

素数の|C++でLife Game|電気学|3を $\sqrt{5}$ 進数
探し方|を実装してみた|の発展|で表す

APC POSITRON

APC ポジトロン 日本版

電子工作や、物理のお話など

総集編

四足歩行ロボット
3Dプリンター

超電導
&
音波
etc...

目次

		難易度	ボリューム
01	ゲームを作るうえで大切なこと	☆☆	☆☆
04	世界一小さな物語	☆☆☆☆☆	☆☆☆☆
09	プログラミング入門の言語は何がいいのか	☆☆	☆☆☆☆
13	生体認証とは？	☆☆	☆☆
18	コラム 動的計画法		
20	投擲 <small>とうてき</small> について	☆☆☆	☆
22	ゲームプログラミングと数学が出会う時	☆☆☆☆☆	☆☆
26	いったいどのくらい大きな声を出せば ブラジルの人に声が届くのか	☆☆☆	☆
28	3を $\sqrt{5}$ 進数で表すと？	☆☆☆☆☆	☆
30	ライフゲームを C++で実装してみた話	☆☆☆☆☆	☆☆
34	電気学の発展	☆☆	☆☆☆☆
37	コイルとコイルガン	☆☆☆	☆☆☆☆
40	Zen とプロセス微細化のお話	☆☆☆☆☆	☆☆☆☆
44	素数の探し方	☆☆☆☆☆	☆☆☆☆☆☆☆
57	電子工作や、物理のお話など総集編	☆	☆ × 11

ゲームプログラムと数学が出会うとき

吉田 行人

物理部中 2PC 班の吉田(行人)デス。我々 PC 班はプログラミングをして、ゲームやソフトを作っている訳ですが、僕たちがゲームを制作していく中で大事なことが幾つかあります。

ここでは、その大事なことを幾つか紹介したいと思います。

① ユーザーに分かりやすい画面

いわゆる UI(User Interface)ですね。この UI がしっかりとしないとユーザーはゲームやソフトの画面を理解できません(理解しにくいです)。この UI をしっかりとするための職業もあるぐらいです。昔は CUI ソフト(コマンドプロンプトを使ったソフト)がほとんどでしたが今ではほとんどのソフトが GUI ソフト(ユーザーに分かりやすいように画像などを使ったコンピュータの操作を容易にしたソフト)になっています。

これは今ではたくさん的人が PC を使うようになったのでユーザーに対して分かりやすいソフトの需要が急激に増えたことによると思われます。

なので我々物理部員も、この風に乗って画面をできるだけ見やすいようにゲームを作っています(~~気にしない人間もいますが~~)。

UI をしっかりとするために日常生活で実践できことがあります。それは—お母さんからよく言われるようなことですが—自分の伝えたいことを他人にしっかりと理解してもらえるように生活する事です。

これを普段から行えば、UI はしっかりとしたゲームとなります。

皆さんも、もっとユーザーがあっと驚く作品をいつか作りたいのであれば、上記のことを試してみては如何でしょうか?

② ゲームが、人をゲームに夢中にできる理由

前置きしておくと、ここでいう「ゲーム」とは堂々巡りですが「人を夢中にできるゲーム」の事です。ここでは、なぜ「人を夢中にできるゲーム」が人を夢中にできるのか、という事についてお話しします。

少し難しい話になりますが、人がゲームに夢中になる時、モチベーションの問題が大きく関わってきます。モチベーションについての研究で(僕が知っている範囲で)最も有名なのは、恐らく「マクレランドの欲求理論」だと思います。

マクレランドは、人が抱えている欲求を次のように分類しました。

○達成欲求:一定の目標に対して、達成し成功しようと努力する欲求

○権力欲求:他者に対して影響力を与え、コントロールしたいという欲求

○親和欲求:友好的で親密な対人関係を結びたいという欲求

○回避欲求:失敗や困難な状況を回避したいという欲求

この欲求を満たせば、基本的には人はゲームに夢中になれると言われています。

例えば、達成欲求だったら、ゲームをクリアするために主人公を強化する要素をゲーム内に入れたり、権力欲求だったら、主人公をプレイヤーによって不自由なくコントロールできるようにしたり、親和欲求だったら、ユーザーとユーザーが共にプレイできる要素を加えたりとか、回避欲求だったら、どんな技も回避出来たりカウンターできたりするようにする、等があげられます。

一つのゲームに夢中になっている人なら、上の4つの欲求をそのゲームが満たしていることが分かると思います。少し考えてみてくださいね。

この事について、日常生活で気を付けなければならることは特にならないと思います。

強いて言うのであれば、周りの人の人間観察をすることです。これをすれば「あっ、あの人今達成欲求が強くなっているな」みたいなことです。

※周りの人の迷惑にならないようにやりましょう(あと、普通の人から見たら変人に見えてしまうので度を越えないようにしましょう)。

③ 日常生活の1つ1つのモーション

最後の3つ目です。私たち人間は脳から電気信号を送り、それを元に体を動かしています。

この事は、それなりに有名なので知っている人もいると思います。人間が行っているモーションは全てプログラムで表せると言っても過言ではありません。要するに、私たちは四六時中プログラムを実行していると言っても(厳密には少し違いますが)過言ではないということです!

日頃の自分の腕の動きを想像してみてください、あるいは、今そこで自分の腕を動かしてみてください。

例えば腕を90°曲げるには、自分の肘を中心として、三角関数を使い座標の値を色々いじつてたりして、ようやく曲げることが出来ます(三角関数などについては理解していないなくてもいいです。ここでは腕の動きをプログラムで表せることを分かってくれればいいです)。

少し難しいですが、体の動きのほとんどは(何度も言う通り)プログラムで表せます。ただ、日常生活でいちいちそんな事を考えていたら埒が明きません。

そこであげられる物が一つ。物事を論理的に考えてみましょう。

熱いコーヒーを飲むことは、

「息で冷ます→カップを手に取る→カップを自分の口辺りに近づける→カップを傾け自分の口にコーヒーが入るようにする(→香りと味を嗜む)→飲み込む(→息をつく)→カップを元の場所或いは自分の近くに置く」

という一連の動作によって成り立ちます。これでご理解いただけたでしょうか?このように、一つの動作を論理的に表すことによって、ゲームプログラミングがある程度やりやすくな

ります。皆さんも、日常生活で慣れれば楽にできることなので、試してみては如何でしょうか？

終わりに

この3つの事を実践することで(2つ目はやってもやらなくてもいいですが)かなりゲームプログラミングがやりやすくなり、またプログラミングをこれからやるという方もタメになると 思いますので、この文章を読んだその日から、是非実行してみてください！

世界一小さな物語

高二 檸檬色のトラウマ@自称数学研究会の永久機関

注意:このレポートは筆者の力量不足で非常に読みづらい上、中身も一般的に難解とされています。クオリティとしてはこれを読んでもせいぜい内容が分かった気になれるくらいのものです。

それでもかまわない方はこのまま読み進めてください。

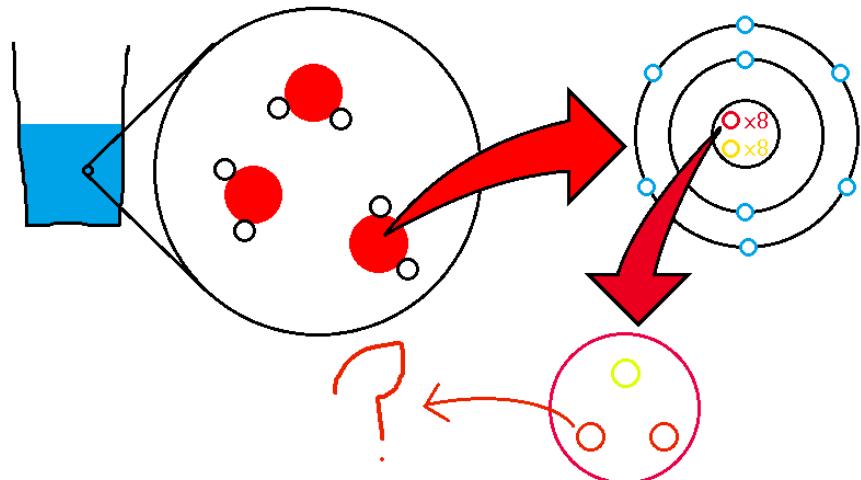
・序章

今手元にコップか何かに入った水があるとしよう。その水は H_2O 分子の集まりである。さらに各分子は酸素原子と水素原子二つに分けられる。また、その原子も陽子と(水素原子の大半以外は) 中性子からなる核と電子に分割できる。物理的な意味で、物質は我々の手でここまで分解できるのである。だが、実際のところ陽子と中性子はさらに細かい部品からなっていることが分かっている。もっとも、この部品レベルに陽子と中性子を分解することは不可能ではあるが。要するに下図のとおりである原子の中心は陽子と中性子 8 つだが、描くのが面倒なので略されている。

こういうサイズになると、物質は直観では想像できないような振る舞いをする。たとえば、何かの拍子に壁をすり抜けられたり、実際に粒子がどこにあるのかは知ることができなかつたりとかする

わけだ。こういうのを見ると寒気がするタイプの人も多いと思うが、もちろん私は一目ぼれである。だからこそ知ったかぶり然とした内容でもこれを書きたくなってしまうのである。

というわけで、これから私は小さな「別世界の」話をしようと思う。



・原子、そしてその欠片

原子の存在に行き着く経緯については、「ブラウン運動」とか、「ティッシュに撃ち込んだ弾丸が跳ね返ってきた」とか、語りたいことが腐るほどあるが、残念ながらそんな余白など与えられそうにないので割と大事なその二つについて軽く語ってあとは検索してもらう、という方法をとることとした。

ブラウン運動は水の入ったビーカーに微粒子を入れると「無作為に」運動する現象だが、それを説明したのは実はアインシュタインだったりする。実はこれ、水分子が微粒子に衝突して起こっているというのである。実際に原子を見せるなんて真似ができなかつたことから、実はアインシュタインによりこれが説明されるまでは原子説は実証されていない、というわけで20世紀までは原子説は今ほど広まっていなかつたのだ。彼はこの論文を出した年に他にもいくつか論文を出している。特殊相対性理論とか $E=mc^2$ とか光量子仮説とか、とにかくすごい論文の数々については、また「余白」の都合でカットである。申し訳ない。

電子は元々真空中に電気を通していたところ何か負の電荷をもつた粒子が中を通っているということで発見され、その電荷を帶電した油滴の電荷の最大公約数を計算して求めたりもされている。後にこれが原子由来のものと判明するわけである。

「ティッシュに撃ち込んだ弾丸が跳ね返ってきた」とは、ラザフォードが自身の行った実験に対して述べた感想である。実験は金箔(めっちゃ薄い)にアルファ粒子(ヘリウムの原子核)をたくさん撃ち込むというもので、原子は均等に正電荷を持ち、中に電子が埋まっている(すでに電子の存在は実証されている)、という仮説のもとアルファ粒子は少しずつズレると予想したラザフォードだが、この予想は大いに裏切られたのだ。実は中にあるのは原子核という非常に小さな正電荷の塊で、あとは何もない。おびただしい数の粒子の一つまみだけが反対側に返ってきたのだ。そんなスカスカな原子の構造に対する驚きを始めた一言がまさに「ティッシュ(以下略)」なのである。

ほかにも色々と原子やほかの諸々については土台となる研究があった。私が紙を節約せざるを得ないため、申し訳ないが、詳しくはグーグル先生にでも尋ねてほしい。

・素粒子

電子は実は素粒子である。

質量が存在するとかで近年話題になったニュートリノも素粒子である。

ただし、陽子や中性子はさらに「アップクォーク」と「ダウニクォーク」に分けられるので、素粒子ではない。

他の素粒子は不安定ですぐ他のものに変化するので自然界で圧倒的に多いのはこの「アップクォーク」と「ダウニクォーク」、電子、ニュートリノ 3 種類くらいであろう。ものによっては存在を仮定しないと自然界の事柄と合致しないという理由で存在することにされたものもあるが、のちに加速器を用いた実験で実在すると証明されているものも多い。昨年言及した力を伝える粒子のうち重力子は存在こそ予言されているが観測されておらず、そういう未観測粒子も数多くある。

実はもっと細かい粒子があつたり、どの粒子もある減茶苦茶小さなひもの一形態だったり、より細かい構造が存在するという説もあるが、正直私にもよくわからないので割愛する。

・奇妙な世界

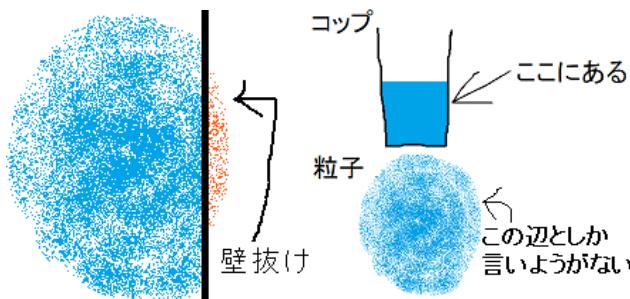
ここまで小さな粒子の話ともなると、我々の常識は通用しない。この常識はずれな世界を取り扱う学問が量子力学である。例えば常識の範囲内では物は常にある場所にしっかりと「存在」する。例えば、先述のコップはどこか、と訊かれたとして机の上、と断言できるようなものだ。しかし、量子力学では、あの粒子はどこか、と訊かれても断言できないのである。「ここかもしれないし、あそこかもしれない。でも、こっちにある確率の方が高い。」くらいにしか断言できないし、知りようがないのである。そのため、観測はある程度ぼんやりしていた「ここかもしれないゾーン」を狭めるくらいしかできないのだ。だが、観測をやめるとすぐに「ここかもしれないゾーン」は元に戻る。

さらに言うと、粒子がどれくらいの速さで動いているかさえもはつきり分からないのである。これは場所を知るために必要な観測という動作特有のものもある。たとえばりんごを見るとき、私たちはりんごで反射した光を見ていることとなる。ただし、物が小さいと光によって運動が乱されたり、場所を知るにはエネルギーが足りなかったりするのである。だからどっかは分からなくなるのだ。

「測定において位置の誤差と運動量(速度×質量で表される値)の積は一定以上になる」という原理が存在する。現在は反例が示されているが、大まかな方向性は合っているようである。

これだけではかなり分かり辛いだろうが、かいつまんで言うとこうなる。

「実際に観測するまで粒子の位置が分からないうえに、観測しても誤差からは逃れられない。運動量についても同様。次頁の図のとおり。」



こうなるとだいぶ厄介なことが起こる。「ここかもしれないゾーン」は時に壁(十分薄い)を超えるのである。その場合、何回か観測するうちに粒子が壁抜けするのである。これはトンネル効果と呼ばれるが、これは量子力学を代表する奇妙な結論の1つといえよう。上に例を示す。

・世界一美しい実験

この「ここかもしれないゾーン」だが、場合によりこんなことも引き起こす。

二つの隙間が開いた壁に電子を打ち込んだと仮定しよう。このとき、電子はいずれかの壁を通ることとなる。しかし、観測するまでは二つの隙間のどちらを通った確率も存在する。その

場合、この電子は二つの隙間を同時に通過したことと同義に解釈できるのである。粒子の検出確率は波のようなグラフを描くので、この状態を「波の状態」と呼ぶこともあるのだが、この「検出率の波」は干渉しあうこともあるのだ。

それにより、電子自身の検出率の波同士が干渉することで何度も電子を打ち出した際に独特のパターンができる。つまり、自分自身により電子の検出率が変わるわけである。

「で、電子はどっちを通ったんだ？」と思う方も多いだろうが、実際にそれを観測すると、電子の飛び方は直線的なものに戻るのである。つまり、同時に通ったからこそこんなパターンになるのだ。

私の能力上細かい計算はできないが、要するに次頁の図のとおりである。下のグラフの色のついた部分と上の同心円は対応するので、同じ色の部分の重なった箇所は検出率が上がり、色の違う部分の重なった箇所からは粒子が全く検出されない。何度も電子を打ち出した際のパターンが右側に示されている。

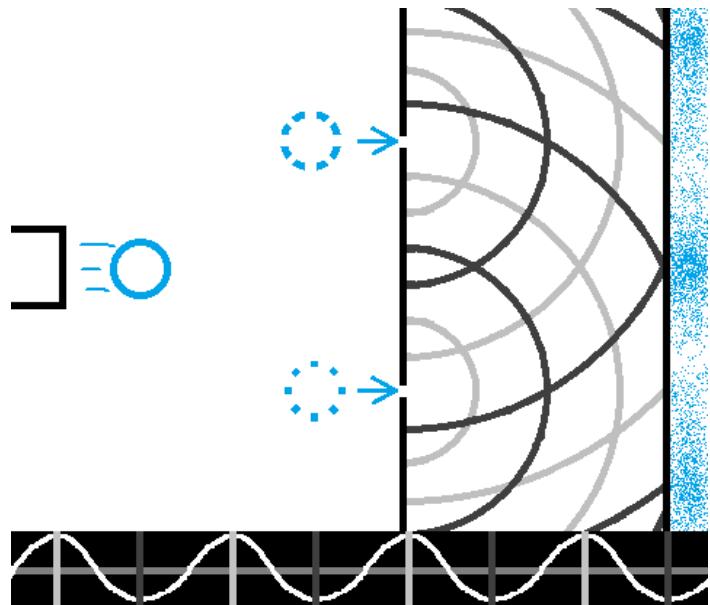
こんな常識はずれな結論を示した実験だが、ちゃんと名前もある。「電子の二重スリット実験」だ。ある科学誌で行われた「世界一美しい実験は何か」という人気投票では我々の物質観を見事に、明白な方法で打ち碎いたという点を評価され堂々の一位を取っている。

・補記および何が言いたかったのか

ここまで論じたことは確かに本当だが、そのスケールは非常に小さいものなので、我々の世界ではこういう効果を気にする必要性はほとんどない。しかし、天文学的確率とはいえ、なんともなしに寄りかかった壁を誰かがすり抜ける可能性は一応存在するのである。

我々の世界とは一見無関係に見えるこの学問だが、実はコンピューターが量子力学の賜物だったりするなど、我々の生活とどんどんと密接になっている。実際のところ他にいい例が思いつかないが、これからの中時代、ここで述べたような結論が生活上の様々なことに役立つことだろう。

そういうものを支えている実に奇妙な法則を知ることも、もしかすると読者の生活を充実したものにする一助になるかもしれない。



・出典

<https://ja.wikipedia.org/wiki/不確定性原理>: アクセス日 2018/07/14
数々の本を読んだときの記憶(2014年以降)

・おまけ

私は一応数学研究会の代表者である。そんなこともあり、余白ができた以上、宣伝をしなくてはならない気がしたので、自身の欲求に従うことにした。これが検閲を免れ皆さんの中に触れることを祈る。（ということで、余白ができるように編集してみました。~~偶然そうなっただけ~~
~~せず~~ by Positron 編集部）

というわけで、以下の問題を解いて数学研究会に持ち込むと粗品がもらえます。

問、正四面体を $T(0)$ とする。任意の多面体 $P(n)$ の辺の中点を結んでできる立体を $P(n+1)$ 、各辺の中点を結んでできる立体が $P(n+1)$ である立体のうち $P(n)$ でない方の立体を $P'(n)$ とするとき、 $T'(1)$ と $T(2)$ を下のスペースに作図せよ。

プログラミング入門の言語は何が良いのか問題

こんにちは、中学二年の永田と申します。ここでは、最近「プログラミング教育」が注目される中で、入門に適したプログラミング言語は何なのかを私の考えで綴っていこうと思っています。(※プログラミング言語=コンピュータをプログラムするときに使う専用の言語。人間にもわかりやすく、コンピュータにも分かりやすいように設計されている。)

1. プログラミングの目的

今、世界中で私達のようなこどもにプログラミングをさせる動きが広まっています。日本ではおおよそ文部科学省の新学習指導要領 2020への改訂で、小学校でのプログラミングが必修化されたことからプログラミング教育が注目されるようになったと言われています。今回必修化される予定なのはあくまで小学校で、中学校技術科の学習指導要領を見ると、既に「プログラム」の文字があります。では何のために文部省はそんなことをしたのでしょうか。文部省はその目的として以下のようにまとめています。詳細は文部省の WEB サイトを参照してください。

http://www.mext.go.jp/component/a_menu/education/micro_detail/_icsFiles/afIELDfile/2018/03/30/1375607_01.pdf

1. プログラミング言語を覚えたり、プログラミングの技能を習得したりといったことではない。
2. 論理的思考力を育むとともに、プログラムの働きやよさ、情報社会がコンピュータをはじめとする情報技術によって支えられていることなどに気付き、身近な問題の解決に主体的に取り組む態度やコンピュータ等を上手に活用してよりよい社会を築いていこうとする態度などを育むこと。
3. 教科等で学ぶ知識及び技能等をより確実に身に付けさせること。

どうやら、「論理的思考を育み、情報社会がコンピュータによってさえられていることを理解する手段」としてプログラミングが選ばれたようです。

これ以外にも、アメリカのオバマ元大統領が在任中(2014年末)に、プログラミング教育の重要性について全国民に訴えた動画が公開されたことも、世界での動きを加速させる原因になったといわれています。[\(https://youtu.be/JDw1ii7aKwg\)](https://youtu.be/JDw1ii7aKwg)

これとは別に、単純に IT 企業のプログラマーになりたいだとか、ゲームクリエイターになりたいだとか、そういう理由でプログラミングを学ぶ人もいます。先に挙げた場合と、この場合では、学ぶべきことが全く違ってくると私は考えています。次頁からの文では、それについて私の考えを書いていこうと思います。

2. 「論理的思考」を身につけるためのプログラミング

先に挙げた資料で、文部省は小学校におけるプログラミング教育の目的を「論理的思考力を育む」などと言っていました。論理的思考、というのは何でしょうか。論理的な、考え方ということなので、それぞれの意味を Wiktionary(Wikipedia の系列プロジェクト・辞書)で調べてみます。論理は「議論や思考を進めていく筋道」、思考は「論理に従い考えること」とあります。思考そのものが実は論理でした。

文部省は、プログラミングを使って「ものごとを行う際に筋道を立てて考える力」を育てたいということなんだと思います。この場合、プログラミングはどういう形で効果を発揮するのでしょうか。実は、プログラムでコンピュータに何かをさせるときには、先に書いた「筋道・論理」が必須になります。プログラミングで「筋道・論理」が必要だということは、NHKの番組「10min.ボックス テイクテック」内のコーナー、「プログラムテック」を見ると分かりやすいです。NHKのウェブサイト、NHK for School で視聴することができます。

第一回放送「流れを操る」のプログラムテックから、自由に指令ができるロボットを使って二分以内にトムさんのところへジュースを運ぶミッションをもとに論理を組み立ててみます。
(http://www.nhk.or.jp/gijutsu/taketech/?das_id=D0005250011_00000)

番組では、まずジュースを持つところまではできました。それを持ち上げて歩いていけばよいのですが、「持ち上げて」と指令を出すと、頭の上まで腕を大きく振り上げてしまい、ジュースがこぼれるという展開になりました。ロボット(=コンピュータ)は人間のように「察する」事ができません。人間なら「持ち上げて」と言われたら普通胸の高さまでだと理解できるでしょう。ロボットをプログラミングするなら、もっと明確に、筋道を立てて司令を出していかないといけないです。この例なら「ジュースを、胸の高さまで、持ち上げて」とします。これが、文部省が小学生に求める論理的思考力だと思います。

このとき、学習に最適なプログラミング言語は何でしょうか。小学生(または中学生も)へのプログラミング教育に盛んに取り入れられている言語として、MIT(マサチューセッツ工科大学メディアラボ)製の「Scratch」というものがあります。この言語の最大の特徴は、ほとんど文字を書かなくて良いことです。Scratch は、「ブロック」という、それぞれに指令としての意味が含まれたものをくっつけていくことでプログラムを作ります。さらに、ブロックの表示は日本語にすることができるので SF 映画などでありがちな小難しいアルファベットの羅列を書く必要がありません。文字を打つことが必要になるのは、キャラクターにセリフを言わせたり、~~度回転させたりするときだけです。

ただし、Scratch と言えど中身はコンピュータです。(実際ロボットを Scratch で動かすのはいろいろと面倒なのですが)ロボットに「ジュースを運べ」と言うだけでジュースを運んでくれるほど簡単ではありません。Scratch でも、「ジュースを持て」、「持ち上げて」、「体を回転させて」、「トムさんの前まで歩いて」と順に指令を出す必要があります。つまり、日本語を使って論理を組み立てることができるのです。これはほとんど日本語しかわからない小学生(+中学生)に論理的思考をさせるのに非常に都合が良いです。



