Turn it Over
Nada School Festival 2022

2022
05/02・03

部誌 2022
サンプル

灘校パソコン研究部
npca

あいさつ

皆さんこんにちは、灘校パソコン研究部 (NPCA) 部長の Sug です。本日は第 76 回灘校文化祭にご来場いただき誠にありがとうございます。

灘校パソコン研究部では、競技プログラミング[*1]や機械学習、Unity[*2]などによるゲーム作成、CTF[*3]やセキュリティ、サーバ構築など情報系の様々な分野に日々取り組んでいます。

新型コロナウイルスの影響がありつつも去年に比べて規模が拡大され、より一層パワーアップされた企画と部誌をどうぞお楽しみください。

この冊子について

この冊子は、部誌の本編から内容を一部抜粋したサンプルです。

部誌の本編（完全版）は灘校パソコン研究部の公式サイト (`https://www.npca.jp/2021/`) で公開しています。少しでも興味を持っていただけましたら、ぜひご覧ください。スマートフォン等からのアクセスは裏表紙の QR コードが便利です。

執筆者の表記について

各記事にはタイトルと執筆者、そして執筆者の学年が表記されています。

執筆者名として、本名ではなくハンドルネームが記載されています。これは、部内では本名を呼ぶことが滅多になく、ほとんどの場合部内ではハンドルネームで互いを呼び合うという慣習によるものです。

また、灘校の文化に従い、学年ではなく何回生かが記されています。2022 年 5 月現在、各回生は次の学年に相当します。

- 75 回生：高校 3 年生
- 76 回生：高校 2 年生
- 77 回生：高校 1 年生
- 78 回生：中学 3 年生
- 79 回生：中学 2 年生

[*1] 与えられた課題を解決するプログラムをいかに素早く正確に記述するかを競うプログラミングコンテストの総称

[*2] ゲームを作るためのアプリケーション (ゲームエンジン) の一種

[*3] Capture The Flag（旗取りゲーム）の略
情報セキュリティの技術を競う競技

謝辞

部誌の円滑な執筆及び Web 版（HTML 版）の制作のために、組版システム Re:VIEW[*4]を 5 年前から利用しています。開発者の皆様、ありがとうございました。

[*4] https://github.com/kmuto/review

目次

あいさつ 2

第 1 章　応用情報技術者試験対戦記 5

第 2 章　ゼロから作る量子コンピューター 7

第 3 章　ニキシー管で適当に遊んでいく 12

第 4 章　i3 window manager を使ってみよう 16

第 5 章　TOTP の生成と Base32 の変換 in C# 21

第 6 章　万年カレンダーを実装する 24

第 7 章　初めての web アプリ作成 26

第 8 章　レシート記述言語で遊ぶ 31

第 1 章

応用情報技術者試験対戦記

77 回生 Sug

1.1　はじめに

こんにちは！ 77 回生のすぐです。普段はこの部の部長をしています。入学したと思ったらいつの間にか中学を卒業しており時間の流れに驚いています。留年しないように頑張ります。さて、文化祭が始まるちょっと前の 4/17 に応用情報技術者試験を受けたので、それに関するいろいろを書いていこうと思います。

1.2　応用情報技術者試験とは

応用情報技術者試験とは、情報処理推進機構が実施している国家試験の一つで、毎年春期と秋期に実施されています。応用情報技術者試験ではネットワーク、セキュリティ、システム監査、経営戦略など様々な内容が出題されていて、基本情報技術者試験の上位として位置づけられています。情報処理推進機構が実施している国家試験には、ほかにネットワークスペシャリスト試験やシステムアーキテクト試験などがあります。

1.3　試験前

僕は大体 1 か月前から勉強を始めました。昔買ったまま積んでいた試験のための本を読み始め、2 週間前までにテクノロジ系（ネットワーク、セキュリティなど）を読み終わりました。その後過去問道場というサイトでひたすら過去問を解きました。マネジメント系とストラテジ系は問題文の日本語を読めば解ける問題も多く、また本を読んでもあまり情報がなかったため、過去問を解いて日本語の勘をつけていきました。結局本番前日にようやく 60% に乗りました。

1.4　試験本番

午前

情報処理推進機構の試験を受ける人は朝起きられない人が多いらしく、受験時間に間に合わない人も数多くいるようですが、なんとか間に合いました。午前の試験を受けて、途中で消しゴムがないことに気づきシャーペンの蓋についてるので対処しました。それ以外は割と順調で、1 時間半くらいで試験を終え退出しました。午後の試験までは 2 時間弱あったので近くをうろつきながら午前の解答速報を見て採点しました。午前の試験でも各サイトでの解答は割と異なってお

り、一応何個かのサイトで答え合わせをすると 60% ギリギリだったので落ちてるか受かってるかはわかりませんがとりあえず気にせず午後の試験を受けに行きました。消しゴムは結局買い忘れました。

午後

午後の試験ではセキュリティ、プログラミング、ネットワーク、組込みシステム、システム監査を選びました。組込みシステムとシステム監査は問題文の日本語を読んで答えるという割と簡単めの問題だったのですんなり解けました。またプログラミングは数独を解く問題で、まさしく競プロでやるような問題でした。記述の文字数がかなり少なく困ってしまいましたが割と順調に解けました。セキュリティは必須なので逃げられません。頑張って記号が合うようにお祈りをしました。最後にネットワークですが、1 時間半も日本語を読み続けたため日本語を読むのに疲れており、またデータベースは図がごちゃごちゃでやりたくなかったので消去法みたいな感じで選びました。図から読み取る問題はそれっぽく解いてあとはお祈りです。今思うと午後の問題初見で本番に向かったのは戦略が立てられないという点ではよくなかったなと思います。試験が終わって家に帰った後、公式の午前の解答が出ていたため答え合わせをすると 51/80 で通っていました。あとは午後ですが、文化祭現在午後の答えも試験結果も出ていません。ドキドキしています。

1.5 最後に

Web 版もぜひご覧ください。また、宣伝ですがパソコン研究部では下の画像のように Unity で鬼ごっこのゲーム AI を動かしている（はず）ので、興味のある方はそちらの方もぜひ見てみてください。

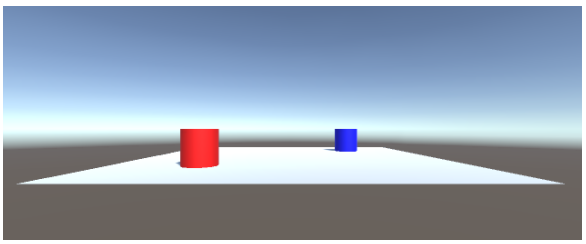

図 1.1

第 2 章

ゼロから作る量子コンピューター

76 回生 Chito

2.1　はじめに

76 回の Chito です。いつの間にか高 2 になってしまい焦っています。普段は、Python や C++ を使ったり、Linux で遊んだりしています。最近、量子コンピューターについて興味を持ち始めたので、今回は、量子コンピューターの計算の仕組みを Python3 でシミュレートしてみながら量子コンピューターの計算方法を自分の分かる範囲で紹介してみようと思います。

