2023
春号

百萬石
Hyakumangoku Vol.60

Published by MMM

部長挨拶

gae

gae@mma.club.uec.ac.jp

「百萬石」をお手に取っていただき，ありがとうございます．弊誌 (以下，部誌) は，電気通信大学の公認サークル MMA が新歓期 (春号) と弊学が開催する調布祭時期 (秋号) の年 2 回発行・頒布しているものです．MMA とは，”Microcomputer Making Association”の頭文字に由来しており，1975 年に創立されたサークルです．現在では計算機に関連することを中心に，ネットワークやセキュリティなどの分野でも活動しています．部誌には，MMA の部員が各々に行なっている活動を記事にしたものが掲載されています．内容は多岐に渡り，高度なプログラミングの話から旅行記など，MMA の活動の自由さが感じられるものとなっております．

さて，この「春号」は新歓期に頒布されるものです．昨年度に引き続き，今年度も MMA は新歓に対面で参加するため，ありがたいことに直接部誌をお渡しできます．オンライン参加時代に作った，これまでの歴代の部誌がすべて pdf で読めるようになっている，部員が作成したサイトもあります．先輩方の歴史とともにお楽しみいただければと思います．

今回は春号ということもあり，新入生が多く読むと思うので，新入生向けに少し話しましょう．MMA にはすごい人たちが集まっているというイメージを持つ人もいるでしょう．確かにすごい人たちはいますが，未経験者も多く入部しています．全くの未経験で入部してから学び始める人も多くいます．私も 1 年生のときには完全な初心者でした．多くの経験者がいるため，初心者は経験者に教わりながら学ぶことができます．**計算機 (コンピュータ) を使っていれば，活動内容は完全に自由です．**部内でもさまざまな活動が行われているため，質問すれば誰かしらが答えてくれます．

しかし，「好きなことややりたいことができる」ということは「やることを探さなくてはならない」ことでもあります．やりたいことがない人は，なにをやりたいのか見つけることから始めてみましょう．昨年度までも，コロナ禍ということもあり，入部したのはいいものの，何をすればいいかわからない部員が幽霊と化していくことは珍しくありませんでした．

そこで我々は**新入生講習会**を用意しました．この講習会の目的は単なる勉強だけでなく，先輩との接点を持ち，雑談のような会話をしていく中で，自分の興味はどこにあるのかを探してもらい，やりたいことを見つけてもらうところにもあります．是非参加して，今後の自分のやりたいことや大学に役立つことなど，いろいろなことを見つけて学んでいってください．

2023 年 4 月吉日

百萬石 Vol.60

Hyakumangoku Vol.60

2023 Spring

MMA

目次

第 1 章

履修一覧表を自動生成してみた

Udon

1.1 はじめに

お疲れ様です。Udon です。

自分が電通大、ひいてはこのサークルに来てからもう 1 年経ったと考えると長かったような短かったような不思議な気分です。

名古屋に夜行バスで行って早朝名古屋駅から名古屋城に徒歩で行ったあと名古屋大の友達と RRR キメた話でも良かったんですが[*1]、流石に技術的なことを書こうと思ったので、今回も GAS を使ってみた話を書きます。

前回の部誌でやったことを前提としてやってみた感じなので、併せて読んでいただければと思います。Web とか部室に 2022 年度秋号があります。

1.2 GAS を使って何作ろう

さて、GAS を使うと言ったんですが、GAS はご存じ Google Apps Script の略称で、ガスだとかギャスって言われているものです。機能としては Google が提供する各種アプリケーションを紐づけし、色々な面で自動化を図るというものがメインとなっています。例えば Google フォームが送信された瞬間に「回答ありがとうございます。」というメールを送信者に自動送信する、なんてことができます。また、Slack に紐づけることもできるので[*2]、想像以上に汎用性は高いと言えます。

前回は Google フォームの回答から Web ページ更新のための HTML テキストを自動生成するやつを考えました。しかし、Web ページの更新はこちらが手動で行うため、更新結果の様子が送信者目線ですぐに分からないって課題がありました。というわけで今回はフォームを送信するとすぐにレスポンスが返ってくるものを作りたいと思ったのでそういうものを作りました。

というわけでフォームを送信するとメールが来るようにして、そのメールにフォームから自動生成した何かを添付すればええやんと思い、今回はそういう仕様のものを実装[*3]してみようと思います。

[*1] RRR 観てね!!

[*2] 実際に MMA の Slack にも実装されています

[*3] 「実装」ってかっこいいですよね

1.3 履修表の作成

せっかく GAS を使って自動化するんだから、一定の需要がないとつまんないよねと思ったので、新学年シーズンだからということで履修表を作ってみようと思いました。

というわけで、次のような流れを考えました。

1. Google フォームで取る科目を聞く
2. それに応じて履修科目を一覧表にしてスプレッドシートを生成する
3. そのスプレッドシートを Google フォームで取得したメアドに送る

履修表にしたのは、せっかく GAS を組んでつまらない産業廃棄物を生成しても意味がないので、一定の需要がありそうなものを作りたいと思ったからです。

では、さっそく実装していきたいと思います。

1.3.1 設計理念

一番楽な方法は履修する科目をフォームに記入してもらってそれをスプレッドシートに貼り付ける感じなんですが、それだと送信者がスプレッドシートを記入しているのと変わらないので、とりあえず必修科目は自動入力されるようにしようと思いました。

つまり、選択科目だけ入力してもらえばいいというわけですね。あとはシートを受け取るメアドと作りたい学期を聞けばよいです。

作りたい学期に応じて質問を変えれば、幅広い人に対して表を提供することができそうです。

新歓時期ということもあるので、1 年前期の履修表を作成するものを優先的に実装することにしました。それ以降のものはゆくゆく追加してみるつもりです。

以下、開発過程について解説していきます。

1.3.2 フォームの作成

まずは履修表を作成するためのフォームを作成しました。フォームでは以下の内容を尋ねます。

- メアド
- 留学生かどうか
- 作る学期
- 取る第二外国語 (必修)
- 取る第二外国語 (選択)
- その他選択科目を取るかどうか
- 取る場合の選択科目
- 教職科目を取るかどうか

- 取る場合の教職科目

フォームの回答はフォームに紐づけられているスプレッドシートに記録されます。そのスプレッドシートに新規シートを作り、とりあえず最新の回答を整理する領域とします。

まず、INDIRECT 関数を用いて回答収集シートから別シートに必要な情報だけをピックアップさせます。シートを学期ごとに分けておくことで、最初の方の質問で聞いた「生成したい学期」を基準として回答を仕分けることができるわけですね。