Scratchで作られたマウスポインターから逃げるプログラム。日本語がわかれば意味がわかるくらい簡単。

ちなみに、最近は Scratch のようにブロックを使ったプログラミング学習環境が増えてきていて、文部省が Scratch をもとに自分で作った「プログラミン」や、Life is Tech!とディズニーがコラボした「テクノロジア魔法学校」などです。また、マイコン(手に乗るような小さなコンピュータ)用に作られたものでは、モータや LED をつないでちょっとした機械を作ることができます。英放送局 BBC のマイコン「micro:bit」に向けて作られた「micro:bit JavaScript ブロックエディター」や、有名な教育用マイコン Raspberry Pi 向けの「Scratch GPIO」などです。モータや LED などの現実世界との接続ができるのなら、情報技術学習よりもさらに広げることができそうです。

以上のことから、「文部省が進める、論理的思考力を身につけるためのプログラミング教育」には、「Scratch を始めとする日本語のブロックプログラミング言語」が最適だというの私が考えです。

3. プログラマーを目指す人のためのプログラミング

プログラマーを目指すなら話は別です。目的がそもそも違います。小学校で取り組むプログラミングの目的は「論理的思考力を身につける」ことなどですが、プログラマーを目指すならそれだけでは足りません。プログラマーが行っているのは実務作業ですから。筋道を立てた思考、論理的思考力はもちろん必須でしょうし、実際にプログラムを書く力が求められます。それに、技術的な知識もたくさん覚えなければいけません。この辺りはしばらく真面目に勉強していれば覚えていけると思いますが、学習に適した言語は何でしょう。

Scratch を考えてみます。Scratch はなかなか有能です。基本的な考え方ほぼ学ぶことができますし、オンラインコミュニティのおかげでわからないことはすぐに質問することができます。ただし忘れてはいけないのが Scratch はあくまで教育用であるということ。例えば Scratch は単体のアプリにできません。Scratch のソフトの中では動かないのです(外部ツールを使えばできないこともないです)。また、ゲーム向けに作られているのでそれ以外のプログラムを作るのは苦手です。このように、Scratch は実用的な言語ではありません。現に Scratch で制作されて、世の中で販売されているソフトは見たことがありません。

世の中ではブロック型の言語はまだ主流ではなく、SF 映画でよく見るような文字を書くタイプのものが多いです。有名所では C(C++&C#), Java, Python などです。プログラマーを目指すなら、このような広く使われている言語を一つくらいは使えないと思いません。入門

用としてどの言語を使うかは自由でしょう。はじめから実用的な C などの言語をやるものも手ですが、もし小学生がプログラマーを目指すなら、はじめは Scratch などで考え方を身に着けてから本格的なプログラミングをするのが良いと私は思っています。Scratch のよいところは、機械音痴でも少しのパソコンの基礎(マウス操作や、キーボードのタイピングくらい)を覚えてしまえば簡単にプログラムが作れるというところです。はじめに難しい言語を使って、自分には向いていないと挫折するより、プログラムの根本の部分は難しくないということを理解したほうが続けやすいと思います。

以上のことから、プログラマーを目指すなら Scratch にとどまらず、少し基礎を理解したら積極的に C,Java,Python などの汎用的なプログラミング言語を学ぶのが良いというのが私の考えです。

実は私も前まで Scratch をやっていましたが、3 年以上 Scratch にどっぷり浸かっていたせいでなかなか新しい技術を取り入れられずにいます。ですが、Scratch で得た感覚は今も役立っています。

4. まとめ

最後まで読んでいただきありがとうございました。近頃テレビやネットでやたらとプログラミングプログラミング言われているので、プログラミングと一概に言ってもそれをするための手段(言語)はたくさんあるよな、と思い私の考えを文章にしました。

2 つ目で書いた Scratch は、実際多くのプログラミング教室などで採用されているようです。小学校で必修化されるのは 2020 年からなので、すでに取り組み始めた学校もありますがまだどうなるかは分かりません。

3 つ目で書いた C,Java,Python についてちょっと余談です。私は Python が好きなので Python の宣伝をします。最近流行りの人工知能なんかは Python で作られることが多いです。Python は文法が書きやすい設計なので、初心者の入門におすすめです。

画像は ScratchBlocks と Twemoji を使った自作のものです。

入り切らなかったので参考にしたウェブサイトはそれぞれ文中に散りばめられた URL を辿っていただければと思います。

生体認証について

村上渉

0. 生体認証とは？

生体認証とは自分の指紋や虹彩、静脈などの情報をデジタル情報化して端末のロック解除、サインイン、さらには決済までできてしまうというすごいものなんだ！

一言に生体とは言っても指紋、虹彩、顔、静脈…と沢山あるので今回はパソコンやスマートフォン、ATMなどで使われる身近なものに絞ってまとめたいと思います

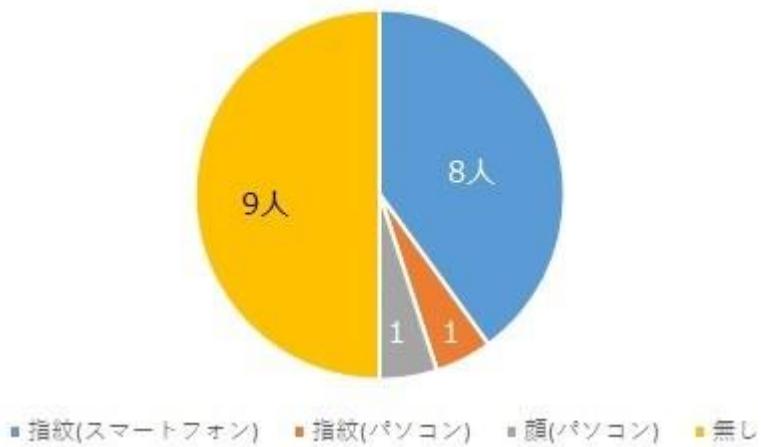
1. デバイスの所有

浅野では先生のほとんどが学校から支給される Surface という顔認証付きタブレット PC をつかっていますが、物理部ではいったい何人の人がそれらの機器を所有しているのでしょうか。物理部でどれくらいの人が持っているのかというのが次のグラフ(図 1)です。(正式に調査したものではなくあくまで把握している分です。)

指紋と顔で約半数の人がいずれかの機能を使えるということです。意外と多いのではないですか。大きな要因は iPhone が指紋認証(Touch ID)を搭載しているからでしょう。

2018 年、実に 10 億台もの指紋認証付きスマートフォンが出荷される予定で約 70% の端末に搭載される見通しとなっています。物理部で使っている人はいないと思われますが、電子マネーの普及が結果的に指紋認証付きスマートフォンの普及につながったのでしょう。

図1



2. 仕組みについて

この様に身近な生体認証ですがどんな仕組みなのでしょうか

2.1 指紋認証

指紋認証は生体認証の中では歴史が古いものです。そのため一言に“指紋認証”とは言っても沢山の種類があります。

1. 画像を読み取るタイプ

これは比較的古いものでスライド式のセンサーで指紋の画像を取り込み特徴点と呼ばれる指紋のスタート地点、切れ目、分岐点などで認識します。特徴点が多ければ多いほど精度は向上します。(20~40 個程度で比較は可能)



2. 電極が埋め込まれているタイプのセンサー（静電容量方式）

これはスマホでも使われるもので、スライドしなくてもよいというメリットがあります。平らな認証部分の下には何万個も電極が埋め込まれていて指紋の凹凸部分を読み取るというものです。これは指がかすかに汗をかいているので凸部分の方がセンサーの電荷がたまりやすいという特性を利用したものです。そのあとはスライド方式と同じように特徴点を読み取り認証します。

尚、シリコン製が一般的ですが、JDI(株式会社ジャパンディスプレイ)がガラス基板を採用して透明にするだけでなく曲面やディスプレイへの埋め込みも可能にしました。(2018年1月発表)

(写真はスマートフォン SH-M05 の指紋認証センサー)



←横に長くかなり小さい

3. 汗孔(汗が出る小さな穴)を第三次特徴として使えるもの

これはまだ開発されたばかりの仕組みで生体認証システムの DDS と東京大学が開発しました。

特徴点を第一次特徴の「指の凹凸によって模様になった渦状紋」、第二次特徴の「分岐点や線の始まり」の他に「汗が出る小さな穴」を第三次特徴とすることで認証の精度を約 10 倍に高めました。

静電容量方式では汗孔を(十分な解像度がなく)読み取ることができません。

コスト面が課題ですがスマホメーカーや自動車や家の鍵にも採用を目指すそうです。

この他にも Qualcomm の超音波センサーを使ったものなどがあります。

2.2 虹彩認証

虹彩認証は精度も高く、2015 年に富士通のスマホ ARROWS NX にスマートフォンとして世界で初めて搭載されて話題になりました。虹彩は一卵性双生児でも区別できます。(DNA の塩基配列によって決まらないため。)その特徴は高い認証精度です。網膜と比べて眼球の表面にあるため撮影も容易で、虹彩パターンの濃淡値のヒストグラムを用います。さらに指紋とは違い直接触れることがないので抵抗も少ないです。

問題点は、目に赤外線を当てて認証するのですが、赤外線の LED を悪意のある人がそれを取り換えると目に障害が残るということです。

2.3 静脈認証

虹彩に次いで精度が高い方式で、ATMなどに採用され利用件数も増えています。近赤外線を当てて認証するので虹彩と同じく機械に触れる必要はありません。近年はセンサーの小型化も進みタブレットPCでも採用されています。



写真は LIFE BOOK S937/S (FMworld 法人より)

2.4 顔認証

この中では認証精度は低く、簡易的なものに用いられます。

赤外線カメラを用いたものと普通のカメラを用いたものがあり、前者が主流です。双子を見分けることが難しく眼鏡や明るさ、顔の表情、加齢などによっても精度は下がります。

ちなみに物理部で顔認証は一人でしたがそれは私です。家族共用のパソコンなのですが父のアカウントに入ることはもちろんできませんし5歳下の弟は私のアカウントに入ることはできません。簡易的とはいってもある程度の精度はあり眼鏡をかけていても読み取ってくれます。



←ロック画面の様子

2.5 声

これは声紋を利用したものがよく知られています。これも簡易的なものでその日の声の調子によっては正しく認証しないことがあります。例えばAndroidのスマートフォンの機能としてSmart Lockというものがありますが、それは「OK Google」と発音してロックを解除できます。

3. まとめ

この様に生体認証は沢山あって便利な物ですが欺瞞^{きまん}の方法も数多く編み出されています。例えば指紋認証だとピースサインが映った写真から指紋を特定される危険性があります。(浅野の教頭先生は記念撮影の時指を曲げて見えなくしていました(笑))生体認証だけに頼らず複雑なパスワードと組み合わせるのが良いでしょう。

参考文献

- <https://ja.wikipedia.org/wiki/生体認証>
- <https://ja.wikipedia.org/wiki/虹彩認識>
- <https://www.nikkei.com/article/DGXMZO29549890Y8A410C1XY0000/>
- https://next.rikunabi.com/tech/docs/ct_s03600.jsp?p=000647
- <https://iphone-mania.jp/news-187532/>
- <https://time-space.kddi.com/ict-keywords/kaisetsu/20160601/>
- http://www.ratocsystems.com/products/subpage/security/srexfsu2_sensor.html
- <https://iphone-mania.jp/news-150996/>
- <https://japanese.engadget.com/2017/06/28/fingerprint/>
- <https://japanese.engadget.com/2018/01/23/jdi/>
- <http://www.fmworld.net/biz/fmv/lifebook/s937s/>
- <http://www.fmworld.net/product/phone/f-04g/>
- <https://www.ipa.go.jp/files/000013735.pdf>
- <http://www.hkd.meti.go.jp/hokim/20170217/data02.pdf>

2018/7/20 , 2018/7/21 アクセス

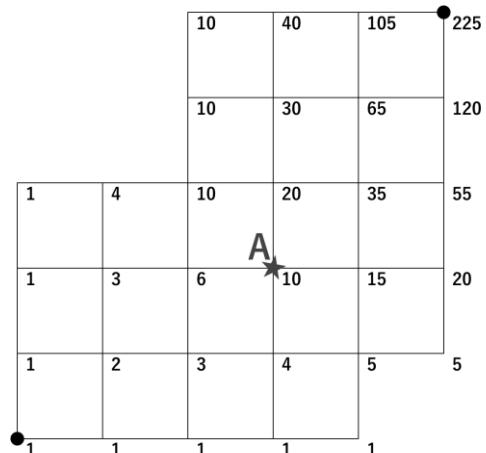
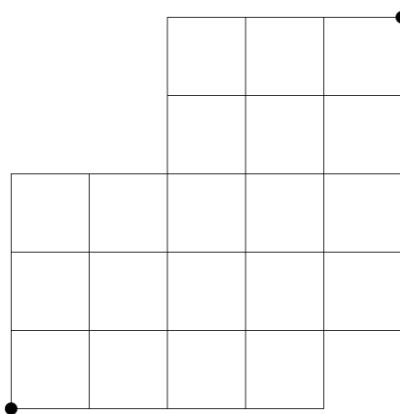
コラム 動的計画法

例えば、探し物が家の中にあるはずなのに見つからないとしましょう。そのときに、まず家を部屋ごとに分けます。そして、その各部屋を、また細かく分けます。例えば部屋を 50cm 四方くらいに分けていくとします。すると、その 50cm 四方の中に探し物があるかどうかは簡単に分かると思います。それを繰り返していくと、その部屋に探し物があるかがわかるはずです。

というように、いきなり答えを出すのが難しい問題（家に探し物があるか？）を解くときに、その問題を小さな問題（50cm 四方の中に探し物があるか？）に分け、その小さな問題をすべて解くことで最終的な問題の答えを出す方法を、分割統治法といいます。

この分割統治法は、数学の問題を解くのにも使えます。中学受験をしようとしている、または経験した人なら、次のような問題を解いたことがあると思います。

問. 図のような道があったとします。このとき、左下から右上まで遠回りしないで行く道順は何通りありますか。（左の図）



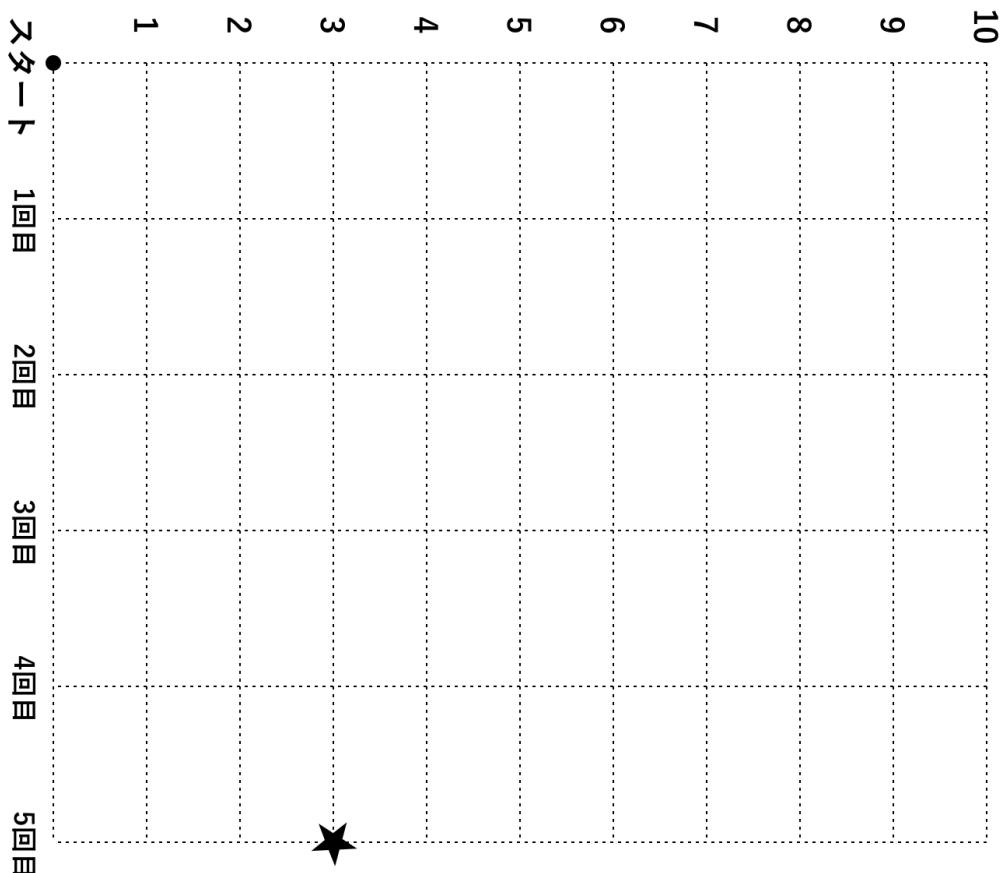
この問題は、交点それぞれに、その交点まで行く道順の数を書き込んでいくことで答えが出せます。（右の図）例えば、図の点 A では、点 A の左から来るのは 6 通り、下から来るのが 4 通りなので、点 A に行く道順は $6+4=10$ 通りです。これを繰り返すと、右上まで行く道順は 225 通りと出ます。

この場合は、「全部で何通りあるか」という問題を、「ある交点まで行く方法は、それぞれ何通りか」という問題に分割しています。また、このように小さな問題を解くときに、他の結果を再利用（左と下の道順の数を利用して次の道順の数を求める）するような分割統治法のことを動的計画法といいます。

では、動的計画法を使って、1つ問題を解いてみましょう。

問。Aくんは、（1から6までのサイコロだと問題が複雑になって、ただでさえ狭いこのページの余白で計算できなくなってしまうので）1から3の数字が書いてあるサイコロとごろくを用意し、Aくんの駒をスタートに置きました。Aくんの駒は、1から3の数字が書いてあるサイコロを振って、1が出たら1つすすみ、2が出たら2つすすみ、3が出たら2つもどります。ただし、スタートより前には戻れないので、そのときはスタートにとどまります。

このとき、サイコロを5回投げたら、スタートから3つ進んだところにAくんの駒があるようなサイコロの目の出方は何通りあるでしょう。必要なら下の図を使ってください。（答えと下の図の使用例はこの部誌内にあります。）



とうてき 投擲について

有馬慎太郎

1. はじめに

何故投擲について書こうとしたのかというと、歴史で世界大戦について学習しており、過去の戦いについて大きな役割を果たした投擲武器について興味が湧いた故、書いていきたいと思う。

2. 扱いや特徴

まず投擲の道具（投擲具）と言われて何を思い浮かべるであろうか。円盤、槍のようなものはたまたま石を思い浮かべる人もいるかもしれない。実はこれらすべてが投擲具であるのはもちろんの事、手裏剣や催涙弾、漁網までもが投擲具なのである。

そもそも投擲具は人類の発展によって狩猟用にうみだされたものである。

はじめこそ石を投げるような単純かつ威力の低いものではあったのだが、長い年月を重ねた末に投げナイフ、カタパルトといった殺傷能力のある凶器へと姿を変えていった。

それは間接的にいさかいを生み、後々に手を付けられなくなるのではあるが一先ず先に行くとしよう。

3. 投擲道具のすばらしさ

ここでは投擲道具の代表例としてオリンピックで扱われている砲丸投げですばらしさを伝えたいと思う。

ここで唐突に物理の話に入るとしよう。

ここでは男子の砲丸の重さ（8kg）で計算する。（空気抵抗や風を考えないものとする。）世界記録は約 20m 飛ばしているから、水平距離 20m、角度 θ 、時間 t 、初速度 v とすると、
$$\tan\theta = \frac{v}{\sqrt{v^2 + 4g t}}$$
となる。

世界記録を出したバーンズ選手は身長 194cm と大きく、体重も 140kg そしてとても「手が大きい」ので、約 35 度で投げたものと思われる。

大抵の選手は首下のあたりから投げるので、 $10\tan\theta = x$ よって約 7m である。

よってこれは首下までの高さが 1.5m であるから、8.5m からの垂直落下に等しい。

よって地面に与える力は $8.5 \times 9.8 \times 8.0 = 666.4$ 。そしてこれは軽自動車一台分に相当し、かなりの重さになると分かっていただけたであろう。

4. 投擲の扱われ方

3. によって投擲のすばらしさはわかつてもらえたと信じている。

一方、投擲の発明によって人を殺めることが容易になってしまったともいえるであろう。

例えば、カタパルトである。カタパルトと恰好良くいっても投石機の事である。石を投擲して、敵の城や敵自身を攻撃するため攻城兵器と呼ばれていた。その頃はまだよかつたのであるが、第一次世界大戦から、科学技術の発展により、フランスは手榴弾を投擲するようになってしまったのである。これによってドイツ軍に大量の死者が出て、白熱させ戦争をこじらせてしまったのは言うまでもないであろう。

5.まとめ

それからというもの投擲はオリンピック競技での平和の象徴として、一種の競技としてだけでなく、紛争地での手榴弾投げ込みのため、または、デモ鎮圧のための催涙弾の投げ込みのための技術として使われるようになってきている。

我々は、現代の技術が戦争を通じてきたものであり、何のために使うべきなのかしっかりと考えていくべきである。

ゲームプログラムと数学が出会うとき

シューティングゲームばかり作っている中2の中野です。今回はいろいろな物理・数学に関する事をここに記したいと思います（内容は至って真面目です。期待しないでください）。

1:微分の基礎

第一章では微分の基本中の基本について説明します。まず、微分とは読んで字の如く“微小部分で考える”ということです。たとえば正比例などのグラフの場合は直線なので、傾きが一定です（傾きの求め方： $\frac{\Delta y}{\Delta x}$ （この Δ というのは変化量という意味です））。しかし直線じゃない線を考えた場合に、 Δx の範囲を変えると Δy が変わってしまいます。では例として $y = 4.9x^2$ を考えてみましょう。

①: $x=2$ から $x=3$ の変化を考える場合

$$\frac{\Delta y}{\Delta x} = \frac{(4.9 \times 3 \times 3) - (4.9 \times 2 \times 2)}{3 - 2} = 44.1 - 19.6 = 24.5$$

②: $x=2$ から $x=4$ の範囲を考える場合

$$\frac{\Delta y}{\Delta x} = \frac{(4.9 \times 4 \times 4) - (4.9 \times 2 \times 2)}{4 - 2} = 29.4$$

上記より Δx の範囲によって傾きが違うことがわかります。では、いよいよ本題に入りましょう。上記の変数を使って考えてみましょう。 $x = 2$ から $x = 2 + h$ の範囲まで考えることにしましょう。

$$\frac{\Delta y}{\Delta x} = \frac{4.9(2 + h)^2 - 4.9 \times 2^2}{(2 + h) - 2} = \frac{19.6h + 4.9h^2}{h} = 19.6 + 4.9h$$

微分とは“微小部分で考える”という意味でした。要はここで言う $\Delta x = h$ を極限まで小さくする（0に限りなく近づける）ということです。この h を限りなく0に近づけることを数学で

$$\lim_{h \rightarrow 0} 19.6 + 4.9h$$

と書きます。そして、その式に $h = 0$ を代入します（限り無く近づけるということです）。

よって

$$\lim_{h \rightarrow 0} 19.6 + 4.9h = 19.6$$

この19.6のことを俗に“微分係数”と呼びます。ここまで微分の導入（基本中の基本）をやってきました。これで数学に興味のある人が増えたら幸いです。

2:ゲームにおける重力の考え方

皆さんはマ○オとかのゲームでジャンプするシーンを知っていますよね。本章ではそのようなゲームにおける、ジャンプのプログラムについて考えてみましょう。まず有名なこちらの2式を見てください。

$$y_t = v_0 t - \frac{1}{2} g t^2$$

$$v^2 - v_0^2 = -2gy_{max}$$

注： y_{max} = 最高点[m]， y_t = 時間によって変化する y 座標， v = 時間によって変化する物体の速さ， v_0 = 物体の初速度， $g = 9.8[m/s^2]$ (重力加速度)， t = 時間

基本的にはこれで求めることができます。しかし、プログラミングでは初速度を考えづらい状況がほとんどですよね。よって速さの概念を無くしましょう。移項などを繰り返すと、結局 $y_t = \sqrt{2gy_{max}t} - \frac{1}{2}gt^2$ という式が作れます。また、x 軸において、放物線は等速運動なので一定数の値を増やすことにします。ではこれをプログラミングで表現してみましょう。ちなみに言語は C 言語、テキストエディタ・デバッガは Visual Studio 2017 Community、ライブラリは DX ライブラリを使っています。Visual Studio は Micro Soft 社から無料でダウンロードできます。<https://docs.microsoft.com/ja-jp/visualstudio/install/install-visual-studio> ここに詳しいことが書いてあります。DX ライブラリは「画像を表示する」とか「特定のキーを監視する」ときに便利なライブラリです。<http://dxlib.o.oo7.jp/> 詳しいことはここからどうぞ。

```
#include "DxLib.h"
#include <math.h>
#define g 9.8067
#define y_max 2.000
int x = 0;
int y = 480;
int time1;
int time2;
double t;
int flag = 0;
int WINAPI WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,LPSTR lpCmdLine,int nCmdShow){
    ChangeWindowMode(TRUE);
    if (DxLib_Init() == -1){
        return -1;
    }
    while (CheckHitKey(KEY_INPUT_ESCAPE) == FALSE && ProcessMessage() != -1) {
        if (CheckHitKey(KEY_INPUT_SPACE) == 1 && flag == 0) {
            flag = 1;
            time1 = GetNowCount();
        }
        if (flag == 1) {
            DrawPixel(x, 480 - y, GetColor(255, 255, 255));
            time2 = GetNowCount();
            t = (double)(time2 - time1) / 1000.000;
            x += 6;
            y = (int)((sqrt(2.000*g*y_max)*t - 0.500*g*t*t)*480.000 / y_max);
        }
        if (y < 0) {
            ClearDrawScreen();
            flag = 0;
        }
    }
}
```

```

        y = 0;
        x = 0;
    }
    ScreenFlip();
}
DxLib_End();
return 0;
}