2.2　量子コンピューターとは

まず、量子とは「粒子と波の性質をあわせ持った、とても小さな、物質やエネルギーの単位のこと」[*1] で量子力学の法則に従っています。そして、量子の物理状態を利用したコンピューターを「量子コンピューター」といいます。

2.3　コンピューターの種類

量子コンピューターに対して、普段私たちが使っているコンピューターを「古典コンピューター」といいます。さらに、任意の量子状態から任意の量子状態へと十分に高い精度で変換できるかとうかという「万能性 (エラー耐性)」、計算能力が古典コンピューターよりも上回っているかという「量子の優位性」、量子特有の物理状態を使用して計算を行うかどうかという「量子特有の物理状態」の 3 つの特徴の有無によって次のように分類できます。以降、特に注釈のない限り量子コンピューターは「万能量子コンピューター」を指すものとします。

種類		万能性 (エラー耐性)	量子の優位性	量子特有の物理状態
量子コンピューター	万能量子コンピューター	○	×	○
	非万能量子コンピューター	×	○	○
	非古典コンピューター	×	×	○
古典コンピューター	古典コンピューター	×	×	×

[*1] (1)量子ってなあに？：文部科学省： https://www.mext.go.jp/a_menu/shinkou/ryoushi/detail/1316005.htm

2.4　量子ビット

量子ビットと古典ビット

まず、古典コンピューターと量子コンピューターの大きな違いは、コンピューターで扱う最小単位の「ビット」の仕組みにあります。

古典コンピューターで扱うビット (いわゆるビット) は 1 ビットにつき「0」又は「1」のどちらかの状態しか持つことができないが、量子コンピューターで扱うビット (これを qbit という) は 1 ビットに「0」と「1」の両方の状態を確率的に「重ね合わせ」て持つことができ、量子状態を「測定」した時に初めて「0」か「1」かが決まります。

この「重ね合わせの」の性質によって、量子コンピューターは古典コンピューターと異なり一度に複数の数の状態を保持できるので、桁違いに高速な計算が可能となることもあります。

ただ、数値が確率的に存在するという量子コンピューター特有の性質により、得られる結果もまた、確定的なものではなく確率的なものとなってしまいます。

2.5　量子ビットの表し方

量子の状態は私たちには見ることができませんが、目に見えない量子の状態を視覚的、数学的に分かりやすくした表記方法があり、代表的なものに「ブロッホ球」と「ブラケット記法」というものがあります。では、それぞれ順番に紹介していきます。

ブロッホ球

ブロッホ球とは、1 量子ビットを視覚的にわかりやすく図示したものです。球面上の矢印のさす場所が量子ビットの状態を表し、矢印が真上を指すとき「0」、真下を指すとき「1」を表します。ただ、複数量子ビットになるとその分ブロッホ球をたくさん描かないといけないのでかえって見づらくなります。

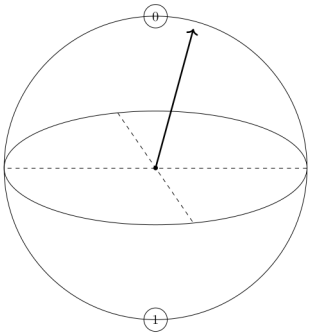

図 2.1

ブラケット記法

ブロッホ球の他に、量子ビットを表す際に「ブラケット記法」がよく使われます。ブラケット記法とは、量子の状態をベクトルを用いて表記する記法です。こうすることで、ベクトル同士の演算を見やすく表現することができ、ブロッホ球とは異なり複数ビットの場合でも分かりやすく表記することができます。また、1 量子ビットの重ね合わせの状態 $|\psi\rangle$ をブラケット記法で表記すると下の様になります。

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\ (|\alpha|^2 + |\beta|^2 = 1, \alpha, \beta \in \mathbb{C})$$

α、β を複素確率振幅と言い、$|\alpha|^2$、$|\beta|^2$ はそれぞれ測定した時に、「0」を示す確率、「1」を示す確率を表しています。

多量子ビット

多量子ビットの場合も基本的には 1 量子ビットの場合と同じです。例えば、3 量子ビットで 1、2、3 番目の量子ビットがそれぞれ $|0\rangle$、$|1\rangle$、$|0\rangle$ と確定している時、$|0\rangle|1\rangle|0\rangle$ と表すことができます。また、これを $|010\rangle$ と省略することがよくあります。一般に n ビットの量子ビットをブラケット記法で表すと以下のように表すことができます。

$$\sum_{k=1}^{2^n} \alpha_k\,|m_{1_k} m_{2_k} ... m_{n_k}\rangle$$

$$(\text{ただし、}\alpha_1, ..., \alpha_{2^n} \text{は} \sum_{k=1}^{2^n} |\alpha_k|^2 = 1 \text{ を満たす複素数で } m_{1_k}, m_{2_k}, ...m_{n_k} \in 0, 1)$$

2.6　量子ビットを Python で

二度目ですが、1 量子ビット $|\psi\rangle$ は次のように表せます。

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\ (|\alpha|^2 + |\beta|^2 = 1,\ \alpha, \beta \in \mathbb{C})$$

従って、$(|\alpha|^2 + |\beta|^2 = 1,\ \alpha, \beta \in \mathbb{C})$ を満たす α、β が見つかると 1 量子ビットが表現できそうです。適当に α、β を決めてこの α、β の組が $|\alpha|^2 + |\beta|^2 = 1$ を満たすようになるまで、繰り返すという方法もありますが、これでは時間がいくらあっても足りないので、論理的にプログラムを組んでいきます。「論理的に何かを作るときは逆の流れを考えるといい」みたいなことがよくあるので、それに倣って逆の流れを考えます。

まず、元の流れは、「α、β を決める」⇨ 「$|\alpha|^2 + |\beta|^2$ を求める」⇨「この結果が 1 だと嬉しい」なので、逆を辿ると、「$|\alpha|^2 + |\beta|^2 = 1$ だと嬉しい」⇨「$|\alpha|^2 + |\beta|^2 = 1$ となる $|\alpha|^2$、$|\beta|^2$ の

値を求める」⇨「$|\alpha|$、$|\beta|$ の値を求める」⇨「α、β の値を求める」この順でプログラムを組むと分かりやすそうです。

List 2.1: 1 量子ビットの確率複素振幅の生成

```
import numpy as np

# 正の最大値
POSITIVE_INFINITY = 2147483647

# 足して1になる0以上の実数の組み
def occupancy():
    samples = np.random.randint(0,POSITIVE_INFINITY)
    return samples/np.sum(samples)

# 絶対値(の組)がcomplex_absとなる複素数(の組)をランダムで一つ返す
def fixed_abs_random_complex(complex_abs):
    rotate_angle = np.random.random(2) * 2 * np.pi
    return (complex_abs + 0j) *\
     (np.cos(rotate_angle) + 1j * np.sin(rotate_angle))
```

さて、これで $|\alpha|^2 + |\beta|^2 = 1$ を満たす複素数の組を作ることができ、1 量子ビットを扱うことができましたが、私たちはその量子ビットが $|0\rangle$ を示すのか、$|1\rangle$ を示すのか観測してみるまでは分かりません。そこで、量子ビットの示す値を観測するための observe 関数を作ります。