仕分けた回答は蓄積されていくわけですが、VLOOKUP 関数を用いることで最下行の回答を取得することができます。これで最新の回答をあるセルに表示させ、フォームの回答があるたびにその内容を変えることができるようになったわけです。

図 1.1 最新の回答の参照

1.3.3 GAS の実装

というわけで、ここからは GAS を用いてこのセルたちを扱っていきます。コードは以下の通り。

ソースコード 1.1 generate.gs

```
function generate() {
    let ss = SpreadsheetApp.getActiveSpreadsheet();
    let sheet=ss.getSheetByName('1年前期');
    let sheet2 = ss.getSheetByName('生成物');
    sheet2.clearContents();
    range=sheet2.getRange('A1');
    range.setValue('1年前期 履修表');
    range=sheet2.getRange('A3');
    range.setValue('必修科目');
```

```
        range=sheet2.getRange('D3');
        range.setValue('選択科目');
        range=sheet2.getRange('G3');
        range.setValue('教職科目・その他');
        range=sheet2.getRange('A29');
        range.setValue('合計単位数');
        range=sheet2.getRange('D29');
        range.setValue('合計単位数');
        range=sheet2.getRange('G29');
        range.setValue('合計単位数');
        range=sheet2.getRange('C29');
        range.setValue("=SUM(C5:C28)");
        range=sheet2.getRange('F29');
        range.setValue("=SUM(F5:F28)");
        range=sheet2.getRange('I29');
        range.setValue("=SUM(I5:I28)");
        range=sheet2.getRange('A5');
        range.setValue("微分積分学第一");
        range=sheet2.getRange('C5');
        range.setValue("2");
        range=sheet2.getRange('A7');
        range.setValue("線形代数学第一");
        range=sheet2.getRange('C7');
        range.setValue("2");
        range=sheet2.getRange('A9');
        range.setValue("数学演習第一");
        range=sheet2.getRange('C9');
        range.setValue("1");
        range=sheet2.getRange('A11');
        range.setValue("物理学概論第一");
        range=sheet2.getRange('C11');
        range.setValue("2");
        range=sheet2.getRange('A13');
        range.setValue("化学概論第一");
        range=sheet2.getRange('C13');
        range.setValue("2");
        range=sheet2.getRange('A15');
        range.setValue("ASEI");
        range=sheet2.getRange('C15');
        range.setValue("1");
        range=sheet2.getRange('A17');
        range.setValue("AWEI");
        range=sheet2.getRange('C17');
```

```
53        range.setValue("1");
54        range=sheet2.getRange('A19');
55        range.setValue("コンピュータリテラシ");
56        range=sheet2.getRange('C19');
57        range.setValue("2");
58        range=sheet2.getRange('A21');
59        range.setValue("基礎科学実験A1");
60        range=sheet2.getRange('C21');
61        range.setValue("1");
62        range=sheet2.getRange('A23');
63        range.setValue("基礎科学実験B1");
64        range=sheet2.getRange('C23');
65        range.setValue("1");
66        range=sheet2.getRange('A25');
67        range.setValue("健康・体力づくり実習");
68        range=sheet2.getRange('C25');
69        range.setValue("1");
70        range=sheet2.getRange('A27');
71        let range2=sheet.getRange('M1');
72        let val=range2.getValue();
73        range.setValue(val);
74        range=sheet2.getRange('C27');
75        range.setValue("1");
76        range=sheet.getRange('K1');
77        val=range.getValue();
78        if(val=="はい"){
79            range=sheet2.getRange('G5');
80            range.setValue("基礎数学演習第一");
81            range=sheet2.getRange('I5');
82            range.setValue("1");
83            range=sheet2.getRange('G7');
84            range.setValue("基礎物理学演習第一");
85            range=sheet2.getRange('I7');
86            range.setValue("1");
87            range=sheet.getRange('O2');
88            val=range.getValue();
89            if(val=="はい"){
90                range=sheet.getRange('Q2');
91                val=range.getValue();
92                range2=sheet2.getRange('G9');
93                let range3=sheet2.getRange('I9');
94                range2.setValue(val);
95                range3.setValue("2");
```

```
                range=sheet.getRange('R2');
                val=range.getValue();
                if(val!=""){
                    range2=sheet2.getRange('G11');
                    range3=sheet2.getRange('I11');
                    range2.setValue(val);
                    range3.setValue("2");
                    range=sheet.getRange('S2');
                    val=range.getValue();
                }
                if(val!=""){
                    range2=sheet2.getRange('G13');
                    range3=sheet2.getRange('I13');
                    range2.setValue(val);
                    range3.setValue("2");
                    range=sheet.getRange('T2');
                    val=range.getValue();
                }
                if(val!=""){
                    range2=sheet2.getRange('G15');
                    range3=sheet2.getRange('I15');
                    range2.setValue(val);
                    range3.setValue("2");
                    range=sheet.getRange('U2');
                    val=range.getValue();
                }
                if(val!=""){
                    range2=sheet2.getRange('G17');
                    range3=sheet2.getRange('I17');
                    range2.setValue(val);
                    range3.setValue("2");
                }
            }
        }else{
            range=sheet.getRange('O2');
            val=range.getValue();
            if(val=="はい"){
                range=sheet.getRange('Q2');
                val=range.getValue();
                range2=sheet2.getRange('G5');
                let range3=sheet2.getRange('I5');
                range2.setValue(val);
                range3.setValue("2");
```

```
139                 range=sheet.getRange('R2');
140                 val=range.getValue();
141                 if(val!=""){
142                     range2=sheet2.getRange('G7');
143                     range3=sheet2.getRange('I7');
144                     range2.setValue(val);
145                     range3.setValue("2");
146                     range=sheet.getRange('S2');
147                     val=range.getValue();
148                 }
149                 if(val!=""){
150                     range2=sheet2.getRange('G9');
151                     range3=sheet2.getRange('I9');
152                     range2.setValue(val);
153                     range3.setValue("2");
154                     range=sheet.getRange('T2');
155                     val=range.getValue();
156                 }
157                 if(val!=""){
158                     range2=sheet2.getRange('G11');
159                     range3=sheet2.getRange('I11');
160                     range2.setValue(val);
161                     range3.setValue("2");
162                     range=sheet.getRange('U2');
163                     val=range.getValue();
164                 }
165                 if(val!=""){
166                     range2=sheet2.getRange('G13');
167                     range3=sheet2.getRange('I13');
168                     range2.setValue(val);
169                     range3.setValue("2");
170                 }
171             }
172         }
173         range=sheet.getRange('N1');
174         val=range.getValue();
175         if(val!="選択第二外国語は取らない"){
176             range=sheet2.getRange('D5');
177             range.setValue(val);
178             range=sheet2.getRange('F5');
179             range.setValue("1");
180             range=sheet.getRange('O1');
181             val=range.getValue();
```

```
            if(val=="はい"){
                range=sheet.getRange('Q1');
                val=range.getValue();
                range2=sheet2.getRange('D7');
                let range3=sheet2.getRange('F7');
                range2.setValue(val);
                if(val=="キャリア教育基礎"){
                    range3.setValue("2");
                }else{
                    range3.setValue("1");
                }
                range=sheet.getRange('R1');
                val=range.getValue();
                if(val!=""){
                    range2=sheet2.getRange('D9');
                    range3=sheet2.getRange('F9');
                    range2.setValue(val);
                    range3.setValue("1");
                }
            }
        }else{
            range=sheet.getRange('O1');
            val=range.getValue();
            if(val=="はい"){
                range=sheet.getRange('Q1');
                val=range.getValue();
                range2=sheet2.getRange('D5');
                let range3=sheet2.getRange('F5');
                range2.setValue(val);
                if(val=="キャリア教育基礎"){
                range3.setValue("2");
                }else{
                range3.setValue("1");
                }
                range=sheet.getRange('R1');
                val=range.getValue();
                if(val!=""){
                    range2=sheet2.getRange('D7');
                    range3=sheet2.getRange('F7');
                    range2.setValue(val);
                    range3.setValue("1");
                }
            }
```

```
        }
    }
    function mail() {
        let ss = SpreadsheetApp.getActiveSpreadsheet();
        let sheet = ss.getSheetByName('生成物');
        let key = ss.getId();
        let gid = sheet.getSheetId();
        let token = ScriptApp.getOAuthToken();
        let url = "https://docs.google.com/spreadsheets/d/" + key + "/export?
            gid=" + gid + "&format=pdf&portrait=false&size=A4&gridlines=false&
            fitw=false";
        let pdf = UrlFetchApp.fetch(url, {headers: {'Authorization': 'Bearer
            ' + token}}).getBlob().setName("履修表"+".pdf");
        let sheet2=ss.getSheetByName('1年前期');
        let range=sheet2.getRange('J1');
        let Email=range.getValue();
        let Subject="履修表の自動生成が完了しました";
        let Body="自動生成した履修表をpdf 形式で添付します。";
        let fromAddress = "pgm_udon@mma.club.uec.ac.jp";
        let options = {from:fromAddress,attachments:pdf};
        GmailApp.sendEmail(Email, Subject, Body, options);
    }
    function exe(){
        generate();
        mail();
    }
```

