータ「創世記」。

## ♣ フォン・ノイマンの生涯



その恐ろしいまでの底知れない知力で、悪魔とも火星人とも呼ばれ、AINシュタインをして、「人類最高の知性」と言わしめた科学者、ジョン・フォン・ノイマン。数学基礎論、計算機科学、ゲーム理論、数値気象学、そして核兵器…、その53年の生涯で残した150篇もの論文は後の科学と社会を基礎づけたと言っても過言ではない。本書はノイマンの足跡を関係者への丹念な取材と数多くの文献を通して明らかにし、「巨大な知性」誕生の背景と人となりを詳らかにする。決定版的ノイマン評伝。

## ♣ ノイマン・ゲーデル・チューリング



今日のコンピュータの礎を築いたジョン・フォン・ノイマン、不完全性定理で数学・論理学の歴史を根底から変えたクルト・ゲーデル、思考する機械への道を拓いたアラン・チューリング。いずれも今日の科学と哲学に多大な影響をもたらした天才たちである。同時代に生きた彼らは、互いに触発され、時に議論し、相互に意識しながら実に多くの業績を残した。比類なき頭脳と個性をもった三人は、いかに関わり、何を考え、どう生きたか。それは今日の世界にいかなる意味を持つのか。彼ら自身の言葉からその思想の本質に迫る。

## A.2

## 計算機科学に関する一般教養を身に付けたい方の為に

### ♣ 教養としてのコンピューターサイエンス講義



デジタル時代で活躍するための「教養」をこの1冊で身につけよう。

プリンストン大学の一般人向け「コンピューターサイエンス」の講義が一冊に。デジタル社会をよりよく生きるために知識を伝説の計算機科学者がやさしく伝えます。(著者ブライアン・カーニハン氏は、C言語の発明者です)

本書は、わたくしたちの世界(デジタル社会)が、どのように動いているのか、なぜそのしくみになっているのかをもっとも明快かつ簡潔に説明しています。

### ♣ みんなのコンピューターサイエンス

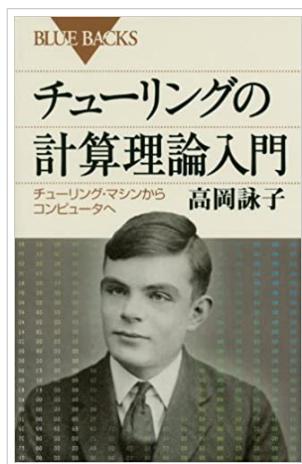


コンピュータなしには生活が立ち行かなくなる水準に達しつつある現代社会。その圧倒的な力を課題解決に援用するには小手先の知識では追いつきません。とは言え無闇に全方位に知識を求めるには、その世界は広すぎ、効率も悪すぎます。

本書は計算機科学が扱う「基礎」「効率」「戦略」「データ」「アルゴリズム」「データベース」「コンピュータ」「プログラミング」という8つのジャンルにしぶり、その精髄と背景となる考え方を紹介します。

ステップアップしたいエンジニアや、ライトに全体像を俯瞰したい学生にも最適な1冊です。

### ♣ チューリングの計算理論入門

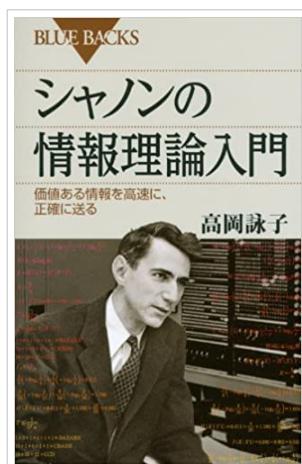


イギリスの数学学者チューリングは、ヒルベルトの「決定問題」解決のために、万能計算機の数学的モデル「チューリング・マシン」のアイディアに至った。この「チューリング・マシン」こそが、コンピュータの万能性を保証する数学的基礎になった。

チューリングは、「チューリング・マシン」を使って、計算という行為を徹底的に検証した。そして、手順を示すことと、計算ができることが同じであることを示した。その手順はアルゴリズムと呼ばれ、いまではソフトウェアと言われている。

本書は、コンピュータの原理としてのチューリング・マシンを解説とともに、決定問題を解決した有名な「チューリング・マシンの停止問題」も分かりやすく説明します。さらに計算量と、7大難問の一つ「P=NP 問題」についても、わかりやすく解説します。

### ♣ シャノンの情報理論入門



情報は、なぜデジタル化できるのだろうか？ 現代の巨大な情報社会を支える情報科学の基礎はシャノンによって作られた。形のない情報をどのように表現し、情報の価値をどのように表すのか？ シャノンの築いた情報理論を分かりやすく解説する。

情報とはなにか？ どのように量るのか？ 情報エントロピーとは？ 圧縮とはなにか？ 高校生でも分かる、シャノン情報理論の入門書

### ♣ プログラマの数学



プログラミングに役立つ「数学的な考え方」を身につけよう。

プログラミングや数学に関心のある読者を対象に、プログラミング上達に役立つ「数学の考え方」をわかりやすく解説しています。数学的な知識を前提とせず、たくさんの図とパズルを通して、平易な文章で解き明かしています。

二進数から人工知能に至るまで、ていねいに説明しています。

プログラミングや数学に関心のある読者はいうまでもなく、プログラミング初心者や数学の苦手な人にとっても最良の一冊です。

### ♣ 数学ガール



本書は、三人の高校生が数学の問題に挑戦する物語。題材は「素数」「絶対値」という基本的なものから「フィボナッチ数列」「二項定理」、「無限級数」や「テイラー展開」、「母関数」まで多岐にわたっています。

