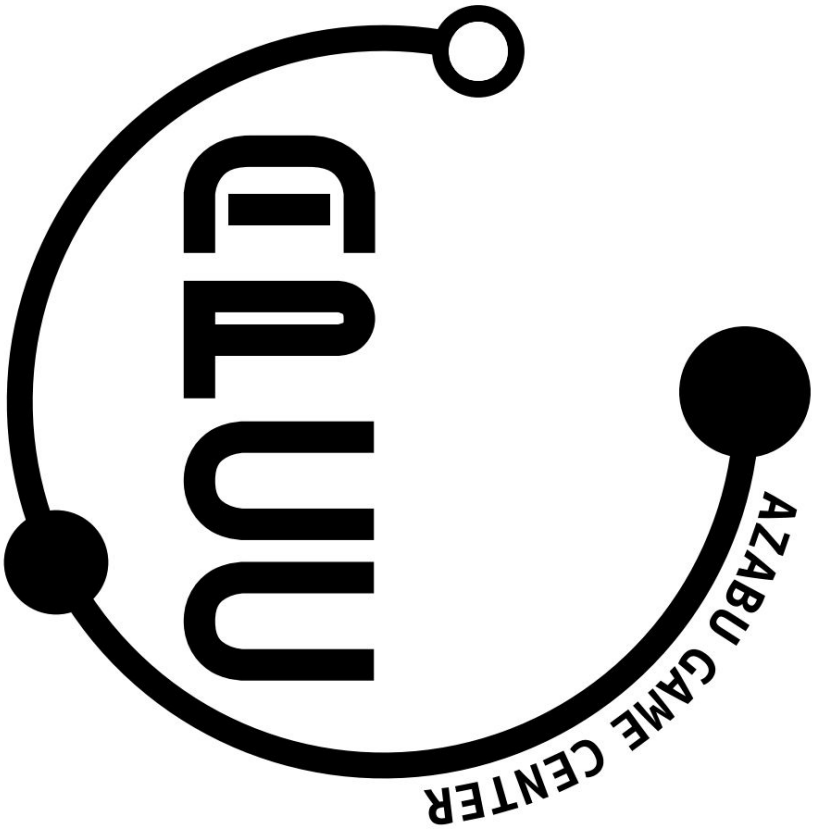


NMI 2019

目次

ゲームをAIする

H2 SugarDragon5

こんにちは。この度はAZABU GAME CENTERにお越しいただき、誠にありがとうございます。今回展示責任者を務めました高校2年のSugarDragon5と申します。

この記事では、日常にありふれた多種多様な「ゲーム」について、それらにおける必勝戦略とその導き方について「カモでもわかるカモ!?」[1]を目標に考えていきたいと思います。

ゲームとは

みなさんは「ゲーム」という言葉を聞いて何を思い浮かべるでしょうか。AZABU GAME CENTERでは様々な形態、ジャンルのゲームを展示しているわけですが、一般的に「ゲーム」と言った場合に指すものはこれに限りません。例えば紙と鉛筆で行うマルバツゲームであったり、道具を用いず己の手のみを使うじゃんけんだっていわゆるゲームです。それに対してどのようなものがゲームであって、どのようなものがゲームでないか、昔から哲学者や偉い人たちは上手な定義をしてきました。それに対して今回は次の条件を満たしたものを指して「ゲーム」という言葉を使い、考えていくものとします。

- プレイヤーが二人である
- 片方のプレイヤーが一定の利益を得ると、もう一方は一定の損害を受ける
- 有限の手番でゲームが終了する
- ランダムな要素が存在しない
- プレイヤーに隠されている情報がない

具体的には、先程挙げたマルバツゲームや、チェスや将棋は今回考えるゲームで、じゃんけんやバックギャモンはそうでないということになります。

さて、これらの条件を満たすゲーム[2]において、理論上プレイヤーは終了までの手をすべて読み最善の手を決定することができます。そして双方のプレイヤーが最善手を指し続けたとき、ゲームの結果は次のどれかに定まることが知られています。

- 先手必勝　　ex.）五目並べ(初期)
- 後手必勝　　ex.）6x6オセロ
- 引き分け　　ex.）マルバツゲーム(3x3)

---

[1] 今年の記事はカモには少し難しいかもしれないのでタイトルから「カモでもわかるカモ!?」は撤去されました

[2] ゲーム理論の言葉では「二人零和有限確定完全情報ゲーム」と言うそうです

したがって、今回考えるゲームではどのようなものであっても先手後手のどちらかには「少なくとも負けない戦法」が存在するわけです。そのような戦法(以降必勝法と呼びます)がどちらにあるのか、そしてそれはどのような戦法なのかということを具体的なゲームをいくつか見ていくことで考えていこうと思います。

碁石取りゲーム

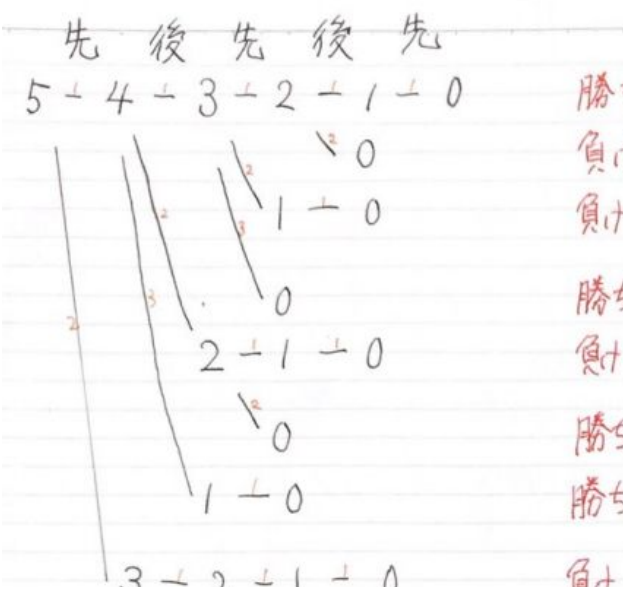
ルール：碁石21個からなる山があります。二人のプレイヤーは交互に山から碁石を取っていきます。最後の一つをとったプレイヤーの勝利です。ただしプレイヤーが一度に取れる碁石は1～3個です。

まずはこのゲームの必勝法を考えてみたいと思います。その前に、このゲームが先程の条件を満たしていることを確認しましょう。

- プレイヤーが二人である　→OK
- 片方のプレイヤーが一定の利益を～～　→OK(片方が勝つときもう片方は負ける)
- 有限の手番でゲームが終了する　→OK(最大でも21回取れば終わる)
- ランダムな要素が存在しない　→OK
- プレイヤーに隠されている情報がない　→OK

よってこのゲームは先程の条件を満たしています。更にこのゲームではかならず勝敗が決する(引き分けがない)ことから、先手か後手のどちらかに必ず必勝法が存在することになります。

ではその必勝法を探して行きたいですが、闇雲に様々な戦法を試しても簡単に発見できるものではありません。そのようなときに有効とされる手法の一つがゲーム木(樹形図)を書いてみることです。簡単のため21個の碁石を5個にしてゲーム木を書くと右のようになります。



これを眺めてみるとその時の手番の最善手が見えてきます。つまり、ある手を選んだとき、その勝者が自分自身しかない時、それが最善手です。そのような最善手を選び続けたときプレイヤーは勝ち、そのような手が無く、相手が最善手を選び続けたならば負けます。このようにして任意のゲームは解析することで最善手及び必勝の条件を見つけることができます。

しかしこれには問題があります。右の図で21個あるはずの石を5個に減らして、その上で省略していることからも察しが付くように、複雑なゲームにおいて、そのゲーム木は膨大になるのです。

その程度を見積もってみましょう。碁石の数をnとしたときの状態数(あり得る状態の数)をf(n)と置くと、f(k)=f(k-1)+f(k-2)+f(k-3)、f(1)=1なのでこれは有名なトリボナッチ数列となります。したがって一般項f(n)は

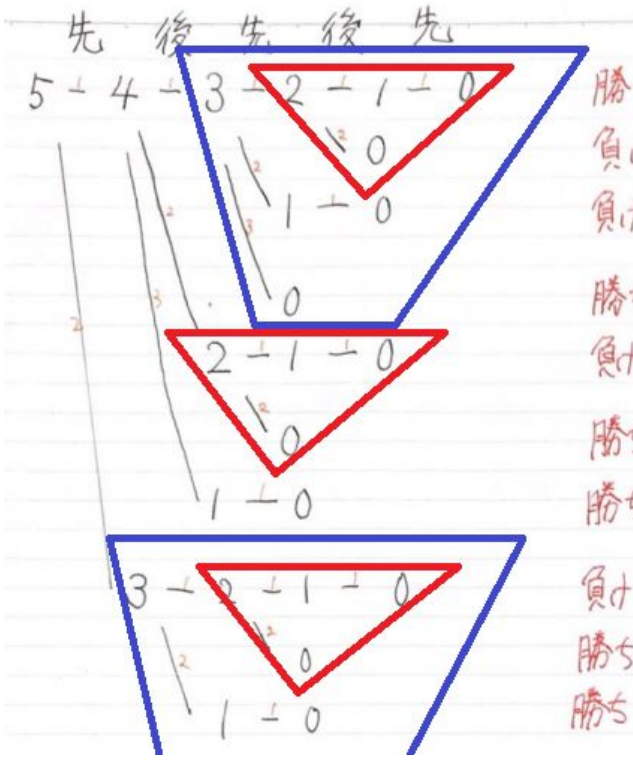
$$f(n)=\frac{3((a+b+1)/3)^{n+3}}{a^2+b^2+4}\qquad (a=(19+3\sqrt{33})^{1/3}, b=(19-3\sqrt{33})^{1/3})$$