1.3.4 機能

この GAS でどんな機能が実装されているのかを軽く説明しようと思います。

さて、この GAS は大きく分けて 3 つの部分から構成されていることが分かると思います。

generate 関数

コンセプトその 1 であり、この企画の肝です。

まずは用いるスプレッドシートと「1 年前期」「生成物」という 2 つのページを取得しておきます。

前あったデータは clearContents メソッドを用いて消去します。このメソッドは書式や枠線やセルの結合状態などは消去しないため、最初に設定しておいたフォーマットを保持することができます。

ここから、「セルを指定→記入」という操作を何回か繰り返します。見出しや必修科目など、誰がやっても必ず記入されるようなセルはここでまとめて処理します。

ここで、回答者が留学生かどうかで分岐します。留学生の場合は 2 科目受ける教科が増えるので、そ

の処理をします。その後、教職科目を記入していきます。スプレッドシートではカンマ区切りの回答を複数のセルに SPLIT 関数を用いて分割しているため、回答の数だけそれらを参照して生成物に書き込むことで処理ができます。留学生の場合は上から 3 段目から、そうでない場合は最上段から記入しなければ空欄ができてしまうので、先ほど説明したような分岐を用いています。

そして、選択第二外国語を取るかどうかで再度分岐します。取る場合は選択科目の欄の最上段に科目名が記入されます。その後その他の選択科目を教職科目の時と同じ要領で取得し、記入していきます。分岐の有無でその他の選択科目を記入し始める位置が変わってきます。

以上の処理の間、科目名の隣に単位数が併記されます。最後にそれらは各列で和を計算され、下の方に計算結果が出ます。

mail 関数

コンセプトその 2。スプレッドシートを取得し、「生成物」のシートを pdf 形式に変換します。そしてそれをフォームの回答で受け取ったメールアドレスに送信します。

GAS には sendEmail というメソッドがあり、4 つの引数それぞれに「送信先」「件名」「本文」「オプション」を渡すことができます。オプションでは細かな設定ができ、今回は変換した pdf ファイルの添付と送信元アドレスの設定を行いました。送信元は Udon の MMA でのメールアドレスとなっています。

exe 関数

GAS に仕込んだ関数はトリガー機能を用いると一定の条件下で自動で実行されるようになっていますが、上で紹介した 2 つの関数をトリガー設定すると、どちらが先に実行されるか分かりません。よって、まだ更新が済んでいないのにメールが送られてしまう可能性もあるわけです。

それを回避するために、exe 関数内で generate → mail の順で関数を実行するように設定しておき、トリガーを exe 関数のみに設定します。これによって、トリガーが発動した瞬間、目的通り「一覧表を更新し、更新したものをメールで送信」という動きができるわけです。

トリガーは「フォームの送信時」に設定したため、誰かがフォームを送れば数秒後には履修表が送付されてきます。

1.3.5 できたもの

これが完成したフォームのリンクとなります。

図 1.2 フォームのリンク

最後まで回答すると、履修リストが pdf 形式でメールを用いて送付されます。

1.4 おわりに

というわけでなんとか履修リストを自動生成するためのツールを実装することができました。

現時点では 1 年前期の物のみを取得できるなんとも限定的な仕様になっていますが、そのうち~~やる気があれば~~ 3 年後期くらいまで対応させたいな、とは考えています。

ただ、類分けプログラム分けとかで複雑な分岐があるので、一筋縄では行かなそうだと思ってます。履修登録期間内には実装したいですね。

あと、その教科の区分とかも併記できたらもっと便利かなとも思います。改良の余地はかなりあります。

今学期受ける科目をリストアップしたいときとかに、ぜひ使ってみてください。

ではまた、次の世界で。

参考文献

[1] https://www.yukibnb.com/entry/gas_send_email_from_another_email_address
[2] https://jetb.co.jp/8759
[3] https://tatsuya-note.com/gas-autoreplymail-forms/
[4] https://blog.synnex.co.jp/google/options-sending-email-by-gas/
[5] https://www.yukibnb.com/entry/gas_sheet_clear
[6] https://teratail.com/questions/117695
[7] https://tetsuooo.net/gas/1107/
[8] https://hajiritsu.com/spreadsheet-gas-setformula/

第 2 章

自作カラーテーマ「necodark」と自作ジェネレータ「Sccg」

ryota2357

こんにちは. 2 年の ryota2357 です.