関数の詳細は省略しますが、今回の作った量子ビットを 10000 回観測してみると下のグラフのような結果になり、0 が約 0.4 の確率、1 が約 0.6 の確率で観測されるような量子ビットだったと予測できます。

今回の場合は、量子ビットが作られた時点で 0 と 1 の出る確率が決まっているので、他の量子ビットを作ったならば、また別の結果が出るかもしれません。

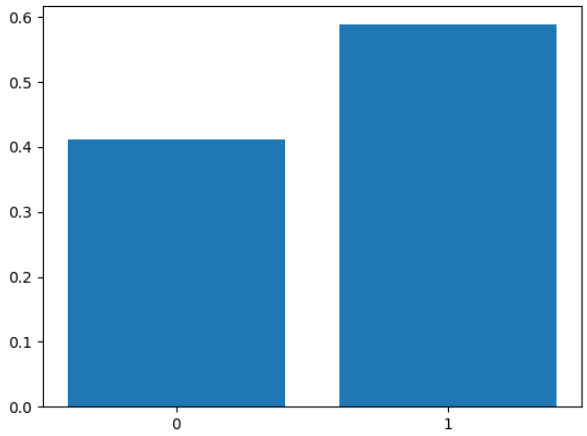

図 2.2

では、次に複数量子ビットですが、紙面の都合によりこれ以降は本編で詳しく話します。

2.7 最後に

かなり数学寄りの記事になってしまいましたが、最後まで読んでいただきありがとうございます。結局量子コンピューターがどのような働きをするのか、どう扱えば良いのかは分からずじまいだったかもしれませんが、これを機会に量子コンピューターやプログラミング、数学にも興味を持つ方が一人でも現れると嬉しいです。

2.8 参考文献

- 「(1) 量子ってなあに？：文部科学省」
  $https://www.mext.go.jp/a\_menu/shinkou/ryoushi/detail/1316005.htm$

第 3 章

ニキシー管で適当に遊んでいく

75 回生 DualXeon

3.1　はじめに

昨年度部誌をサボった DualXeon です。カスです。ごめん…

5 年前に入学したと思ったらもう灘校生活も終わりです。普段は NPCA にてシステム管理者をやっていました。NPCA の部室には恐らく他校のパソコン研究部にはないであろう業務用のネットワーク機器やラックサーバーがあるのでインフラ好きの自分にはいい経験になりました。~~まぁ導入して部室を狭くしたの私なんですけどね。~~高 3 にもなると部活にほとんど関われないので少しばかり寂しさを感じています。

ところで前々から在学中に一度は部誌を書いてみたいと思っていたのですが、もう最終学年になってしまいました。しかも特に書くようなネタがありません……。そこで今回は僕が昨年執筆しようと思っていた (が、断念した) ネタについて書いてこうと思います。~~使い回し~~

3.2　ニキシー管ってなに

概要

1950 年〜60 年あたりに主流だった表示器です。形は真空管に結構似ていますが中身は真空ではありません。構造としてはネオンランプに近いです。約 170V 以上の電圧をかけると電極の周りのガスがオレンジ色に光って、電極の形に応じて数字や文字を表示します。図 3.1, 図 3.2 のように 0〜9 の数字の形の電極が封入されているものが多く、数値のデジタル表示に使われていました。その独特の雰囲気が魅力です。

図 3.1: 内部構造

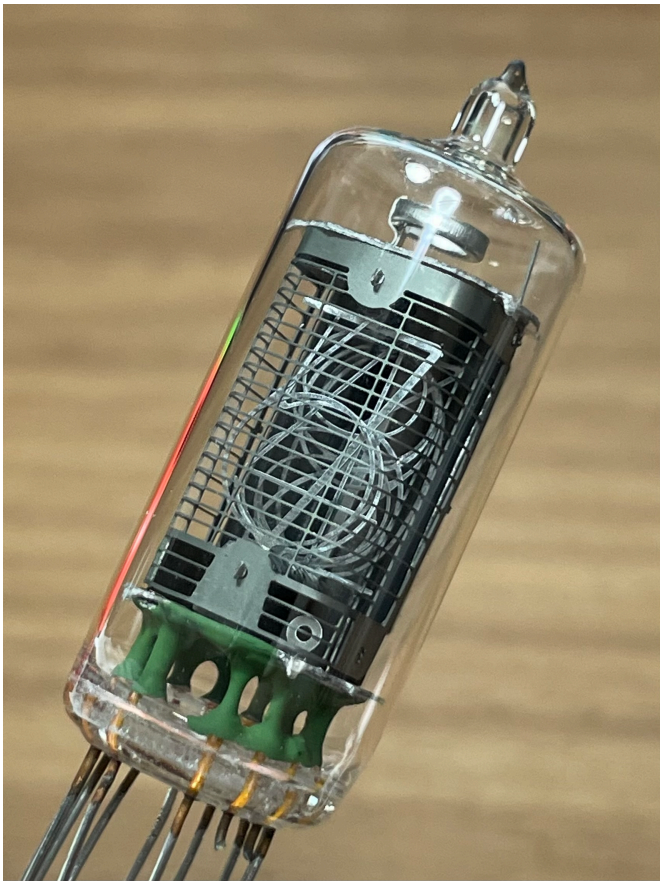

図 3.2: ニキシー管

ニキシー管の登場当時は電子的にデジタル表示を行える表示器がほかに電球くらいしかなく、電圧計や周波数計等の計測器、電子計算機や自販機などさまざまな用途に使われたようです。構造が精密かつ複雑だったり特許料の支払いが必要だったりで高価、駆動に高電圧が必要、といった理由で、その後登場した VFD 管や LED 表示器に置き換えられていきました。1990 年くらいまでは生産が続けられていたようです。ちなみに自分が生まれていない時代の代物なので、私は外でニキシー管が使われているところを一度も見たことがありません。

種類とか

ニキシー管には大小さまざまなサイズがあり、また表示部分も上向きのものと横向きのものがあります。小さいものでは小指程度の太さしかありませんが、大きいものでは手のひらサイズのものまであります (図 3.1 は世界最大のニキシー管、CD47 です。ちなみに岡谷電気という日本メーカー製です)。

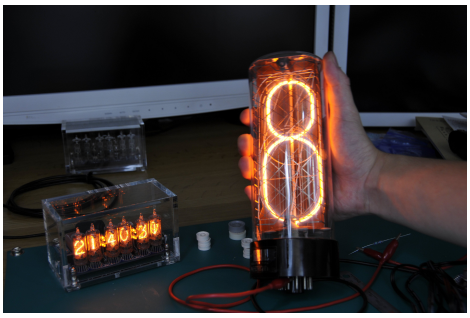

図 3.3: CD47

どうやって動かすの

中身の構造は複雑ですが、光らせるのは簡単です。点灯したい数字に対応したアノード端子と共通のカソード端子との間に 180V 程度の電圧をかけるだけです。流す電流は基本的に直流の数 mA 程度なのですが、電圧は 180V とかなり高いので電子工作とかしたことないよ！ という方にはあまりおすすめできません。もしニキシー管を光らせたいという方がいましたら十分安全に配慮しつつ自己責任でお願いします。