になります。これはnが増加するにつれて指数関数的に増大し、解析にかかる時間もまた爆発的に長くなります。具体的に、f(100)=1803963808151009012141576 39となり、毎秒10^9個探索できると仮定して57億年ほど必要です。苦痛ですね。

つまり、このようにすべての状態を探索して最善手を見つけられるゲームというのは非常に単純なもののみなのです。もう少し上手な方法を考えなければ必勝法の導出は困難でしょう。

効率的なゲーム木探索

では、愚直な探索を工夫して高速に最善手を導くことを考えます。再び先程のゲーム木を眺めてみましょう。ある数字が異なる文脈で出たとしても、その後の木の形は全く同じものになります。取れる個数はそこまでにどのようなとり方をしてきたかに関係ないからです。よって一度計算した区間に関してその結果を保存して再利用できれば再び計算する必要はなくなります。このような考え方を**動的計画法**といいます。



残りの個数がn個で手番が回ってきて、その後両者ともに最善手を選んだときの勝敗をP(n)で表し、P(n)=1ならば勝ち、P(n)=0ならば負けであるとします。(ただし便宜上n≦0ではP(n)=0であるとします)

このとき、n個からはn-1個、n-2個、n-3個に遷移できるわけですが、実は

P(n)=1 ⇔ P(n-1)=0 又は P(n-2)=0 又は P(n-3)=0

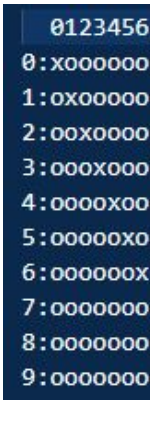
が成立します。自分が勝つためには、相手に必敗の盤面を渡せばよく、候補に一つでも必敗の盤面があれば自分は必勝にできるからです。
これとP(1)=1を利用すると、nが小さい方から順番にP(n)を求めることができ、n回の計算で盤面が必勝か必敗か確認することができます。また、必勝であるときの最善手も簡単に計算することができます。必勝ならばP(n-1)、P(n-2)、P(n-3)のどれかは0なので、0であるようなものを選び続ければ勝利できます。
このように状態に対して不変な条件を見つけるとすべて探索しなくても最善手を見つけることができるのです。
実は、もう少し観察してみるとより簡単な勝敗の判定法を見つけることができます。P(n)をnが小さい方から順に並べてみましょう。

n	1	2	3	4	5	6	7	8	9	10	11	12
P(n)	1	1	1	0	1	1	1	0	1	1	1	0

このように、P(n)は{1,1,1,0}が繰り返された数列になります。よって、実はnを4で割った値が0ならば後手必勝、そうでなければ先手必勝なのです。

碁石取りゲームII

ここまでで、山の個数が1つで一度に3個まで取れる碁石取りゲームは石の個数を4で割ったあまりを考えることで必勝かどうかを判定することができるとわかりました。ではこれをもう少し複雑にして、次のゲームを考えてみましょう。
ルール：碁石の山がN個あり、k個目の山はAk個の碁石からなります。二人のプレイヤーは交互に「碁石のある山を選び、1個以上碁石をとる」という操作を行います。この操作が行えなくなった方の負けです。
このゲームは「Nim」と呼ばれているものですが、今度はこのゲームについて必勝法を考えていきたいと思います。勿論このゲームは先程の碁石取りゲームより複雑(状態数が多い)なので、愚直なゲーム木探索で探索するのは困難です。したがって今回も盤面を評価できる式を立てるのが目標になります。まずN=2で実験してみます。探索の結果、右のような結果が得られました。2つの山AとBに対して、A=Bならば負けで、そうでなければ必勝になります。次の考察からこれは証明できます。
(証明)
・A=B≠0の時、どのように取っても相手にA≠Bが回る
・A≠Bのとき、A=Bとなるように大きい方から|A-B|個取ることで相手にA=Bを回せる



よって、A≠Bが回ってきたとき、その後は常に相手にA=Bを回すことができます。A、Bは単調減少していくため、いつかはA=B=0が相手に回ることになり、相手は負けます。従って最初の盤面がA≠Bならば必勝、そうでなければ必敗になります。　　（Q.E.D.）

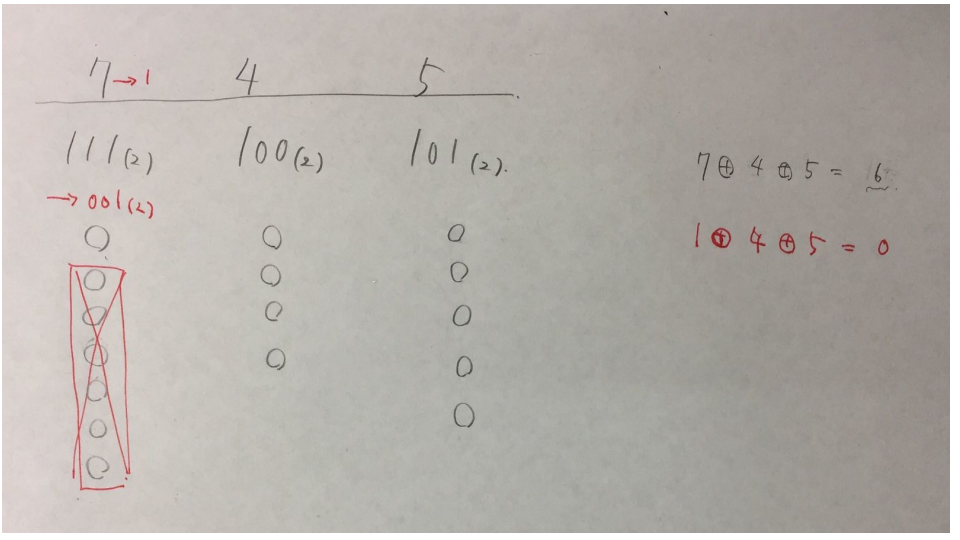
さて、以上でN=2のときは必勝法がわかったわけですが、今このゲームでは一般のNを考える必要があります。N=2の結果からなんとなく、石の個数に対する演算の結果が0かどうかで必勝かどうかわかりそうな気がしてきますが、具体的な演算を自力で発見するのは少し難易度が高いです。よって、今回は結果のみを参照します。

$$A_1 \oplus A_2 \oplus A_3 \oplus \ldots \oplus A_N \neq 0 \Leftrightarrow \text{先手必勝}$$

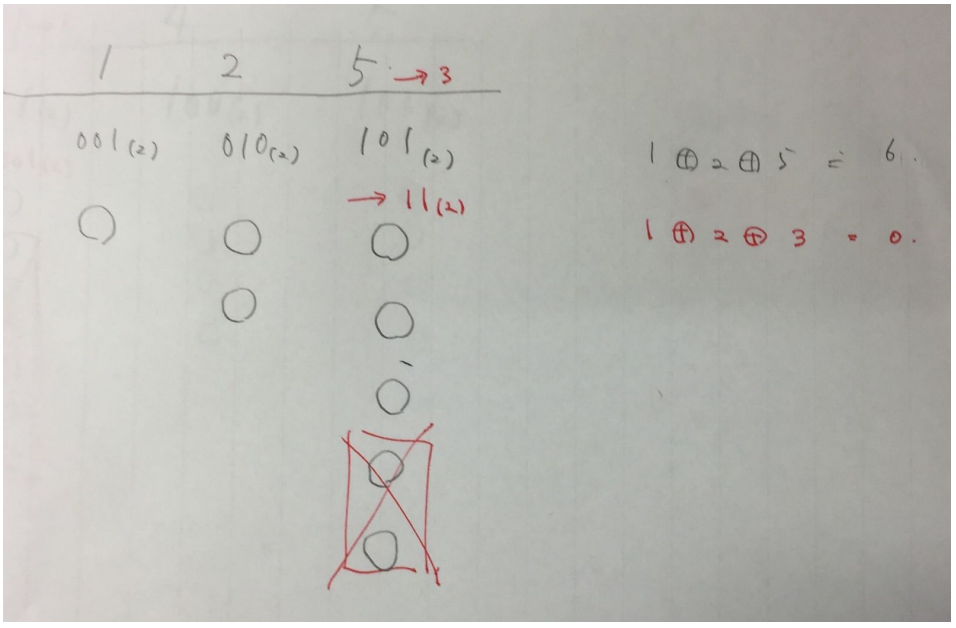
ただし、演算子$\oplus$は2進数表記における排他的論理和を示します。つまり、2数を2進数表記をしたときに各桁についてそれらが異なっていれば1、同じなら0を返す演算です。

この式では少し分かりづらいと思うので、操作の方法とともに例示したいと思います。

N=3, {7, 4, 5}を考えます。全体の排他的論理和が0になるように相手に回すために、7個の山を1個にします。



次に後手が2番目の山から2個取ったとして、このとき排他的論理和は0でないため、再び0になるようにして渡します。



これを繰り返すと相手から排他的論理和が0の形を渡し続けることができ、最終的にすべて0の局面が渡ります。このようにして勝利することができました。

Grundy数

ここまでで確認したとおり、Nimは山ごとの碁石の数の排他的論理和を取ることで盤面が必勝であるかどうかを判定することができます。排他的論理和という操作は日常生活において馴染みが薄い演算であり、ゲームの必勝法に突然登場することに驚かれるかもしれませんが、ゲーム理論では排他的論理和を用いて判定することは有名な手法です。

実は今回扱ったようなゲームについては、一般に次のように盤面に数を割り当てることで簡単に勝敗の判定を行うことができます。

1. これ以上ゲームが進行しない盤面に0を対応させる。
2. ある盤面Aから進むことができる盤面の数の集合をA’とし、AにはA’に存在しない最小の非負整数を対応させる