```

これを Visual Studio にコピペして実行してみてください。点が放物線を描いているように見えるはずです。

3:二乗を表現する方法

1 2 3 4 5 6 7 8 9...などという数を二乗するとそれぞれ 1 4 9 16 25 36 49 64 81...となりますよね？本章はこれらの数列をプログラミングで表現してみたいと思います(すごく原始的です)。まず上の数列を見ると“差”は、3 5 7 9 11 13 15 17...ですよね？そうです、2ずつ差が増えているのです。これを再現すれば皆さんでも簡単に表現することが可能です。

<ソースコード>

```

#include <stdio.h>
int main(void) {
    int a = 1; //二乗した数
    int b = 1; //二乗する前の数
    int s = 3; //二乗した数たちの最初の差
    while (a <= 10000) { //二乗した数の上限は 10000 まで
        printf("%d の二乗は%d です\n", b, a); //二乗した数を表示する(改行付き)
        a += s; //次の二乗した数に差を足す
        s += 2; //今の差に2を足して次の差にする
        b += 1; //二乗する前の数に1ずつ足す
    }
    return 0;
}

```

これをコピペして Visual Studio で実行してみてください。右の図みたいなものができるはずです。

1の二乗は1です
2の二乗は4です
3の二乗は9です
4の二乗は16です
5の二乗は25です
6の二乗は36です
7の二乗は49です
8の二乗は64です
9の二乗は81です
10の二乗は100です
11の二乗は121です
12の二乗は144です
13の二乗は169です

4:展示作品

- ・シューティングゲーム

<概要>上から降ってくる緑の弾を方向キーで回避する遊び。

<考察>弾が降ってくるときに最初の方だけ、ガガガッってなる、というエラーがある(未解決)。

- ・アクションゲーム

<概要>Zキーで攻撃、Xキーでガード、方向キーで移動。

<考察>あまり完成の目処が立っていない為、文化祭で発表されない確率が微小レベルで存在。

- ・放物線

<概要>スペースキーの押下で、等間隔で放物線を描く。

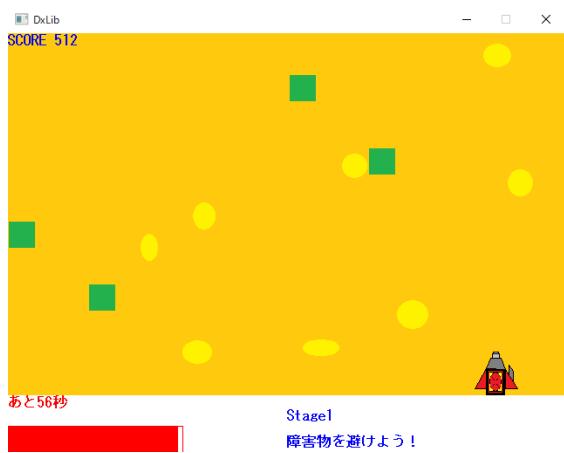
<考察>本部誌の第二章参照。

- ・スクリーンセーバー

<概要>物理部と書かれた楕円が色を変えながら反射する。

<考察>背景もRGBで次第に青っぽく、赤っぽくするようにした。思いの他、難しかった。また、楕円のサイズの関係上、物理部の端っこが楕円の色と重なり気持ち悪くなつた。

↓シューティングゲームの様子



今年の僕の部誌はこれでおしまいです。来年また部誌でお会いしましょう。

いったいどのくらい大きな声を出せばブラジルの人に声が届くのか

1はじめに

皆さんは、「ブラジルの人、聞こえますかー！？」というネタをご存知でしょうか？

サバンナというお笑いコンビの八木(高橋じゃないほう)のギャグで、地面に向かって叫び、上記のセリフを言うというもので、数年前に流行って(?)いました。そして、そのギャグを見た時誰もが思ったであろうことの一つとして、「いったいどのくらいの声を出したらブラジルの人に声が届くのだろう」というのが有ると思います。なので今回は、実際にどのくらいの声を出せばいいのか、計算し導いていきたいと思います。

2計算過程

実際にどのような計算をするかが、以下になります。

①距離減衰を求める

まず、距離減衰について簡単な説明をすると、「音の大きさが遠くに伝わっていく間にどんどん小さくなること」、これが距離減衰です。また、減衰は音の発生源(音源)によって違うので、まずはその数値を出します。ここでは、ブラジルが地球上において日本のちょうど反対に位置するとし、地球の半径を(6371→)6400km とする

と、距離減衰量(A) = $20 \times \log_{10}(r / r_0)$ となります。

※ r は距離、 r_0 は、音源からの基準となる距離
なのでこの場合、 $r_0=1$ とすると $A = 20 \times \log_{10}((12800000 \times 3.14 \div 2) \div 1) = 146.06 \dots \rightarrow$ 約 146dB と求められ、これがこの場合の距離減衰となります。

②出す必要がある dB の量を求める

①で減衰量を求めたので、右の図より、日常的な会話の大きさである 60dB を聞こえさせるとすると、 $60 + 146 = 206$ dB となり、これがブラジルの人に声を届けるのに必要な声の大きさということです。ちなみに、どのくらいの大きさかを比べるのに飛行機付近の音と比べると、飛行機のエンジンから 30m 離れた場所での音の大きさが 120dB なので、差が約 86dB であり、

0dB		二十歳の最小可聴値	聞こえなくても問題なし
10dB		安静時の呼吸音	聞こえなくても問題なし
20dB		木の葉の触れ合う音	聞こえなくても問題なし
30dB		ささやき声	これが聞こえないと軽度難聴
40dB		図書館内	これが聞こえないと軽度難聴
50dB		エアコンの室外機	これが聞こえないと中度難聴
60dB		普通の会話	これが聞こえないと中度難聴
70dB		セミの鳴く声	これが聞こえないと身体障害者6級
80dB		救急車のサイレン	これが聞こえないと身体障害者4級
90dB		カラオケ	これが聞こえないと身体障害者3級
100dB		電車の通るガード下	これが聞こえないと身体障害者2級

$dB=20\times\log_{10}(\text{音の倍率})$ なので $86(dB)=20\times\log_{10}(\text{音の倍率})$ であるため、音の倍率は約 20100 倍となります。

つまり、八木氏は飛行機のエンジンから 30m 離れた場所で聞く音の 2 万倍もの音量の音を出そうとしていたわけです。

3まとめ

ということで、実際に音の大きさを計算してみたわけですが、予想通りとても大きいことが分かりました。八木氏は、ネタの中で地球にぽっかりと穴をあけようとしていたようですね。

4 編集後記

今回は合同部誌ということで、計算は自分(清水)が、そして本文は柴生田が書くという形で進めました。最初、音のことについてほぼ何も知らず、計算方法を調べるところから始まったのですが、なかなか自分の要望にあったページが見つからず、かなり苦労しました。計算自体は意外と簡単だったのでよかったです(笑)。

今回、自分たちとしては初めての部誌だったので至らぬ点もあったかと思いますが、生あたたかい目で見てくださいと幸いです m(_ _)m

最後まで読んでいただき、ありがとうございました。

【参考にさせていただいたサイト】

『騒音の基礎知識』

<http://www.city.gifu.lg.jp/secure/6589/soukiso.pdf>

『dB の話 音の大きさ』

http://www.geocities.jp/fkmtf928/dB_sound.html

『音の大きさ 「dB(デシベル)」』(図はこちらから転用させていただきました。)

https://www.sakura-shukugawa.com/2016/06/01/音の大きさ-d_b-デシベル/

『音速 - ウィキペディア』

<https://ja.wikipedia.org/wiki/音速>

『騒音予測計算の紹介』

<http://www.idemitsu.co.jp/content/100136813.pdf>

3を $\sqrt{5}$ 進数で表すと？

吉田安紀彦

まずは 0.1 を 2 進数で表してみる。

整数を n 進数で表す方法は学校で習った通りなのですが（知らない人はごめんなさい）、小数を n 進数で表すのはなぜか習わないので、その方法を考えてみました。

2 進数と 10 進数を分けて考えます。10 進数で 2 倍にすることは、2 進数で 10 倍にすることに相当します。また、整数部分が 2 未満なら整数部分だけ比べるなら同じなので、図 1 のように計算でき、このように計算した結果が下です。

$$0.1 = 0.000110011001100110\dots$$

$$0.2 = 0.001100110011001100\dots$$

$$0.3 = 0.010011001100110011\dots$$

$$0.4 = 0.011001100110011001\dots$$

$$0.5 = 0.1$$

$$0.6 = 0.100110011001100110\dots$$

$$0.7 = 0.101100110011001100\dots$$

$$0.8 = 0.110011001100110011\dots$$

$$0.9 = 0.111001100110011001\dots$$

10進数

$$\begin{array}{r} \times 2 \\ \boxed{0.1} \\ \times 2 \\ \boxed{0.2} \\ \times 2 \\ \boxed{0.4} \\ \times 2 \\ \boxed{0.8} \\ \times 2 \\ \boxed{1.6} \\ \times 2 \\ \boxed{0.6} \\ \times 2 \\ \boxed{0.4} \\ \times 2 \\ \boxed{0.8} \\ \times 2 \\ \boxed{1.6} \\ \times 2 \\ \boxed{0.6} \\ \times 2 \\ \boxed{0.4} \end{array}$$

小数部分だけ取り出す

2進数

$$\begin{array}{r} \times 10 \\ \boxed{0.000110011001100110\dots} \\ \times 10 \\ \boxed{0.001100110011001100\dots} \\ \times 10 \\ \boxed{0.01001100110011001100\dots} \\ \times 10 \\ \boxed{0.011001100110011001100\dots} \\ \times 10 \\ \boxed{0.10011001100110011001100\dots} \\ \times 10 \\ \boxed{0.1011001100110011001100110\dots} \\ \times 10 \\ \boxed{0.00110011001100110011001100\dots} \\ \times 10 \\ \boxed{0.0110011001100110011001100\dots} \\ \times 10 \\ \boxed{0.1100110011001100110011001100\dots} \\ \times 10 \\ \boxed{0.11100110011001100110011001100\dots} \\ \times 10 \\ \boxed{0.001100110011001100110011001100\dots} \\ \times 10 \\ \boxed{0.011001100110011001100110011001100\dots} \\ \times 10 \\ \boxed{0.110011001100110011001100110011001100\dots} \\ \times 10 \\ \boxed{0.11100110011001100110011001100110011001100\dots} \end{array}$$

【図 1】

では、3を $\sqrt{5}$ 進数で表すと？

さっきのように、まずは整数部分を $\sqrt{5}$ 進数に直してから、小数部分だけ取り出して、それを $\sqrt{5}$ 倍し、その整数部分を書いて、小数部分だけ取り出して、...ということを繰り返すことで $\sqrt{5}$ 進数が計算できます。このように計算した結果が下です。

$$\begin{aligned}
1 &= 1 \\
2 &= 2 \\
3 &= 10.1110110012001001001200011100112 \dots \\
4 &= 11.1110110012001001001200011100112 \dots \\
5 &= 12.1110110012001001001200011100112 \dots \\
&\quad = 20.1002000002000011112010010010101 \dots \\
&\quad \quad = 100 \\
6 &= 21.1002000002000011112010010010101 \dots \\
&\quad = 101 \\
7 &= 22.1002000002000011112010010010101 \dots \\
&\quad = 102 \\
8 &= 110.1110110012001001001200011100112 \dots \\
9 &= 111.1110110012001001001200011100112 \dots \\
10 &= 112.1110110012001001001200011100112 \dots \\
&\quad = 120.1002000002000011112010010010101 \dots \\
&\quad = 200 \\
11 &= 121.1002000002000011112010010010101 \dots \\
&\quad = 201
\end{aligned}$$

$$\begin{aligned}
12 &= 122.1002000002000011112010010010101 \dots \\
&\quad = 202 \\
&\quad = 1000.1112000112001110020011200112010 \dots \\
13 &= 210.1110110012001001001200011100112 \dots \\
&\quad = 1001.1112000112001110020011200112010 \dots \\
14 &= 211.1110110012001001001200011100112 \dots \\
&\quad = 1002.1112000112001110020011200112010 \dots \\
&\quad = 1010.1011001200100100120001110011200 \dots \\
15 &= 212.1110110012001001001200011100112 \dots \\
&\quad = 220.1002000002000011112010010010101 \dots \\
&\quad = 1011.1011001200100100120001110011200 \dots \\
16 &= 221.1002000002000011112010010010101 \dots \\
&\quad = 1012.1011001200100100120001110011200 \dots \\
&\quad = 1020.011110000101111000011110012000 \dots \\
100 &= 111102.2000101101100001101112000000100 \dots \\
&\quad = 11110.1100120010010012000111001120010 \dots
\end{aligned}$$

複数ある場合があるのは、整数部分を $\sqrt{5}$ 進数に直すときに、複数の表し方があるからです。

ついでに π 進数で表してみる。

さっきと同じように π 進数で表してみました。

$$\begin{aligned}
1 &= 1 \\
2 &= 2 \\
3 &= 3 \\
4 &= 3.3011021110020221130001020002102 \dots \\
&\quad = 10.2201220211211103010000101100100 \dots \\
5 &= 11.2201220211211103010000101100100 \dots \\
6 &= 12.2201220211211103010000101100100 \dots \\
7 &= 13.2201220211211103010000101100100 \dots \\
&\quad = 20.2021120021000000300201212221000 \dots \\
8 &= 21.2021120021000000300201212221000 \dots \\
9 &= 22.2021120021000000300201212221000 \dots \\
10 &= 23.2021120021000000300201212221000 \dots \\
&\quad = 30.1212011100221300120300222121221 \dots \\
&\quad = 100.0102212222112112200111121020120 \dots \\
11 &= 31.1212011100221300120300222121221 \dots \\
&\quad = 101.0102212222112112200111121020120 \dots \\
12 &= 32.1212011100221300120300222121221 \dots \\
&\quad = 102.0102212222112112200111121020120 \dots
\end{aligned}$$

$$\begin{aligned}
13 &= 33.1212011100221300120300222121221 \dots \\
&\quad = 103.0102212222112112200111121020120 \dots \\
14 &= 110.30100111210121210203010212000221 \dots \\
15 &= 111.30100111210121210203010212000221 \dots \\
16 &= 112.30100111210121210203010212000221 \dots \\
100 &= 2322.0023000110210220200020212220122 \dots \\
&\quad = 3013.2200000230101000010102211213010 \dots \\
&\quad = 3020.20122130010112211202110101020 \dots \\
&\quad = 10002.1220120100000111301021202102110 \dots \\
1000 &= 232310.3001112202000200000022113003010 \dots \\
&\quad = 233002.202103000120111020020000222110 \dots \\
&\quad = 302123.21101200122002202012120030022 \dots \\
&\quad = 302130.1301011023000100212120230023003 \dots \\
&\quad = 302200.0122022021200120110021112220211 \dots \\
&\quad = 1000322.2021202200301011202102130002020 \dots \\
&\quad = 1001021.1000022221100021203001010110123 \dots
\end{aligned}$$

$\sqrt{5}$ 進数は使えるのか

$\sqrt{5}$ 進数では、足し算する時も上のような計算をしなくてはならないときがあります。例えば、 $\sqrt{5}$ 進数で $1 + 2$ は、 $1+2=10.1110110012001001001200011100112\dots$ です。掛け算も、 $2\times 3=6$ を $\sqrt{5}$ 進数で表すと、 $2\times 10.1110110012001001001200011100112\dots=21.10020000020000111120100100101\dots=101$ です。2つ以上の表し方がある上に、足し算や掛け算もできません。なので、実用性はほぼないと思います。

おわりに

思い付きでまとめてみたのですごく短くなってしまいましたが、ここまで読んでいただいてありがとうございました。個人的には、これを思いついたとき $\sqrt{5}$ 進数ってどうなるのだと思ったのですが、複数の表し方があったり、意外な発見があって良かったと思いました。

ライフゲームを C++ で実装してみた話

高校一年 片山 悠哉

1. はじめに

実は「実装してみた」要素の少ない紹介記事であるがお許しを(盛大なタイトル詐欺)。

2. Life Game(ライフゲーム)とは?

ライフゲーム (Conway's Game of Life) は 1970 年にイギリスの数学者ジョン・ホートン・コンウェイ (John Horton Conway) が考案した生命の誕生、進化、淘汰などのプロセスを簡易的なモデルで再現したシミュレーションゲームである。単純なルールでその模様の変化を楽しめるため、パズルの要素を持っている。

(Wikipedia より)

というもの、と言ってもわからないと思うので、ひとまずルールを書いておく。説明が簡潔で理解しやすかったため、これもまた Wikipedia から。(一部視認性のために改変済み) 最後に完成形のスクリーンショットを貼っておくので、イメージとしてはそれを参照して欲しい。

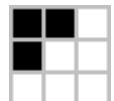
ライフゲームでは初期状態のみでその後の状態が決定される。碁盤のような格子があり、一つの格子はセル (細胞) と呼ばれる。各セルには 8 つの近傍のセルがある (ムーア近傍)。各セルには「生」と「死」の 2 つの状態があり、あるセルの次のステップ (世代) の状態は周囲の 8 つのセルの今の世代における状態により決定される。

セルの生死は次のルールに従う。

また、中央のセルにおける次のステップでの生死の例を示す。生きているセルは■、死んでいるセルは□で表す。

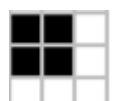
誕生

死んでいるセルに隣接する生きたセルがちょうど 3 つあれば、次の世代が誕生する。



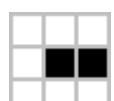
生存

生きているセルに隣接する生きたセルが 2 つか 3 つならば、次の世代でも生存する。



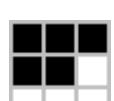
過疎

生きているセルに隣接する生きたセルが 1 つ以下ならば、過疎により死滅する。



過密

生きているセルに隣接する生きたセルが 4 つ以上ならば、過密により死滅する。



3. 取り敢えずやりたいこと

さて、ライフゲームを作るに当たって、画面の確保、描画などは DX ライブラリに丸投げする予定なので、処理機構を考える。

やりたいこととしては、

すべてのセルを見る

→周りにある、生きているセルのカウント

→基準のセルが生きているか確認

→カウントに基づいて生死を決定する

こんな感じ。

セルは 2 次元配列を用意すれば良いが、取り敢えず vector でいいだろう。ここで、セルのカウント後セルの状態を変更するのだが、変更後の状態をそのまま適応すると次のセルについて確認するときに影響してしまうので、コピーを用意してそちらに書き込むことにする。

つまり、セルの状態を保存するモノを 2 つ用意して、1 つ目を見て生きたセルのカウントをし、その結果を 2 つ目に書き込んで最後に反映する。

```
vector<vector<bool>> map(ARRAY_SIZE, vector<bool>(ARRAY_SIZE, false));
vector<vector<bool>> map2(ARRAY_SIZE, vector<bool>(ARRAY_SIZE, false));
```

4. 書いてみよう

とりあえず周辺の 8 つをチェックしたいが、取り敢えず自分を含めた 9 個を for で走査し自分自身を除くことにする。

配列外参照チェックの処理を書くのが面倒なので、配列外参照をしたときに例外を投げてくれる at メンバ関数で若干楽をしてみる。

下に挙げるるのはライフゲームの核となる部分。

```
for (int y = 0; y < ARRAY_SIZE; y++) // ARRAY_SIZE: これの自乗個の ↘
    for (int x = 0; x < ARRAY_SIZE; x++) // マス上で動かす。
        int count = 0;
        for (int y2 = -1; y2 < 2; y2++) {
            for (int x2 = -1; x2 < 2; x2++) {
                try
                {
                    if (map.at(y + y2).at(x + x2)
                        && !(x2 == 0 && y2 == 0)) count++; // 周辺の生きたセルの
                                                // 数
                }
                catch (const std::exception&) {} // ここで若干楽をする
            }
        }
        if (map[y][x]) // ↓ 過疎および過密判定
            if (count <= 1 || count >= 4) map2[y][x] = false;
        else if (count == 3) map2[y][x] = true; // 誕生部分
```

```
    }  
map = map2;
```

5. 改善

ここまでにおいては、最初のパターンを起動時に配置していたのだが、自由に操作するため書き込みモードと実行モードの2つのモードを用意することにした。

enum で WRITE,MAIN の2つを宣言し switch-case で管理する。

また垂直同期待ちのトグル、マウスホイールによる更新頻度の調整機能も付けた。

ここでとある先輩から上下左右の端を繋げてはどうかとアドバイスが来たので、先程の周囲の8つをチェックする部分をこのように書き換えた。

```
if (map.at((ARRAY_SIZE + y + y2) % ARRAY_SIZE).at((ARRAY_SIZE + x + x2) % ARRAY_SIZE) && !(x2 == 0 && y2 == 0))count++;
```

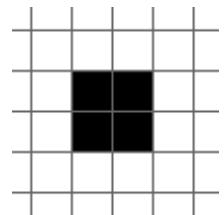
これで範囲内に収まるようになった。

6. パターンの紹介

ライフゲームのルールは前述したようにかなり簡素なものだが、これだけで様々なセルの動く様子を観察することができる。Wikiにもたくさん例があるのだが、ここでも取り上げておく。

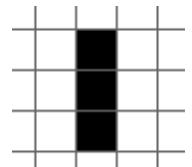
- ・固定物体

名前の通り、変化しないパターンである。その代表格が「ブロック」これはそれぞれの生きている(黒)セルの周辺に3つ生きているセルがあるので、どれも生存し続ける。



- ・振動子

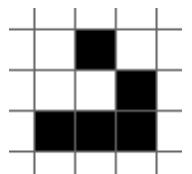
一定の周期で元に戻るパターン。代表格としては「ブリンカー」誕生と過疎を繰り返し、縦と横交互に縦棒ができる。



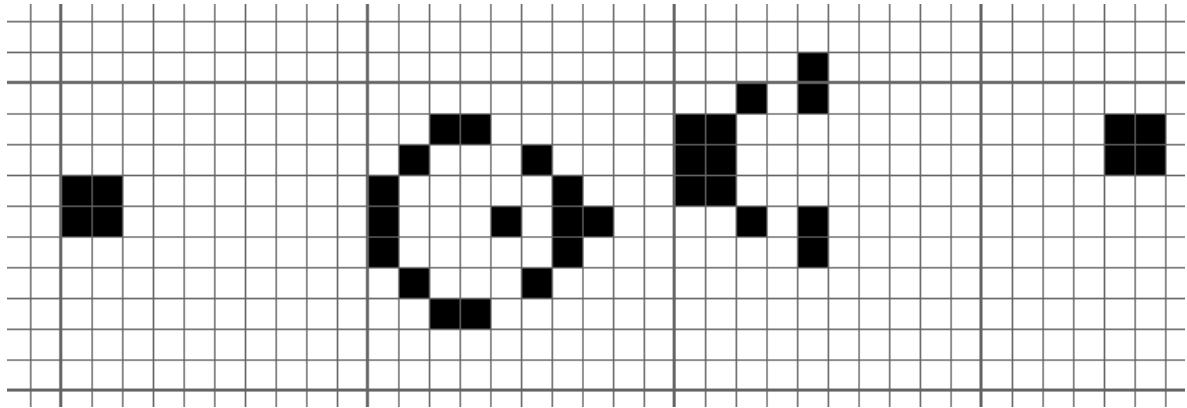
名前の由来は、画面の更新頻度が高いモニタでこれを観察するとかなり点滅して見えるからである。

- ・移動物体

同じような形を保ったまま移動するパターン。代表格が「グライダー」高確率で発生しやすく、最小であるため大人気。



・繁殖型



スペースさえあれば無限にセルが増え続けるパターン。代表格は「ゴスペル(又はゴスペー)のグライダー銃」これは無限にグライダーを打ち続ける機構で、初めて発見された増殖型であるため発見者の名前がつけられた。

7.まとめ

ライフゲームを楽しめる Golly というフリーソフトがあり、こちらはかなり高速に動かせるのでオススメしたい。後はニコニコ動画に「ライフゲームの世界」というタイトルの分かりやすく面白い動画があるので、こちらを見るとなおライフゲームの面白みが分かるかと思う。