数学クイズが好きな一般の方から、理系の大学生、社会人まで幅広い読者に楽しんでもらえる数学物語です。数式が苦手でも大丈夫。登場する高校生自身も数式で悩み、ああでもない、こうでもないと読者と思いを共有します。数式が追えなくても「旅の地図」と称した概念図で読者さんの理解を助けます。「数学は、時を越える」をテーマにおいた本書は本格的な数学の奥深いおもしろみをすべての読者に提供するでしょう。

### ♣ アルゴリズム図鑑 絵で見てわかる33のアルゴリズム



基本的な33のアルゴリズム+7つのデータ構造をすべてイラストで解説。アルゴリズムはどんな言語でプログラムを書くにしても不可欠ですが、現場で教わることはめったになく、かといって自分で学ぶには難しいものです。

本書は、アルゴリズムを独学する人のために作りました。はじめて学ぶときにはイメージしやすく、復習するときには思い出しやすくなるよう、基本的な33のアルゴリズム+7つのデータ構造をすべてイラストで解説しています。

よいプログラムを書くために知っておかなければいけないアルゴリズムの世界を、楽しく学びましょう。

### ♣ キタミ式イラスト IT塾 ITパスポート



可愛いイラストでとてもわかりやすい解説を行っているため、ITパスポート試験にとって、まず大切な「解説書を一冊読み、用語や計算に慣れる」ことができる書籍です。

## A.3 プログラミングを始めたい方に

ご自身で始められたい方、また親子で楽しみたい方にお勧めの書籍です。

### ♣ 初めてのプログラミング 第2版

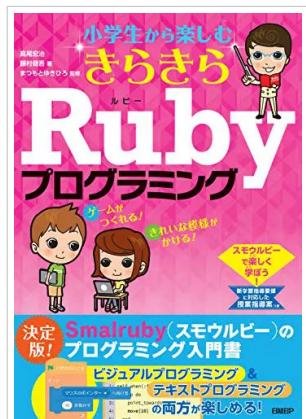


初めてプログラミングを学ぶ入門者を対象に、プログラミングの基礎をていねいに解説した書籍。

教材には、誰でもどんな環境でも気軽に使えるRubyを使い、実際に簡単なコードを書きながら理解を深めます。プログラミングとは何かを無理なく理解してもらうために、要点をひとつひとつていねいに解説。簡単な概念から始めて、かなり高度なプログラミングの知識まで身に付けられます。

プログラミングを学ぶ最初の一冊に最適な入門書です。

### ♣ 小学生から楽しむ きらきら Ruby プログラミング



スマウルピー解説書の決定版! ブロック(ビジュアル)プログラミング言語「Scratch」とテキストプログラミング言語「Ruby」の両方の特徴を持つ「Smalruby」を使ったプログラミング入門書です。Scratch同様に簡単にプログラミングを始められ、さらにテキスト言語への移行もスムーズに行えるよう、ブロックとテキストの両方でプログラムを書く方法を丁寧に解説。新学習指導要領に対応した、実際の授業でも使われている授業指導案も付属。

この本では小学校でするプログラミングの内容を、音楽、社会、算数、理科といった各教科に分けてできるようになっていて、プログラミングがはじめての人にも経験している人にもバッチリな内容。

### ♣ ルビィのぼうけん こんにちは！ プログラミング



親子で楽しくプログラミングに触れる絵本『ルビィは大きな想像力を持つ女の子。ルビィの好きな言葉は“どうして？”』。ルビィの世界では考えたものがなんでも実現します。ある日、ルビィはパパからの手紙を見つけます—「宝石を5つ、かくしたから、さがしてごらん。ぜんぶ見つけられるかな？」。でもどう探せばいいのか書かれていません。まずはヒントを探しあげると・・・秘密の数字が書かれた紙きれを発見！ ここからルビィのぼうけんが始まります』

プログラミングを、子どもたちが身近に感じ、楽しく学んでいける本を——との思いから生まれたのが「ルビィのぼうけん」です。

続編として「コンピューターの国のルビィ」「インターネットたんけん隊」「AIロボット、学校へいく」もあります。

# 終わりに

本書では、算盤から iPhone に至るまでの歴史を俯瞰、計算機科学の基礎知識に触れ、HTML / CSS / JavaScript によるウェブアプリを作成、公開いたしました。

全部で、108行のじゃんけんプログラム、要所要所にコメントも付けていますので、今まで学んできた知識で読解できるはずです。ぜひ、遊んでみてください。自分で作ったプログラムの体験はいかがでしょうか？いろいろ創意工夫して、さまざまなアプリを作っていけそうですね。

「福祉」。「福」「祉<sup>\*2</sup>」どちらも「めぐみ、さいわい」という意味を持ちます。

「熱き心、<sup>たくま</sup>逞<sup>かいな</sup>しき腕、冷静な頭脳」

学生時代に言われた言葉ですが、福祉を生きる者は、人としての熱い思い、暖かい心を持ち、その上で、冷静な判断力を以て、力強く行動するのだと。

「工学」の「工」は、「天の 理<sup>ことわり</sup>を、地に下ろす」意味です。

技術の産物としての社会ではなく、世界を 耀<sup>かがや</sup>かせるために技術を用いてください。技術に使われるのではなく、技術を使いこなし、人の道に役立てる人となってください。

令和の御世を生きる皆さんのが素晴らしい人生を生き、素晴らしい日本を創ることを願って筆を置きます。

いやさか  
彌榮

---

\*2 「祉い」と書いて、「さいわい」と読みます。天からの恵みがその身に止まる意味です。

# 計算機の歴史と働く仕組み

暮らしに生きるコンピュータ

---

令和五年五月五日

著 者 アトリエ未来

発行者 早乙女 遙香

連絡先 contact@atelier-mirai.net

<https://atelier-mirai.net>

---

© 令和五年 アトリエ未来