このようにして盤面に対応付けた数をGrundy数と呼びます。これを用いて勝敗を確かめるために、先程使った排他的論理和を使います。盤面ごとのGrundy数の排他的論理和によって勝敗が確認できるのです。

先ほどのNimを考え直してみます。まず山ごとにGrundy数を決定したいわけですが、これは極めて単純で、m個碁石が残っているときその時のGrundy数はmです。これは簡単な数学的帰納法で示すことができます。そして、山ごとに求まったGrundy数に対して排他的論理和を取ると勝敗

が決定します。結局、碁石の個数とGrundy数が一致することから勝敗を判定したければ碁石の個数の排他的論理和を取ればよかったわけです。

練習問題

Grundy数を用いると様々なゲームの勝敗を決定することができるということを確認するために、プログラミングコンテストで実際に出題された問題を載せておきますのでぜひ考えてみてください。

パソコン甲子園2018本選 問10 石遊び

幸四郎くんと浮子さんは白と黒の石を使った遊びをしている。この遊びは以下のような内容である。

１．遊びの開始時に、いくつかの場を作り、それぞれの場に白石と黒石をそれぞれ１つ以上置く。

２．幸四郎くんと浮子さんは交互に、場をどれか選び、その場に対して以下の中から一つを選んで行動 する。

（ア）白石を場から１つ取り除く。

（イ）白石の数を超えない個数だけ、黒石を１つ以上場から取り除く。

（ウ）黒石を１つ白石と交換する。このとき白石は石入れから持ってくる。石入れには十分な数の 白石が入っているものとする。

３．先に２ができなくなった人が負け。

おわりに

ここまで、いくつかの具体例をもとにゲームの必勝法を編み出す方法を考えてきました。オセロのようなより複雑なゲームにおいてGrundy数など今回扱った数学的な手法が直接使えることはまずありませんが、ゲーム木の探索や動的計画法の利用などの部分では大きく役に立ちます。もし今回の記事でゲーム理論に興味を持っていただけたのであれば他の様々なゲームについて調べてみるのが良いと思います。拙い文章ではありましたがここまでお読みいただきありがとうございました。

Vimのすすめ

H2 魔王の嫁

いんとろだくしょん

ついに5年間も魔王の嫁を名乗っていて、自分の頭大丈夫かみたいな気分になっています。魔王の嫁ともなると毎日のパソコン雑務が多いので(ホントか？)、今年は文章の入力を効率的にする方法をお教えしたいと思います。なお、タイピング練習については詳しく言及しないので各人練習いただければ幸いです。僕は主にe-typing[3]で練習しました。タイピング練習をするときは、どんなときでも速度より正確さを重視したほうがいいですよ。ホームポジション[4]の乱れは心の乱れです。レベルが同じくらいの友達がいるならTyping War[5]で対戦してみるのもいいと思います。

さて、この記事を読んでくれている小学生の諸君や、保護者の方の中にはパソコンを使ってよく文章を書く、という人もいると思います。そんなとき、2行前に間違いを見つけたらどうしますか？矢印を使う？マウスやタッチパッドを使う？そのどちらも解決策の一つだとは思います。しかし、それって指をホームポジションから動かすことになってちょっと時間がもったいないな、と思いませんか？かく言う僕も中学生の頃ずっとこの問題に苦悩していました。そう、Vimに出会うまでは。

Vimって何？

実は去年後輩の記事にちょろっと登場しているので見たことがある人もいるかもです(パー研に2年来てくれたあなたにありがとう)。Vimとは文章を編集するテキストエディタの一つで、矢印キーもマウスもなかった時代に原型が開発されました。矢印キーやマウスで辟易としてるなら、それがない時代に作られたものを使ってやればいいという寸法ですね。

---

[3] https://www.e-typing.ne.jp/ 僕の最高スコアは413pt。
[4] 左手をASDFに、右手をJKL;に。両手の指を使う最良の方法と考えられている。
[5] http://typingwar.trap.games/

Vimでは、「コマンドモード」「編集モード」と二つのモードを使うことで矢印を使わずに文章を操作することを可能としています。「コマンドモード」では打ったキーボードの文字が「コマンド」として認識され、「編集モード」では打ったキーボードの文字がそのまま入力されます。...何のことやらわからないでしょう。具体的な操作を軽く見ていきましょう。

頼れる相棒、コマンドモード

Vimを起動すると最初はこのモードです。文字を打っても入力されなくて、初心者が一番最初に困り果ててしまうのがこの場所です。

- :q!<Enter> 保存せずに終了するコマンド。:はLの2個右にあります。"Quit"(やめる)の頭文字と覚えておくとよいです。終わり方を知ってればいじってみるのもこわくない！
- i 編集モードに移ります。新しい文章を作りたいのに文字が出てこない…と困った時に真っ先に押すべきボタン。
- h, j, k, l それぞれ←↓↑→に動くコマンド。急に難易度が上がった。ぶっちゃけいっぱい使うくらいしか上達法はないとは思いますが理屈をこねておくと、ホームポジションの人差し指が押せるJが一番良く使う操作の↓で、残りは直感的に覚えられるかなと思います。
- x 文字を消すボタンです。設定しないとおなじみBackSpaceキーが使えなかったりで、結構お世話になります。
- :w 保存するコマンド。"Write"(書き込む)の頭文字と覚えるといいですね。
- :wq 保存して、終了するコマンド。"Write"+"Quit"です。

もっとたくさんコマンドはあるのですが、初めて使うならこのくらいで十分だと思います。hjklで動くのだけでも大変なのに、たくさんのコマンドを覚えようとしたらパンクしちゃいます。無理せずゆっくり、です。

いつもの挙動、編集モード

コマンドモードから特定のボタン(紹介した中ではi)を押すと移るモードです。基本的には入力した文字がそのまま入力される普通のモードだと思ってください。

- Esc コマンドモードに戻る。Escなんて聞いたこともない人がいるかもしれない、大体のパソコンでは一番左上の辺りにあります。

だいたいEscだけあれば十分です。というのも僕が編集モードで使えるコマンドにそこまで熟達してなくても使えてるので、ハードルの高さを感じることはないかなと思います。Ctrl+R→=とか押すと計算したりはできます。

秘伝のタレ、.vimrc

Vimのもう一つの特徴は、その高いカスタマイズ性にあります。.vimrcというファイルをいじることで、Vimはあなたの色に作り変わっていき、手放せないエディタとなるでしょう。ここで一言だけ。ネットを調べたりするとプラグインを入れることを推奨する記事をよく見かけるかもしれませんが、僕個人としてはあまりおすすめできないかなと感じています。なぜなら、どんな機能があるのかもわからないままいろいろと付け足してもどのプラグインが効果を発揮しているのかわからなくなってしまうからです。プラグインが増えれば増えるほど管理は大変になりますし、起動も遅くなります。一通り触ってみて、なんで標準機能にないんだろう、と思えるもののみインストールするべきです[6]。

標準の機能で行えるカスタマイズで、僕の作業効率を爆上げしているものをひとつだけ紹介します。

[6] https://postd.cc/vim3/ この記事の考え方に感銘を受けて上記のように考えています。プラグイン紹介としても良い記事です。

- inoremap <silent>jj <ESC>[7] 入力モードでjjと入力するとEscと入力したことになります(=コマンドモードに戻ります)。ホームポジションに指を置いたまま入力できるのがVimの強みなのにEscが超遠いというジレンマを見事に解消してくれます。代償としてjjが一拍置かないと入力できなくなりますが、普段の生活でjjと打ちたくなること、ないでしょう？(お察しの通り、今僕はこの記事にjjと入力しています。すげえ手間取ってます。)

おまけ: Vim vs. Emacsの構図について

Vimに興味を持ってくれて、ウェブを調べてくれたならVimと似たような設計思想で作られたEmacsとの「宗教戦争」、「エディタ戦争」という単語をたまに見かけるかもしれません。ですが、僕がEmacsについて書いてないのは単純に使ったことがないからで、大概はネタの範疇であり本当に互いのことを貶めあっているのは見たことがありません。Vimを合わないと感じたならEmacsを使ってみてもよいし、どちらも合わないならAtomやVisual Studio Codeに行けばいいと思います。あなたが「自分の好き」を見つけて、ストレスフリーな文章編集を行えることを祈っています。

拙文をお読み頂き、ありがとうございました！

[7] https://routecompass.net/vim-jj/ 見たとき疑ってたんですけど便利すぎる。感動しました。

誰も知らないマイナーブラウザの世界

H2 ありゃお

ブラウザってなんだ？

みなさんは、「ブラウザ」と言われてピンとくるだろうか。パソコンを持っている方なら聞いたことがあるという人も多いと思うが、意味はよく知らないという方もいるかもしれない。「ブラウザ」とは、簡単に言えば「ネット上のパソコンにしか読めない言語を翻訳してくれる」ものである。WindowsユーザーならInternet ExplorerやMicrosoft Edge、MacやiPhoneユーザーならSafariなどの標準ブラウザは、誰しも一度は目にしたことがあるだろう。

一方最近では、性能の高さや拡張機能の豊富さからGoogle ChromeやFirefoxなどのブラウザがシェアを伸ばしており、ブラウザ市場はこれらの有名ブラウザが独占していると言える。

しかし、有名ブラウザは既に使っている方も多く面白みに欠けるため、本記事では決して有名とは言えないブラウザにも焦点を当てたいと思う。

有名ブラウザ

まずは先ほど挙げたメジャーなブラウザについて簡単に説明していく。

・Internet Explorer

通称IE。Windowsの標準ブラウザ。Microsoft社がWindowsと抱き合わせで販売することで爆発的にシェアを伸ばしたブラウザで、インターネットを黎明期から支え続けてきた。Windows10の発表に際して開発が止められ、代わりにMicrosoft Edgeが標準ブラウザとなった。