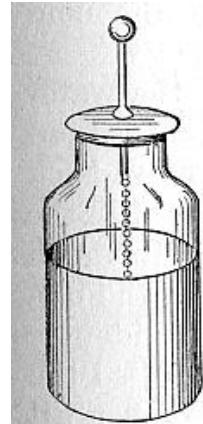
電気学の発展

中学二年 藤山

突然ですが皆さん、勉強などの、作業の間の息抜き、きちんとできていますか?この部誌では他の人が物理のことについて真面目に述べているので、この記事はその息抜き的な記事にしようと思って書いています。ということで僕たち、電子工作班員達の活動の基礎の基礎となっている、電気学の発展の概観を見ていきたいと思う。

1 ~ 静電気蓄電池 ライデン瓶 ~

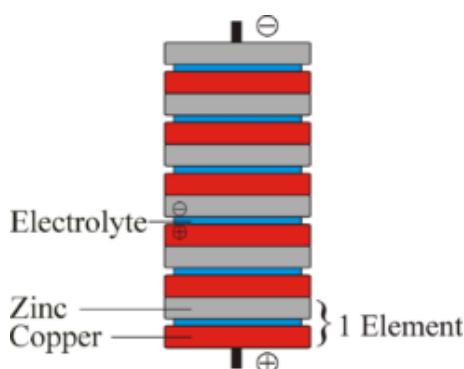
1746年、オランダのピーテルがライデン大学で静電気を貯める「ライデン瓶」を発明した。ライデン瓶はガラス瓶(絶縁の要素)の内側と外側を金属(主に鉛)でコーティングし、内側のコーティングは金属製の鎖を通して終端が金属球となっているロッドに接続されているつくりになっており、瓶の上の球に電圧加えることによってガラスによって絶縁された導体の表面に電気が溜められるのである。溜まった電気は静電気であるが、このライデン瓶の発明によって、人々が静電気を蓄えられるようになったのである。



2 ~ 電池の発明 ガルバーニとボルタ ~

1799年にイタリアの物理学者、ボルタが電池を発明した。そして、そのきっかけは1791年のガルバーニが発表した動物電気の研究に遡る。ガルバーニは解剖されたカエルを使って実験をしていたところ、偶然不思議なことに気が付いた。二種類の異なる金属を接合してつくった物の両端がカエルに当たったところカエルが痙攣を起こしたかのように動き足したのである。これを発見したガルバーニはこれを動物から生み出された「動物電気」と言って論文にし、発

表した。しかし実はこの動物電気、動物から生み出された電気ではなかったのである。二種類の金属が湿った物に触れた際に生じた電圧の差が原因であったのだ。そしてカエルの痙攣はただの検電の動きにしか過ぎなかったのである。



そこで、ボルタは銅と亜鉛を用いて電池を発明した。この発明は強い電流を安定して長時間供給することができるようになり、今まで未開拓な領域に留まっていた電気の研究を急速に発展させていくことになる。

3 ~ センスの塊 マイケル・ファラデー ~

ここで、電磁気学の礎を築いた人たちの内の一人である、イギリスの科学者、マイケル・ファラデーである。少しファラデーの生い立ちと、前半生を見てみよう。ファラデーが生まれた家庭は、あまり裕福な家庭ではなく、高等な教育を受けておらず、数学の知識を身に着けてい

なかった14歳の時から、製本業に就いていた。ファラデーが製本業に就いたのはとても偶然なことであったが、この偶然がファラデーの人生と、科学の歴史を大きく変えることになるのである。

ファラデーは仕事の合間を縫って製本を依頼された本を読んだのである。特に電気と化学の分野に興味を持った。そして、このこともまた、一つの偶然であった。なぜなら、電気と化学の分野はそのころまだ発展途上であって、力学のように、理論で完全に一般化された分野とは違い、未開拓の分野であり、高等教育を受けていなく数学がまったくと言ってもいいほどできないが、実験のセンスに優れているファラデーが活躍できる余地が少なくなかったのだ。ファラデーの勉強の意欲に火がつき、なんとかして実験ができる環境に身を置きたいと考え、1812年に20歳になったファラデーは、電気分解を駆使して多くの元素を発見したデービーの講演を熱心に聴講し、デービーの行った実験をスケッチをまじえてノートに書き留め、デービー宛ての手紙と製本職人が製本したデービーの講義のノートを同封して、デービーに送ったのである。これが功を奏し、1813年3月からデービーのもとで働くようになり、自由に実験を行えるようになり、数々の功績を残していくことになる。

その後、ファラデーは様々なことを発見していったが、中でも有名なのは、1831年の磁力線の着想と、同年の電磁誘導の法則の発見だろう(厳密に言うと、少し先にジョセフ・ヘンリーが発見している)。電磁誘導というのは、磁界が変化している環境にある導体に電圧が生じる現象(簡単に言うと”発電”)であり、この現象は発電機や、変圧器などの多くの電気機器の動作原理となっている。もちろん、この電磁誘導の法則も高等な数学は一切使われていなかった。

～ 理論家 マクスウェル～

その年6月、後にファラデーの行った実験を理論にする、マクスウェルが生まれた。マクスウェルは大地主跡取りに生まれ、ケンブリッジ大学を卒業した、典型的なエリートだった。彼は数学に長けていてそれを駆使して理論を築き上げるのである。貧困層に生まれ、あまり学歴のないファラデーとは実に対照的な人物である。

マクスウェルは若いころから才能を發揮している。1856年、マクスウェルは25歳で「土星の環の構造と安定性」について取り組み論文を提出し、翌年にアダムズ賞を受賞している。また、マクスウェルは1855年から1856年にかけて、「ファラデーの力線について」と題した研究をし、論文にまとめると、真っ先にそれを力線の提唱者ファラデーに送った。これを機に、後にファラデーとの交流が生まれていく。

さらに、マクスウェルは理論的に電磁波の存在を予想し、そして、マクスウェルは1864年に、ファラデーの電磁誘導の法則、アンペール=マクスウェルの法則、電場に関するガウスの法則、磁場に関するガウスの法則の4つの法則を結合し、今日、”マクスウェル方程式”と呼ばれる電磁気学の基礎方程式が確立したのである。この方程式は、アインシュタインが、特殊相対性理論の基本原理となった、光速度不变の原理を提唱する際にも役立っており、電磁気学の

非常に重要なものとなっている。

ちなみに、それからマクスウェルはエーテルという、光が伝播する際に媒質となる、仮想媒質があるのを前提として波動方程式を求めたわけなのだが、このエーテル、今ではエーテルは存在しないと証明されたのである。しかし、この方程式そのものは正しいものである、という面白い展開になっている。このようなことも歴史を学ぶ上での面白さである。

このように、興味のある分野の歴史、あるいはそれに関わったたくさんの偉人達の生涯(特に、幼い頃は自部が後世に名を残すことなんて思ってもいなかった人がほとんどであるため、素の人間が見えて、より面白い)や考え方を知ることは、僕たちにいろんなことを考えさせてくれる、また、考える時の参考にもなるのである。そして何より、学校の歴史の勉強とは一味違う面白さで、調べていて楽しい!皆さんも、息抜き程度に調べてみるのはいかがですか?最後まで読んで下さりありがとうございました。

参考文献:

『光と電磁気 ファラデーとマクスウェルが考えたこと 電場とは何か? 磁場とは何か?』 小山慶太 著 発行所 株式会社講談社

<https://ja.wikipedia.org/wiki/電磁波>

<https://ja.wikipedia.org/wiki/マクスウェルの方程式>

http://www.xjzyz.com/old/electromagnetic_wave.html

<https://ja.wikipedia.org/wiki/相対性理論>

[https://ja.wikipedia.org/wiki/エーテル_\(物理\)](https://ja.wikipedia.org/wiki/エーテル_(物理))

<https://ja.wikipedia.org/wiki/ジェームズ・クラーク・マクスウェル>

<https://ja.wikipedia.org/wiki/ハンフリー・デービー>

<https://ja.wikipedia.org/wiki/電気分解>

<https://ja.wikipedia.org/wiki/マイケル・ファラデー>

<https://ja.wikipedia.org/wiki/ライデン瓶>

<https://ja.wikipedia.org/wiki/ボルタ電池>

第一章「コイルの用法！コイルガンの逆襲」

こんにちは。中二の鈴木です。今回の物理部展ではコイルガンを出展しましたが、それと関連しコイルの原理を物理的な観点から説明し、その応用としてコイルガンの解説をします。難解な語句も頻出しますが、そこは注訳を参考にして読み進めて頂けると幸いです。

1.コイルとは何か

そもそもコイルとは何か？コイルとは紐状の物を渦巻き状、らせん状に巻いた物をそう呼ぶ。コイルはバネとして利用する時、圧縮コイルの場合は圧縮すると自己の弾性力で抵抗する力が、引張コイルの場合も引っ張ると自己の弾性力で抵抗する力が生まれる。コイルばねは荷重に比例して弾性力も増加し、なおかつ生産が安価で可能な為幅広い分野で利用されている。一昔前までは乗り物等にも客室の動搖を低減するために利用されていた。（今は空気ばねが主流だが…）

電磁石としての利用

電磁石として利用する場合、導体に電流を流すと僅かな磁力が発生するが、その力をコイルとして巻かれた導体で発生させる。すると纏まった磁力、要は磁性体を近づけると反応したりするような強さの力になる。これは巻き数を増やせば増やすほど増大する（注：これは導線が電気抵抗のない理想的な状態でのことであって、実際には巻き数を増やすと電気抵抗が増大するため電流が減り、実際の磁力が単純に増えていく訳ではない）。

電磁石の利用法としては、主にモーター、リレー（継電器とも呼ぶ）、ソレノイド等が一般的に挙げられる。あまり一般的ではないが、ここでは電磁石の利用法としてコイルガンを解説することにしよう。

2.コイルガンとは何か

コイルガンとはその名の通りコイルの磁力で弾丸を加速させ、対象物に発射するものである。中空状のコイルに磁性体の弾丸を押し込み、同時に電流を流すと、コイルが磁気を帯びるため弾丸は中央方向に吸引される。しかし、そのままでは弾丸が中央で留まってしまい、弾丸を発射することは出来ない。そこで、弾丸が中央付近まで吸引されたと同時に電流を切り、そのまま慣性で発射してしまうのだ。この仕組みを実現するために主に大容量、高耐電圧のコンデンサーを半導体スイッチで一瞬だけ電流を流す。機械的なスイッチでは損失が大きく、また大電流で接点部分が溶着してしまうことも考えられるためこのような大電流を扱う分野で利用することは少なくなっている。

3.コイルガンの効率の低さと、その解決法

この単純なコイルガンでの効率は5%程度しか出せない。理由としては主にコイルの磁力不足、コイルの電気抵抗による損失、コイルと弾丸との距離やデンサーの通電時間とコイルの長さが合わず、弾丸が引き戻されてしまうことが挙げられる。

コイルの巻き数を増やす

コイルの磁力不足の場合、コイルの巻き数を増やすと磁力が増し、その分弾丸を吸引する力は増大する。しかし、その場合電気抵抗が増大し損失が増え、また通電時間が長くなりコイルの長さを調整しないと弾丸が中央を越えても通電し続け弾丸が引き戻されてしまう欠点がある。

弾丸を誘導しコイルを巻くパイプを薄いものにする

パイプを薄いものにすれば、その分弾丸に働く磁力は増大するが、パイプの強度が下がりコイル巻きの途中や使用中に折れてしまう危険がある。そこで薄く、丈夫なアルミパイプを使うと強度は上がるが渦電流が流れ効率が下がる原因になる。渦電流とは、伝導体を磁場内で動かしたりして生ずる渦状の誘導電流である。これが物体の運動を妨げる力として働いてしまい、コイルガンにとってはやっかいなものだ。極端な話モーターの鉄芯と同様にケイ素鋼板を積層させたものを使う等の対策をすれば良いのだが、それを個人レベルでやるのはほぼ無理であるため、大抵はアクリル等のプラスチック製パイプを利用する。アクリルだからと言って、弾丸の動きは速すぎて見ることも出来ないことは断っておこう。

コイルの抵抗を減らして通電時間を短くする

コイルの電気抵抗を減らすとそもそも電気抵抗による損失が減り、その影響で電流が増大し結果的に通電時間も短くなる。しかし、一般的にコイルに利用されるエナメル線の導体は銅。これ以上電気抵抗を減ずるには金か銀を利用するしかないが、当然高価すぎるので無理な話である。導体断面積を大きくすれば電気抵抗は減るが、巻き数が減るため磁力も減る。超伝導を使うなんて夢のまた夢だ。

コイルをコンデンサーの容量、電圧に合った長さにする

コイルの長さを調整すれば、通電時間が長すぎてコイルが引き戻されることは避けられる。それが一番確実である。

コイルとコンデンサーの回路を増やし多段式とする

また、多段式とすると一つ一つのコンデンサーを小容量化しても威力を確保することが可能になり、総じて制作価格を低減させられる。さながら一つのコイルガンで加速した弾丸を次のコイルガンで加速しているようである。しかし、二段目以降のコイルは電流を流すタイミングが重要になり、タイミングを誤ればかえって効率を低下させる原因にもなりかねない。対策と

しては、タイマー回路やマイコン制御で通電間隔を調整する、或いは赤外線ダイオードとフォトトランジスタで弾丸を検出することが考えられる。しかし前者は調整がとても面倒で、後者は赤外線の扱いの難しさがある。

4. コンデンサーを効率良く利用するための工夫

コンデンサーの容量は静電容量×電圧²で、電圧が高い方がより多くのエネルギーをためることが出来る。その高電圧を得るため大抵のコイルガンでは昇圧回路を挟み、250~400V程度に昇圧してからコンデンサーに電力をためる。昇圧回路には昇圧チョッパ回路が利用される。昇圧チョッパ回路は直流電源とインダクタ、またインダクタ～GND方面と負荷の方面で切り替えられるスイッチで構成される。実際はそのような機械的なスイッチは損失が大きいため片方で半導体スイッチにより On/Off を制御し、もう片方はダイオードで逆流を防ぐ。スイッチが On の時、電源～インダクタ～GND が導通しインダクタに電流が蓄えられる。スイッチを Off にすると電源～インダクタ～負荷～GND が導通するが、インダクタに蓄えられた電流はすぐに無くなることがないので、ダイオードを介して高圧側に注入される。これで昇圧回路が成立する。

このように、一通りコイルガンの仕組みについて解説してみた。まだ経験の浅いこともあり違和感のある説明になってしまったかもしれないが…

参考にさせていただいたサイト

EML 制作記録(仮)『コイルガンの作り方というか考察』

<http://emllaboratory.blog.fc2.com/blog-entry-9.html>

多段式コイルガン『多段式コイルガンの仕組み』

<http://yorozulab.web.fc2.com/msc/mscoil.html>

Energy Chord 『昇圧チョッパ回路の原理イメージ』

http://energychord.com/children/energy/pe/dcdc/contents/dcdc_boost_intro.html

第二章 「AMD 反撃！シェアを犯される Intel」

どうも、二回目ですが中二の鈴木です。それだけですが…

ここ最近 CPU 市場は Coffee lake や Zen+に沸いていますね。今後も Intel の 8 コアや 7nm Zen が楽しみですね。ここでは、以前の Intel の Sandy Bridge ベースのプロセッサや、AMD の Bulldozer ベースのプロセッサとは違う Zen ベースコアの仕組みについて解説します。

Zen アーキテクチャは何が違うのか

そもそも Zen とは、AMD が Core i で Intel にシェアを奪われる中 2012 年に開発が始まったアーキテクチャだ。Zen の開発にはかの有名な天才開発者、ジム・ケラーが関わっており、AMD にとっては Bulldozer 以来のアーキテクチャの更新となった。Zen ベースの CPU は競合する Core i 系のプロセッサより多コア、低価格なことが一般的に言われている。まずもってそれはどうしてなのか、という話である。

回路構成の効率化

Zen アーキテクチャは Intel の競合する価格帯の製品よりダイサイズが小さい。半導体を製造する上でダイサイズは非常に重要で、歩留まりにも大きく関わってくるのだから小さければ小さいほど良いのだ。

「ロードマップでわかる！当世プロセッサー事情」では、「そもそも昨今では、14nm プロセスと言いつつも、実際には 14nm の寸法になっている部分は 1 つもない。その代わりに Fin Pitch(プレナー型トランジスタの場合は Transistor Pitch などとも呼ぶ)と CPP を使ってプロセスの大きさを判断することが普通だ。

これに関しては以下の経験則(ASML Formula : ASML の法則)がある。

$$\text{ノードサイズ} = 0.14 \times (\text{CPHP} \times \text{MMHP})^{0.67}$$

CPHP は CPP の半分、MMHP は FinPitch の半分をそれぞれ示す値で、上の数字を使うと Ryzen(GlobalFoundries の 14LPP)は 13.70nm、Skylake(インテルの 14nm)は 11.66nm 相当になる。注1 と分析している。

プロセスの微細化という面では Intel の方が攻めた構成を採用しており、AMD の方が微細化のみで見れば Intel よりダイサイズが大きくなるはずである。理由はどういうことか、それは、アーキテクチャの更新で不要な回路を設計段階で省き (Intel のメインストリーム用途プロセッサではハイエンドデスクトップ用途やサーバー用途の CPU で必要な回路を無効化して居る場合が多い)、同じ回路でもダイサイズを小さくすることを成功させているのだ。こうした効率化で低価格化を実現しているのだと推測する。Zen アーキテクチャの欠点として、コア間レイテンシが挙げられるが、実測値でそこまで処理速度に悪影響を与えてはいるのではないと

されている。また、シングルコア性能の低さも挙げられるがそれは今後のアーキテクチャの改良によって改善されていくことだろう。実際、Ryzen 1000 シリーズより 2000 シリーズでは微増とはいえシングルコア性能が改善している。Zen2 にも期待だ。個人的にはこの Bulldozer の失敗を生かしコンピューターでの開発に頼ることなく、人の手で設計したのも大きい感じている。実際、Bulldozer はアーキテクチャとしての出来上がりは微妙な物だった。また、近年のアーキテクチャの微細化、効率化でダイサイズは組み込み用途で利用されるアーキテクチャの大きさに近づいている。微細化で製造コストが上がったのも一つ理由だが、わざわざ開発リソースを割き組み込み用とのアーキテクチャを開発するより、メインストリーム用途のアーキテクチャをそのまま組み込み用途にも利用できた方が得策と判断したのだろう。Zen アーキテクチャの消費電力を考えれば十分可能な話だ。

Zen 発表を受けてからの Intel の対応

一方で、Ryzen で反撃された Intel も黙ってみているわけではない。元々 Ryzen 1000 シリーズ(Summit Ridge と呼ばれる)が出るまではハイエンドデスクトップ用途は Core i7 68xx,69xx(Broadwell-E と呼ばれる)で対応していたのが、Ryzen 7 1800X は Core i7 6900K と、Ryzen 7 1700X は Core i7 6850K と、Ryzen 7 1700 は Core i7 7700K と太刀打ちできるレベルと宣伝されてしまったが為にいくらかシェアを AMD に奪われてしまった。実際、Intel は AMD が Ryzen の販売前に製品のレベルが低いと自作でも、メーカーでも採用する例が減ってしまったが為に Intel のシェアが増え、Intel の独占状態が続き Intel も価格をつり上げていた状態があり、AMD はうまいことその点を突いたのだ。筆者も自作機を組む予定で最初は Core i7 6850K を使う予定で居たのが Ryzen 発表で Ryzen 7 1800X で組むことにしたのだ。1 年半経った今では Coffee lake や Pinnacle Ridge といったより優れた製品が市場に送り出されているが、当時としたらたいそう驚いた物であったことを記憶している。並列処理については依然強いので例えばゲーム 2 つ、ブラウザ、動画再生なんてことも楽々出来ている。Intel は対抗製品として Sky lake-X や Kaby lake-X を発表し、それが Intel 特有の高い消費電力と、何しろ今まで消費者を舐めきっていたのかと思わせるほどのプライスダウンだった。結果。Intel と AMD の価格競争構図が生まれ Ryzen を買わないユーザーでも恩恵を受けたのは大きい感じている。今後もこの競争には注目できる。

今後のプロセス微細化の展望

話が変わるが、今度はプロセスルールの微細化の話だ。現在半導体メーカー各社はプロセスの微細化を続けており、現在は 16nm(NVIDIA)、14nm(Intel)、12nm(AMD)程度まで微細化が進んでいる。この微細化競争の背景にあるのは「ムーアの法則」と呼ばれる経験則が根底にある。（よくムーアの法則は物理法則と誤解されることがあるが、ムーアの法則は集積回路上

の部品辺りのコストが最小になるような回路の複雑さを定義した経験則に基づいた将来予測であり、誤解してはならない）しかし今日「ムーアの法則は終わった」と言われることがあるが、それはどういうことなのだろうか。

これまでの Intel の微細化

Intel は元々ムーアの法則に基づき、Broadwell で 22nm から 14nm へ微細化させ、Sky lake でそれを最適化した回路を構成し、Cannon lake で 10nm に微細化し、Ice lake で最適化、Tiger lake で…俗に言う Intel Tick-Tock である。

10nm で製造された製品は果たして発表されるのか

Skylake までは何とか Intel Tick-Tock を維持させてきた（厳密に言えば Broadwell もかなりずれ込んで一回 Haswell-Refresh を挟んでいる）が、Sky lake の先、Cannon lake、もとい 10nm の開発が遅れたことや 14nm の設備投資の回収が進んでいないことから 2016(2017)年の新製品は Kaby lake で 14nm+として出荷されることになった。おまかせ。と Intel が出してきた次期プロセッサは…「14nm++しか無かったけど…良いかな？」そう、本命の 10nm Cannon lake ではなく Coffee lake だった。選ばれたのは、14nm++でした。（実際 10nm よりも 14nm++の方が、性能が良くなるという話もあったことにはあったのだが…）Intel の発表では、かなり大きな壁に当たったこともあり 10nm は当分延期になるようだ。また、Coffee lake は Ryzen への対抗製品の意味合いもあり、プロセスの微細化よりも性能向上を優先させメインストリーム系プロセッサでの 6 コア化を実現させた。そして、現在に至るのだ。

これまでの AMD の微細化

対して AMD は、Bulldozer 以来停滞していたアーキテクチャ開発を新規で行い、Intel に少し遅れをとったものの 14nm 化を果たした。14nm 化で低消費電力を実現、また、ダイサイズの最適化を行った。AMD の資料だと、2017 年発表の Zen で 14nm、2018 年発表の Zen+ で 12nm、その後 2020 年まで Zen2 で 7nm、Zen3 で 7nm+となっている。要は 10nm はスキップし、7nm まで直接行ってしまうのだ。どうやら CPU ではなく GPU で Vega 7nm のサンプル品の製造が出来ているようなので、そっちの心配は要らないと思われる。

プロセス微細化は今後も最善の手段となり得るのか

そもそも、半導体の微細化は回路の複雑さを向上させ、なおかつ製造コストを下げるための最善の手段として今まで行われてきたことであり、他の手法が見つかれば微細化に拘る必要は無い。半導体メモリは 3D-NAND という手法が考案され微細化競争は下火になっている。

ムーアの法則を維持させるために（ムーアの法則を維持するのが目的ではないが）、各社は様々な技術を投入し微細化、もとい回路の複雑さを進めてきた。今まで何度も「ムーアの法則は終わった」と言われ続けながら…

注1 「ロードマップでわかる！当世プロセッサー事情 — 第398回 Ryzenが消費電力を削減できた仕組み」<http://ascii.jp/elem/000/001/449/1449995/>

素数の探し方

吉田安紀彦

1. はじめに

素数と聞いて、「2,3,5,7,11...のことだよね」と分かった人も多いと思いますが、では、現在までに見つかっている最も大きな素数はご存知ですか？じつは、その素数は2千万桁以上にもなります。どうやってその数が素数であることがわかったのか、考えてみるとふしぎではありませんか？2千万桁の数を2からひとつずつ割っていくわけにもいかないし。ということで、ここでは素数の判定方法などをまとめてみました。なお、この文章での「現在」とは2018年7月22日のことです。

2. 素数とは

2,3,5,7,11,13,17,19,23,29,31…というような数のことです。

素数の定義は色々ありますが、簡単に言うと「その個数のおはじきを長方形に並べることができない数。ただし1列に並べるのは除く」と言えると思います。

3. 素数の記録

2017年12月26日、今までに見つかっている素数の中で一番大きな素数の $2^{77232917} - 1$ が発見されました。これは23249425桁の素数で、1ページびっしりに印刷しても719ページの本になります。（実際にAmazonで売られていたり…）

このような素数の記録は「The Prime Pages」(<http://primes.utm.edu/>)で管理されています。

4. 素数の種類

素数と言っても、ただのランダムな素数や、きれいな数式で表される素数、ある数列の中に出てくる素数など様々な種類があります。といっても整数の種類はたくさんあるのでここで紹介するのはほんの一部です。