先日, 「necodark」というカラーテーマをリリースしました. このカラーテーマは 「Sccg」というカラーテーマジェネレータ (私が開発し, 先日 v0.2.1 をリリースしました) を用いて作成しました.

本投稿では「necodark」と「Sccg」について書いていきます.

図 2.1　necodark (GitHub)

図 2.2　Sccg (GitHub)

図 2.3　Sccg Document

2.1　necodark

次のスクリーンショットは Neovim にて necodark を利用した際の画像です.

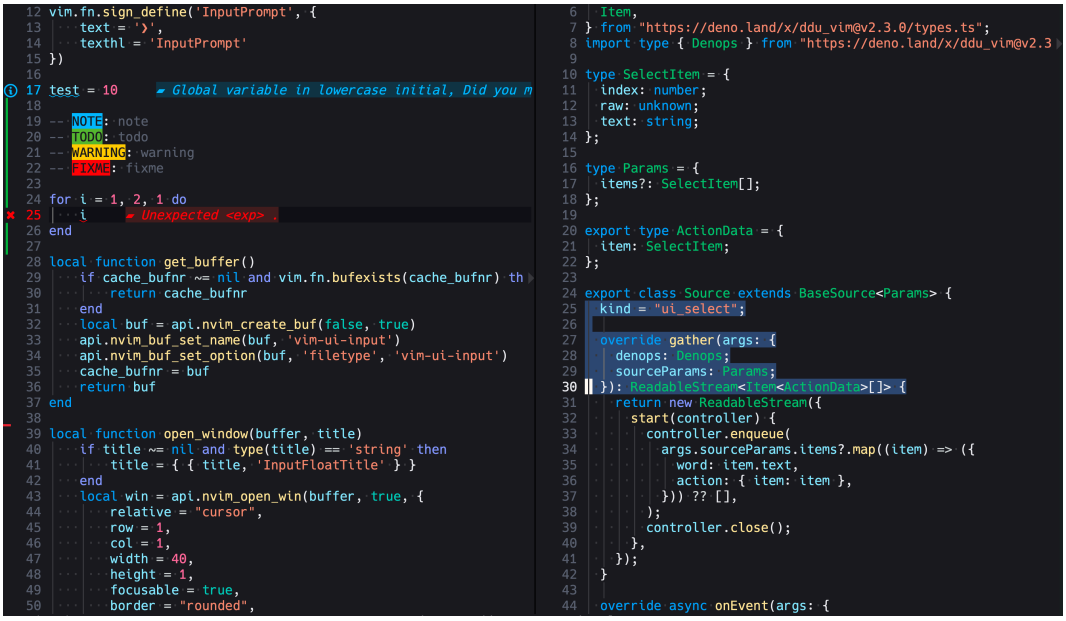

図 2.4

現在 (2023/04/04) Vim, Neovim にて利用できます. 現在 VS Code 対応の作業中です. 今後 CotEditor や IntelJ IDE, ターミナルミュレータ (iTerm2 や Alacritty) でも利用できるようにしていきます. 4 月中には VS Code と CotEditor でも利用できるようになる予定です.

このテーマは Visual Studio Dark (以下 vsdark) や Visual Studio Code Dark(以下 codedark) の配色を参考にして作成しました. (vsdark や codedark にて使用されている色を使ったわけではありません)

青・緑を中心に構成し, boolean 値には青緑 (Teal), 数値にはピンクを採用しました. 全体的にはっきりとした色 (彩度が高めの色, ビビット/ブライトあたりのトーン) を使っています.

カラーパレットは `https://github.com/ryota2357/necodark/blob/main/src/Necodark/Palette.cs` にて確認できます. 使用している主な色は次の通りです.

図 2.5

2.1.1 necodark の目的と特徴

necodark を作成した目的は主に次の 2 点です.

- 長時間のコーディングによる目の疲れがあってもしっかりコードを読める
- 馴染みのある配色である

1 つ目の目的のために, 背景には少し暗めの黒 (#1a1b1f) を使い文字色の彩度・コントラスト比を上げました. 2 つ目の目的のためベースに vsdark や codedark を採用しました.

necodark は「制御 (if や for)」「リテラル (boolean, 数値)」に特徴的な配色を行っています.

「制御」に充てている色は インディゴ (#8a99ff) です. この色はキーワード色として使っている青 (#379df1) に近い色を持ちます.

```
if (a == 2) {
    for (int i = 0; i < a; ++i) {
        b -= 1;
        continue;
    }
} else {
    while (true) {
        b -= 1;
        if (b < 0) break;
    }
}
```

図 2.6

vsdark では「制御」と「キーワード」には同じ色が当てられています. necodark ではこれらの色を分けました. 一般にキーワードはコード内で重要なマーカーとなります. そして「制御」もコードのロジックを示すのに重要なマーカーとなります. 重要なマーカーとしての意味であれば同じ色を当てても良さそうです. しかし necodark では上に挙げた目的の 1 つ目「長時間のコーディングによる目の疲れがあってもしっかりコードを読める」を達成するため色を分けました.

「制御」と「キーワード」はコード中に頻繁に登場します. そのため, もしこれらの色を同じにしてしまうと同じ色がコード内に比較的多く登場することになります. 通常時はとても良いマーカーになるかもしれませんが, 疲労時には同じ色が多いと感じて目が滑ってしまうかもしれません. これを解決するため necodark では「制御」にインディゴを当てました.

necodark で使用したインディゴと青は近い色です. しかしインディゴは青よりは目立たないでしょう. 先ほど「制御」は重要なマーカーと言ったのにキーワードのようには目立っていません. しかしこれで良いのです. 「制御」となる文字は `if` や `for`, `loop`, `raise`, `break` などが挙げられます. これらの文字はコード内でインデントや } などの付近で使用されています. そのため文字色で目出させずにも識別することは十分可能だと考えました. また, 特に `if` や `for` ではリテラルと共に使用されることも多いです. necodark ではリテラルにも目立つ特徴的な配色がされています. このことからも「制御」にはキーワードよりは目立たないインディゴを割り当てるのが適当だと考えました.