・Microsoft Edge

通称Edge。IEに代わるブラウザとして開発され、Windows10から搭載されたブラウザ。IEとの互換性を保ちつつ、不要な機能を大量に排除し、直感的な操作が可能になっているらしい。正直あまり使わないためよくわからない。

・Google Chrome

ちょろめと呼んだり呼ばなかったりする。かのGoogle様が提供するブラウザで、国内・世界ともにトップシェアを誇るブラウザ。とりあえずこれを使っておけば間違いはないと思われる。普及率の高さがそのまま魅力になっていると言える。

一方メモリの使用量が大きめであるなど、環境によっては使いにくい場合もある。

・Firefox

通称は火狐など。世界シェアはChromeに次ぐ第2位のブラウザ。オープンソースであるため拡張性が抜群に高い。一方で脆弱性もあって、いろいろな意味で多少上級者向け。何故か多くの信者とアンチを抱えており、ウェブブラウザ論争の中心にいがち。筆者の父親はこれ。

擬人化されると大抵狐耳巫女になる。かわいい。

・Safari

iOSの標準ブラウザ。iPhoneの普及に伴ってスマホブラウザにおけるシェアも高い。筆者はMacPCを持っていないためよくわからないが、メモリの使用量が少なく、直感的な操作ができるのが利点らしい。ただ拡張機能が多くなく、自由度は低いブラウザと言える。

マイナーブラウザ

ここからは、シェアの高くないブラウザについて紹介していく。

・Opera

北欧産の古参ブラウザ。根強いファンに支えられている。標準で広告ブロックの機能が搭載されており、これにより読み込み速度が速い。操作感がChromeに近く、比較的乗り換えはしやすい。その上「Chrome Extention」という拡張機能を用いることでChromeの拡張機能が使えてしまう。ここまでするならChromeを使えというのは内緒。

日本での普及率は決して高くないため、Chromeユーザーに「俺はOperaだけどね」と言って差を付けたい人向け。

・BLISK

web開発者からの支持を多く集めるブラウザ。ページのさまざまなUAにおける挙動が瞬時に確認できるのが売り。UIもChromeと似ているため、かなりとっつきやすい。web開発者からすると神のようなブラウザらしい。

・Vivaldi

筆者が今メインで使っているブラウザ。比較的新しいブラウザで、Operaの元CEOが開発している。デフォルトでマウスジェスチャなどの便利機能が搭載されている。タブの休止機能により動作を軽くできたり、新しく開くタブのURLを指定できたりする。贔屓のブラウザだけに利点を挙げればきりがないのだが、特に気に入っているのがウェブパネルという機能。これは指定したサイトを画面の端に表示してくれる機能なのだが、ここにTwitterを入れると、なんと縦画面に適したモバイル版を表示してくれるのである。これによりネットサーフィンとTwitter巡回

を同時に行えるという悪魔的ブラウザである。(ちなみにグラブルなどのスマホ向けブラウザゲームも快適にできる)
　欠点はTwitterが覗くせいで調べものが捗らないこと。

・Sleipnir

　通称プニル。理由は「スレイプニル」だと不適切な文字列が含まれるため。
　これも筆者の好きなブラウザ。珍しく純国産のブラウザで、ナショナリズムが煽られる。魅力は何と言ってもアイコンがかっこいいこと。これは他にはない大きな利点である。こちらもマウスジェスチャがデフォルトで搭載されており、拡張性も高い。起動時に、前回の終了時に開いていたページがそのまま復元されるのも便利である。また「100個開いてもすぐ見つかるタブ」を謳う通り、ページのプレビューが表示され、瞬時にどんなサイトだったかがわかる。
　モバイル版のアプリもリリースされており、タブのプレビュー表示はそのままに、高速スワイプができるなどこちらもなかなか使いやすい。

・Lunascape

　通称ルナたん。こちらも国産ブラウザであるが、知名度ではSleipnirに劣る。業界初の3つのレンダリング・エンジンを搭載したブラウザであり、これで読めないサイトはないと思われる。GoogleからCookPadまであらゆる検索エンジンが元から用意されており、登録することでワンタッチで呼び出せる。またマウスジェスチャやUAの変更もデフォルトで搭載されており、人によってはかなり使い勝手がよく感じるだろう。

まとめ

　ここまでいくつかのブラウザを紹介してきたが、まだまだこれ以外にもウェブブラウザは数多く存在する。何が言いたいかというと、OSの標準ブラウザでもトップシェアのブラウザでもないウェブブラウザにも、興味を持って欲しいということである。知らなかったブラウザの知らなかった機能によって、この記事を読んで下さった皆さまのインターネットライフが充実したなら幸いである。

定数倍高速化について

QCFium

プログラムを高速化したい場合、まずアルゴリズムの改良を考えるべきですが、そのような最適化以外の高速化について書いていきます。

競技プログラミングと定数倍高速化

競技プログラミングは定数倍高速化と関係があるので書きました。
競技プログラミングとは与えられた問題を解くプログラムを書いて提出する競技です。(競技ではなくオンラインゲームとする説あり)
国内ではAtCoder社などがほぼ毎週コンテストを開いています。
ある程度難易度が上がってくると、問題を指定されたままに解くプログラムをそのまま書いたのでは実行時間制限(2秒程度が一般的)を超過してしまい不正解となる問題が出現してきます。
このような問題のほとんどはより効率的に解くアルゴリズムが存在し、それを考えるのが本質となります。定数倍高速化とはこのようなアルゴリズムの高速化**__ではなく__**、アルゴリズムを本質的に変えないままプログラムを高速化することです。ごくたまに非効率的なアルゴリズムを定数倍高速化することで実行時間制限に間に合い、正解できることがあるのです。

実際のソフトウェア開発では定数倍高速化はアルゴリズムの改良をし終えてもなお高速化が必要な際に試されることが多いです。

実例

定数倍高速化によって実行時間制限に間に合った例を紹介します。
__情報オリンピック 2019 春合宿 1日目 1問目(問題名：試験(Examination))__
情報オリンピックの春合宿は例年情報オリンピック本戦の成績上位20人程度が参加し、国際情報オリンピックの日本代表を決めるものです(僕はなれませんでしたが)。4日間に渡って1日3問程度を解くのですが、これは1日目の1問目です。

問題概要は以下のとおりです：

N人の人間が数学と情報のテストを受けた。全ての人間の数学と情報の得点が与えられる。
以下の質問にQ回答えよ(X,Y,Zがそれぞれ Q個ずつ与えられる)：
　数学がX点以上、情報がY点以上かつ数学と情報の合計がZ点以上の人間は何人いるか？

Nは10の5乗以下、Qも10の5乗以下
各点数は正で、10の9乗を超えない
実行時間制限：3秒

普通に解くと、各質問に対して全ての人間を調べて条件を満たすかをチェックして条件を満たす人間をカウントして答えを出力するという感じになります。
ところがこの解法では条件をチェックする回数は全部でN × Q回、すなわち最大10の10乗回になり、さすがのコンピュータでも3秒で処理し切ることはできません。
この問題にはこれより遥かに効率的なアルゴリズムが存在し、これが想定解となっています。この解法はかなり難しいのでここでは紹介しません、知りたい方は
https://www.ioi-jp.org/camp/2019/2019-sp-tasks/index.html から公式解説を見ることができます。
ところが後の項で紹介するベクトル化とループアンロールという手法を使って定数倍高速化することで普通の解法を実行時間制限に間に合わせることができました。

これで春合宿の楽しみが2倍くらいになりました。

手法

実際に定数倍高速化に使われる手法を紹介します。

ベクトル化

最近進化している技術です。
コンピュータの処理をするCPUという部品は、普通は足し算、掛け算、比較などの処理を4バイトや8バイトごとに行います。ところが、CPUが

ベクトル化に対応していると適切な命令を与えてあげれば16バイト,32バイトや64バイトの処理を一度にすることができます。

ベクトル化は上からわかるように似たような処理を何度もする場合に高速化が効きやすいです。
上の情オリ春合宿の例でいうと「全ての人間に対して条件をチェックする」の部分が「条件をチェックする」という処理を「全ての人間に対して」、つまり何回も行っているためこの方法が効きやすいのです。
どうベクトル化の命令をCPUに与えるかが問題ですが、簡単な方法は**コンパイラに任せる**です。CやC++で、ソースコードの先頭に以下の二行を追加すると結構ベクトル化してくれます。(コンパイラとしてGCCを仮定します)

```
#pragma GCC target("avx")
#pragma GCC optimize("O3")
```

一行目はavxというベクトル化命令セットを(可能なら)生成するという意味です。avxの進化系としてavx2というものもあり、これはおおむね第四世代以降のCPUにおいて使用できます。avx2の方が速くなるので、avx2が使用可能なら一行目の**"avx"**を**"avx2"**に変えると良いです。
avx2のコードをavx2に対応していないCPU上で動かそうとするとクラッシュするので注意してください。
実は更に進化系のavx512というものがありますが、現状これに対応しているCPUはほとんどないので使われることはあまりありません。

二行目は、コンパイラに強い最適化を指示するもので、これを指定しないと一行目があってもベクトル化の命令を生成してくれないことがあります。

ループアンロール

小さい処理をたくさんループさせる場合にループを一定回数ごとに展開すると速くなることがあります。
具体的には

```
for (int i = 0; i < 10000; i++) {
  sum += a[i];
}
```

というループを

```
for (int i = 0; i < 10000; i += 4) {
  sum += a[i] + a[i + 1] + a[i + 2] + a[i + 3];
}
```