メルセンヌ素数 (Mersenne Prime) $2^n - 1$

先ほどの $2^{77232917} - 1$ がその例です。現在50個のメルセンヌ素数が発見されています。また、現在発見されている素数の大きい方から10個のうち、9個がメルセンヌ素数です。
 $n=2,3,5,7,13,17,19,31,61,89$ の時などがメルセンヌ素数です。

フェルマー素数 (Fermat Prime) $2^{2^n} + 1$

n が少し大きくなるだけで $2^{2^n} + 1$ は大きくなってしまうので、探すのが大変な素数の一つです。

n が 32 のとき (1292913986 枠) までは素数かどうかが分かっていて、そのうち n が 0, 1, 2, 3, 4 のときのみが素数です。 n が 4 よりも大きいとき素数になるかどうかはわかっていないのですが、初めの 5 個だけ素数で、その後には素数がまだ見つかっていないっていうのはちょっと不思議な感じがします。

プロス素数 (Proth Prime) $k \cdot 2^n + 1 (2^n > k)$

メルセンヌ素数のところで、現在発見されている素数の大きい方から 10 個のうち 9 個がメルセンヌ素数だと書きましたが、その残りの 1 個がこのプロス素数の $10223 \cdot 2^{31172165} + 1$ です。

双子素数 (Twin Primes)

隣り合っている 2 つの奇数 (p と $p+2$) の両方が素数である数です。

現在見つかっている中で最大の双子素数は、2016 年 9 月に発見された $2996863034895 \cdot 2^{21290000} \pm 1$ (388342 枠) です。

階乗素数 (Factorial Primes) $n! + 1, n! - 1$

一瞬、この数は n までは割れないので、全てが素数なように見えますが実は少なく、 $n! - 1$ は 27 個、 $n! + 1$ は 22 個しか見つかっていません。現在見つかっている最大の階乗素数は、2016 年 7 月に発見された $208003! - 1$ (1015843 枠) です。

2018/7/22 現在 $n = 250976$ まで探索が終了しています。

(http://prpnet.primegrid.com:12002/server_stats.html)

素数階乗素数 (Primorial Primes) $n\# + 1, n\# - 1$

2 から順に n までの素数をすべてかけた数を $n\#$ と表します。

この数も、見た目よりはるかに素数になりにくく、 $n\# - 1$ は 20 個、 $n\# + 1$ は 22 個しか見つかっていません。現在見つかっている最大の素数階乗素数は、2012 年 3 月に発見された $1098133\# - 1$ (476311 枠) です。

2018/7/22 現在 $n = 2551729$ まで探索が終了しています。

(http://prpnet.primegrid.com:12008/server_stats.html)

レピュニット素数 (Repunit Prime) $(10^n - 1)/9 = 11111111\dots11$

1 だけでできた数、Repeated unit (反復単位数) を縮めてレピュニット (Repunit) と呼び、この数が素数の時、レピュニット素数といいます。

現在見つかっている最大の素数 (確率的素数) は、2007 年 7 月に発見された $(10^{270343} - 1)/9$ です。

現在、 $n = 2500000$ までの素数の探索が完了しています。

(<http://www.elektrosoft.it/mathematica/repunit/status.htm>)

ニアレピュニット素数 (Near Repunit Prime)

レピュニット数の数字をひとつだけ別の数に置き換えた数です。

現在見つかっている（たぶん）最大のニアレピュニット素数（確率的素数）は 2014 年 12 月に発見された $(64 \cdot 10^{762811} - 1)/9 = 711111\dots111$ です。

ニアレプディジット素数 (Near Repdigit Prime)

すべての位が同じ数のことをレプディジット数とよびます。そのレプディジット数の数字をひとつだけ別の数に置き換えた素数です。現在見つかっている最大のニアレプディジット素数は、2013 年 9 月に発見された $2 \cdot 10^{1059002} - 1 = 199999\dots99$ です。

回文素数 (Palindrome Prime)

上から読んでも下から読んでも同じ素数です。

現在の最大は 2014 年 11 月に発見された $10^{474500} + 999 \cdot 10^{237249} + 1 = 100000\dots00000999000\dots00001$ です。

フィボナッチ素数 (Fibonacci Prime)

フィボナッチ数列とは、1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144…のことです。このうち素数のフィボナッチ数のことです。現在 51 個の確率的素数が見つかっていて、その最大は 3340367 番目の 698096 衍の数です。また、この 51 個のうち 34 個は素数であることが証明されています。

Smarandache 素数 (Smarandache Prime)

1234567891011121314151617…というふうに 1 から順に自然数を並べた素数です。

(1, 112, 123, 1234, 12345, 123456, 1234567, 12345678, 123456789, 12345678910, 123456789101 1...) この形の素数はすぐに見つかりそうですが、現在、少なくとも 344869 まで探索され、まだ一つも見つかっていません。

ちなみに、12345678901234567890123456789012…が素数なのは、 $n = 171,277,367,561,567,18881$ です。（これ以降は 400000 まで探索しましたが、まだ見つけていません）

Wall-Sun-Sun 素数 (Fibonacci-Wieferich 素数)

同じ素数に 2 つも名前がついていますが、現在まだ見つかっていません。

Wilson 素数 (Wilson Prime)

p を素数として、 p^2 が $(p-1)! + 1$ を割り切るとき、その素数 p を Wilson 素数といいます。現在、20,000,000,000,000 までの素数 675,895,909,271 個が調べられましたが、 $p = 5,13,563$ しか見つかっていません。（<https://arxiv.org/abs/1209.3436>）

Wieferich 素数 (Wieferich Prime)

p を素数として、 p^2 が $2^{p-1} - 1$ を割り切るとき、その素数 p を Wieferich 素数といいます。現在、607,131,900,000,000,000までの素数 15,208,298,813,191,373 個を調べられましたが、 $p=1093,3511$ しか見つかっていません。http://u-gf.de/PRPNet/wrange_stats.php?proj=WFS

5. 素数探索の歴史

1588年、イタリアの数学者 Pietro Cataldi が $2^{17} - 1 = 131071$ と $2^{19} - 1 = 524287$ を発見しました。なんと手計算で、しかもローマ数字で 743 までの素数表をつくり、それで 1つ1つ割って発見しました。

1772年、18世紀を代表する数学者のオイラーが、 $2^{31} - 1 = 2147483647$ を発見しました。 $2^{31} - 1$ の約数は $248 \cdot n + 1$ か $248 \cdot n + 63$ の形でなければならぬことを見つけ、実際に割ってみて発見したといわれています。

1867年、Landry が $(259 \cdot 1)/179951$ が素数であることを見つけました。この素数はメルセンヌ数以外の素数として、長く記録に残りました。

1876年、リュカ(Lucas)が $2^{127} - 1 = 170141183460469231731687303715884105727$ を発見しました。リュカ数列を使って 15 歳のときから手計算で素数判定を始め、19 年後ついに素数であることを確かめました。これが、手計算で見つけられた最大の素数です。

1951年、Ferrier が $\frac{2^{148}+1}{17} = 20988936657440586486151264256610222593863921$ を、機械式電卓を使って発見しました。これでコンピューター以前の時代は終了し、その年のうちに Miller と Wheeler がイギリスのコンピューター EDSAC で 79 枚の素数 $180 \cdot (2^{127} - 1)^2 + 1 = 5210644015679228794060694325390955853335898483908056458352183851018372555735221$ を見つけました。

その翌年の 1952 年には、Raphael Robinson がアメリカのコンピューター SWAC を使って $2^{521} - 1, 2^{607} - 1, 2^{1279} - 1, 2^{2203} - 1, 2^{2281} - 1$ を見つけました。このうち最初の 2 つの素数はプログラムを走らせ始めたその日のうちに見つかりました。

その後、1957 年には $2^{3217} - 1$ 、1961 年には $2^{4423} - 1, 2^{4253} - 1$ 、1963 年には $2^{9689} - 1, 2^{9941} - 1, 2^{11213} - 1$ というように順調に記録を伸ばし続けています。

上の $2^{4423} - 1, 2^{4253} - 1$ の 2 つの素数について、面白いエピソードが残っています。この素数を発見した Hurwitz は、2 つのうち大きい方の素数 $2^{4423} - 1$ を $2^{4253} - 1$ より数秒前に知りました。出力用紙が重なっていたからです。素数が「発見された」と言うためには人間がその結果を見る必要があるのか、それともコンピューターが知っていればいいのか微妙なところですが、出力結果を人間が見たときに発見したと言うとすると $2^{4253} - 1$ が最大の既知素数になることはありませんでした。

6. 素数判定法

素数の判定方法は、大きく分けて決定的素数判定法と確率的素数判定法の2つがあります。決定的素数判定法とは絶対に素数であると証明できる判定方法で、確率的素数判定法は素数である確率が非常に高いと言えるが、100%素数であるとは言えない判定方法のことです。他にも、特殊な数に対して判定できる方法などがあります。

試し割り（決定的素数判定法）

\sqrt{N} までのすべての素数で割ってみて、そのどれでも割り切れなかったら N が素数ではないと示すことができます。この方法は最も手軽にできる方法ですが、 N が20桁を超えるくらいになると、2から1000000000000くらいの素数の全てで割らなければならず、現実的な時間で終わらないので、試し割りができるのはせいぜい15桁くらいまでだと思います。

フェルマーテスト（確率的素数判定法）

大きな数の素数判定をするのに最も一般的に使われる方法が、フェルマーの小定理を使ったフェルマーテストです。

フェルマーの小定理とは、

p を素数、 a を p の倍数でない任意の整数とすると、 $a^{p-1} \equiv 1 \pmod{p}$ が成り立つ。
という定理です。

この定理を逆に使って、 n が素数かどうか判定することができます。 a を適当に選んで $a^{n-1} \pmod{n}$ を計算してみて、1ではなかったら絶対に素数ではなく、1だったらおそらく素数だろうと言えます。その1のときの n を、 a を底とする概素数(probable prime)と呼び、a-PRPと書きます。

例えば、77が素数かどうか調べてみます。

$$2^{77-1} = 2^{76} = 75557863725914323419136 = 77 \times 981270957479406797651 + 9$$

余りは9なので、77が素数ではないことが分かりました。

しかし、この方法は100%素数とは言えません。例えば、341は2-PRP（つまり $2^{341-1} \pmod{341} = 1$ ）ですが、 $341=11 \times 31$ は素数ではありません。ほかにも、1,000,000,000,000以下に2-PRPだけでも素数ではない数は101,629個あります。それでも、1,000,000,000,000以下の素数は37,607,912,018個あるので、それと比べると成功率は非常に高いのが分かります。また、複数の底 a に対してテストをすればもっと正確に判定できます。

ただし、 n の約数以外の底に対してはすべてフェルマーテストを通ってしまう、Carmichael数と呼ばれる数があります。このような数は1,000,000,000,000までに8241個と少ししかありませんが、無限個存在することが示されています。

ミラー・ラビン判定法（確率的素数判定法）

Gary Miller氏が発表したMillerテストをベースにして、1976年にMichael Rabin氏が発明した素数判定法です。細かい説明は省略しますが、 n がミラー・ラビン判定法を通った時、 n は a を底とする強概素数(strong probable prime)と呼び、a-SPRPと書きます。

この方法でもおそらく素数だと間違えて判定してしまう可能性はありますが、
1,000,000,000,000 以下の合成数（素数ではない数のこと）のうち 2-SPRP は 22407 個（2-
PRP は 101629 個）、1,000,000,000,000 以下の合成数のうち 2, 3, 5, 7, 11-SPRP は 0 個（2,
3, 5, 7, 11-PRP は 5718 個）というように、フェルマーテストよりも優れています。さらに、
フェルマーテストの時の Carmichael 数のような数が無いことがわかっています。

リュカ擬素数テスト（確率的素数判定法）

リュカ擬素数テスト、Lucas-Lehmer 法、Lucas 擬素数判定法、Lucas pseudoprime test、
Lucas-Selfridge とも呼ばれています。ここで、擬素数(Pseudoprime)とは素数ではないのにお
そらく素数だと判定されてしまった数のことを言います。

リュカ擬素数テストの中でも、複数の判定法があります。大きく分けて I と II の 2 段階があ
り、3 以上の合成数 n に対して I を満たす n をリュカ擬素数(Lucas Pseudoprime)、II を満たす n を
強リュカ擬素数(Strong Lucas Pseudoprime)といいます。さらに、 $Q = -1$ のとき III が成立
する合成数 n を超強リュカ擬素数(Extra Strong Lucas Pseudoprime)といいます。このとき、
 U_t を無視すると、ほとんど超強リュカ擬素数(Almost Extra Strong Lucas Pseudoprime)とい
います。

それぞれの 1,000,000,000,000 以下の合成数のうちの擬素数の個数は、リュカ擬素数は
116928 個、強リュカ擬素数は 25542 個、ほとんど超強リュカ擬素数は 17245 個、超強リュカ
擬素数は 16231 個です。また、Bruckman-Lucas 擬素数は 43039 個、フィボナッチ擬素数は
77188 個、ペル擬素数は 89714 個あります。

Baillie-PSW 法（確率的素数判定法）

BPSW 法とも呼ばれています。この方法は、上 2 つのミラー・ラビン判定法と強リュカ擬素
数テストを組み合わせた判定法です。ミラー・ラビン擬素数で、かつ強リュカ擬素数であるよ
うな合成数しか失敗しませんが、このような数はまだ見つかっていません。

ペラン擬素数判定法（確率的素数判定法）

ペラン数列 (Perrin Sequence) の性質の一つの、 q が素数のとき p_q が q の倍数になることを
逆に使い、 p_q が q の倍数のとき q はペラン擬素数(Perrin Probable Prime)といい、特に合成数の
ペラン擬素数をペラン擬素数(Perrin pseudoprime)といいます。これは非常に少なく、
1,000,000,000,000 以下の合成数のうちペラン擬素数は 285 個しかありません。

Frobenius テスト（確率的素数判定法）

Grantham's Frobenius test、Quadratic Frobenius test、Frobenius-Grantham 法などと
も呼ばれています。

この判定法の中でも複数あり、1,000,000,000,000 以下の合成数のうち、(1,-1) の Frobenius 擬
素数は 37527 個、(3,-5) の Frobenius 擬素数は 1532 個、(P, 2) の Frobenius 擬素数は 0 個、

Frobenius-Underwood 擬素数は 0 個で、この中で(P, 2)の Frobenius 擬素数と Frobenius-Underwood 擬素数はまだ発見されていません。

Kraitchik-Lehmer 法（決定的素数判定法）

ここからは 100% 素数だと言える決定的素数判定法を紹介していきます。

Kraitchik-Lehmer 法は、Lucas の判定法、Lucas primality test、 $n-1$ 判定法とも呼ばれています。素数判定をしたい数 n に対して、 $n-1$ の素因数分解ができていれば、確定的に素数かどうか判定できる、つまり素数であることの証明ができるという判定法です。しかし、これは $n-1$ の素因数分解が完全にできていることが条件です。 $150209!+1$ のようにもともと素因数分解されている数に 1 を足した数についてはこの方法が使えますが、そうではなくてランダムに選んだ大きな数は素因数分解するのが大変なので、そのような数に使うのは厳しいです。

ポクリントンの判定法（決定的素数判定法）

ポクリントンの判定法は、一般 Pocklington 法、Pocklington の判定法、 $n-1$ 判定法、Pocklington-Lehmer test とも呼ばれている方法です。

これは、 $n-1$ の素因数分解が 50% 以上できている、つまり $n-1 = a \times b \times c \times \dots \times R$ (a, b, c, \dots は素数、 R は合成数) というふうに素因数分解できているとき、 R が $\sqrt{n-1}$ 以下のときに使える方法です。その他にも 33.3% 以上素因数分解できている、つまり R が $\sqrt{R/F}$ 以下の素因数を持たないときに使える方法もあります。

モリソンの判定法（決定的素数判定法）

Morrison's test とも呼ばれています。これは、ポクリントンの判定法とは逆に $n+1$ の素因数分解が 33.3% 以上出来ているときに使える方法です。

PPLS 法（決定的素数判定法）

PPLS 法は、Proth-Pocklington-Lehmer-Selfridge 法、 $n-1/n+1$ 法、複合判定法、combined method、combined $n^2 - 1$ test とも呼ばれている判定法です。

これは、ポクリントンの判定法とモリソンの判定法を組み合わせたような判定法で、 $n-1$ と $n+1$ の素因数分解されている部分の積が n の立方根より大きい場合に使える方法です。

ECPP 法（決定的素数判定法）

これは楕円曲線素数判定法、Elliptic Curve Primality Proving 法、ECPP 法とも呼ばれます。この方法は、 $n-1$ と $n+1$ の素因数分解が十分にできていない数 n が素数である証明をするときに最も一般的に使われている判定法で、Goldwasser-Kilian の素数判定法、Atkin-Morain の素数判定法、高速楕円曲線素数判定法などがあります。

AKS 素数判定法（決定的素数判定法）

AKS 素数判定法 (Agrawal-Kayal-Saxena 素数判定法) は、Manindra Agrawal、Neeraj Kayal、Nitin Saxena の三人が 2002 年に発表した判定法で、与えられた自然数が素数かどうか

かを決定性多項式時間でできるという衝撃的な方法でした。ただし発表された当初は、この方法では6桁の素数判定に10秒くらいかかるようなとてつもなく遅い方法でした。現在ではそれが改良されて100桁の素数判定に1秒くらいでできるようになりましたが、まだ橿円曲線素数判定法に比べると遅いです。

ヤコビ和法（決定的素数判定法）

ヤコビ和法は、Cohen-Lenstra法、Jacobi sums test、APR判定法とも呼ばれています。Leonard Adleman、Carl Pomerance、Robert Rumelyが発表した決定的素数判定法で、後にHenri CohenとHendrik Willem Lenstraが改良したものはAPR-CL判定法と呼ばれています。

それをもとに円分判定法（円分環素数判定法、cyclotomic primality testsとも呼ばれている）も作られました。

Lucas-Lehmer判定法（特殊な数に対する判定法）

ここからは、特殊な数に対する判定法を紹介していきます。

Lucas-Lehmer判定法は、L-L法、リュカ・レーマーテスト、Lucas-Lehmer testとも呼ばれている、メルセンヌ素数に対する判定法です。1930年に、D. H. Lehmerがエドゥアール・リュカの判定法を改良して作ったため、こう呼ばれています。 $2^{521}-1$ 以降のメルセンヌ素数はすべてこの方法で見つけられました。最初の $2^{77232917}-1$ もこの方法で素数だとわかりました。この判定法はコンピューターで計算しやすいため、巨大なメルセンヌ素数がいくつも発見されています。

Pepinの判定法（特殊な数に対する判定法）

フェルマー数 $F_n = 2^{2^n} + 1$ に対する判定法です。しかし、フェルマー数は n が少し大きくなるだけでとてつもない大きさになるので、この判定法に当てはめて計算することは難しいです。例えば、まだ素数かどうかわかつていない F_{33} は2,585,827,973桁の数です。これは今見つかっている中で一番大きな素数 $2^{77232917}-1$ の23,249,425桁の100倍以上の桁数です。

Prothの判定法（特殊な数に対する判定法）

プロス数 $h \times 2^k + 1$ （ただし $2^k > h$ ）に対する判定法です。プロス数という名前は、フランスの独学の数学者で農家の Francois Proth に由来します。

LLRテスト（特殊な数に対する判定法）

Lucas-Lehmer-Rieselテストは、 $h \times 2^k - 1$ （ただし $2^k > h$ ）に対する判定法です。メルセンヌ素数の判定法のLucas-Lehmerテストをベースにして Hans Riesel が発展させました。

7. 素数判定法の速度比較

実際に複数の素数判定法の速度を調べてみました。最近の CPU でシングルスレッドでの実行時間です。

10 行から 100 行

全て mpir というライブラリを使用。試し割りは素数に対する判定。

L-L とは Lucas-Lehmer 判定法のことです。

試し割り	2-PRP	2-SPRP	ECPP	L-L
10 行	0.0003162 秒	0.00000207 秒	0.00002015 秒	0.00156 秒
15 行	0.1001246 秒	0.00000241 秒	0.00001970 秒	0.01515 秒
20 行	202.321935 秒	0.00000345 秒	0.00002327 秒	0.03909 秒
30 行		0.00000531 秒	0.00002534 秒	0.26259 秒
40 行		0.00000914 秒	0.00003315 秒	0.78847 秒
50 行		0.00001044 秒	0.00003722 秒	1.47968 秒
60 行		0.00001672 秒	0.00005238 秒	3.37809 秒
70 行		0.00001966 秒	0.00007992 秒	7.95976 秒
80 行		0.00002969 秒	0.00008309 秒	7.75192 秒
90 行		0.00003428 秒	0.00008861 秒	9.15327 秒
100 行		0.00004711 秒	0.00012416 秒	17.60249 秒

100 行から 1000 行

ECPP の 300 行以降は PRIMO を使用した結果を利用。

	2-PRP	2-SPRP	ECPP	L-L
100 行	0.0000471 秒	0.0001242 秒	17.6 秒	0.0000806 秒
200 行	0.0003169 秒	0.0003149 秒	111.3 秒	0.0001596 秒
300 行	0.0008869 秒	0.0008142 秒	8.5 秒	0.0003368 秒
400 行	0.0018953 秒	0.0017754 秒	10.2 秒	0.0005779 秒
500 行	0.0036472 秒	0.0041140 秒	20.7 秒	0.0000323 秒
600 行	0.0061181 秒	0.0072429 秒	52.8 秒	0.0000248 秒
700 行	0.0093117 秒	0.0108036 秒	107.1 秒	0.0000286 秒
800 行	0.0131810 秒	0.0163689 秒	139.1 秒	0.0000246 秒
900 行	0.0181532 秒	0.0232536 秒	293.4 秒	0.0000358 秒
1000 行	0.0244537 秒	0.0286438 秒	388.8 秒	0.0000429 秒

1000 行から 10000 行

2-PRP は PFGW を使用して計測、ECPP は PRIMO を使用した結果を利用。

2-PRP	ECPP	L-L
-------	------	-----

1000 行	0.0130 秒	6.5 分	0.007 秒
2000 行	0.0432 秒	74.3 分	0.042 秒
3000 行	0.1031 秒	499.7 分	0.108 秒
4000 行	0.1585 秒	27.5 時間	0.238 秒
5000 行	0.2715 秒	89.5 時間	0.430 秒
6000 行	0.3864 秒	183 時間	0.656 秒
7000 行	0.5845 秒	13.6 日	1.021 秒
8000 行	0.6637 秒	21.0 日	1.487 秒
9000 行	0.9299 秒	60.1 日	1.731 秒
10000 行	1.2008 秒	31.6 日	2.175 秒

10000 行から 100000 行

2-PRP は PFGW を使用して計測、ECPP は PRIMO を使用した結果を利用。
ECPP の 30000 行の結果は、30950 行の数の素数証明をした結果。

	2-PRP	ECPP	L-L
10000 行	1.20 秒	31.6 日	2.182 秒
20000 行	6.48 秒	242.4 日	11.321 秒
30000 行	12.71 秒	1409.4 日	31.687 秒
40000 行	26.77 秒		67.202 秒
50000 行	35.46 秒		126.009 秒
60000 行	53.08 秒		216.691 秒
70000 行	76.20 秒		285.263 秒
80000 行	99.18 秒		420.646 秒
90000 行	129.23 秒		447.456 秒
100000 行	155.39 秒		573.296 秒

100000 行から 1000000 行

2-PRP は PFGW を使用して計測、L-L は Prime95 を使用した時の推定値。

	2-PRP	L-L
100000 行	2.59 分	0.4 分
200000 行	12.57 分	1.3 分
300000 行	32.90 分	3.2 分
400000 行	65.45 分	7.3 分
500000 行	122.65 分	11.2 分
600000 行	187.86 分	18.9 分

700000 行	281.54 分	22.6 分
800000 行	384.21 分	29.5 分
900000 行	498.08 分	36.6 分
1000000 行	688.06 分	50.3 分

1000000 行から 10000000 行

2-PRP は PFGW を使用して計測、L-L は Prime95 を使用した時の推定値。

	2-PRP	L-L
1000000 行	11.5 時間	0.8 時間
2000000 行		3.0 時間
3000000 行		7.0 時間
4000000 行		14.1 時間
5000000 行		21.8 時間
6000000 行		31.0 時間
7000000 行		42.4 時間
8000000 行		59.0 時間
9000000 行		74.7 時間
10000000 行		90.6 時間

8. 素数判定法の比較

試し割り

素数判定をするのには向いていませんが、大きな数の素数判定をする時、はじめに小さめの素数で割ってみて割り切れないかどうか確認するのにつかえます。

フェルマーテスト

100%素数だとは言えませんが、間違って素数だと判定してしまう擬素数は意外に少なく、大きな数ほど擬素数は少なくなるので、 ± 1 をしても素因数分解できない大きな数についてはこの方法を使うのが一般的です。ミラーラビンテストは比較的小さい数には有効ですが、大きな数についてはフェルマーテストで十分だと思います。