3.3 とりあえず光らせてみよう

とりあえず 170V に昇圧した電源に繋いでみます。数 mA 程度に調整する為に抵抗を繋ぐのを忘れないようにしましょう。過電流はニキシー管の寿命を著しく縮めます。

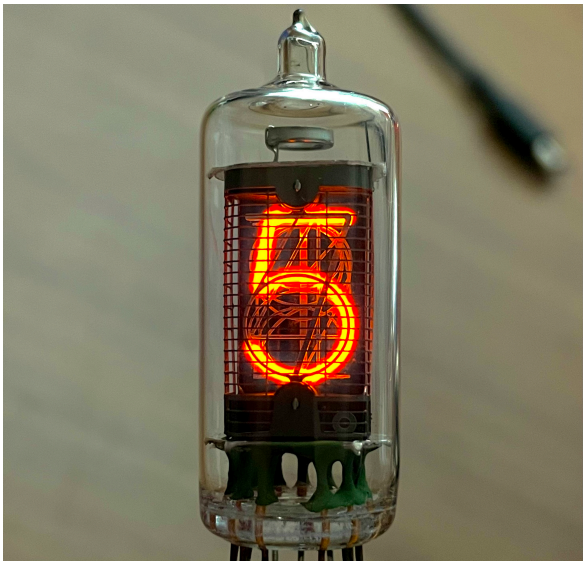

図 3.4: 光った (喜び)

次に実際の製作 (?…) に移りますが、紙面の都合によりこれ以降は Web 版にてご覧下さい。

3.4 終わりに

締め切り 2 時間前に書き始めたのですが、まぁなんとかなったでしょう。(なんとかなっていますか？) この記事を読んで少しでもニキシー管に興味を持って下さった方がいたら幸いです。

お付き合い頂きありがとうございました！

3.5 参考文献及び引用

- 2.2 - 概要：Q61.ORG Blog

  https://q61.org/blog/2019/08/28/what-is-a-nixie-tube-nixie-tube-faq/

- 図 3.3：ゆな でぃじっく氏[*1]の Twitter 投稿より引用

  https://twitter.com/yuna_digick/status/678964075263430656

- 図 3.1：マイコン活用シリーズ マイコンと表示器をつなぐ 10 の方法 (書籍) より引用

[*1] 生産の終了したニキシー管を作り直し復刻しようという試みに個人で取り組んでおられる凄い方です。

第 4 章

i3 window manager を使ってみよう

76 回生 maguro

4.1　挨拶

こんにちは。76 回生（高 2）の maguro です。普段は競技プログラミングと CTF をしています。気が付いたらもう高校生活の終わりが近づいており、焦りが出始めてきました。

今回は、前に使って思ったよりも良かった i3 window manager を紹介したいと思います。

4.2　i3 window manager ってなに？

i3 window manager(i3wm) とは、**Linux 上で動くタイル型ウィンドウマネージャ**です。多分ほとんどの人が分かってないと思うので、とりあえずデスクトップ画面を見せたいと思います。これは Manjaro i3wm Edition です。[*1]

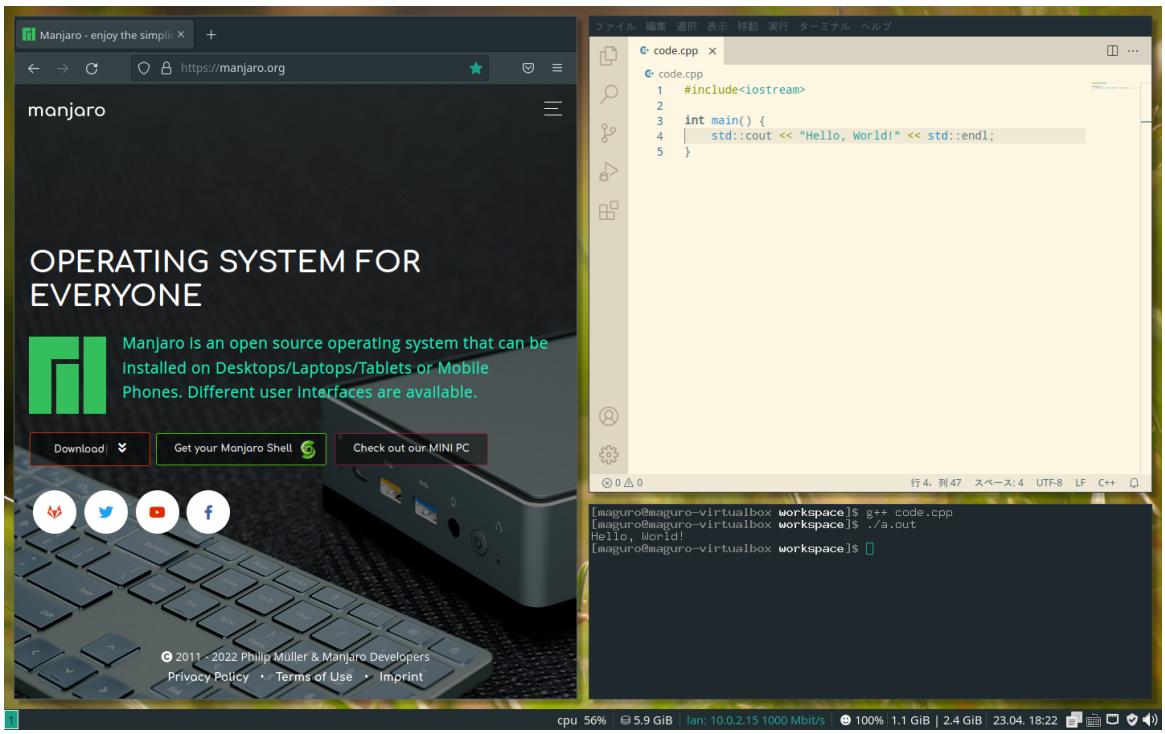

図 4.1: デスクトップ画面

[*1] 筆者は Windows をメインで使っているので、これは仮想環境で撮影しています。

これらの画面をキーボードだけで表示することができます。慣れるとマウスで操作するよりも早くできます。

ここからちょっと詳しく説明します。分からなかったら飛ばしてください。

Linux とは

OS（コンピュータを動かす土台になるソフトウェア）の一種で、Ubuntu や Arch Linux などのディストリビューション（OS のコアになるカーネルと、その他もろもろのソフトを含めて配布する形態）の総称です。

タイル型ウィンドウマネージャとは

ウィンドウマネージャ（ウィンドウの配置や見た目を管理するプログラム）の一種です。

ウィンドウマネージャはウィンドウの描画や更新の手法で大きく

- コンポジット型ウィンドウマネージャ
- スタック型ウィンドウマネージャ
- **タイル型ウィンドウマネージャ**

の 3 種類に分けられます。タイル型ウィンドウマネージャは他の 2 つのウィンドウマネージャと違い、ウィンドウ同士が重ならないように分割して表示します。

「ウィンドウ同士が重ならない」以外に、i3wm の特徴として

- **基本的にキーボード操作のみでウィンドウを配置できる**[*2]
- タブ・スタック化機能（Web ブラウザーのタブのような感じでソフトを整理する）
- ワークスペース（仮想デスクトップ）機能