というループに展開します。
こうするとi < 10000のような継続条件の判定回数が減り、速くなります。「1回処理して1回条件判定」を10000回繰り返すより、「4回処理して1回条件判定」を2500回繰り返す方が条件判定の回数が減ります。但し条件判定に対して一回の処理が十分に大きいと、条件判定のコストが相対的に小さいため、あまり速くなりません。
今回は10000が4で割りきれるためこれだけで十分ですが、余りがでる可能性がある場合余った部分は別に処理をする必要があります。

実はこれも手動ではなくコンパイラに任せることができます。以下の一行をソースコードの先頭に追加してください。

```
#pragma GCC optimize("unroll-loops")
```

入力

プログラムは標準入力から入力を受け取ることがありますが、これが思ったよりも時間がかかります。
例えばC++で**std::cin >> n;**のような形で整数を標準入力から読み込むことを10の7乗回繰り返すと20秒ほどかかります。(これは環境によってだいぶ差が出ますが、WindowsのMinGW-w64/Core i7 4600Mではこのくらいの速度になりました、以下同じです)
scanf("%d", &n)というようにC言語の入出力関数を使うと10秒ほどになります。
更にgetchar関数を使って一文字ずつ入力を読んで自力で整数をパースすると3.7秒まで速くなります。
また、Windowsには_getchar_nolock関数というものがあり(Linux等ではgetchar_unlockedがこれに相当する)、これはgetchar関数と同じですがスレッドセーフではありません。その代わりに動作が速くなります。getchar関数の代わりに_getchar_nolock関数を使うと1.1秒になります。

実はほとんどの競技プログラミングサイトのように入力がファイルからのリダイレクトのような場合、更に速くすることができます。
fread関数で入力を文字列として一括で読み込み、後はgetcharと同様に自力パースすると0.3秒まで縮まります。
最初のstd::cinを使った方法から60~70倍も入力が速くなりました。
但しこの方法は最初にファイルからのリダイレクトなどで全ての入力が与えられないと使えないという欠点があります。

速くなるにつれコードは複雑になったり環境に依存したりするようになってしまいますが、これによって得られる速度も無視できません。競技プログラミングでは計算量が入力の個数と変わらないような解法ではほぼ確実に入出力がボトルネックとなります。それによって実行時間制限を超過することは稀ですが全提出での実行時間最速などを取りたい際には避けて通れない道です。

SVGの便利さに触れてみよう！

H1 すばる

はじめまして

　はじめまして、H1のすばるといいます。文章を書くことは苦手なので、読みづらい部分もありますがよろしくお願いします。
　さて、タイトルにあるとおり、今回のNMIでは、SVGの便利さについて触れていくわけですが、私自身まだSVGについて詳しいとは言えないので、間違っているところやぼかしているところもあるとは思います。どんどん心の中で指摘しちゃってください笑。

SVGの概要

　まず、SVGとは何なのかですが、”Scalable Vector Graphics”の略であり、W3Cという団体によって規定された言語です。SVG1.1の勧告書には次のように記されています:

> SVG は二次元グラフィックスを XMLで記述するための言語である。 SVG は３種類のグラフィックスオブジェクト：ベクター形式のグラフィック（例えば直線と曲線からなるパス），画像, テキストを扱う。 オブジェクトはグループにまとめたり, スタイルを付けたり, 変換を加えたり, すでに描画されたオブジェクトに組成することができる。 SVG の特能には、変換の入れ子, クリッピングパス, アルファマスク, フィルタ効果, ひな型オブジェクトも含まれる。
> (引用元:SVG1.1　日本語訳 - 概要)

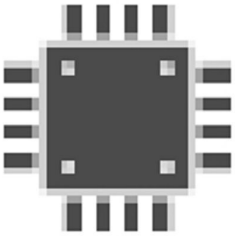
なにやらいろいろ書いてありますが、要は

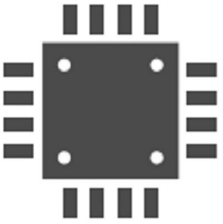
-XMLを使って2D画像が描けるよ
-ベクター形式, 画像, テキストの３つのオブジェクトで出来ているよ
-オブジェクトは自由に扱うことができるよ

といった感じです。

なんでSVGを使うのか

1. 画質劣化がない





SVGのもっともいいところは、画質劣化が発生しないことです。ベクター形式で管理されているため、自由に拡大縮小できます。右の図を見ると、その違いは明らかです。

上段は、16px司法のpng画像を、約４倍に引き伸ばしたものです。拡大した影響で、全体的にぼやけた見た目となってしまっています。

下段は、svg画像です。どんなに引き伸ばしたとしても、このようにくっきりした状態を保ってくれます。

2. 場合によってはファイルサイズが小さくて済む

通常のpng画像などの規格で大きなサイズの画像を用意しようとした場合、基本的にファイルサイズが大きくなってしまいます。しかしsvg画像には基本的に解像度という概念がないため、大きくても小さくても同じファイルサイズです。

例えば、文化祭で使うパーカーを、業者に発注するとしましょう。印刷物ですから、高い解像度が必要です。ここでは、2560px四方のpng画像を用意しました。

png	193.7 KB
svg	45.5 KB
pdf	7.2 KB

ここでpngとsvgのがいるのサイズを比較してみます。その結果が左の表です。約158KBもの差が出来てしまいました。そして、いくら高解像度にしたとしても、png画像では緑のギザギザ、すなわちジャギーが生まれてしまいます。svgの圧勝です。

さて、もうお気づきになった方もいらっしゃると思いますが、表の一番下にpdfのファイルサイズが記載されています。実はpdfも、オブジェクトによって構成されるベクター形式なのです。

3. css.jsで自由に操作できる

ここからは、私自身まだ触れていない領域もあるので信用度は低いです。あまり過信せずに読んでください笑。ここまで読んでくださっている方がいるかわかりませんが…。

svgをhtmlに埋め込んで表示する際、いくつかの方法がありますが、**<img src="hoge.svg">**や**background-image: url("hoge.svg");**を使うと画像として読み込まれてしまい、pathに対してcss.jsが適用されません。

また、**<object data="hoge.svg">**だと、そのファイルに対してcss.jsを適用しないといけないので面倒です。

一番いいのはajax()で動的に追加することです。これなら同じファイルに追加されるので、css.jsも適用されます。同じファイルにあればいいということであれば、html内に直接記述しても動きます。面倒ならこれでもいいでしょう。

実際にどのように扱うかは、普通のhtmlの要素と同じです。ここで説明すると長くなってしまうので、ググるなりなんなりしてください。

終わりに

さて、ここまで読んでくださりありがとうございました。この文章は文化祭延期でモチベがだだ下がりのときに書いているので、最後はとてもぐだぐだになってしまいました。ごめんなさい…

ただひとつ言いたいのは、svgはとても便利だということだけです。私自身その便利さに惹かれて、最近は何かとsvgを多用しています笑。

これを読んでくださったみなさん、ぜひぜひ試していただければと思います！！

VPS のすゝめ

一家に一台 VPS[8]

H1 にこなのにふわわあ

0. 注意

この記事は中・上級者向けです.

1. VPS とは

Virtual Private Server を略して VPS,個人用のレンタルサーバーです.

Web ページを公開する目的のレンタルサーバーも存在しますが,それとは比べ物にならない便利さがあります.

それとの比較も含めて VPS の魅力を紹介していきます.

1.1. VPS の特徴

簡潔に述べると次のようなものが挙げられます.

- 自分専用のマシンとして使える
- 常時稼働させることができる

[8] 正しい VPS の数え方って何なんでしょうかね...?

詳しくはこの記事では触れません.

今回は筆者の VPS の使い方を書きつつ魅力を伝えていけたらと思います.

2. VPS の用途

2.1. ゲームのサーバー

某 3D サンドボックスゲームではユーザーが自由にサーバーを構築してマルチプレイをする

ことができます.

これが筆者が VPS を借り始めたきっかけで,

24 時間常に起動をしておくことができるほか.

自 PC への負荷の影響が小さく済み,ストレスになりません.

2.2. Web サーバー

いわゆるホームページを公開するサーバーです.

冒頭で触れたとおり無料の Web レンタルサーバーもありますが,

自分の好きなようなカスタマイズができたり,

サーバー側でプログラムを実行するようなものも使えたりします.

自分で Web アプリを開発するときには VPS があると大幅にハードルが下がって開発が捗ります.

次のような言葉があります.

質の低いものでもたくさんのプロジェクトを開発して公開していくことが自分の技術を磨くことに繋がります.

Done is better than perfect.

— Mark Zuckerberg

VPS は開発者にとても向いていると言えます.

2.3. 放置ゲー

見出しとして適切な語が見つからなかったのでこれにしましたが,ゲームではありません.

例えば,長時間を要する計算などの処理をサーバーに代わりに行わせることで,PC を起動しておく必要がなくなります.

ただ,下手に重い処理を投げると管理者から怒られるなどする *かもしれない* のでご注意ください.

2.4. リモートで作業

サーバー用途の環境であるためネットワークが結構高速です .(実測:上下 90Mbps 程度,ping 10ms 程度)

また,現在では Linux[9]に対応したソフトウェアも多く,

開発環境をサーバー上に構築することができます.

これによりどこからでも同じ環境を使用でき,

[9] サーバーにはよくこのOSが使われます。

携帯回線を使っていても大きなファイルをやり取りすることに苦痛を感じません.(サーバー利用者の多くはこれによる ***得した気分*** を経験したことがあるはず.)

3. おすすめの VPS

VPS を契約できる会社はいくつかありますが,おすすめ 3 つを紹介していきます.

3.1. ServersMan@VPS

価格 : 格安

スペック : まあまあ

運営 : ドリーム・トレイン・インターネット(DTI)

URL : https://dream.jp/vps/

一部機能制限がありますが初期利用では問題ありません.