「リテラル」に当てている色は, 真偽値には青緑 (#4dcbbb), 数値にはピンク (#fb749c) です.

```
let a = 10u8;
let b = vec![0, 1, 2, 3, 4, 5];
let c = 0.1;
let d = true;
```

図 2.7

コード内に登場する数値リテラルは基本的に何かの固定値であり, 変更する時には注意を払う必要があ

ります．そのため数値には他のところでは使用していない色であり, 赤系に近いながらも警告の意味を持たないような色を当てました．

真偽値リテラルも分岐を決定する重要なものですが, 分岐の決定には文学的な方法 (メソッド名や変数名を用いて真偽を抽象化し意味を持たせて分岐する) が多いでしょう．そのため重要度はそこまで高くないかもしれません．しかしリテラルであることには変わりはなく, 重要ではあるので通常の文字色を当てることはできません．そこで青・緑と白の中間的な色を使おうと考えました．水色は「パラメータ」などの色に当てられているので緑と白の中間の色を採用しました．

2.2 Sccg

ここまで necodark の「見た目」について説明しました．ここからは necodark の「実装」について書いていきます．

necodark は C# 製のカラーテーマジェネレータ, Sccg を用いて作成されています．Sccg は v0.2.1 であり, 今後も開発は進められていく予定です．v0.2.1 ですが十分に実用に耐えれるものだと思ってます．

2.2.1 Sccg の目的と特徴

Sccg の README からそのまま引用しましょう．

> **S**criptable **C**olor **C**onfiguration **G**enerator.
Sccg is a tool to generate color schemes for your terminal, editor, etc.
Goals and Features
- Fully scriptable
  - the each generation logic, `Source`, `Formatter`, `Converter` and `Writer`.
  - can write your own `Source`, `Formatter`, `Converter` and `Writer` with C#.
- Unified API and Cross-platform
  - Each platform has its own color scheme format.
  - Sccg provides a unified API to generate color schemes for each platform.
  - You can set the color with `Set` and can link to other color with `Link`.
- Foolproof
  - There are may color groups, such as 'Comment', '@type.builtin', 'Ansi 0 Color', etc.
  - All color groups are defined in the `Source`, so you get completion, type check and so on.
  - Cycle reference will be detected by `Source`, then Sccg warn you or failed to build.

まとめると Sccg は, C# でカスタマイズ可能であり, 多くのプラットフォームに統一された方法でカラーテーマを作成することができ, typo や色の循環 link を抑制することができることで, 安全に効率的にオリジナルのカラーテーマを作成することを可能にするカラーテーマジェネレータです．(このことを目標に開発してます)

続いて Sccg にて採用した概念である, `Source`, `Formatter`, `Converter`, `Writer` の説明と関係を説明します. README にもあるように, これらを用いてカラーテーマの生成ロジックをスクリプト可能なものに分割しました. 次にこれらの関係を図示します ((README からの引用)).

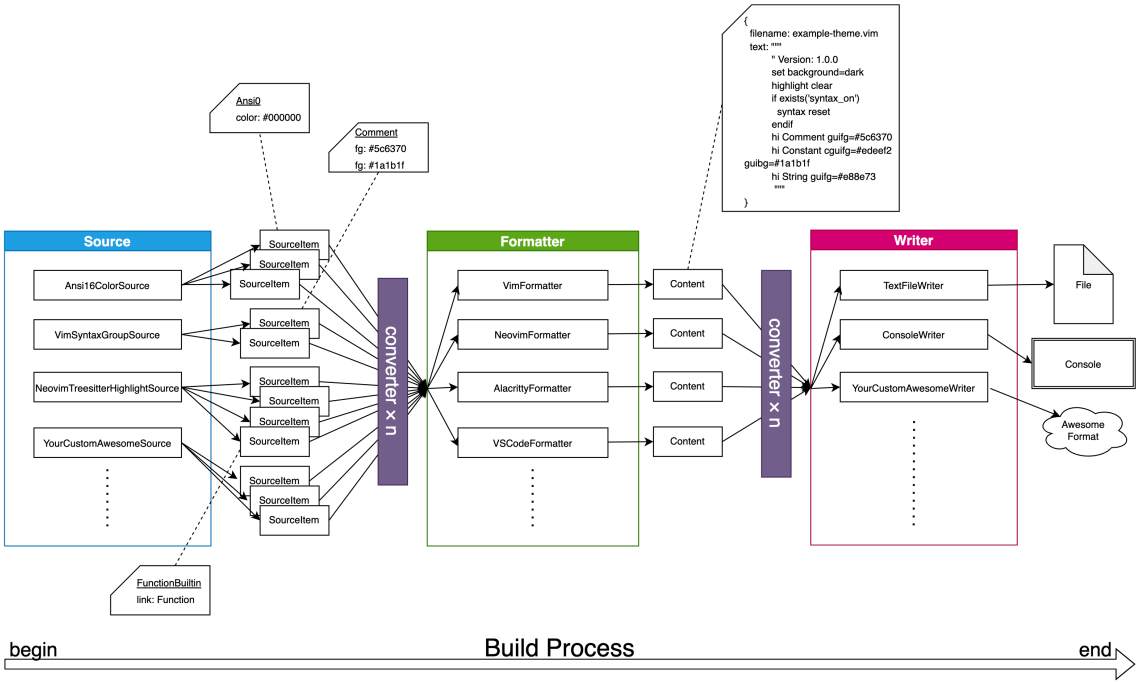

図 2.8

`Source` から `SourceItem` というものが生成され, それが `Formatter` によって `Content` に加工され `Writer` によって何かしらの形をとって出力されます. 途中 `Converter` が `SourceItem` や `Content` を加工することができます.

ここからは necodark の実装と合わせて, それぞれの概念がどのように使われ, 働くことでカラーテーマが生成されるのかを見ていきましょう. 個々の概念の説明も書いていきます.

2.2.2 Program.cs

necodark のメインスクリプト部分は次のようになっています.

```
using Sccg;
using Sccg.Builtin.Formatters;
using Sccg.Builtin.Writers;

var builder = new Builder
{
    Metadata = Metadata.Default with
    {
        ThemeName = "necodark",
        Author = "ryota2357"
```

```
11     },
12 };
13
14 // source
15 builder.Use<Ansi16>();
16 builder.Use<VimSyntaxHighlight>();
17 builder.Use<VimEditorHighlight>();
18 builder.Use<VimFiletypeHighlight>();
19 builder.Use<NeovimEditorHighlight>();
20 builder.Use<NeovimTreesitterHighlight>();
21 builder.Use<NeovimLspDiagnosticHighlight>();
22 builder.Use<NeovimPluginHighlight>();
23
24 // formatter
25 builder.Use<VimFormatter>();
26 builder.Use<NeovimFormatter>();
27
28 // converter
29 builder.Use<FilenameConverter>();
30
31 // writer
32 builder.Use<TextFileWriter>();
33
34 builder.Build();
```

1 行目から読んでいきます. using 部分を説明するために, Sccg が 2 つの部分に分けられていることを先に説明します. Sccg はコアな部分 (Sccg の核となる部分) とそれの付属部分 (標準ライブラリ的な部分) の 2 つに別れています. コアな部分を単に Sccg といい, 付属部分を Sccg.Builtin と呼びます. `using` にてコア部分といくつかの付属部分をインポートしています.