ECPP

± 1 をしても素因数分解ができない大きな数の素数の証明をするならこの方法です。しかし、特別な理由がない限り 2 万桁以上の素数をこの方法で証明するのは現実的とは言えず、これまでに 3 万桁以上の素数は 4 個しかこの方法で証明されていません。

Lucas-Lehmer テスト

とりあえず大きな素数を見つけるならこの方法です。確率的素数判定法のフェルマーテストよりもけた違いに早い上に素数だと証明ができる判定法ですが、メルセンヌ数に対してのみにしか判定できません。

n+1 法、n-1 法

素数であることの証明が、フェルマーテストの1倍から数十倍くらいの計算時間でできますが、 ± 1 が素因数分解できないとこの方法は使えません。

どの素数判定法を使えばいいのか

素数であることの証明をしなくともいいときは、15桁以下くらいなら試し割りをして、それ以上でもある程度まで試し割りをし、その後フェルマーテストを何回かするのがいいと思います。

素数であることの証明が必要なときも、まずは上の方法を試して素数である可能性が高いことを確認してから、 ± 1 をすると素因数分解できる数は n±1 法で、できないなら最終手段として ECPP をするのがいいと思います。

9. おすすめのフリーソフト

OpenPFGW (Prime Form/GW)

どんな形の数でも高速にフェルマーテストや n+1 法、n-1 法ができます。

LLR

$K \times b^n \pm 1$ の形の素数判定が高速にできます。

NewPGen

何種類かの形の素数について、一度にたくさんの数を高速に篩（試し割り）ができます。

Prime95

Lucas-Lehmer テストが高速にできます。1997 年以降に見つかっているメルセンヌ素数はすべてこのソフトによって発見されました。

Yafu

素因数分解などができます。

10. さいごに

実生活で直接は関係ないように思われるこのような素数判定法は、実はインターネットで通信する時の暗号化でも使われていたり、意外と身の回りで使われています。

今回、これをまとめるにあたって色々とネット上を調べましたが、日本語の情報がすごく少なく、さらに英語の情報も少ししかないことがよくあったり、ひとつの判定法にいろいろな名

前がつけられていたりしました。できるだけ正しい情報をまとめたつもりですが、間違っていることもあるかもしれません。その内容も、大学レベルの数学ばかりで正直ぜんぜん理解できていなかつたので、細かい説明や証明を書かずに大雑把にまとめてしましました。わかりにくいうところも多々あったと思いますが、この文章を読んでいただいてありがとうございました。

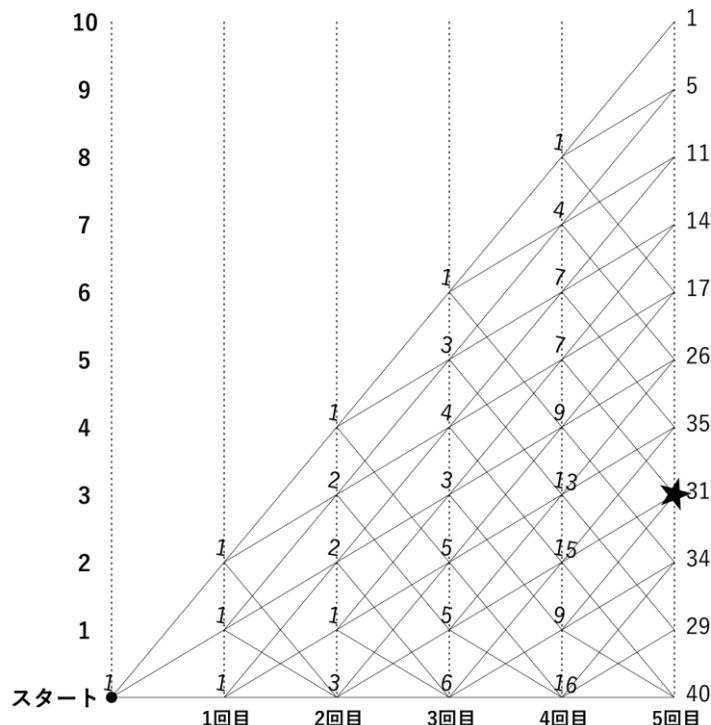
11. 参考サイト・文献など

<http://primes.utm.edu/prove/merged.html>
https://www.ipa.go.jp/security/fy14/crypto/prime_num/invest.pdf
<http://factordb.com/>
<http://oeis.org/>
<http://stdkmd.com/nrr/>
<http://mathworld.wolfram.com/>
<http://www.prime-numbers.net/prptop/prptop.php>
<https://books.google.co.jp/books?id=8KZ4RQufxhYC>
<http://stdkmd.com/blog/2003/04/0301.html>
<https://www16.atwiki.jp/projectpn/pages/42.html>
その他多数。

【コラムの解答】

31通り

例えば、次のように線を引いて、それぞれの点の先のもとにある数をすべて足すということを繰り返すと求まります。



電子工作や、物理のお話など総集編

高1 安藤一生

0. はじめに

こんにちは、こんばんは。副部長兼会計兼電子工作班班長、高1の安藤です。今回が3回目となります。2年前は壁に沿って走るロボットの製作記（懐かしい...）、去年はArduinoについてまとめ、早いものでこれを書くのも来年が最後になってしましました。毎年読んでくれる方がいるのであればただただうれしい限りでございます。この3年間もいろいろありましたね。いつしか、3つも役職を持つとは....。

さて、今年の部誌の内容ですが、これを書いている時点でまだはっきりと決まっておりません（大丈夫なのか？6月5日←提出期限まであと1か月）。はっきりとは言っても去年と違い「はじめに」が「0」なので、方針は決まってはいます。

今年の部誌は短編集形式で行きたいと思います。弁解をさせてもらうと、後に述べる四足歩行のせいで、まとまにはんだ付けをしていない期間が長く、そのブランクのために、1つのことで深く掘り下げるものもなく、また書きたいことも多かったからです。

長い長い前置きもここらで閉じさせてもらおうと思います。

参考文献については、最後にまとめて書いておきます。

0-1 目次

0. はじめに(57p)
1. 4足歩行ロボット製作記(57p~)
2. 超電導（超伝導）について(61p~)
3. 超音波について(63p~)
4. 3Dプリンタについて(64p~)
5. 電子工作で使われる主な部品について(67p~)
6. 全体についてのまとめ(76p)
7. 参考文献まとめ(76p~)

1.4 足歩行ロボット製作記

1-1 はじめに

最初に部品を買い始めて約1年と4か月が過ぎた先日、数々の困難を乗りぬけて、ついに4足歩行ロボットが正常動作しました。マジ大変だった。ここまで來るのに。もう少し早く終わっていれば、他のことにもさらに精を出せたのになあ、とか今になって思ったりもする。この部誌を読むにあたって、部品などの基礎知識については、ある程度知っていることが前提です。電子部品について、全くわからないのであれば、先に5の内容を読むことをお勧めします。

1-2 材料

Arduino Uno（今回は互換品である、Marduino Unoを使用）×1

RC サーボモーターS03N-2BBMG/F ×9個

測距モジュール GP2Y0A21YK ×1

1/4W 10kΩ 抵抗×2

積層セラミックコンデンサ 0.1μF50V ×1

タクトスイッチ×2

ピンヘッダ オスL型(40ピン)×1

ピンヘッダ オスシングル型(40ピン) ×1

DC ジャック MJ-179P 2.1 mm 標準 ×1
ユニバーサル基板（大きさ的には ICB-93S がちょうどいい）×1
タミヤユニバーサルプレート ×1
乾電池 9V ×1
乾電池単3型 ×4
電池ボックス単三型 4本 ×1
電池スナップ DC プラグ付き ×2 （電池ボックスやスナップは自分の好きなタイプを選ぶほうが良い）
アルミ板 適量
ビス、ナット 必要な分だけ
ワッシャー 100個入りを買うとよい。あるとかなり便利
両面テープ、ゴム板など

部品について軽く説明をします。4足歩行ロボットを作るというのに材料はこれだけ？と思う人もいるかもしれません。展示しているのをよく見てもらうと、基板の部分についている部品が少ないことに気づくと思います。これは、4足歩行するロボットの原理がそこまで難しくないことを意味しています。それを1から設計するとなると、部品や回路の知識であったりが必要なのですが、今回はまねて作ったものなので、作ってみて、自分でもよく原理を理解しながら作されました。サーボモーターを一気に9個動かしながら、センサーでは、壁との距離が近すぎると離れるようにするという回路を作つて、同時に信号をArduinoに送るという感じの回路です。

これを作っている期間が長かったので、他の団体の展示会とかにも行ってみたりしたのですが、よく展示されているロボット、たとえば2足歩行ロボットなどは、動きをすべてサーボモーターでやっていました。普通のよく見るモーター（DCモーターというのですが）は、限度がなく、基本的にずっと回り続けますよね？しかしながら、サーボモーターは永遠に回り続けるというものではなく、角度を調節しながら、動いたり、止まったりすることのできる、いわばDCモーターの上位互換です。ただし、回ることのできる角度には大体制限があって、DCモーターのようにずっと回り続けることを目的に作られた部品ではありません。そのため、基本的に制御が必要な部品です。そのため今回ではArduinoを使っています。

そして電源なのですが、9Vの乾電池と、単三乾電池4本、すなわち、6Vを電源にしています。これは、考えればわかるのですが、Arduinoを動かすための電源として9Vを使い、モーターで6V使うということです。9個もの、サーボモーターを動かすとなると、さすがに、9Vの乾電池1つでは足りなくなるので、別電源としています。ちなみに、この電池、減るのがとても速いので、電池が原因で動かなくなるというのもよくあります。対策としては、コンセントからの電源をDCジャックに変換して供給するか、エネループなどの充電式の電池を使うほうがいいと思われます。

アルミ板に関しては、綺麗に切る方法を熱心に探していた期間も長かったので、かなり資材を浪費してしまいました。面積的には、よく、ホームセンターの木材売り場の近くにある、銅板や、鉄板や、ステンレスの板などとともに売っている、金属板の売り場にある1000円くらいのアルミ板の大きさで十分です。

ワッシャーは、あるととても便利です。アルミ板を綺麗に切断してもなお、素人のできる正確さには限度があると、自分でも痛いほど身にしみてわかっています。なので、その時のそれを、特にねじ止めするときに、修正できるのがワッシャーです。必須とまでは言いませんが、あつたらとても助かると思います。

軽くとか言いながら、これだけ説明してしまいました。これだけ苦労して時間もかけて作つたので言いたいことがたくさんあるんだなあと思って流してくれれば幸いです。でも、割とあって助かる情報も書いてあると思います。

（追記）詳しい話は、5の部分でも書いてあります。

1-3 回路

これが、回路です。（次ページ）本にあったものをスキャンして、加工したものです。見ての通り、回路は割とシンプルです。プログラムするときは、各々のサーボモーターをArduino

にさした時に、どこを動かす部分のサーボモーターかを覚えておかなければいけません。あと、注意すべきところは、電源が2種類あるというところです。

1-4 製作の様子

(1) 回路について

参考にしている本が、買った当時はまだ発売されたばかりで（今は改訂版も出されている）、回路図を当時あまり参考にしていなく、基板の図を見ながら作っていて、その図が間違っていたために、基板が完成するのにとても時間がかかってしまった。

(2) 筐体の組み立てについて

ここが、最も苦労したところであり、そして最も時間がかかったところである。これだけで、軽く10か月使っている説はある（諸事情により、中3の3学期は、部活にきて作業することが少なかったのだが）。アルミ板を切るだけではなく、曲げ加工をするのもとても大変だった。

(2-0) 加工初期

作業を始めた当初、それも去年の文化祭が始まる前、本にはアルミ板で、4足歩行ロボットを作ると書いてあったので、楽しそうだなあという感情もあって、挑戦してみることにした。ここからが、中学校生活終盤から高校1年生の中間考査が始まる前まで続く地獄の始まりだった。

(2-1) 第1関門

まず、何もわからない状態だったので、とりあえず、先輩に聞いて何で加工するか考えたところ、リョービのジグソーがあった。結果は、うまく切断できず、汚いし、うるさかったので、違う方法を考えることにした。糸鋸で切ったり、いろいろ部活にあるもので試してみたが、工業高校がよくやっているような、綺麗なものには程遠いモノしかできなかった。

(2-2) 金属のこぎり全盛期

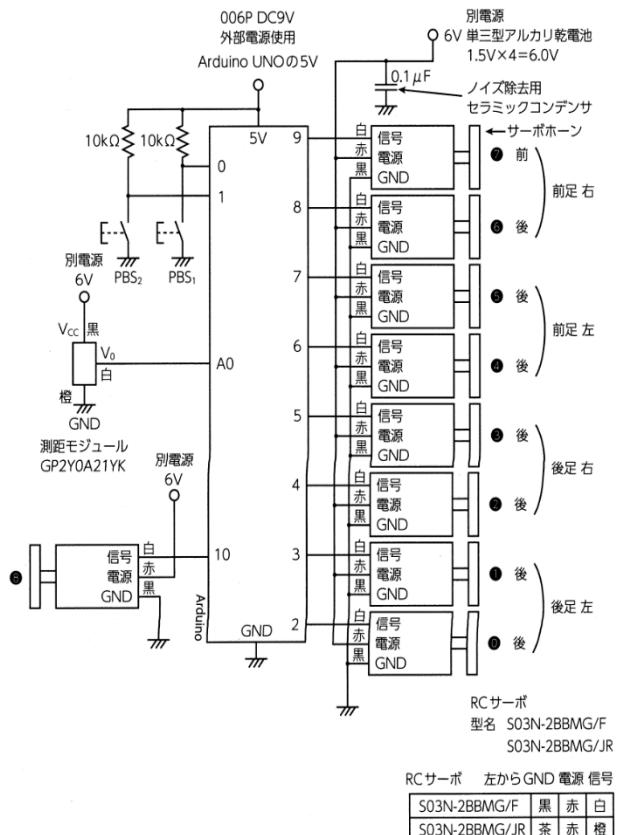
川崎のUnidyに行って、のこぎりのところを見ると、金属用のこぎりがあったので、それを購入して、加工してみた。ついに救われると思った。しかしふたを開けてみたら、切りづらく、そして糸鋸でやった時と汚さは相変わらず、それでも無理やり加工して頑張っていったん完成させてみたが、全くダメだった。

(2-3) 暗黒時代

この時代は闇でしかなかった。前述したように回路もおそらくおかしい、筐体はとてもひどい状態、文化祭は終わって、前では、遊んでいるPC班、先輩はいなくなつて一気にかかるプレッシャー、研修旅行の課題がうまくいかず自暴自棄になるなど、とても荒れて、作業のやる気も全く起きなかつた時代だった。

(2-4) 冬休み、そして3学期の暗黒時代再び

考査も終わり、ほっとして、冬休みはFFばかりやっていて、部活もなかつた。このころネットで、カッターや金切りばさみを使って加工するという動画を見て、やってみようという気になつた。これが救いだった。しかし3学期は、英語のプレゼンの、グループ課題に追われ、班員は全員サボり、自分一人でただただ作業していた日が続き生活のリズムが乱れまくつたせいで、部活もサボつて、日々課題に追われるという、つまらないことに明け暮れてい



RCサーボ	左からGND	電源	信号
S03N-2BBMG/F	黒	赤	白
S03N-2BBMG/JR	茶	赤	橙

た。万力バイスも買ってとりあえずの形で完成してはいたが、見た目も汚いし、うんともすんとも言わないし、動いてもくれなかった。

(2-5) 高校入学式後

本格的に作業も進め、アルミ板の加工も、GW 明けにはほぼすべて終わっていた。そして、回路も修正し、中間考査明けに完成して、今に至る。

1-5 アルミ板の加工方法まとめ

~~なんか、製作記書いてたらFFについて語りたくなっちゃったよ。~~
今までのものは、自分の体験談を語っただけなので、読み飛ばしてもらって構わない。というわけで、自分が苦労し、時間もかなりかけたので、アルミ板の加工方法について、お勧めの方法をここで詳しくまとめていきたいと思う。

○用意すべき道具

- ・作業台（自分はごみ捨て場から拾ってきた机の天板を使っていた。）
- ・万力バイス（4000円程度の物、大きいほうが良い）
- ・カッター（1000円くらいの大きい物）
- ・アルミ板
- ・ボールペン（油性ペン）
- ・定規（長さは加工するアルミ板の大きさによる）

○アルミ板の切断の仕方

(1) アルミ板を切断したい形にボールペンで何度も同じ場所をなぞってアルミ板に傷をつけるように書く。この時、油性ペンを使ってもよいが、見た目はボールペンのほうがきれいになるので、絶対に勧めない。（ネジ穴を通す場所まで記しておくとよい）ちなみに、鉛筆、シャーペンは痕すら残らないので、戦力外。できるだけ誤差の内容に書く。

(2) カッターで、ある程度書いた線に沿って、傷をつける。ここで、傷を丁寧につけると、傷がはみ出ないので、見栄えがとてもよくなる。とにかく、ここは丁寧に作業したほうが、後の見栄えもよくなる。ここで、むしろカッターだけを使って切断することも可能。2~30回は最低でも同じ場所を傷つける。

(3) ある程度傷がついたら、万力バイスも使って傷をつけた場所に沿うように丁寧に曲げる動作を繰り返す。

(4) 切断完了

○アルミ板を曲げるとき

(1) 切断の仕方の(1)と(2)をやる。ただしこの時、あまり傷はつけすぎないほうが良い。また、傷はアルミ板の両面につけると良い。大体5回程度傷をつけるだけで十分。

(2) 曲げるときもわかるとは思うが1回だけで十分だし、曲げなおしたりすると、逆に切断されてしまうので注意が必要。

○作業場の写真



万力バイス

1-6 作り終わった後の感想など

最初に作り終えて、急に4足歩行ロボットが動いたときは、かなり驚いたし、感想としては、生き物みたいで気持ち悪いという印象があった。9個のサーボモーターを使って、動物のような動きをしていた。当たり前といえば当たり前ですが、苦労して作った電子工作の作品とあって、正常に動いてくれた時の感動って、テストで100点取るより感動しますよ。今回は、そんな感動と、気持ち悪いという2つの感情だったので、少し複雑ではありましたけど。

2. 超電導（超伝導）について

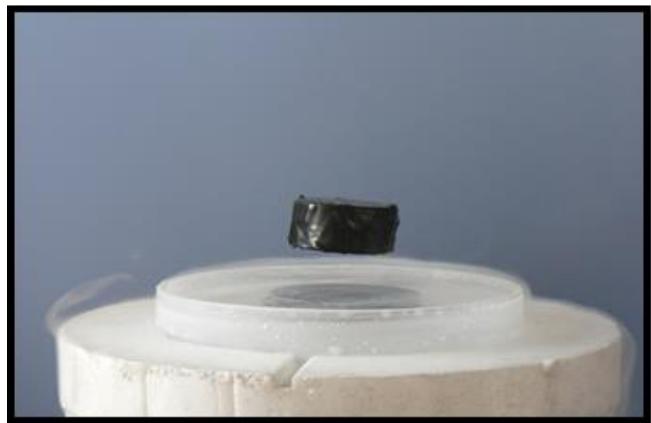
2-1 はじめに

これを書こうと思った動機は、とある大学の文化祭で、見て面白いと思ったのがきっかけで、今回に至った。今年はもう無理だろうと思っているが、来年度の文化祭では、間違えなくなんかしらの形で超伝導の発表をしたいと思っているので、来年にご期待ください。

2-2 そもそも、超電導とは何ぞや？

「超伝導」と聞いても何のことかわからない人は、これを読んでいる人もいるでしょう。説明だとわかりづらいと思うので、導入部分で印象を持ってもらいたく、画像をのっけます。というわけでどうぞ。

こんな感じのものですが、見たことある人もいると思います。次の項では、詳しく説明したいと思います。



2-3 仕組みなど

そもそもその定義とは何だろうか。参考にしたサイトにはこう書いてある

- ・電気抵抗がゼロである
- ・マイスナー効果が観測される。

と。「この2つのことが観測されることが超伝導には必要である。」とも。電気抵抗がゼロ？ そう思った人も中に入ると思いますので、説明していきます。

【電気抵抗について】

電気抵抗というのは、電流が流れるのを妨げるのを妨げるで電気の一部が熱に代わってしまいます。例えば、電化製品を使っているときにもそのことを感じられる時があるでしょう。ドライヤーや電気ポットなどは諸、その性質を使っています。またコンセントにさしていた掃除機などのコードが熱くなるのもこの性質によるものです。これが、大きなエネルギーの損失になります。その電気抵抗がゼロということは、抵抗がないので夏を発生せず、エネルギーの損失もないで、流れている電流は永遠に流れるというものです。

そしてこの状態は、ある特定の物質を混ぜ合わせて作った材料を冷やすと、電気抵抗がなくなるという状態になります。これを、超電導現象といいます。

物質によってなるものとならないものがあり、金、銀、銅などの物質は、冷やすことによって電気抵抗は小さくなるが、ゼロにはならず、超電導にはなりません。

【マイスナー効果について】

聞いたことがないワードだとは思いますが、こっちのほうがどちらかというと有名だと思います。

簡単に言います。あの画像では、なんか得体のしれないものが浮いていましたよね？ はい、そうです。あれは、「マイスナー効果」が原因の一つで起きる現象です。

普通の電導体（常電導体）、ここではわかりやすくするために、通常の温度での磁石を想像してください。では、磁力線が常電導体を通るのですが、超伝導体ではその磁場が超伝導体から排除されるという効果です。

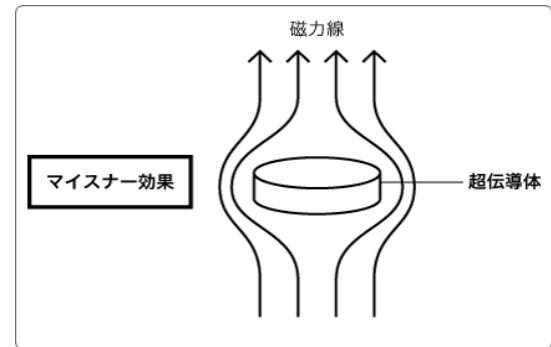
それでは、あの浮遊している磁石はなんでそうなっているのか？

それは、もう一つの重要な性質で、ピン止め効果と呼ばれるものがあります。この二つの効果により、超伝導体と磁石の位置関係が固定されて、安定して浮上しているのです。

【臨界温度について】

超伝導状態になる温度のことを、臨界温度といいます。

超伝導というものが発見されてからはずっと -243°C を超えることはありませんでしたが、現在では -138°C で超電導になる物質も発見されています。



2-4 補足説明

よく、「超伝導」と聞くと大体次いで、「液体窒素」という言葉を耳にすることがあると思います。ちょっと興味深い話も聞けたので、大学生から聞いた話の内容も混ぜながら、書きたいと思います。

【温度の概念について】

中学に入つてからの理科（高校化学基礎かも）では、こんな単位を習うかと思います。K（ケルビン）

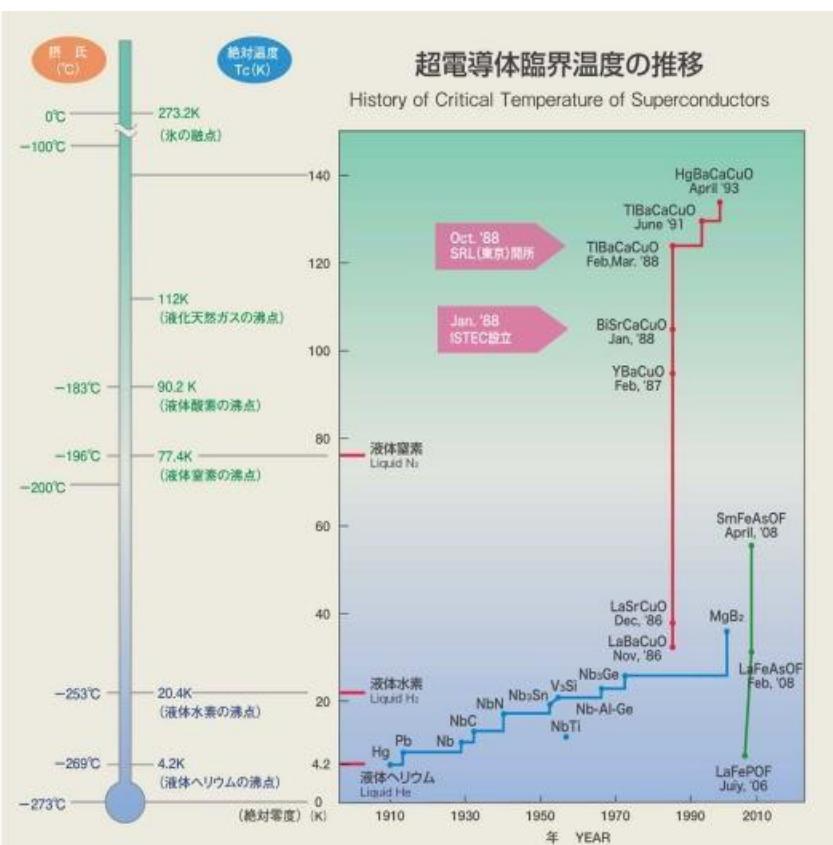
Kと°Cの関係について、式を書くとこうなります。

$$T(K) = t(^{\circ}\text{C}) + 273$$

Kとは、原子・分子の熱運動というものがほとんどなくなる温度を0Kとして定めた温度のことを言い、これを絶対温度といいます。 $^{\circ}\text{C}$ はセルシウス温度といい、1気圧の下で氷が溶ける温度を 0°C 、水が沸騰する温度を 100°C と定めた温度の事を言います。熱運動というのは、目には見えない原子・分子が行う運動のこと、1気圧というのは、およそ1023hPaのことです。