価格面では文句のない安さなのでお試しにちょうどいい

3.2. さくらの VPS

価格 : それなり

スペック : 良い

運営 : さくらインターネット

URL : https://vps.sakura.ad.jp/

十分な機能と少し安めの価格設定,安定した運営により高く評価されています(偏見を含む

かもしれません)
2018/09 の北海道地震の時には石狩データセンターを自家発電によって停電を乗り切った

という話もあるので,

運営はとても信頼できます.(参考)

3.3. ConoHa VPS

価格 : まあまあ

スペック : 良い

運営 : GMO インターネット

URL : https://www.conoha.jp/vps/

手厚いサポートとかわいい公式キャラクターこのはちゃんにより人気は高い.

初心者が取っつきやすいようなサポート,Minecraft サーバーを構築するためのプランが用

意されているなど

ハードルは低いかと思います.

4. 締め

「一家に一台 VPS」は大げさすぎるかもしれませんが,それくらい VPS は便利で素晴らしい

ものです.

サーバーがほしいな,などと思ったら選択肢に入れてみてください.きっといいパートナーに

なると思います.

2 冪をおぼえよう

H1 にこなのにふわわあ

1. 2 冪ってな〜に?

数学における冪演算(べきえんざん、英: 独: 仏: Exponentiation)あるいは冪乗は、底 (base) および冪指数 (exponent) と呼ばれる二つの数に対して定まる数学的算法で、その結果は冪 (power) と呼ばれる。自然数 n を冪指数とする冪演算は累乗(るいじょう、英: repeated multiplication) に一致する。 — 冪乗 - Wikipedia

つまり 2 冪というのは 2 の自然数[10]乗のことをいいます.

実際の値は最後の方に表がありますが,それなりのスピードで値が大きくなっていきます.

2. なんで 2 冪なの?

小学校で自由帳にひたすら 2 冪を書いていませんでしたか?筆者はそういう人種でした.

2.1. コンピュータと 2 進数

コンピュータでは数が 2 進数で表されることがほとんどです.(そうでないものもあるので言い切っていません)

つまり,2 冪と非常に相性がいいのです.

コンピュータでは決められた領域内でデータを完結させる必要がありますが,

n 桁では 2^n 未満の自然数を扱うことができます.

[10] 自然数については派閥が分かれますがここではどちらでもそんなに困らないと思います.((

つまり,プログラムを組む上ではこの値を覚えておくととても便利なのです.

2.2. 私はプログラムを書かない

こういう読者もいるかと思います.
頑なに覚えるのを拒否するのは構いませんが,知識として頭の片隅に存在を置いておくことをおすすめします.

プログラム上で扱いやすい数はユーザーの目に入るところにも現れやすいことは自明でしょう.

たとえば,RAM/ROM(メモリ,SD,USB メモリなど)の容量が 2 冪であることです.
SD カードが調べやすいと思いますが,ラインナップは 4 ,8 ,16,32,64 ,...*GB* となっています.

3. 身近に潜む 2 冪

PC のストレージ(HDD/SSD)の容量がスペック情報と表示で異なっていることがあります.

これは,数の扱い方に違いがあります.

GB(Giga Bytes)という単位は,

SI 接頭辞の定義によると 10^9 Bytes を表します.

一方,コンピュータ内ではよく 2^{30} Bytes を表すことがあります.

この単位の扱い方によって相違が発生するのです.

3.1. GiB(ギビバイト)

単位の表記がどちらであるかわからなくなってしまったことを踏まえて ,新しい単位の表記法が作られました.

接頭辞の後に小文字 i を入れることで,情報系でやさしい 2 冪をもとにした単位であることを表しています.

これは他の接頭辞に関しても使われていて,次のようになっています.

i なし	係数	i あり	係数
K(キロ)	10^3	Ki(キビ)	2^{10}
M(メガ)	10^6	Mi(メビ)	2^{20}
G(ギガ)	10^9	Gi(ギビ)	2^{30}
T(テラ)	10^{12}	Ti(テビ)	2^{40}
(続きは省略)			

4. さあ,2 冪をおぼえよう!

表を載せました.無理はしないでね.

n	2^n	n	2^n
1	2	17	131,072
2	4	18	262,144
3	8	19	524,288
4	16	20	1,048,576
5	32	21	2,097,152
6	64	22	4,194,304
7	128	23	8,388,608
8	256	24	16,777,216
9	512	25	33,554,432
10	1,024	26	67,108,864
11	2,048	27	134,217,728
12	4,096	28	268,435,456
13	8,192	29	536,870,912
14	16,384	30	1,073,741,824
15	32,768	31	2,147,483,648
16	65,536	32	4,294,967,296

5. 覚えたところで何に役立つの

...いい質問ですね!((

ほとんどは趣味の領域なので役にはあまりたちません.

ただ,数学とかでたまーに使えると少し計算が速くなったりします.

筆者は受験算数で少し使うことができました.

ドラムパターンで遊ぼう！

H1 山内　康平

　初めまして、高校1年の山内と申します。最近は曲を書いたりしています。早速ですが皆さん「DTM」ってご存知でしょうか？「Desk Top Music」の略らしいです、簡単に言えばパソコンで曲を作ることですね。やってます？---やってないですか、そうですよね。
やりません？---難しそう？そうですよね、いろんな音が鳴っていて複雑ですからね。
ところでドラムってご存知です？---知ってる、そうですよね。

　ここまで決めつけで話を進めてきたわけですが、おそらく「曲を作るのは難しそう」と考える人が多いと思います。ですが、ドラムって曲のなかでもかなり重要なパートであるにも関わらず、音程が(ほぼ)存在しないので、リズムさえ決めれば終わりなんですよね。例えば曲の途中でいったんドラムを消したら静かなパートになりますし、だんだん早くなるように置いたらサビ前の盛り上がりとして使えますし。ぱっと見た感じパート自体が多いように思えますが、とりあえず2種類、「キック」と「スネア」さえあれば何とかなる気がしています。もちろんほかのパートが入ったほうが豪華にはなりますが。

　ということで、個人的な備忘録もかねていろんなドラムパターンをキックとスネアに限定して載せてみようと思います。これをちょっとだけいじったりするときっといい感じになりますし、ぜひここからいろいろ試して頂きたいなと思っています。

キック |Oxxx|Oxxx|Oxxx|Oxxx|（「|」でかこまれた四文字が1拍分を表し、四つ
スネア |xxxx|Oxxx|xxxx|Oxxx| でー小節です。Oが音符、xが休みを表します。）

　シンプル。4つ打ちです。拍が強調されるので曲のリズムがわかりやすい(気がします)。
ちなみに4つ打ち自体はキックが拍に合わせてなるものなので、スネアのパターンは実は関係ありませんし、別にこれじゃなくても大丈夫です、とにかく「キックが拍に合わせてなっている」ものであれば4つ打ちになるはずです。個人的にこのパターンは汎用性が高い代わりにあまり特徴的ではないイメージがあるので、そこまで好きではなかったりします。
ただ、割と何にでも合います。本当に使いやすいです。

キック |Oxxx|xxxx|xxOx|xxxx|
スネア |xxxx|Oxxx|xxxx|Oxxx|

ドッタン、ドタンというリズムですね、ドラムンベースと呼ばれるジャンルではよく使われてます。疾走感があるというたとえ方をされることが多いですね。これを少しアレンジしようとするなら、スネアは触らず、キックをちょっと増やすことをお勧めします。スネアを触ると大体リズムがぐちゃぐちゃになります。ちなみにこのパターンの2つ目のキックのように少し後ろにずらすことでもかなりちょっとしたことでも印象が変わります。

キック |Oxxx|OxxO|xOxx|Oxxx|
スネア |xxOx|xxOx|xxOx|xxOx|

一気に複雑になりました、ドンダンドンダドンドダンドンダンです。読みにくい。かなり速いですね。キックとスネアが交互になる部分が特徴的なパターンで、その部分で体感の拍が少しずれて感じるのではないでしょうか。これを使うとちょっとノリがよくなるイメージがあるので割と使ってたりします。ただしメロディーが複雑だとリズムが混ざってしまうのでこれに限らずドラムが複雑な時はメロディーがシンプルだといいと思います。

ほかにもまだまだパターンはありますが、とりあえずこの3つを紹介させていただきました。これらをよく見るとわかるのですが、個人的にドラムパターンをつくるときのコツとして「スネアのリズムは一定にする」というのが一つあると思っています。もちろんそうでないものもありますが、スネアのパターンが決まっているとリズムがわかりにくくなってしまうことが少なくなると思います。そのかわりキックに関してはかなり複雑にしても問題はないと思います。少し参考にしていただければなと思います。

ということですが、どうですか？やってみましたか？もしちょっと楽しいと思っていただけたなら、ぜひハイハットも入れていただきたいのですが、そこまで書くと長くなってしまうのでそこはほかの方々に任せようと思います。きっと調べれば出てきます。このようにドラムだけならかなりとっつきやすいですし、今ならパソコンやスマホでポチポチするだけでリズムは出来上がるのでぜひまだやっていない方は一回試して頂きたいです。これを機会にいろんな音楽に興味を持っていただけたらいいなと思います。

ぜひ興味を持ったならDTM、やってみましょう。楽しいですよ。というところで終わりにしようと思います、ここまで読んでいただきありがとうございました！

圧縮ファイルと解凍について

H1 山田寛人

１．圧縮ファイルとは

皆さんは圧縮ファイルについてご存知ですか？
インターネットからダウンロードする場合によく見かけるので知っている方も多いと思いますが、ファイルを一定の法則(圧縮アルゴリズム)に従って書き換えることで、中の情報を維持したまま、サイズ(データ量)を縮小したファイルのことです。
それに対して解凍とは、その縮小したファイルを元のファイルに戻す作業のことを言います。
音声や画像などを圧縮しているため.mp3や.jpgも圧縮ファイルと呼称する場合もあるのですが、本記事では元のデータを完全に復元できる可逆圧縮ファイル(.zipファイルなど)について触れていきます。