`using` の次では `Builder` クラスのインスタンス builder を生成しています (`Builder` クラスは Sccg に定義されています). `Builder` クラスは登録された `Source` や `Formatter` などを適切に利用し, カラーテーマを生成するロジックをもったクラスです. インスタンスの生成時にメタデータを記述しています. これにより, いつくかの `Source` や `Formatter` にメタデータを注入することができます.

続いて, builder に Source を登録しています. `Use<T>()` メソッドを用いて `T` クラスを builder に登録できます. 上記スクリプトでは Vim・Neovim 用の様々な Source を登録しています.

`Source` の登録の次に `Formatter` を登録しています. Vim と Neovim 用のテーマを作っているので `VimFormatter` と `NeovimFormatter` を登録しています. `Formatter` の次は `Converter` , そして最後に `Writer` を登録しています. 必要なものが全て登録しおわったら builder の `Build()` メソッドを呼び出すことでカラーテーマが生成されます.

`Source`, `Formatter`, `Converter`, `Writer` は 0 個以上の任意の個数登録することができます. 上のコードでは順番に登録してますが順番に登録する必要はありません.

ちなみに, Sccg は `Source` や `Formatter` などは 1 つも定義していません. Sccg.Builtin にて定義されています. `Source` や `Formatter`, `Writer` はただのクラスなので継承によってカスタマイズ・拡張することも可能です. もちろん自分で 1 から作成することも可能であり, さほど難しくはありません.

2.2.3 Source

1 つの Source は 0 個以上のカラーグループ (トークン) と, 各カラーグループの設定 (色やフォントスタイル) を持っています. 私たちはカラーテーマを作成するために Source を作る必要があります. 1 から Source を作るのは大変なので基本的には Sccg.Builtin から提供される Source を用いると良いでしょう.

Sccg.Builtin が提供する Source はカラーグループは持っていますが, 各カラーグループの設定は持っていません. 私たちはそのカラーグループに色やフォントスタイルを設定することができます. 私たちは色・フォントスタイルを 0 個以上設定したものを Builder に登録します. 登録された Source は私たちの設定に基づき, 0 個以上の SourceItem と呼ばれるものを生成します.

実際にコードを掲載します. 例えば Neovim の Editor ハイライトを設定したい場合は次のようにします.

```
class EditorHighlight : NeovimEditorHighlightSource
{
    protected override void Custom()
    {
        Set(Group.lCursor, fg: "aaff00", bold: true);
        Set(Group.Title, bg: "000000");
        Set(Group.Menu, none: true);
        Set(Group.ColorColumn, underline: true, bold: true);
        Link(Group.Conceal, Group.ColorColumn);
    }
}
```

この例は Sccg のテストで使っているコードです. NeovimEditorHighlightSource を継承した EditorHighlight クラスの Custom() メソッド内で Set() 関数や Link() 関数を用いて設定を書くことができます (NeovimEditorHighlightSource は Sccg.Builtin から提供される Source です). lCursor というグループに 前景色を#aaff00, スタイルを bold に設定したり, Title グループの背景色を #000000 に設定したり, Conceal グループを ColorColumn グループにリンクさせている, , などがわかると思います.

次に necodark にて実際に使用しているコードを載せました (同じく Neovim の Editor ハイライトを設定しているところです).

```
public class NeovimEditorHighlight : NeovimEditorHighlightSource
{
    protected override void Custom()
    {
        Set(Group.ColorColumn, bg: S.HideText.Foreground);

        Set(Group.Conceal, S.HideText);

        Set(Group.CurSearch, fg: S.NormalText.Background, bg: S.
            BooleanLiteralTeal.Foreground);
        Link(Group.IncSearch, Group.CurSearch);
        Set(Group.Search, S.SelectRangeBlue);

```

```
        Set(Group.Cursor, fg: S.NormalText.Background, bg: S.NormalText.
            Foreground);
        Link(Group.lCursor, Group.Cursor);
        Link(Group.CursorIM, Group.Cursor);
……
```

先ほどのように Set() で直接色・フォントスタイルを設定するのではなく, S.*というように, スタイル設定をまとめた変数を用意して Set() の第二引数で指定しています.

2.2.4 Formatter

1 つの Formatter は Source により生成された 0 個以上の SourceItem を受け取り, Content 1 つだけを生成するものです. SourceItem を参照するだけで消費はしません (できません).

実装は普通 Formatter が最も複雑になり, 異なる Formatter 間で共通する操作も少ないです. 基本的には 1 つの Formatter がカラーテーマの 1 つのターゲットプラットフォームに対応する形を取ります. Vim に対してビルドしたければ VimFormatter を, VS Code ならば VSCodeFormatter(現在開発中) を Builder に登録することで, 適切な Content が生成されます.

Formatter は受け取った SourceItem のうち 0 個以上を処理して (参照して)Content を生成します. ある Formatter にて処理してほしい場合はその Formatter が処理可能な SourceItem を Source にて生成するようにしなければなりません. 具体的には適切な interface を実装する必要があります. Sccg.Builtin の Source の中には複数の Formatter で処理可能にした SourceItem を生成する Source があります. 例えば Ansi16ColorSource は VimFormatter, NeovimFormatter, Iterm2Formatter, AlactirryFormatter で処理可能な SourceItem を生成する Source です. 処理してほしくない場合は ItemTarget プロパティをオーバーロードすることで無効化することができます. 他にも VimSyntaxGroupSource は VimFormatter と NeovimFormatter で処理可能な SourceItem を生成します. このようにすることで設定を複数プラットフォームで使い回すことを可能にしています.

実装コードを載せようと思ったのですが, どの Formatter も長かったので省略します.

2.2.5 Converter

Converter には 2 つの種類が存在します, SourceItemConverter と ContentConverter です. それぞれ SourceItem, Content を加工する, という役割を持ちます.

SourceItemConverter は n 個 (n は 0 以上の整数) の SourceItem を受け取り m 個 (m は 0 以上の整数) の SourceItem を返します. この時 Formatter とは異なり SourceItem を消費します. そのため加工したくないものに関しては受け取ったものをそのまま返す必要があります. ContentConverter も同様です. Content を消費します.