と、温度の概念について話したところで本題に入りましょう。

先ほど話した臨界温度について。これはどの物質であってもとても低い温度です。冷凍庫にしまっただけでは到底出せるものではありません。そこで磁石などの冷却に使うのが、液体窒素です。よく聞きますよね？



なんで液体窒素なのか聞いてきました。まさに上の図（前ページ）を見ながら説明していたのですが、その図に、いくつかの物質の沸点が書いてあって手あたり次第に聞いてみようということで、聞いてみました。白黒で見づらかったらすいません。

「液体酸素だと何がいけないのですか？」

「液体酸素は、窒素より沸点が高いし、値段も高いので」

「だったら、液体ヘリウムだと何がダメなんですか？」

「液体ヘリウムだと1Lで、3000円くらいするし、研究用で大量の気体を使うから、安いほうがいいです。液体窒素だと1Lで10円（確か）くらいなので」

というかんじで話をしていました。気になって調べてみました。10円とまではいかなかつたものの、割とリーズナブルな値段ではありました。これで、今年の文化祭に出せるかなあと思ったんですが、問題点だらけで、まず、それを保存しておく専用の容器（20万円）が必要。法的な資格はいらないが、保存容器の値段のほかに、取り扱いに注意が必要であるなど、かなり文化祭で出すのは、どうだろうかなあなんて思ってしまいます。しかし、レンタルもやっているということなので、下調べをもっと詳しくしてからにしようということで、今年は見送り、来年出せたらなあと。その下調べの一環でございます。この部誌。

2 - 5 超電導の応用例

この項目も、長くなってしまうので、1つだけ紹介します。よく言われているのが、リニアモーターカーですね。リニアモーターカーでは、高額な液体ヘリウムを使っているということも聞きました。リニアモーターカーには、「線路」という概念がないので、車両を浮かせて、左右にある柵みたいなものの中にある磁石のN極とS極が交互に配置されているので、その反発する力をを利用して車両を前進させています。その時の車両を浮遊させるために超伝導の技術が利用されています。

2 - 6 まとめ

超伝導に関してはこれで終わりとなります。超電導について、ちょっとでも興味を持ってくれたり、面白いと思ってくれればいいなあと思っています。

3. 超音波について

3-1 はじめに

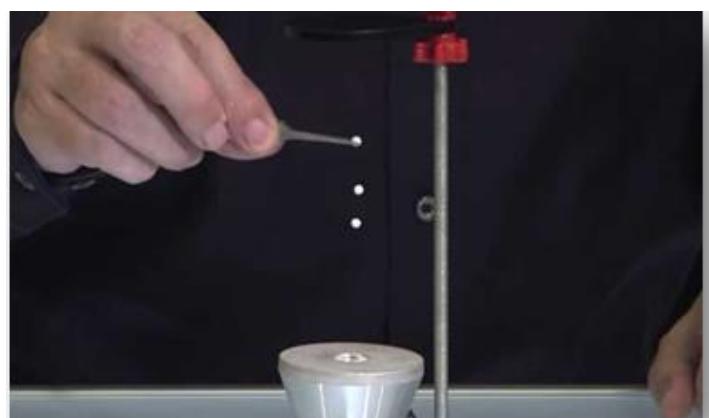
今回もとある大学の文化祭シリーズ(仮)です。これも、少し気になったので、超電導ほどは深入りしませんが、調べたことなどをまとめていきたいと思います。

僕が見たのは、こんなものです。

画像の中央に白い球が3つほど浮いていますがどうやらこれも超音波と関係があるようです。これを見たのがきっかけで調べようかなあと思いました。これも後に詳しく説明したいと思います。

余談ですが、大学の文化祭ってとても面白いですよ。なぜって、中学生、高校生のやる文化祭なんかとは、レベルが全く違うんで。小学生が、中学、高校の文化祭を見ると真新しいことだらけで、面白いと思うけれど、それと同じように、高校生

が大学生の文化祭を見ると、これまた真新しいものを見られて、面白いと思いました。



ちなみに、この超音波は、今のところ来年の文化祭で出すつもりはありませんので、こういう実験とか、研究を間近で見てみたいならば、それこそ大学の文化祭、オープンキャンパスに行ってみることをお勧めします。おそらく費用面でこれを文化祭に出すのは無理でしょうね。

3-2 超音波ってなーに？

超音波とは音の一種です。そもそも人間には 300Hz から 4kHz の周波数を声として発し、聞くことのできる音は 20Hz から 20kHz の周波数といわれています。そして、超音波とは、「人間の耳では聞くことのできない音」とされています。他にも定義があるらしいのですが、ここではこの定義で説明したいと思います。

超音波も音の一種なので、特徴は理科で習うような音と同じです。真空中では伝わりません。光とは違い、非透明な物質中でも伝わることができます。音と同じように、早さも、気体 < 液体 < 個体の順になります。

3-3 Hz とは？

ここでは、自分が確かめたいということもあり少し脱線したいと思います。Hz、すなわち周波数とはそもそも何なのか？これ、実は、電子工作班の立場としては、トランジスタという部品で、高周波トランジスタ、低周波トランジスタとかよく言ったりするので、ここでまとめておきたいと思います。

(以下 Wikipedia からの引用)

周波数（しゅうはすう 英：frequency）とは、工学、特に電気工学・電波工学や音響工学などにおいて、電気振動（電磁波や振動電流）などの現象が、単位時間（ヘルツの場合は 1 秒）当たりに繰り返される回数のことである。

と書いてある。そして、この文章を書いている現在、物理基礎の教科書を見てみると高校の物理基礎でも触れている内容です。どうやら波に関するらしいです。これで解決しましたね。トランジスタでも、高周波と低周波のものが存在するのも、納得がいきます。

トランジスタとは、簡単に言うと、電流を増幅する、増幅作用と、スイッチング作用を持つた端子が 3 つある部品です。しっかりと別のところで説明したいと思います。

3-4 あの写真の解説

さて、それでは、ここまで話したところで、最後に例の写真の原理について説明したいと思います。最後の感想はないです。この現象は「超音波浮揚」と呼ばれるものになります。

そもそも、浮揚とは？

物体が宙に浮くことであり、前の、超電導のところで説明した、磁石のあの画像なんかがいい例だと思います。

下のほうにある円盤のようなもの（ホーン）から、超音波を発生させ、反射板を平行に配置し、球が浮いている場所に定在波音波というものを形成します。そしてこの、気体の疎密差による力を揚力に変えて物体を浮揚させています。詳しくは物理の音の分野を参考にするとよいでしょう。しかし、この程度の揚力では、数mg程度の重さしか浮揚させられないで、あの画像のように、小さい球が浮いているわけです。図のように、大体ピンセットを使って浮揚させることも多いです。

僕が大学のオープンキャンパスに行ったときは、ホーンの部分にもさわることができました。音などは何も聞こえなかったのですが、ホーンの表面は、とてもぬめぬめとした感触となっていました。あと、反射は、床とやったほうが、浮揚は安定していました。また、浮揚中に、ピンセットなどで、球と球の間をつづいたりすると、少しなら何の変化もないのですが、あまりやりすぎると、その後は、浮揚することはなくなってしまいました。

以上、大学でのオープンキャンパスでの体験も含めた、超音波の話でした。

4. 3D プリンタについて

4-1 はじめに

前年度に購入予定だった3Dプリンタは予算の都合上今年度、購入し、今年から運用を始めた。3Dプリンタは、テレビで見て知っている、なんて方もたくさんいると思いますが、ここでは印刷の仕組みや、値段、どんなものがあるのかなどについて踏み込んで話をていきたいと思います。

4 - 2 造形方式の違いと、特徴など

3Dプリンタを語るうえで、これ抜きには話が進まないと思います。なので、まずは、造形方式の違いについて、分類しながら話を進めたいと思います。自分が調べた結果、造形方式の種類は5種類あることがわかりました。自分では、2種類くらいしか知らなかったので、5種類もあってとても驚きました。

【熱溶解積層法】

この方式が、僕を含め、一般人の中では最も知られた方法なのではないかと思います。展示されているであろう3Dプリンタでも取り入れられている方式です。

リール状に巻かれている、樹脂のフィラメント（熱可塑性樹脂）を、ノズルの先端にあるヒーターで加熱し、溶かして、押し出すことで溶かしたフィラメントを使って造形する方式です。

その溶かしたフィラメントを文字通り積み重ねていくことで、造形物が出来上がります。展示しているものを見てもらえばわかると思うのですが、ノズルの高さは基本的に変わらないので、台の部分の高さを変えて積層しているのがわかると思います。また、完成した造形物を見てもらえばわかると思うのですが、造形物が層状になっているのが、感触だけでもわかると思います。ちなみに、造形物が1層分出来上がった後に、台が下がるのですが、その高さのことを、「積層ピッチ」というそうです。そして気づいている人もいると思いますが、この積層ピッチより細かい形状を作ることはできないので、制度を求める場合は、この積層ピッチが細かい3Dプリンタを使うか、他の造形方法などの3Dプリンタを使うのが良いです。

種類	PLA	ABS
ノズルの温度（純正フィラメント）	210°C (180 ~220°C)	230~240°C (210~250°C)
ヒートベッドの温度	40°C	110°C
冷却ファンの使用	あり	なし

自分たちが展示しているであろうX-one2は、その機能が付いただけの造形方式です。

樹脂には、2種類あって、一つは、PLA。2つ目はABSというものです。その違いは表にまとめたいと思います。

見ての通り、樹脂によって、溶ける温度なども違うため、設定なども3Dプリンタによって変えなければなりません。この温度がしっかりとしていないと、樹脂は溶けないし、場合によっては、詰まってしまうこともあります。

自分が知らなかつたことなのですが、複数のノズルを使うタイプもあるようです。比較的安価なものが多くの家庭用3Dプリンタでは、この方式を採用しているものが多いように思います。

ただ、手入れをしなければいけなくて、ノズルにフィラメントが詰まるというのは、よくおこることです。なので、ノズルの部分だけ取り外すか、ノズルを買い替えるというのが良いでしょう。なので、ノズルは、3個入りで、安い値段で、Amazonで売っています。

【光造形法】

自分がかけらも聞いたことのない造形方式の一つです。調べてみたのですが、なかなか、高度な造形方式だなあと思いました。というわけで、まとめていきます。

まず、このプリント方式は、実は、3Dプリンタの期限となるもっとも古くから存在する造形方式で、製造業でもよく利用されています。また、制度もよいので、プレゼンやデザインの確認のモデルとしても、また、真空注型や樹脂型のマスターモデルとしても利用されます。紫外線を当てると硬化する、液体状の、「エポシキ樹脂」という者に紫外線レーザーを当てながら硬化させていく方式です。

ただし、この硬化して作るのが、熱溶解積層法と考え方が似ていて、ここでも積層、ピッチと呼ばれるものが存在します。

材料となる液状のエポシキ樹脂をタンクに入れて、レーザーを当てて1層目を硬化させます。そして、硬化した積層面を、熱溶解積層法と同じように、1段下げて、次の層を硬化させていきます。そして、これもまた熱溶解のものと同じで、何層も重ねて、造形していくというのが、この造形方式です。

使われる材料としては、先ほどの、エポシキ樹脂や、アクリル系樹脂。ただし、エポシキ樹脂は、太陽光に弱く、アクリル樹脂はもろくて弱いという欠点があります。

【インクジェット方式】

こちらもまた、聞いたことのない造形方式ですね。

この方式は、紙に印刷するインクジェットプリンターと同様に液状の樹脂を噴射して印刷する方法です。

特徴としては、高速に複雑で、細かい造形物を作ることができて、造形物の表面が滑らかできれいです。

まず、インクジェットが、移動しながら樹脂やサポート材を噴射していきます。そして、ローラーカッターで指定した1層分の厚みになるように鏡面を削っていきます。そして、いつものように、台を下げる2層目という流れです。

材料としては、ABS樹脂に似ている、ABSライクや、PPライク、ラバーライクなどです。複数のノズルがある場合はこれらの材料を複合した造形物を作ることができるそうです。

【粉末石膏（せっこう）造形】

粉末を使って作るというのは、実は、テレビで見て僕が3Dプリンタを知った最初の出来事だったので、実は、割と最近までは、3Dプリンタの印刷方法の主流はこの方法（または次紹介する方法）なのかなあと思っていた。

この方法では、高速に複雑で、細かい造形物を作ることができ、インクで着色でき、カラーでも印刷できるという、かなりよいメリットがありますが、反面、石膏であるため、きょうどが弱いというデメリットもあるようです。表現が正しいかわかりませんが、諸刃の剣みたいなものですかね。

この粉末石膏造形は材料である粉末を造形テーブルに敷き詰めて、レーザーや接着剤で固める造形方式です。

まずは、ローラーを使って造形ステージに1層分の粉末を敷き詰め、インクジェットヘッドから接着剤を噴射して固めます。そして1層目が完成したらいつものようにステージを下げるという動作の繰り返しです。

石膏はどうやら材料費が安いようです。自分は美術部で、石膏を使った作品を作ったことがあるのですが、かなり大変でした。石膏は、さわったことがある人はわかると思うのですが、落としただけで簡単に割れてしまいます。あと、面白い性質もあって、水に普通溶かして使うのですが、溶かすと、熱を軽く帯びてきます。これら辺の話は科学の分野ですかね。

【粉末焼結造形】

いよいよ最後の造形方法です。この方法だと、金属や樹脂など、さまざまな材料が使えるようで、複雑な形状を作成できるが、表面はざらつくようです。

まずは、金属や樹脂の粉末をローラーにより造形テーブルに敷き詰めて、紫外線レーザーを当てて1層目を固め、テーブルを下げてと、いつもの繰り返しです。

4-3 ABS樹脂とPLA樹脂の違い

この項目は、展示されている3Dプリンタの解説がメインになろうかと思います。家庭用3Dプリンタでよく使われている素材ですが、そもそも、この素材を使うのが、最初で説明した、熱溶解積層法で使われています。

ABS樹脂とはとある英語の頭文字を3つ取った名称です。表面光沢に優れていて、着色することもできるので、デザインを含む外観部品に適しています。また、比較的強度も高く粘りもあることから、力が加わるものにも適用できます。

PLA樹脂は植物由来の樹脂でできているため、プリント中は、樹脂独特の嫌な臭いを発しません。先ほどの表を見てもわかる通り、この樹脂は、プリントするときの樹脂温度がABS

樹脂より低く、高温に弱いです。差のため、弾力性がなく、固いという特徴があります。そのため、サンドペーパーや、ヤスリ等で表面がけをするのが難しく、塗料もなじみにくいです。材料が粘り強いため、大型の造形物を作るのに適しています。

4-4 用途

よく言われている用途は、試作でしょう。本物の材料を使って試作するより、3D プリンタを使ったほうがいいですからね。あとは、意匠確認や、治工具、最終製品を作ることなどに利用できます。

4-5 まとめ

この項での 3D プリンタは、主に 3D プリンタの造形方法などの違いを説明したのがメインであるので、お勧めの 3D プリンタや、値段などについては一切触れていません。あくまで、技術的なお話をと思ってくれると嬉しいです。

5. 電子工作で使われる主な部品について

5-1 前書き

実はこの文章、書き始めが、部誌の提出期限かなりギリギリです（提出期限が過ぎて 7 日ほど）。なので、主に中 2 の部員が書いた部誌の内容は、ほぼすべて内容を見させてもらいました。今年の部誌、ほとんどが自分の趣味（真面目）を書いているか、技術系の話ばかりじゃないですか（怒）？電子工作とか、アニメ系の話とか触れている人が全くと言っていいほどいないんですよ。今年はもう、真面目な方向で統一しようということで、電子工作がらみの話も少ないので、聞こえは悪いかもしませんが急遽入れた内容になります。少し内容も調整したほうがいいかなあとは思ったので。何年か前にも同じようなことを書いていた先輩もいたんですが、それを超えられるようにしたいと思います。あの、長くなりそうな予感がするので、一気に読まないほうがいいかと思います。

5-2 概要（前書きと概要って何が違うんだ？）

さて、では 1 つ 1 つ見ていきませんか。先輩の過去の部誌をあさってみたところ、8 個の部品について説明していました。僕は基板など、基本的な部分から、とにかく使われる部品について何もわからない人にも、1 から説明したいと思います。

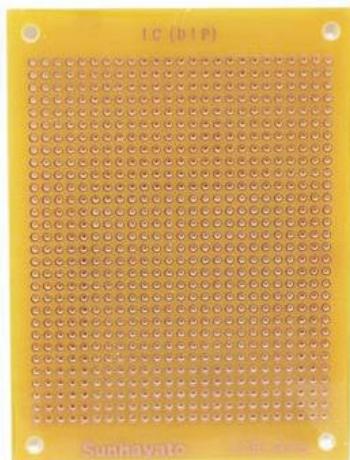
5-3 基板（+ ブレッドボード）

ブレッドボード？ とりあえず無視してください。

基板という言葉は、誰でも聞いたことがあるでしょうか？認知度がどれくらいかはよくわからないんですが、多分聞いたことがないという人は珍しいのかなあ、どうでしょう？

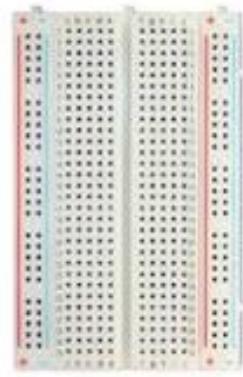
右に挙げたのが、基板の一つですね。実はこれが ICB-93S です。基板の中でも、ユニバーサル基板といいます。この基板には表裏があって、銅箔があるのが、裏です。

基板といえば、緑色を想像する、なんて方もいると思います。それも基板です。間違ひありません。



大体、緑色のものは、プリント基板であることが多いです。プリント基板の場合は、基板の部分で、かなり後に説明するスズメッキ線というものを使わずに基板にハンダづけができます。しかし、この、ユニバーサル基板はスズメッキ線を使う必要がありますが、自由に配線をすることができるので、危険な回路を作るときや、ハンダづけの練習をしたい人には、こっちの基盤をお勧めします。

さて、回路を組む時にはこの基板を使うしかないのでしょうか？はんだ付けができない人にも朗報です。はんだ付けしなくても回路を組む方法はあります。それが右にある、ブレッドボードという部品です。基板買うよりもこっちを買うほうがどう考えてもコスパがいいです。横の2列は縦に、残りの10列は、5列ずつ、横につながっています。



ただし、このブレッドボードを使うには別に、ジャンパー線というものを買わなければいけなく、また、このブレッドボードの穴に入らない部品などもあるので、注意が必要です。



5-4 ハンダ

さらっと行きます。右の図です。ハンダごてを当てて、溶かして、それを基板や銅線や、部品の足につけます。

ハンダは、鉛とスズの合金であることが多く、コンセントにさしただけの半田ごてでも解けるように融点が低くなっています。



5-5 ハンダごて

前述のハンダを溶かすためにある道具です。普段、はんだ付け使うのは、30Wが多いです。場合によっては40Wであることもあります、自分は30W信者です。40Wだと、ちょっと使いづらいという個人的な理由ですが。ただし、コイルガンの製作などで、太い銅線が必要な時には、最大100Wまであるのですが、60Wなどを使うときもあります。

5-6 銅線 (+スズメッキ線+ジャンパー線)

文字通り「銅線」です。自分たちは、白黒で分かりにくいかもしれません、右の図とおそらく同じものをよく使っています。

銅線にもいろいろな太さがあって、ショートしたら危険な回路などを組む時には、図より太い銅線を使うことが多いです。主にコイルガンで使う、昇圧回路ですかね。

黒い部分をはがすと、銅線の本体が出てきます。カッターを使ってはがすことが多いです。



一方、左の図はスズメッキ線と呼ばれるもので、先ほど紹介したユニバーサル基板の裏で、線同士をつなげるときは銅線ではなく、このスズメッキ線を使う場合が多いです。

銅線と違い、ニッパーで切断したらそのまま使うことができます。ただし、ハンダとはなじみづらいので、ラジオペンチなどを使って線同士をつなげるのが良いでしょう。



さらに右の図は、ジャンパー線と呼ばれるものです。主にブレッドボードや、ピンヘッダ（後述）の穴にさして使います。Arduino や RaspberryPi を使うときにも使えます。あると便利な一品です。オスとメスの使い分けによって色々な種類があります。

5-7 抵抗類

ようやく部品に入ります。この項を抵抗類としたのは、抵抗の種類がたくさんあるからです。抵抗は大きく、固定抵抗、可変抵抗、シャント抵抗（ここでは触れない）に分かれます。

まずは、固定抵抗について、抵抗の値が固定されているごくごく一般的の抵抗です。固定抵抗にも、炭素皮膜抵抗や、金属皮膜抵抗などと別れています。また、それらの抵抗は、4~6本の色が書いてあり、カラーコードといいます。

右の表が、カラーコードと呼ばれるものです。この内容は暗記しておいても個人的には損はないと思います。というか、暗記しておいた方が助かる場面が多いです。

1、2番目はそのまま読みます。3番目は、1、2番目に読んだ数字にかけます。それで得た数値に許容差をつけると、得られた数値 \pm 何%という結果が得られます。

試しに、1個だけ例を。1番目の色が、茶色。2番目の色が黒色。3番目の色が赤色。4番目の色が金色だとどうなるでしょうか？

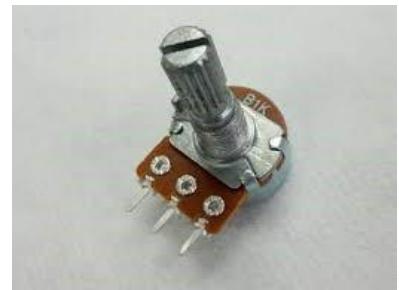
計算結果は、 $10 \times 10^2 (\Omega) = 1000\Omega = 1k\Omega (\pm 5\%)$ という結果になります。なので、この表を覚えておくと、抵抗の抵抗値自分で求めることができます。

そして、右の図が、炭素皮膜抵抗の図です。ちゃんと4本の色の線がみえるとおもいます（白黒）。この図は、茶黒橙金、です。 $10k\Omega (\pm 5\%)$ です。金属皮膜抵抗も同じ感じです。

そして左の図が、可変抵抗です。音量調節のボリュームなんかによく使われています。端子は2本のものと3本のものとあります。こっちにはカラーコードはないので、後ろに書いてある数値で判別しなければいけません。

そして、右のものが、半固定抵抗です。真ん中の部分をドライバーで回して、抵抗値を調節します。端子は3本であることが多いです。可変抵抗と性能に大きな違いはありません。半固定抵抗も、可変抵抗も、大体端子は2本しか使わないことが多いです。

	1, 2番目	3番目	4番目
色	数値	乗数	許容差
黒色	0	$10^0 = 1$	—
茶色	1	$10^1 = 10$	$\pm 1\%$
赤色	2	$10^2 = 100$ (以下略)	$\pm 2\%$
橙色	3	10^3	$\pm 0.05\%$
黄色	4	10^4	—
緑色	5	10^5	$\pm 0.5\%$
青色	6	10^6	$\pm 0.25\%$
紫色	7	10^7	$\pm 0.1\%$
灰色	8	10^8	—
白色	9	10^9	—
銀色	—	$10^{-2} = 0.01$	$\pm 10\%$
金色	—	$10^{-1} = 0.1$	$\pm 5\%$



5-8 トランジスタ

電子工作をするのには切っても切れ離せないくらい重要な部品です。前の超音波でも一部出てきました。根本の部分から説明をしたいと思います。

まず基本的な概念となる N 型半導体と、P 型半導体について。

簡単に説明すると、N型半導体は、電子が余分にあるもの。P型半導体は電子が不足しているもの。という認識だけで大丈夫です。この後も少しだけ出てくると思います。

それらの半導体を3つ組み合わせたものが、トランジスタで、NPN型と、PNP型のトランジスタがあって、型番にも影響します。そこで、周波数の話にもなります。

xxxxには、4つの数字が入ります。自分が良く見る順番としては、2SC>2SA>2SD>2SBの順ですかね。2SBなんて見たことがないと思います。そして、低周波用あまり見かけないし、使われないです。2SC型は最もよく見ます。2SAと2SDはたまに見ることがあります。