2. 圧縮ファイルの仕組み

この項では、圧縮ファイルの仕組みについて書きたいと思います。
基本的には、入力データの統計的冗長性（出現する符号の偏り、規則性）を利用して、情報を失うことなくより稠密なデータに変換する。このような種類の方法は連長圧縮（RLE）と呼ばれる。ハフマン符号という仕組みを例えに出すと、ABCDEのアルファベットで構成された「D A E B C B A C B B B C」のデータを二進数に置き換えるときに本来はA=000,B=001,C=010,D=011,E=100として置き換えていたとする。そうした場合はデータとしては「011 000 100 001 010 001 000 010 001 001 001 010」となり、36桁(ビット)になるが、ABCDEのなかで登場回数の多い順に桁数を短く置き換えていくことにする。例としてはB=0,C=10,A=110,D=1110,E=1111と置き換える。
そうした場合のデータは「1110 110 1111 0 10 0 110 10 0 0 0 10」となり、25桁(ビット)となる。
他にも連続した符号の列に対して符号を割り当てる方法（拡大情報源）といった方法もある。

こちらの例はaaaaaabbccccccdddをa6b2c5d3と書き換えるようなものである。
これらの方法は圧縮率や圧縮に使うメモリや時間が異なるので、使い分けや併用が必要になってくる。

3.圧縮ファイルの利点と問題点

圧縮ファイルの利点は単純明快で、
1.ファイルの保存に必要な記憶容量の削減できる
2.インターネットでファイルをダウンロードする場合などにより高速でダウンロードできる
で、一部のファイル圧縮方式では
3.複数のファイルを1つにまとめて扱えるようにするアーカイブ機能
も備えています。
しかし問題点もあり、ファイルが圧縮されるとそのままではもともとの利用ができなくなります。
なので、ファイルを利用しようとすると、いったん解凍作業をしなければいけません。

4.圧縮ファイルの主な利用方法

前述の利点を踏まえて、どういった場合にファイル圧縮が用いられるかというと、例えば膨大なデータ量のもののバックアップをとる場合や、インターネットにファイルをアップロードしたりダウンロードしたりする場合、ファイルの内部構造が複雑だが、まとめて保管したい場合、などによく使われます。

5.まとめ

圧縮ファイルはもともとのファイルを内部の情報を維持したまま縮小したもので、それをもとのファイルに戻すことを解凍と呼ぶ。
基本的にファイルの中のデータの偏りや規則性をもとに中身を圧縮する。
圧縮の方法にも種類がある。
ファイルを圧縮すると記憶容量を削減できる。

解凍しないと元のように使えない。
バックアップを取るときや、ファイルをアップロードするときに利用すると効果的。
この記事を読んで少しでも役に立ったら幸いです。

一部引用
https://ja.wikipedia.org/wiki/%E3%83%8F%E3%83%95%E3%83%9E%E3%83%B3%E7%AC%A6%E5%8F%B7
https://ja.wikipedia.org/wiki/%E3%83%87%E3%83%BC%E3%82%BF%E5%9C%A7%E7%B8%AE

音 MAD を作ろう

M3 水神

音 MAD って何?

皆さまは「音 MAD」という物をご存知でしょうか。定義が曖昧すぎて 僕には説明できないので、知らない方は「音 MAD」で検索して実物を見て下さい。最初は何が面白いのかさっぱり分からなくても慣れれば楽しいと思います。

音 MAD を作ろう

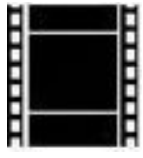
音 MAD ってパソコンがあれば誰でも簡単に(簡単に?)作れるんです。 この記事ではどんなソフトを使ってどんな風に音 MAD を作っている かを簡単に紹介します。細かい事は書ききれないので、本気で作る気に なったら「音 MAD 作り方」で検索して下さい。

使用するソフトと簡単な説明

1.Reaper

たくさんの音を取り込んでミックスできて、ピッチの編集など簡単な音声編集もできるソフトです。 原曲と素材をミックスして使います。

2.AviUtl



動画編集ソフト。やたら重いけど何でもできる。映像部分 を作るのに使います。

3.Vocalshifter

波形音声編集ソフト。取り込んだ音のピッチ(音の高さ)や　ボリュームを曲線で表示してくれて、さらにそれを編集で きるソフトです。音程を合わせるのに使います。

4.Wavetone

取り込んだ曲を解析できるソフト。原曲の音程やリズム、 BPM(音の速さ)などを調べるのに使います。

この四つがあればかなりクオリティの高い音 MAD でも作れるように なります。

音 MAD の作り方

1 素材集め

原曲とそれに合わせる音声と映像(BB 素材とか)を集めます。

2 リズム合わせ

Reaper でカンと音量の波形表示、wavetone の解析結果等をもとに音声素材を原曲に合わせて音声を配置します。

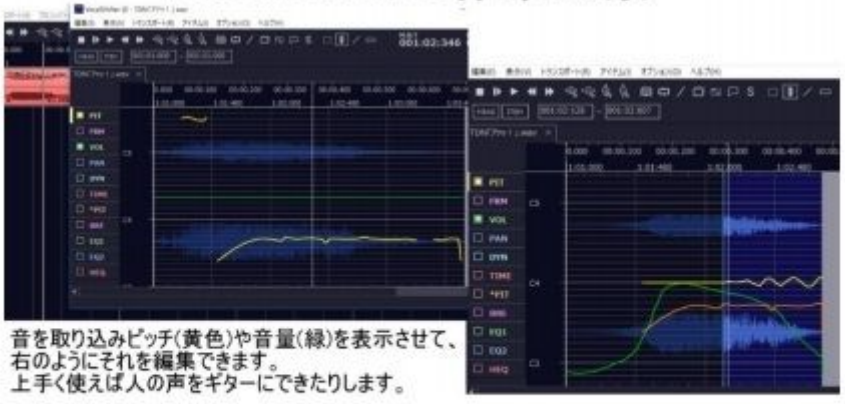
3 音程合わせ

リズム合わせでできた音の集まりをvocalshifterに入れてwavetoneの解析を参考にし音程を原曲に合わせます。先に素材を特定の音に合わ せてreaperで音を合わせる方法もあります。

参考画像　reaperの作業風景

ちなみにこの画像はaviutlで作っています。
wordよりもaviutlの方が使いやすいと思うんです。

vocalshifterの作業風景



音を取り込みピッチ(黄色)や音量(緑)を表示させて、
右のようにそれを編集できます。
上手く使えば人の声をギターにできたりします。

4 映像制作

基本aviutlを使います。何を作りたいかによって作業の内容が変わるので特に説明はしません。

あとがき

I LOVE 音 MAD

かなりニッチな記事だったかなと思います。ここまで読んでいただきありがとうございました。

進歩したVRのあれこれ

M3 由良　大樹

名前にもある通り、VRの進歩の理由について中３レベルの客観的な目から述べさせてもらう。正直部誌という場で客観的に感想を述べているのは僕だけだと思う。

　前置きはさておき、まずはVRから。

VR（バーチャルリアリティー）とは、仮想的な世界を作り上げ、人を刺激させる仮想現実のことを指す。この技術は前からあったのだが、ここ近年で様々な企業がこれに手を伸ばしている。中でも、有名ゲーム機のほとんどがこれを実装していて、専用VRゴーグルを売り出してしまうほどだ。売り出したらすぐに売り切れ、何倍にもなって転売されたことだってあった。ほかにも一昔前に流行ったバーチャルユーチューバー（通称Vtuber）の爆発的な流行もVR技術が流行った一因だろう。（余談だが、僕はVtuberという技術が好きだ。仮想的なものをいかにも現実かのように振る舞う姿はまさにバーチャル文化の発展といえる。）

いかがだっただろうか。本当に個人の意見を述べただけだったが、共感してくれたり、これを読んで満足してもらえたのだったら幸いだ。

はじめてのプログラミング

M2 山城

1 はじめに

タイトルにもある通り、僕はゲームプログラミングに関しては始めてあまり長くありません。（プログラミング自体は１年１ヶ月、ゲームプログラミングを始めたのは半年前です。）

そのため、自分でもよくわかっていない部分もたくさんありますが、それでもミニゲームを一つ作ることはできました。

１年ぐらい（多分向上心のある人なら僕よりもできる）プログラミングをしていれば、プログラミングはそれなりにできるようになります。（特に麻布に入ろうと思っていてプログラミングに興味がある人は是非パーソナルコンピューター同好会へ！）