が挙げられます。

本記事では具体的な入手手順は紹介しません。`i3wm` と検索すると様々な方法が出てくると思います。

4.3 起動から基本操作まで

まずは先ほどデスクトップ画面を見せた、i3wm が使える Linux である Manjaro i3wm Edition を起動して、ターミナル（コマンドが打ち込めるところ）を開けるようにしましょう。ここで i3wm でよく使う mod キーについて説明します。

mod キーとは

mod キーは i3wm で使用する特別なキーで、デフォルトだと Win キー（Mac でいうコマンドキー）や Alt キーが割り当てられています。

[*2] 一応 Windows や Mac のようにマウスで動かせて、他のものと重なるウィンドウも作ることができます。

このキーと他のキーを一緒に押すことで、様々なことを行うことができます。例えば、

- `$mod + enter` でターミナルを開く
- `$mod + shift + q` でウィンドウを閉じる

などです。
とりあえず`$mod + enter` でターミナルを開きます。

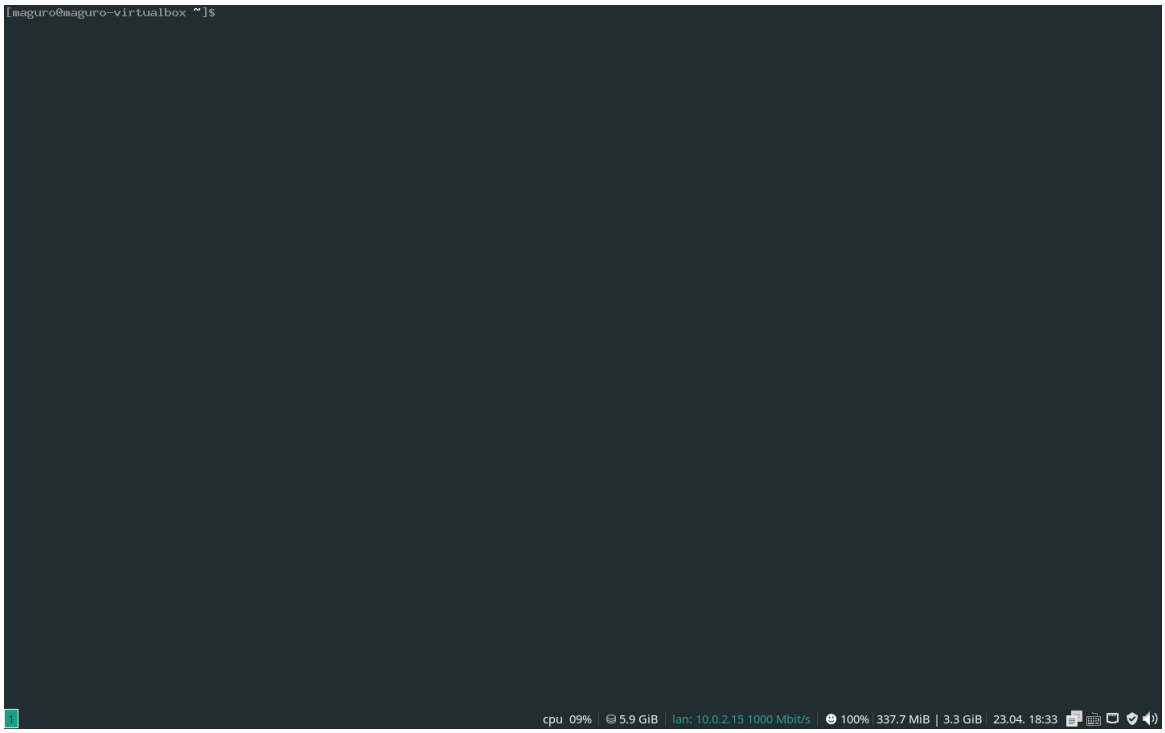

図 4.2: ターミナル（1 画面）

もう一回`$mod + enter` を押すと、画面が分割されてターミナルが 2 つ表示されます。

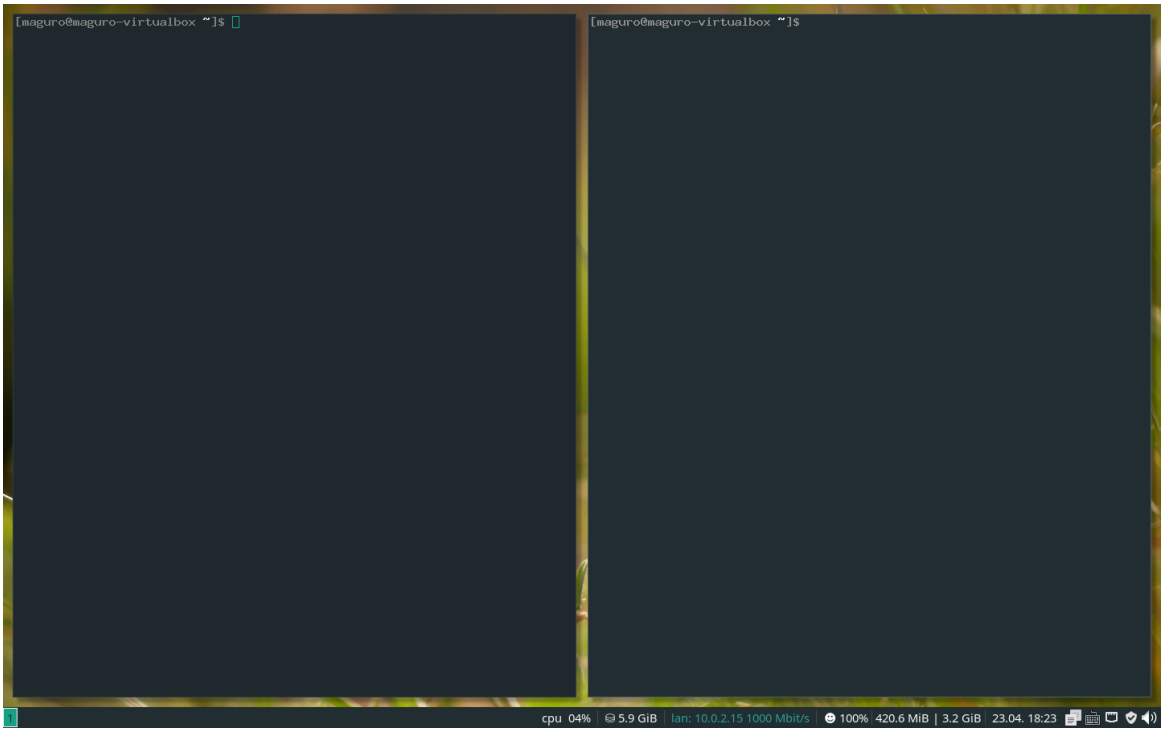

図 4.3: ターミナル（2 画面）

下のショートカットでフォーカスするウィンドウを変えられます。

- `$mod + j` で左
- `$mod + k` で上
- `$mod + l` で下
- `$mod + ;` で右

また、ウィンドウを追加するときにどこに追加するかは下のショートカットで指定できます。

- `$mod + h` で右方向
- `$mod + v` で下方向

例えば、画面に何もソフトが無い状態から`$mod + Enter` → `$mod + Enter` → `$mod + v` → `$mod + Enter` と入力すると下の画面のようになります。