右が、トランジスタの図です。E(エミッタ) C(コレクタ) B(ベース)の3つの端子からなっています。どのトランジスタも、この順だと思います。次はその原理についてです。

【NPN接合の場合のみ】

1. 最初に、E-C間に、Eを-として、電圧をかけます。この時に電流は流れません。

2. ここで、追加で、E-B間にEを-として、電圧をかけます。すると、トランジスタ全体に電流が流れます。軽く説明すると、P型と、N型を合計で3枚くっつけているので、電子の移動を利用して、E-B間に電流を流すと、それに従って、E-C間には、何10~何100倍の電流が流れます。これを、増幅作用といいます。

かなり簡略化した説明がこれになります。まとめると、

【増幅作用】

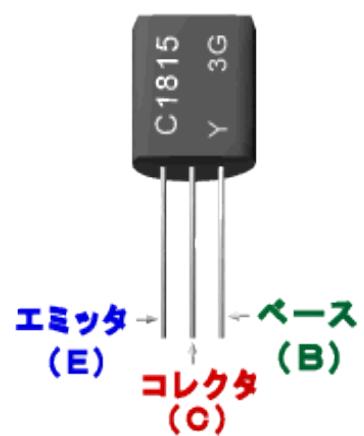
- E-B間にわずかな電流を流すことで、E-C間にその何倍もの電流を流すことができる。その何倍になるかを示す値のことを直流電流増幅率(hFE)という。

- E-B間のわずかな電流変化が、E-C間の電流に大きな変化をもたらします。

【スイッチング作用】

- 増幅作用の仕組みを利用しています。

- 電流の大小ではなく、ONとOFFの違いだけが制御の対象であり、Bに与える小さな信号によってより大きな電流を操作できるので、メカニカルなリースイッチの代わりに利用されることもあります。



5-9 ダイオード類(LED)

ダイオードというのは、右の図にある部品です。本来はトランジスタよりも先に書くべきかもしれません。ダイオードという部品には、極性があり、A(アノード)とK(カソード)の2種類があります。ダイオードは、P型半導体と、N型半導体をくっつけたもので、Aの端子は、P型と、Kの端子は、N型とつながっています。灰色の部分があるほうがKです。勘のいいひとは、ダイオードの特性について、ここで気づいたかもしれません。ダイオードの基本的な特性はずばり、整流作用です。AからKにしか電流を流さないという性質を持っています。Aと、電池の+を、Kと、電池の-をつなげればいいです。要するに、Aが+、Kが-です。



反対側には、ほとんど電流を通さないだけで、電流が通らないことはないのですが、かなりの高電圧を流さなければいけなく、また、その時には壊れてしまうことが多いです。よって、スイッチとしての機能も待っているため、スイッチング作用も持っています。

これを派生したものもありますよね？そう、LEDです。別名は「発光ダイオード」ですかね。ダイオードとつなげ方は同じです。線が長いほうが、Aです。なので、長いほうが+と習った人もいるかもしれません。

5-10 コンデンサ類

コンデンサというのは、電気を蓄えたり、放出したりする部品です。物理部では、ここ数年、個コイルガンが流行っているため、500個ほどをつなげたコンデンサの塊や、超巨大なコンデンサもあったりします。自分は、小学校の理科の実験のキットの中にこれが入っていた記憶があります。種類はいくつもあるのですが、電子回路の中では、なんかしらの部分で必ず使われていることが多いです。



コンデンサには大きく3つの役割があります。

- 1つ目は、充電したり、放電させることで、電圧の変化を吸収し、電圧を安定させます。
- 2つ目は、電気の通り道で、余計なノイズを横道にそらし、ノイズを取り除きます。
- 3つ目は、直流はさえぎり、周波数で信号をより分け、信号を取り出します。

【ノイズとは】

すべての電子回路に存在し、電気信号の無作為な変動のことです。一般的には望ましくはないです。

【信号について】

電気通信や信号処理、さらには、電子工学伝搬において、時間や空間に伴って変化する任意の量のことです。

コンデンサには色々な種類があります。モーターによく取り付けるのは、積層セラミックコンデンサで、コイルガンなどによく使うのは、電解コンデンサです。

5-11 モーター類



モーターにもいくつか種類があります。4足歩行のところでもかなり書きましたが、普通のDCモーターと、サーボモーターというものがあります。DCモーターは、電気が流れている限り、永遠に回り続け、サーボモーターは、マイコンを使って角度をしっかりと、制御して、動かすというものです。左の図がDCモーターです。

以前学校

から最も近い鶴見のタミヤの店に歩いて行ったときに写真をとったのですが、見ての通り、15種類もモーターがあります。あの日は、1年で最も雪が降った日でしたね。かなり辛かったのを覚えています。最も高いものは、1個で800円するものもありますが、これはあくまで、タミヤの物なので、もっとすごいものも世の中にはあるはずです。



これと形が似ているもので、マイクロモーターといふものもあります。これが右の図です。DCモーターと同じですが、シャフトの部分が細いくらいです。

次は、サーボモーターです。これが、下の図です。

これは、よく売っている最も安いサーボモーターです。1個400円くらいです。自分が使ったら物は、1個1000円です。1480円の物や、2000円ほどするものもあります。

見えづらい

かもしれません、モーターのギアの部分に付属の物をつけて、それごと、アルミ板や、アクリル板につけてギミックを完成させます。サーボモーターは、Arduinoなどのマイコンとともに使うことが非常に多く、プログラムもほぼ必須の部品です。関連書籍などもたくさん出ているので、まずはArduinoから始めるといいと思います。メタルギアサーボモーターというものもあります。

モーターは、電子回路の中でもよく電力を消費する部品で、電力(W)がそもそも少ないと動かない部品なので、後述のリレーや、モータードライバーという部品と一緒に使うこともほとんどです。モーター単品を動かす回路ならあまり必要はないのですが。

5-12 リレー（継電器）

右図の部品です。リレーというのは、外部から電気信号を受け取り、電気回路のON/OFFや切り替えを行う部品です。運動会などでよくやるリレーと同じように、このリレーも電気信号を受け取り、スイッチをON/OFFすることにより、次の電子部品に信号を伝える働きをしています。この中にはコイルとスイッチが入っています。コイルに電流が流れると、磁力が発生してスイッチを物理的に動かします。なので、この部品が正常に動作しているとき、時折、カチカチっと音が鳴っているのが聞こえることがあります。これは、エレクトロ・マグネティック・リレーといいます。

この、リレーはもともと、小電力の入力によって大電力のオンオフを制御することが当初の目的でした。今では、安全性、操作性、操作の確実性が増すことなどにより、電力的な増幅の目的だけではありません。

この部品も高く、部品屋に行くと、あまりかさばっていないのに値段が10000越えをするときがあって、その時の原因是、このリレーの大量購入であります。割と値段は1個あたりが高いです。

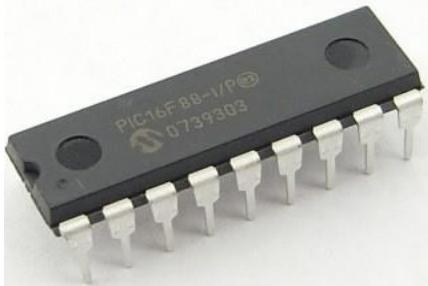


モータードライブICは、かなりの電力を使うモーターの助けになる、モーター専用のICです。ハイパワーなモーターは、電池を大量につなぐのではなく、これをを使います。面倒くさいので、自分はよく、モータードライバーと呼んでいます。左の図は、よく売られている、完成しているモータードライバーで、右の物は、自分で自作しなければいけないモータードライブICです。



自分の意見ですが、どう考へても左のほうを買うほうが楽だと思います。安いし手間もかかりないので。型番もいろいろあるので、使い方は、ネットで調べるのが一番だと思います。配線するのは大変だと思います左の図のTA7291Pというものは、現時点でおそらく最もよく使われていると思うので、いくつもネットに記事が載っています。それ以外でなければ、部品屋の

サイトなどに飛び、データシートで、1本1本の端子の役割（入力、出力、+、など）を調べるといいと思います。データシートとは、部品の情報が、主に英語や、日本語で書いてあるというものです。トランジスタやICなどの部品を使うときなどにはかなり助けになるでしょう。



Arduino のです。Arduino に関しては去年まとめたので、内容かぶりになるので、画像だけ載っけとります。図は、Arduino Uno です。最もよく使われています。

5-14 マイコン類

マイコンとは、マイクロコンピュータの略称です。後述の、IC と非常によく見た目が似ているのが、この PIC マイコンです。詳しくは触れません。プログラムを書き込み、電子回路の制御をするのが、このマイコンのお仕事です。その、最もポピュラーなものが、わたくし



5-15 センサー類

センサーもいくつか種類があります。LED を常に光させておいて、近くまたは、反対側にセンサーを置いて、出した LED の光（赤外線の可能性もあり）の反射の様子により、電流を流さないなど LED の一種であるものから、人を感知するセンサー、温度、湿度を感知するセンサーなどもあります。多すぎるので、画像は省略させていただきます。

5-16 スイッチ類



文字通りスイッチです。人の手で、ON/OFF を操作することのできるものです。押しボタンスイッチや、スライドスイッチ、タクトスイッチなど様々です。仕組みは、スライドすると、端子同士のつながっている部分が変わることで、接続状態が切り替わる。ボタンの先に

は、電気を通すアルミ板などが付いていて、押すと全部の端子とつながるなど、リレーと IC と違って、簡単であることが多いです。

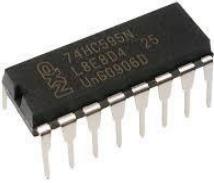
5-17 電源系統

皆さんのが日ごろよく使う、単3、単4電池や、時計などに使われている、ボタン電池、ボックス電池（9V乾電池）や充電式の電池など様々です。物理部では、たまに、コンセントから直接電気をとる、ACアダプタも使います。（右図）



5-18 IC（集積回路）

IC というのは、日本語名では、集積回路といいます。IC は、トランジスタや、抵抗がたくさん詰まって1つの部品になったものです。トランジスタなどを大量に使う回路を作るときには、こっちを使ったほうがいいです。SSI、MSI、LSI とあ



ります。MSI と LSI は集積度が高く、それらが普通に生産されるようになると、そのうち、マイクロプロセッサなど、比較的複雑なものを LSI、汎用ロジック IC など、自分たちが日ごろよく使うものを、IC と大雑把に呼び分ける程度の分類となりました。

電子工作をするときは、この部品が熱に弱いので、ソケットに入れて、ソケットの端子をハンダづけすることが多い、というか、IC をそのままハンダづけするのは、正直バカだけだと思います。

IC というのは、日本語名では、集積回路といいます。IC は、トランジスタや、抵抗がたくさん詰まって 1 つの部品になったものです。トランジスタなどを大量に使う回路を作るときには、こっちを使ったほうがいいです。SSI、MSI、LSI とあります。MSI と LSI は集積度が高く、それらが普通に生産されるようになると、そのうち、マイクロプロセッサなど、比較的複雑なものを LSI、汎用ロジック IC など、自分たちが日ごろよく使うものを、IC と大雑把に呼び分ける程度の分類となりました。

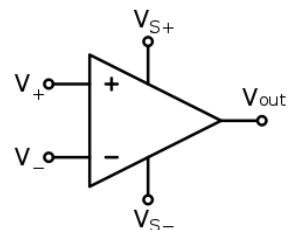
電子工作をするときは、この部品が熱に弱いので、ソケットに入れて、ソケットの端子をハンダづけすることが多い、というか、IC をそのままハンダづけするのは、正直バカだけだと思います。



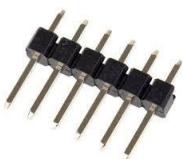
5-19 オペアンプ

集積回路の一種で、自分は、実物を見たことも使ったこともないのですが、よく使う部品らしいです。見た目が IC と似ているものもあります。增幅器、電圧比較器、加算器、発振器、など、多種多様な用途があります。右図が、オペアンプの回路記号です。

本などを読んでいてもかなり深い部品だと思われます。この部品一つで、回路設計の本の約 6 分の 1 を占めていました。

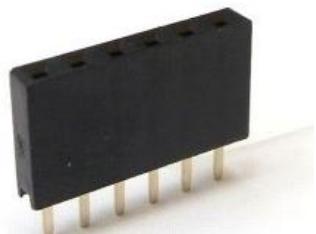


5-20 小物類



まずは、だいぶ前に名前だけ出した、ピンヘッダですね。左が、銅線などをハンダづけるオスのピンヘッダ、右が、ジャンパー線や、オスのピンヘッダなどを指すメスのピンヘッダです。割と汎用性は高いです。

この左の物は、基板に実装はできませんが、アクリル板などにくっつける、ブザーです。ビーという非常に単純な音しか出ません。期待はしないほうがいいです。



右の物は、マイコンのところで紹介しようかどうか迷ったのですが、Raspberry Pi です。Arduino とは少々異なっています。SD カードを別途買って、OS もインストールしなければいけません。最新版は Raspberry Pi3 です。



5-21 タミヤの製品たち

多すぎて、すべて紹介しきるのは無理です。よく使うのは、ツインギアモーターボックスゴ

ムタイヤの4個入り、ユニバーサルプレートですね。ちなみに、ギヤーボックスだけでも左の画像ほどあるそうです。こんなにあって使い分けなんかするんですかね？ ラジコンのほうはよくわからないので、その辺の理由についてはよくわかりません。ただ、ギア比などはあるため、何種類かの使い分けはします。

ほかにも、ブリーザー、水中で使えるギヤーボックス、ユニバーサルアームセット、ラダーチェーンセット、前述した、高性能なモー

項目番号	種類	略称	回転数(rpm)	ギヤ比	回転トルク(gf/cm)
24	遊星ギヤボックスセット	遊星(1/8)	2450.0	4.0	50.0
25	遊星ギヤボックスセット	遊星(2/8)	1960.0	5.0	58.0
38	ミニモーター多段ギヤボックス(12速)	ミニ多段(1/12)	1360.0	4.6	23.0
39	ミニモーター多段ギヤボックス(12速)	ミニ多段(2/12)	1224.0	5.1	25.0
16	シングルギヤボックス(4速タイプ)	シングル4速(1/4)	1039.0	12.7	94.0
20	ハイスピードギヤーボックスHE	ハイスピード(1/2)	880.0	11.6	185.0
8	6速ギヤボックスHE	6速(1/6)	870.0	11.6	161.0
54	ミニモーター標準ギヤボックス(8速)	ミニ標準(1/8)	842.0	7.5	37.0
1	3速クラクギヤーボックス	3速クラク(1/3)	795.0	16.6	122.0
55	ミニモーター標準ギヤボックス(8速)	ミニ標準(2/8)	663.0	9.5	47.0
40	ミニモーター多段ギヤボックス(12速)	ミニ多段(3/12)	647.0	9.7	48.0
26	遊星ギヤボックスセット	遊星(3/8)	612.0	16.0	136.0
41	ミニモーター多段ギヤボックス(12速)	ミニ多段(4/12)	583.0	10.8	53.0
21	ハイスピードギヤーボックスHE	ハイスピード(1/2)	570.0	18.0	290.0
27	遊星ギヤボックスセット	遊星(4/8)	490.0	20.0	156.0
56	ミニモーター標準ギヤボックス(8速)	ミニ標準(3/8)	401.0	15.7	78.0
35	ミニモーターギヤボックス(3速タイプ)	ミニ3速(1/3)	394.0	12.7	69.9
28	遊星ギヤボックスセット	遊星(5/8)	392.0	25.0	185.0
17	シングルギヤボックス(4速タイプ)	シングル4速(2/4)	345.0	38.2	278.0
9	6速ギヤボックスHE	6速(2/6)	338.0	29.8	415.0

ターたち、などがあります。自分は、何年もお世話になっています。

5-22 ハンダごて 以外に使う道具たち

右にあるものを主に使っています...。嘘です。そんなことを言ってみたいですか。いつか。いつか...

この中で使っているものは、真ん中下の電動ドライバーくらいです。あとは、そもそも部活はありません（当たり前）。実は、コンプレッサーは部活にあります。

メインで使っているのは、右下の手動工具類です。ただし、六角レンチは使わないのでなく、スパナも小さいモノしかないです。

その代わり、充実しているのは、接着剤と、ペンチ、ニッパー、ドライバーです。

あとは、ハンマー、のこぎり、カッターバイ丝などです。ニッパー、ペンチ、接着剤はとてもよく使います。

特殊ドライバーもたまにあると便利です。自分は自前のもの持っています。アマゾンで2000円くらいで購入できました。



5-23 主な買い出しをする店

基本的に僕らが買い出で、部品を買うのは、秋葉原です。その日は、交通費だけでおよそ1000円は持ってかれるのが、辛いです。主な店は、

・秋月電子部品

土曜によく行くことが多いので、自分は、この店に入るときは、「満員電車に突っ込んでくる」などといいます。土曜はとても混んでいます。ですが、日曜日は、すいていることが多いです。八潮にも店があって、そっちのほうが、品ぞろえがいいとかいうことを聞いたことがあります。

・千石電商

品数はいいです。いろいろなフロアがあるので、部品を探すのは大変だったりします。

・マルツ 1号店、2号店

自分は、この店は、最後の砦だと思っています。秋月と千石になかったら3つ目のこの店で大体そろうからです。

・Unidy ラゾーナ川崎

工具や、木材などはここで調達してくることが多いです。

5-24 まとめ

これで、自分の思いつく限りの伝えておくべき自分の知識含めた、電子工作をするときに使う部品の説明は以上です。これ以外にもあると思いますが、大体はこれらだけで一通りの電子回路は組めると思います。先輩の物は、4ページで、幅と文字の大きさも大きかったので、自分では、頑張ったなと思います。基本となる部品には、詳しく説明をしたつもりですが、途中の部分からは、仕組みなども、すべて書ききれなかったところもあります。詳しくは自分で調べてみてください。では、電子部品などについては、ここらで締めたいと思います。ここまで読んでくださってありがとうございました。

6. 全体についてのまとめ

「今年は、電子工作をあまりやっていないから、かけることも少ないだろう。だからいくつかに項目を分けて書いたら、いい感じになるんじゃないかなあ」とかわけわからないことをほざいてた昔の自分を殴りに行きたい気分です。今は。正直、こんな超大作になるとは思いもしませんでした。ここまで、最後まで、読んでくれた人は、猛者といってもいいでしょう。すごいと思います。また、ありがとうございました。今年も、物理部展 #2018 に来ていただき、ありがとうございました。ぜひ、Asano the Bestへの投票もよろしくお願いします。

7. 参考文献まとめ

参考文献はここにまとめておきます。コメントを書いているところもあるので、そこも参考にどうぞ。

【4足歩行ロボット製作記】

鈴木美朗志、Arduino でロボット工作をたのしもう！、2014年、313ページ
この本は、5でも参考にしています。

<https://www.youtube.com/watch?v=ttDdMgSXlc8>

7月27日アクセス

【超電導（超伝導）について】

<http://www.istec.or.jp/description/description.html>

<http://www.istec.or.jp/description/movie.html#point-1>

<http://www.linear-museum.pref.yamanashi.jp/about/structure.html>

<https://kimika.net/rr3serusettai.html>

<http://www.miekyo.com/newpage5.htm>

<https://ja.wikipedia.org/wiki/%E8%B6%85%E4%BC%9D%E5%B0%8E>

全て7月23日アクセス

【超音波について】

https://www.honda-el.co.jp/hb/1_1.html

<http://commonpost.info/?p=100243>

https://www.honda-el.co.jp/hb/3_21.html

<https://ja.wikipedia.org/wiki/%E5%91%A8%E6%B3%A2%E6%95%B0>

全て7月23日アクセス

【3Dプリンタについて】

<http://www.ricoh.co.jp/3dp/what/>

<http://d-engineer.com/3dprint/3dprinterzairyou.html>

https://ht-deko.com/3d_printer/

全て 8 月 1 日アクセス

【電子工作で使われる主な部品について】

<https://ja.wikipedia.org/wiki/%E3%83%88%E3%83%A9%E3%83%B3%E3%82%B8%E3%82%B9%E3%82%BF>

<https://ja.wikipedia.org/wiki/%E6%8A%B5%E6%8A%97%E5%99%A8>

7 月 29 日アクセス

[https://ja.wikipedia.org/wiki/%E3%83%8E%E3%82%A4%E3%82%BA_\(%E9%9B%BB%E5%AD%90%E5%B7%A5%E5%AD%A6\)#%E3%83%87%E3%82%A3%E3%82%B6](https://ja.wikipedia.org/wiki/%E3%83%8E%E3%82%A4%E3%82%BA_(%E9%9B%BB%E5%AD%90%E5%B7%A5%E5%AD%A6)#%E3%83%87%E3%82%A3%E3%82%B6)

<https://www.murata.com/ja-jp/campaign/ads/japan/elekids/compo/capacitor>

<https://ja.wikipedia.org/wiki/%E9%9B%86%E7%A9%8D%E5%9B%9E%E8%B7%AF>

<https://www.omron.co.jp/ecb/product-info/basic-knowledge-series/basic-knowledge-of-relays/part1-relay-from-the-beginning/basics/defination>

<https://ja.wikipedia.org/wiki/%E7%B6%99%E9%9B%BB%E5%99%A8>

8 月 1 日アクセス

辻正敏、設計のための基礎電子回路、2017 年、249 ページ

この本は、1 から、回路設計について学びたい人にお勧めの本です。小さい割に 2800 円 + 税ですが、基本的なことは、問題を通しながらでも学ぶことができます。ただし、駅前の大きい本屋さんなどに行かないことが多いです。Book Express などには売っていません。

編集部後記

Positron 編集部

今年の部誌は、僕が知る限り、内容然り、ページ数然り、過去最高のものとなっております。例年よりかなり内容に力を入れたので、是非良いものになっていると信じたいです。

今回、中2たちのやる気がヤバいので、部誌が多めになっております。これからの成長が楽しみです。

中学代表後記

中2の誰か

今年、中2勢では文化祭等の部の仕事をかなり頑張ったつもりです。また、文化祭も心なしか展示数が多いかと思います。それもあってか、今年は部誌の量が凄いです。ここまで全部読み進めて頂いて、ありがとうございました。~~（まさか後ろから読んだなんて…）~~

これぐらいのやる気で今後とも物理部を盛り上げていきたいです。

名 称	部誌
原 材 料 名	陽子、中性子、電子
内 容 量	60063 文字(外装含まず)
賞 味 期 限	欄外に記載
保 存 方 法	空気中で保管する際は、紙の発火点に注意してください
販 売 者	浅野学園物理部

賞味期限：上記欄内に記載

栄養成分表示 1 冊当たり	
食物繊維（セルロース）	レタス 約 30 個分
鉄分	ドライフルーツ 約 1000 個分
エネルギー	質量×c ² + 部員の熱意
ビタミン L	0.0t

※出典：日本食品標準成分表 2015 年版（七訂）より、レタス（土耕栽培、結球葉、生）、フルーツ（乾）の成分を参考に、レタス 1 個を 350g、ドライフルーツ 1 個を 10g として割り出しました。

●この部誌の内容の正確性、信頼性、完全性は保証できません。●水に溶けにくいので、水洗トイレに流さないでください。●本製品を廃棄するときは、法とあなたの良心に従ってください。●洗濯はお避け下さい。特に、他の物と洗濯すると、衣服に細かい紙くずが付着する恐れがあります。●電子レンジで加熱しないでください。万一、電子レンジ内で発火したときは、ドアを閉めたまま電源プラグを抜き、鎮火するのを待ってください。ドアを開けてしまうと、庫内に酸素が入り、勢いよく燃える恐れがあります。●乳幼児が読む際には、誤読に注意してください。●火災の原因になるので、アイロン台として使用しないでください。●一度に多く読みすぎるとお腹が痛くなることがありますのでご注意ください。●まれに誤字や脱字が含まれる場合がありますが、部員の疲れによるものです。そのまま読んでも健康には問題ありません。●体質によりまれに身体に合わない場合があります。その場合は読むことを中止してください。●薬を服用している方、通院中の方は担当専門医にご相談の上、お読みください。●本商品は暇つぶし用として設計されております。高い信頼性を要求されるシステムには応用しないでください。社会的に大きな混乱が発生する恐れがあります。●ごくまれにトマトのヘタが混入している場合がございますが、健康には影響ありません。取り除いてお読みください。●本商品の使用方法をはき違えないでください。

次のようなスペースに放置しないでください。火災の原因になることがあります。

- 膨らんだスマートフォンの上。
- 単体のフッ素の近く。
- SNS に投稿された不適切な言動、またはその返信。