2 C言語

プログラミングをするときに、まず言語というものがあります。

その中で、僕はC＋＋という言語を使っています。（これを使ってソースコードというものを書きます）

言語とはなんなのかと思う人もいるかもしれませんが、そんなものがあるというくらいの認識でも大丈夫です。

3 DirectX

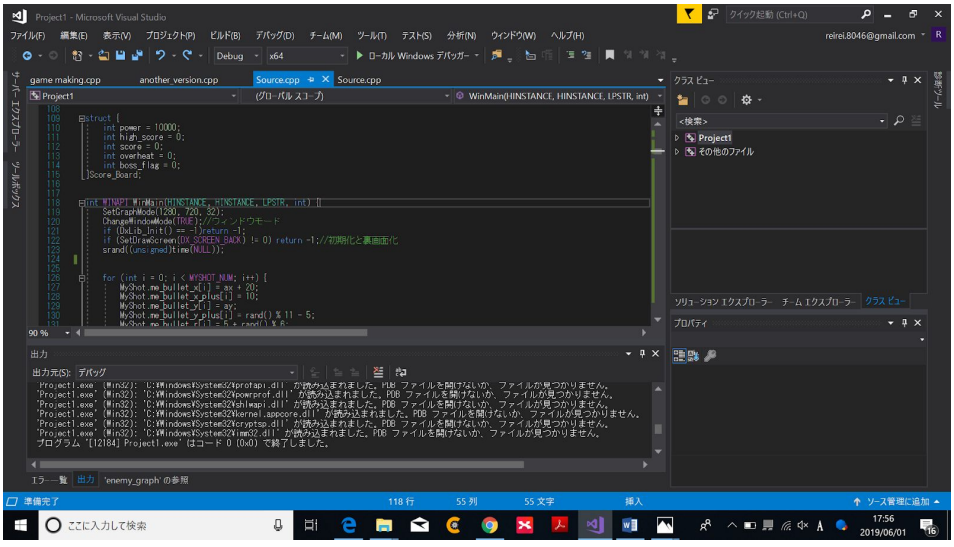
ゲームプログラミングをする上で、DirectXというものを使っています。これを説明していくと非常に長くなるのですが、簡単に言うと画面の描画、入力受付など、ゲームの制作をするにあたって便利な機能が多いものです。

4　超簡単にゲームの裏側解説

　ソースコードというもの（この中に「このボタンを押したらこんな動きをする」ということを書いたりします。）を入力し、そのソースコードの言語に対応したコンパイラ（ソースコードを機械語に翻訳するものです）に翻訳させ、機械語に翻訳されたコードを機械が実行します。

　ソースコードはいろいろな言語（たとえばさっき挙げたC＋＋）で書くことができます。

たとえば、テトリス（のようなもの）をつくることもできたりします（下の画像はソースコード）



5　いざプログラミング

　DirectXを使ってプログラミングをするときに、毎フレームごと（＝６０分の１秒ごと）に操作を反映するために、ソースコードの中に特定の状態（ゲームオーバーのときなど）以外のときに指定した範囲内を繰り返させるもの（この場合Whileというものを使っています）があります。この範囲内に

「このボタンを押したら上に動く」

「このボタンを押したら弾を発射する」

というようなことを書いていくと、操作をしたときにうまいこと動くようになるのです。

これ以外にも、自分の撃つ弾の情報や敵の大きさの情報を書くなど、いろいろなことをした結果としてゲームが出来上がるのです。

6 さいごに

いろいろと書いてきましたが、僕はプログラミングに関してそこまで詳しいというわけでもありませんが、プログラミングは楽しいということは自信を持って言えます。

上にかいたようなことをするとなると、時間もかかり、かかる苦労もひとしおですが、それを上回る楽しさがあります。

この記事をきっかけとしてプログラミングに興味を持つひとがいればいいなと思っています。

キースイッチのお話

M2 midri

そもそもキースイッチとは

はい、皆さん一度はキーボードというものを見たことがあると思います、そして使った事もある方がほとんどでしょう、キーボードって文字が書いてあるとこを押して操作しますよね[11]、そして、押されると電気的な信号を流す部分をキースイッチと言います。
端的にキースイッチと言っても様々な種類があります。なので今回はそのあたりの話をしていきたいと思います。

キースイッチの種類

メンブレン式

お値段が控えめなキーボードでよく使われている。一般的なキーボードの主流。
ラバードームをキートップで（普段叩いてる文字が書いてあるところ）押し込んでキーボード全体に張り巡らされてる上下のシートの電極同士が触れ合って入力する。
安い代わりに軽快さは無い物がほとんど、メンブレン式でも中にはメカニカル式の様な感触があるものもあるが大抵いいお値段がする。

パンダグラフ式

メンブレン式にパンダグラフの支持構造を追加してキートップ全体からムラなく力が伝わるようにしたもの。
ノートパソコンのキーボードでよく使われている、比較的低コストで生産できる。

静電容量無接点方式

名前がなんかもう堅苦しく感じるかもしれませんが、まあお気になさらずに。
まずこの先のメカニカル式とこの静電容量無接点方式のキーボードはそれぞれのキー1つ1つが独立したスイッチを持ちます。何言ってるのか分からないかも知れませんがメンブレン式みたいにキーボード全体にシートが張り巡らされてるのではなくそれぞれのキーにスイッチが付いてます、そのスイッチを押し込んで入力するといった感じです。中でも静電容量無接点方式はラバードームの中に円錐型のスプリングが入っていて心地よいうち心地になっています。更に静電容量無接点方式は機械的な接点がなく静電気で入力を検知するので耐久性が抜群に高くなっています。静電容量無接点方式のキーボードとしてHHKBやREALFORCEなどが有名です。これらは

[11] ~~ソフトウェアキーボードとか言わないでください筆者が倒れます~~

耐久性も高く軽快なうち心地で、上品な打鍵音がするもののかなり高価です。

メカニカル式

静電容量無接点方式と同じように一つ一つのキーが独立したスイッチを持っています。ですがメカニカル式は機械的な接点を持っています。そして最大の特徴として様々なスイッチの種類があり荷重、打鍵感、キーストロークなどが違います。

（荷重･･･キーを押し込むのに必要な力、
打鍵感･･･クリッキー→カチッと音がする、接点の部分で感触がある
タクタイル→音はしないが接点の部分でコクッという感触がある
リニア→接点で音も感触もなく一定の打鍵感
キーストローク･･･キートップを押し込める最大の幅の長さ）

そして、それぞれの軸は色で区別されています、ここでは有名ないくつかの軸を紹介します。（下の表や画像はcherry社のものです）

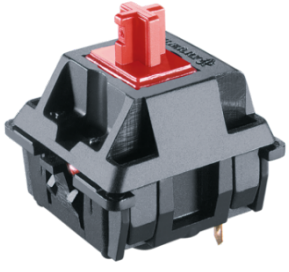
軸（cherry MX）	荷重	打鍵感	キーストローク
青軸	60cN	クリッキー	4mm
赤軸	45cN	リニア	4mm
茶軸	55cN	タクタイル	4mm
静音赤軸（ピンク軸）	45cN	リニア	3.7mm^2
黒軸	60cN	リニア	4mm

(1cNはおよそ1g分の力です)

Cherry社以外のメーカーからもメカニカルスイッチは販売されていますが既製品のメカニカルキーボードではcherry社のスイッチを使用した物が一般的です。

ちなみに静電容量無接点方式のものほどではありませんがcherry社のキースイッチを使用したメカニカルキーボードは高価なものが多いです。

最近だとメカニカル軸なのに接点部分が静電容量無接点方式と似た構造になっていて耐久性も向上しているものまであります。



（左が赤軸、右が青軸）

自作キーボードのお話

本題とはずれますが少しだけ自作キーボードの話をします、付録みたいなものだと思っておいて下さい。

デザインは好きだけど使用している軸が気に入らない、好みに合ったデザ

インのものがない、といったときに「ほならね、自分が作ってみろって話でしょ？そう私はそう言いたいですけどね。」といって自作キーボードというものをする怖い方々（褒め言葉）がいます。
自作キーボードと言ってもキースイッチを作ったりするのではなくPCBと言われる基板やケース、キーキャップ、キースイッチ、プレートなどを組み立てて作ります（中にはそれぞれのパーツも作ってしまう人までいますが、はい、やばいですね）。自作キーボードだとcherry社以外のメカニカル軸が使われることも多いです。
自作キーボードの画像については調べてみてください（個人で制作されているものがほとんどなのでここにのせずらいのです）。

まとめ

必ずしも静電容量無接点方式やメカニカル式のキーボードが良いとは限りません、人それぞれ好みの打鍵感なども違うので要するに自分が一番いいと思ったものを使えればいいですねって話です。まあお値段的にどうしようもないこともあるかもしれませんが。

最後までお読みいただきありがとうございました。

画像出典　https://www.cherrymx.de/en/mx-original/mx-red.html
https://www.cherrymx.de/en/mx-original/mx-blue.html

少しだけでもパソコンを使ってみよう

M2 碓井一成

はじめに

僕は自分のパソコンを持って、初めて触った機能がExcelやWordでした。
どんなものかというとExcelは自分で表を作るもので、Wordは文をまとめるものです。
なんだ、それだけかと思う人が多いと思いますが、僕も初めはそれしか触っていませんでした。具体的にどんなことができるか書いていきたいと思います。

Excelについて

Excelは表づくりだけでなく、代入計算もできます。

1

6	9	=

このように＝を入力します。

2　計算したい表のマス目をクリックで選択します。

3

6	9	15

間に＋などを入れることで計算できます。
（今回は６のマスと９のマスの間に＋をつけます。）

4

7	9	16

この状態で選択したマスの数字を変えると計算結果のマスの数字がかわります。

このような手順を踏むことで打ち間違えて計算を進めていてもすぐに直すこともできます。

16				15	←□□□	□□□□
19				19		
37				37		
1295	←(15+19)*37□□□		→→→	1258		

さらに文字やマス目に色を付けることもできます。

□□□□	□□□	□□	□□	□□

こんな感じです。

Wordについて

WordはExcelのような計算機能などはありませんが、その分文字の強調などができます。
このように文字を斜めにできたり
文字を太くできたり　***色々な組み合わせができます***
その中でも特徴的なのは光彩的な字です。
こんな感じで他よりも強調できます。

タイトルなどに使えそうですね。
こんな風にして多彩な文章を作ってみてください。

終わりに

いかがでしたか？
これらはプログラミングには関係ありませんが、このようなものでもパソコンに興味を持ってくれると嬉しいです。皆さんも触れる機会があったら是非やってみてください。

おまけ

WordにExcelを導入する方法です。
1　Excelで導入したい表の部分を選択する。
2　右クリックでコピーする。
3　Wordにとび、右クリックで貼り付けを選ぶ。
こんな感じです。