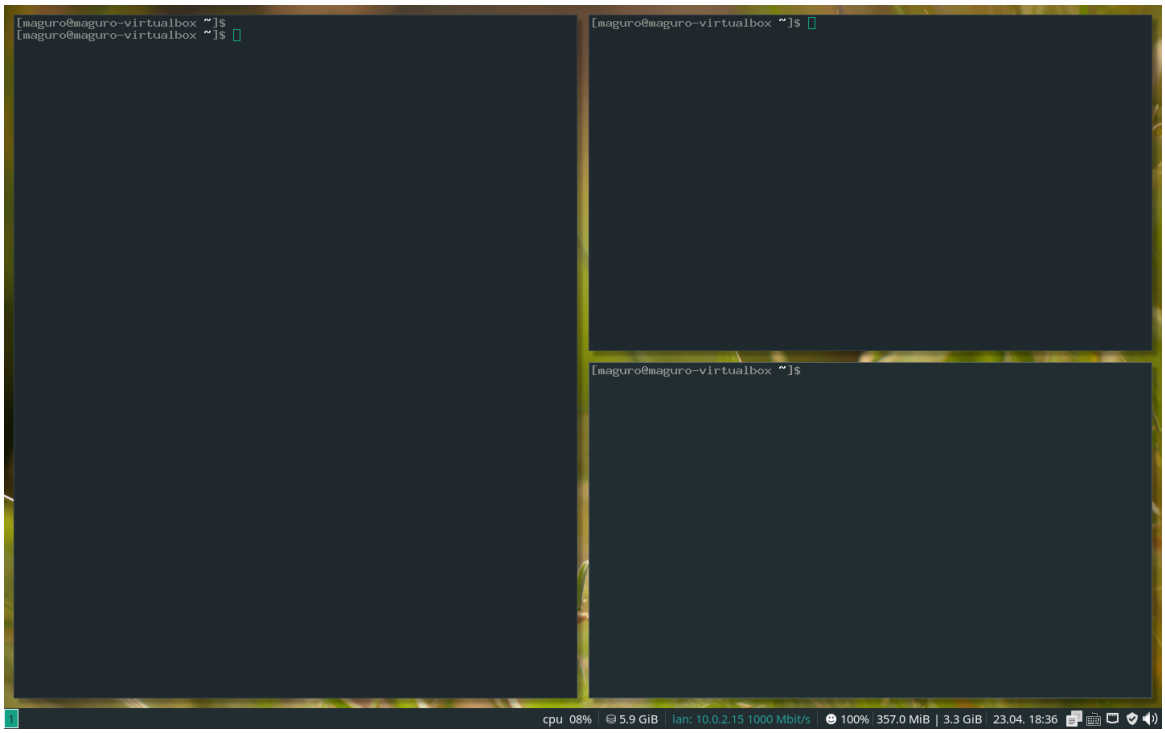

図 4.4: ターミナル（3 画面）

そして、フォーカスしているウィンドウを閉じるためには、`$mod + shift + q` と押します。

4.4 最後に

サンプルはここまでです。本編では、ソフトのインストールや見た目の設定をしていきます。是非ご覧ください。

第 5 章

TOTP の生成と Base32 の変換 in C#

76 回生 デカブツ

5.1　挨拶

76 回のデカブツです。気が付けば学校生活が 2/3 終わりました。今年で 20 周年の C#という言語を触っています。この記事は入部後初めてのものとなりますので、多少拙いところがあると思いますが、暖かい目でご覧ください。

5.2　実装したコードを見たいという方へ

長くなるので一部分は記事中に含めず外に置くことにしました。下のリンクから御覧ください。

`https://bit.ly/3OwTxu4`

5.3　多要素認証に使われる TOTP と HOTP について

多要素認証とは

多要素認証は、IPA(情報処理推進機構) の Web サイトによると、「各種インターネットサービスにおける不正ログイン対策として、複数の要素（記憶、所持、生体情報）を用いた認証方式」[*1] と述べられています。例えば、パスワードを入力した後に SMS や E メールで送られてくるコードを入力するというような Web サイトは、"多要素認証対応" などといった感じで呼ばれます。

HOTP とは

HOTP は、上で述べられた「複数の要素」の内の一つとして使われる形式です。特徴としては、ユーザー側とサービス側で共有した鍵と特定の数字 (サービス側で生成した乱数や生成した回数など) を使い、HMAC-SHA-1 関数により生成される数桁の数字のパスコードを使って認証するというものです。

[*1] 不正ログイン対策特集ページ `https://www.ipa.go.jp/security/anshin/account_security.html`

TOTP とは

TOTP は、先ほどの HOTP の特定の数字を現在時刻から求めたカウンタに置き換え、パスコードを生成するものです。短時間で更新され、生成に使う特定の数字を共有しやすいという点により、HOTP よりもよく使われています。

HMAC-SHA-1 とは

これについての説明は、サンプル版なので非常に大まかな内容しか伝えません。詳しく知りたい場合は Web 版の部誌を見るか IETF という団体の RFC2104 という文書[*2]を見てください。

HMAC-SHA-1 は鍵と入力によって決まるランダムな短い出力を返す関数の一部です。今回は C#に公式でこれを計算する機能 (System.Security.Cryptography.HMACSHA1) があるので、それをそのまま使います。

5.4 TOTP と HOTP の仕様と生成方法について

1 注意書き

サンプル版なので非常に簡略化しています。詳しい内容は Web 版の部誌を御覧ください。

2 生成のための大まかな手順

2.1 TOTP の場合

1. 鍵を読み込みます。
2. 1970 年 1 月 1 日午前 0 時 0 分 0 秒から現在の協定世界時 (UTC) までの秒数を 30 で割りカウンタを求めます。(なおこの時余りは切り捨てます)
3. 先ほどの鍵とカウンタをもとに HOTP を求めます。(下の HOTP の場合を御覧ください)
4. 求めた HOTP を出力します。

2.2 HOTP の場合

1. 鍵とカウンタを読み込みます。
2. 鍵とカウンタを HMAC-SHA-1 関数で変換し、短い特定のデータを取得します。
3. 先ほど取得したデータを DT 関数 (省略、特定の一部だけを切り取るものと考えてください) を使って最大 2^31-1 の整数を求めます。
4. 先ほど取得した整数から特定の桁数だけ切り取り出力します。
5. 6 桁の数字を出力します。