現在 Sccg.Builtin からは Converter は 1 つも提供していません. (VS Code 対応において MultiTextContentSplitter という ContentConverter が追加される予定です)

necodark では複数プラットフォームにビルドする際, ビルドディレクトリ整理のための ContentConverter を作成し利用しています.

```
public class FilenameConverter : IContentConverter
{
    public string Name => "Filename";
    public int Priority => 10;

    public IEnumerable<IContent> Convert(List<IContent> contents, BuilderQuery
        query)
    {
        foreach (var content in contents)
        {
            if (content is SingleTextContent singleTextContent)
            {
                yield return Path.GetExtension(singleTextContent.Filename) switch
                {
                    ".lua" => new SingleTextContent(Path.Join("nvim",
                        singleTextContent.Filename), singleTextContent.Text),
                    ".vim" => new SingleTextContent(Path.Join("vim",
                        singleTextContent.Filename), singleTextContent.Text),
                    \_ => new SingleTextContent(Path.Join("other",
                        singleTextContent.Filename), singleTextContent.Text)

                };
            }
            else
            {
                yield return content;
            }
        }
    }
}
```

Name と Priority というプロパティはここでは無視してください. Builder が適切に動作するために必要なものであって Converter の説明には関係しません. また, SingleTextContent というクラスが登場してますが, これは Sccg.Builtin にて定義されている Content の 1 つです. Filename と Text という 2 つのメンバを持つクラスで, 単一のテキストファイルを表すためのクラスです.

FilenameConverter ではファイルパス (Filename) を見てディレクトリ構造を追加しています. lua ファイルの場合は nvim/下に配置するように, vim ファイルの場合は vim/以下に配置するようにファイル名を変更したものを返しています.

2.2.6 Writer

Writer は 0 個以上の Content をファイルや標準出力など, 何らかの形に出力するものです. necodark では Sccg.Builtin が提供する TextFileWriter を利用して SingleTextContent という Content をファイルに出力しています. Formatter と同様, Content は消費されません (することができません).

Writer の実装は非常にシンプルなものになります. TextFileWriter の実装は次のとおりです.

```
public class TextFileWriter : Writer<SingleTextContent>
{
    public override string Name => "TextFile";

    protected override void Write(IEnumerable<SingleTextContent> contents,
```

```
        BuilderQuery query)
    {
        var outputDirectory = query.GetMetadata().Context.Get("OutputDirectory", "
            build");
        foreach (var content in contents)
        {
            var path = Path.Combine(outputDirectory, content.Filename);
            var directory = Path.GetDirectoryName(path);
            if (directory is not null && Directory.Exists(directory) == false)
            {
                Directory.CreateDirectory(directory);
            }
            File.WriteAllText(path, content.Text);
            Log.Info(\$"Wrote {path}");
        }
    }
}
```

2.2.7 interface

個々の概念を実際のコードにする際に次に示す interface を Sccg にて定義してあります. (名前空間は `Sccg.Core` です)

- `Source` を表す `ISource`
- `SourceItem` を表す `ISourceItem`
- `Formatter` を表す `IFormatter`
- `Content` を表す `IContent`
- `Converter` を表す `ISourceItemConverter` と `IContentConverter`
- `Writer` を表す `IWriter`

さらに, これら interface を実装し, 基本的な実装を加えた抽象 class も定義してあります.

- `ISource` を実装した `Source<TGroup, TItem> where TItem : ISourceItem`
- `IFormatter` を実装した `Formatter<TSourceItem, TContent> where TSourceItem : ISourceItem where TContent : IContent`
- `IWriter` を実装した `Writer<TContent> where TContent : IContent`

Sccg.Builtin ではこれら抽象 class と interface を実装した class, interface を作成し, `Builter` にて目的とするカラーテーマが作成できるようにしています. 詳しくはソースコードを読んでください. 一応ドキュメントコメントも書いてあります.

2.3 補足

2.3.1 なぜ C# で作ったのか

型が厳格であり, かつ柔軟性がある程度あり, リフレクションが使える言語で僕が扱えるのが C# だけだったから.

TypeScript とか Dart とか Rust も考えたけど, どれも条件に当てはまらなかった.

2.3.2 Converter の説明で出てきた Name と Priority プロパティは何をしているか

Name, Priority プロパティは ISource, IFormatter, ISourceItemConverter, IContentConverter, IWriter 全てに実装されているプロパティです. これらのプロパティは Builder クラスにて使われます.

Name プロパティは Builder::Use<T>() で利用され, 重複したものが登録されていないことを確認します. 重複したものが登録された時には例外 (ArgumentException) が投げられます.

Priority プロパティは Builder::Build() で利用され, 例えば, 登録された複数の Source の Source 内での処理順番を決定します. 小さい方が先に処理されます.

次にドキュメントコメント付きの ISource の実装を載せておきます.

```
1  /// <summary>
2  /// Represents a source that provides source items.
3  /// </summary>
4  public interface ISource
5  {
6      /// <summary>
7      /// Gets the source name.
8      /// </summary>
9      /// <remarks>
10     /// <see cref="Builder"/> cannot have more than one <see cref="ISource"/> with the
           same <see cref="Name"/> to avoid
11     /// creating duplicate <see cref="ISourceItem"/>.
12     /// </remarks>
13     public string Name { get; }
14
15     /// <summary>
16     /// Gets the order in which the source is applied. The lower the number, the
           earlier the source is applied.
17     /// </summary>
18     public int Priority { get; }
19
20     /// <summary>
21     /// Collects source items from the source.
22     /// </summary>
23     /// <returns>It is passed to <see cref="IFormatter"/> to format.</returns>
24     public IEnumerable<ISourceItem> CollectItems();
25
26     /// <summary>
27     /// Starts the source customization.
28     /// </summary>
29     /// <param name="query">The means of accessing other sources, etc.</param>
30     public void Custom(BuilderQuery query);
```

```
}
```

2.3.3 コード例で出てきてた BuilderQuery とは

上の ISource のドキュメントコメントにも書いてありますが，他の Source や Formatter, SourceItem などにアクセスするための手段です．necodark でもいくつかの場所で利用しています．例えば Source は 1 つ 1 つが独立して存在しているので別の Source に連動したことをしたい，だとか，Formatter にて暗黙的に ContentConverter を追加したい，など色々 1 つのクラスを飛び越えた連携をするのに使うものです．