と、非常に簡素な仕組みとなっています。上をそのまま実装した場合の C#でのコードは下の

[*2] RFC2104 の参考 URL:https://datatracker.ietf.org/doc/html/rfc2104

様です。

List 5.1: TOTP.cs

```
using System;
using System.Security.Cryptography;
namespace LibraryForTOTP
{
    public static class RFC6238andRFC4226
    {
        public static int GenTOTP(byte[] S, int adjust = 0, int span = 30)
        {
            TimeSpan time = DateTime.UtcNow - new DateTime(1970, 1, 1);
            var counter = (long)time.TotalSeconds / span;
            return GenHOTP(S, counter + adjust);
        }
        public static int GenHOTP(byte[] S, long C, int digit = 6)
        {
            var hmsha = new HMACSHA1();
            hmsha.Key = S;
            var counter = BitConverter.GetBytes(C);
            Array.Reverse(counter, 0, counter.Length);
            var hs = hmsha.ComputeHash(counter);
            return DTruncate(hs) % (int)(Math.Pow(10, digit));
        }
        static int DTruncate(byte[] vs)
        {
            var offset = vs[vs.Length - 1] & 15;
            var P = (vs[offset] << 24 | vs[offset + 1] << 16 | vs[offset + 2] << 8 | vs[
offset + 3]) & 0x7fffffff;
            return P;
        }
    }
}
```

ここより下は、Web 版にてお楽しみください。

サンプル版では載せていない部分の大まかな目次とその要約は下の通りです。

5 Base ○○とは?

0 と 1 の組み合わせではなくアルファベットと数字の組み合わせでデータを表す方法である Base ○○について説明します。

6 Base32 の仕様とその変換方法について

RFC4648 にある Base32 の仕様の解説とその実装を紹介します。

第 6 章

万年カレンダーを実装する

てぃー

6.1　はじめに

こんにちは。78 回生のてぃーです。昔はすしとるなという名前を使用していました。実はこの部の副部長をしています。ですが何も仕事をしていません。ごめんなさい。

そういえば競プロをやめました。中 2 の夏くらいまでは競プロに全てを捧げるぞとか思っていたんですが、凄い人たちを見たり他に自分の好きなこと見つけたりして、そろそろ競プロは良いかなってなりました。

ということで今回は Swift で万年カレンダーの iOS アプリを実装することにしました。カレンダーは様々なアプリで使えるので、自分でカレンダーを作れるようになっておくとカスタマイズも自在にできるので便利だなと思い実装しました。

6.2　ツェラーの公式

万年カレンダーを実装する上で最も重要となるのが、曜日の計算です。この曜日の計算をするための公式がツェラーの公式というものです。まずはこの公式を導出します。

まず前提として、西暦 1 年 1 月 1 日は土曜日です。ここから何日経過したかを求めることで、曜日を計算することができます。閏年がなければ 365 に年数をかけるだけで良いのですが、閏年は、

「4 の倍数かつ 100 の倍数でない。または 400 の倍数である年のみ閏年になる」

というややこしい感じになっているので日数を求めるのが割と大変です。

とりあえず 100 の倍数が鍵となるので、西暦 y 年を $100j + k$ 年に分解します。閏年分以外を計算すると、$365(100j + k - 1)$ 日となります。ここから、基本的に 100 年に 24 回閏年があるので $+24j$ 日、ただし 400 年に 25 回ある世紀があるので $+\lfloor j/4 \rfloor$ 日、下 2 桁分の $+\lfloor k/4 \rfloor$ をすると、西暦 y 年 1 月 1 日の西暦 1 年 1 月 1 日からの経過日数は、

$365(100j + k - 1) + \lfloor j/4 \rfloor + \lfloor k/4 \rfloor$ 日

となります。しかし 365 などは 7 で余りを取った方が小さくなるので、頑張って変形して、

$k - 1 - 2j + \lfloor j/4 \rfloor + \lfloor k/4 \rfloor$ 日

と書き換えることができます。紙面の関係上、ツェラーの公式の続きは Web でご覧ください。

6.3　アプリ概観

最後に、実際にこれを実装したアプリの概観だけ掲載します。

図 6.1

ツェラーの公式の続きや実装については Web でご覧ください。

第 7 章

初めての web アプリ作成

nisnah

7.1　挨拶

78 回生の nisnah です。気づけばもう学校生活の 1/3 が終わっていました。時の流れってはやいですね。それなのにまだあまり進捗が生やせていないことに焦りを感じています。そこで、web アプリというものを作ってみます。

7.2　web アプリとは

web アプリとは、ウェブなどのネットワークから利用するアプリです。また、web アプリには、よく比較されるものとしてネイティブアプリというものがあります。ネイティブアプリとは、手元の機器にインストールして使うアプリです。

7.3　作る web アプリの内容

今回作る web アプリは「ndnd 掲示板」です。名前の通り掲示板として使えます。(とりあえず形にしたので稚拙な部分があるかもしれません…) スレッドにタグをつける機能などをつけます。

7.4　作成手順

今回は、windows でメモ帳を使って作っていきます。html ファイルをメモ帳で開くときはファイルを選択し右クリックして、「プログラムから開く」からメモ帳をクリックすれば開けます。css ファイルをメモ帳で開くときはダブルクリックすれば開けます。

最初に、フォルダを適当な名前で作り、その中に index.html というファイルを作ります。そして index.html をメモ帳を開いて、下のコードを記述します。

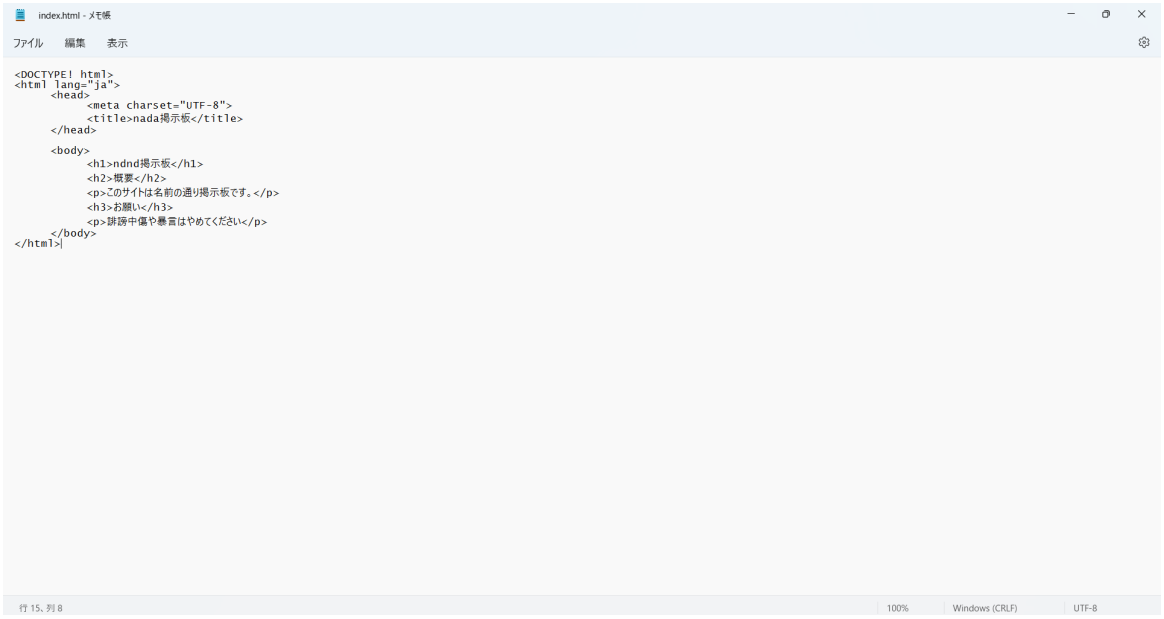

図 7.1

これは HTML というマークアップ言語 (プログラミング言語とは違う！) によって書かれたものです。これが web アプリの土台となります。そして上書き保存をし、ファイルをクリックして開くと画像のようになります。

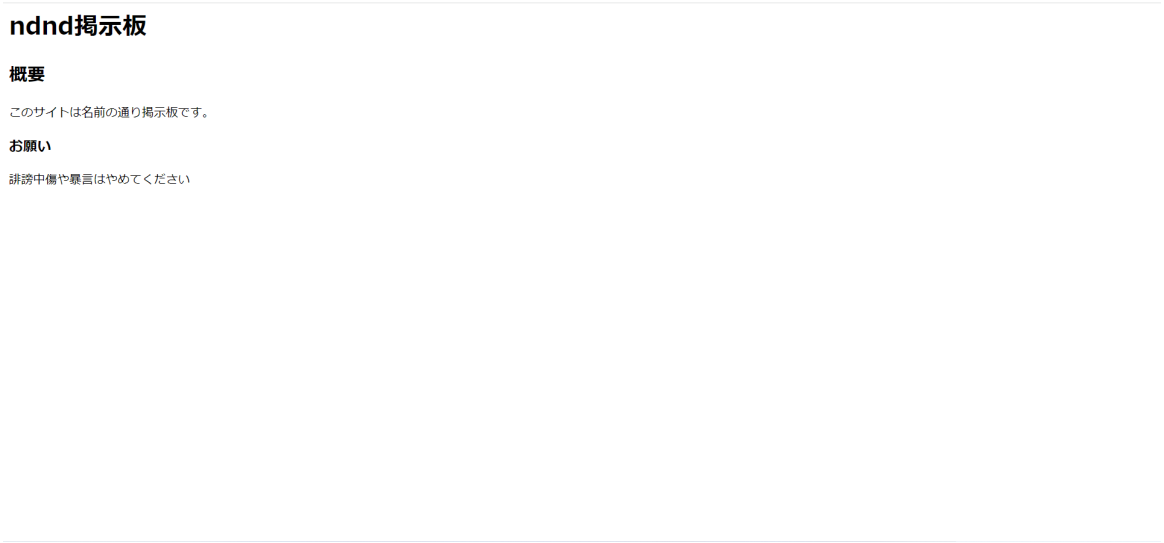

図 7.2

(。´・ω・) ん?と思いましたよね？ 何か味気ないなあと。それは、CSS ファイルを作っていないからです。CSS とは HTML で書かれた文書の見た目を装飾することのできるスタイルシー

ト言語です。ということで、CSS ファイルを作っていきます。まず、先ほど作った適当な名前を付けたフォルダの中に、style.css というファイルを作ります。そして style.css をメモ帳で開き、下のコードを記述します。

図 7.3

そして、上書き保存をします。その次に、index.html をメモ帳で開き、下のコードのようになるよう記述します。

図 7.4

またまた上書き保存をし、index.html を開くと…

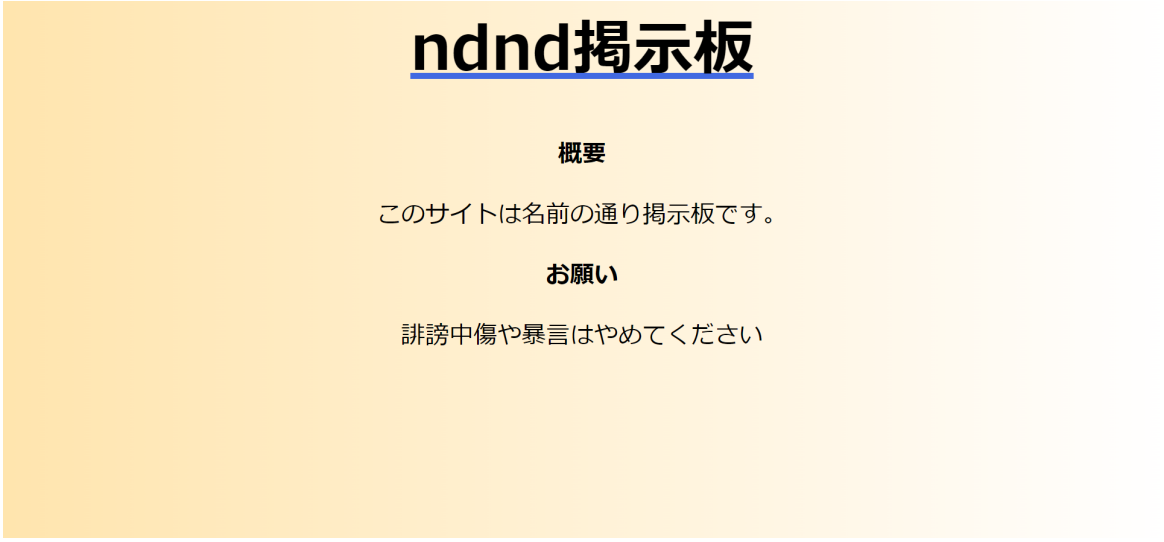

図 7.5

なんと、先ほどよりカラフルにいい感じになりました！ このほかにも、HTML ファイルの中に CSS を記述する方法などもあります。ここまでで、掲示板のトップページのおおよその見た目が完成しました。

サンプルはここまでにします。部誌本編ではこのページの作成の続きや他ページの作成、様々

な機能の実装を書く予定です。ぜひご覧ください。

7.5 参考文献

【図解】WEB アプリケーションとは？ 仕組みと開発言語を解説！ - カゴヤのサーバー研究室 (https://www.kagoya.jp/howto/it-glossary/web/webapplication/)

第 8 章

レシート記述言語で遊ぶ

stranger_86952

8.1　はじめに

78 回生の stranger_86952 です。中 1 でパソ部に入って以来、競プロ、Web 開発、Python、シェルスクリプト、レシート記述言語など色々やってきました。次は何をやろうかとネットをさまよっていたところ、Twitter でレシート記述言語という奇妙な言語の存在を知ったので今回はこいつを使って遊んでいきます。

8.2　言語って何

プログラミング言語のことで、日本語や英語などの喋る言語ではありません。特定の文法に従って書くことでテキストを表示したり、計算させたり、ゲームを作ったりできます。言語によってできることは様々ですが、今回はレシートを書くための専用の言語があると知ったのでこれを使って遊びます。

8.3　具体的に何の言語を使うの

今回使うレシート記述言語は ReceiptLine という言語です。ReceiptLine の公式サイトでコードを実行することができ、なんとサンプルまで置いてくれています。もちろん ReceiptLine の学習サイトなどはありませんが、このサンプルを見ながらレシートを極めたいと思います。

続きは本編で