次にドキュメントからメソッド一覧を載せました．

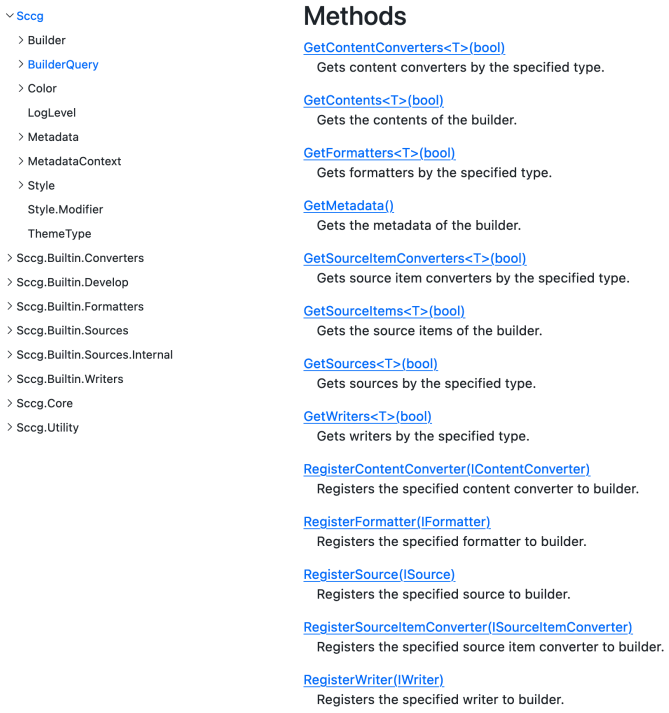

図 2.9

2.4 最後に

個人的に非常に満足できるカラーテーマを作成することができました．また，カラーテーマジェネレータも良いものができていると感じています．今後もカラーテーマ，ジェネレータともに開発を進め，さらに強力なものにしていきます．

第 3 章

MMA 競プロ班の紹介とオタク語り

terry

3.1 はじめに

2021、2022 部長をやってた新 4 年の terry です。AtCoder(Algo) の Highest は 1255(2018) です。いわゆる老人です。

実はゲーム制作もやったことがあるのでその辺興味ある人は聞いてください。

3.2 競プロとは？

競技プログラミングというものがありまして、ソフトやゲームを作るのではなく、問題を正確に解く速さを競うものもあります。[*1]それが競技プログラミングです。日本語だと AtCoder というサイトが有名ですね。イメージとしては $1 \leq N \leq 10^4$ を満たす N について、N 秒は何分何秒ですか？を求めるプログラム作りなさい、とかが簡単な例ですね。

プログラミングやりたいけど作りたいものがないそこの君、**一緒に競プロやろーぜ！** (〇堂守風)

3.3 MMA 競プロ班の紹介

実はそんな者なかったけど人がほしいので今作りました。今。

3.3.1 メンバー紹介

- `sepa_38` この前入青したコーディングスピード爆速最強新 2 年生
- `dyktr_06` 黄色まであとちょっとの高度典型に強い最強新 2 年生
- `terry(Nafmo2)` 部長をやってただけで現在緑のよわよわ新 4 年生

このメンバーに質問し放題で雑談し放題なの圧倒的に良い環境ですね。僕は教えてもらってばっかりで

[*1] 今回は Algo の話をします

後輩にお返しできていなくてごめんなさいになってる。

3.3.2 活動

毎週 Slack で ABC 後は感想戦をしていたりします。sepa,dyktr くんたちはコンテストを主催したりしてくれてます。あとは競プロの話を Slack で一生雑談してたりします。おもろいのでぜひ。

3.4 MMA 競プロ班 2022 年度活動報告

MMA Contest を MojaCoder で 14 回開催しました。偉すぎる。主に後輩が。ほか二人の後輩の仲間がほしいので MMA に入ってほしいなぁ。あとは夏休みに精進バトルをやったりして、AC 数を競ったりしました。時間がないので略（ごめん）

3.5 オレオレ環境構築

僕の環境をさらっとお話します。

3.5.1 UserScript

本当はお話したかったんですが、時間がないので泣く泣く省略。Tampermonkey とかで動かせます。詳しくは聞いてね。雑にスクショとともに紹介

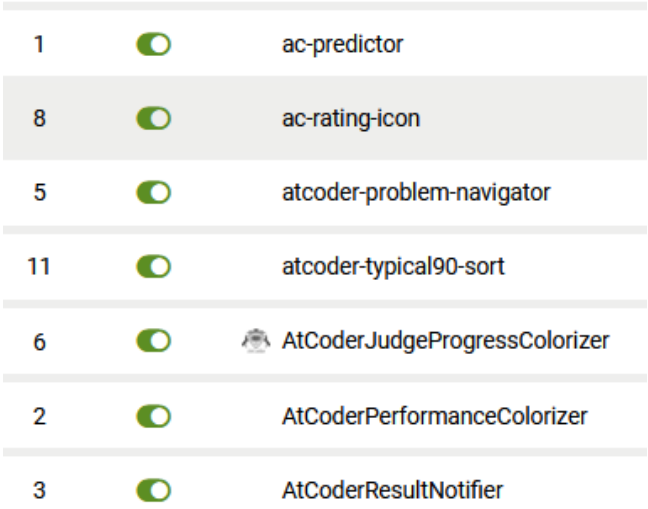

図 3.1 僕の使ってる UserScript の紹介

日付	コンテスト	順位	パフォーマンス	新Rating	差分
2023-04-01(土) 22:40	AtCoder Beginner Contest 296	1125	1373	1164	+26
2023-03-19(日) 22:40	AtCoder Beginner Contest 294	1671	1131	1138	-1
2023-03-11(土) 22:40	AtCoder Beginner Contest 293	1737	1204	1139	+7
2023-03-04(土) 22:40	AtCoder Beginner Contest 292	1129	1384	1132	+32
2023-02-26(日) 22:40	AtCoder Beginner Contest 291（Sponsored by TOYOTA SYSTEMS）	1858	1117	1100	+2
2023-02-11(土) 22:40	Ｓｋｙ株式会社プログラミングコンテスト2023（AtCoder Beginner Contest 289）	2081	1096	1098	±0
2023-02-04(土) 22:40	Toyota Programming Contest 2023 Spring Qual A（AtCoder Beginner Contest 288）	1937	1140	1098	+4
2023-01-28(土) 22:40	ユニークビジョンプログラミングコンテスト2023 新春 (AtCoder Beginner Contest 287)	1442	1262	1094	+21
2023-01-15(日) 22:40	AtCoder Beginner Contest 285	1342	1178	1073	+12
2022-12-03(土) 22:40	デンソークリエイトプログラミングコンテスト2022 Winter(AtCoder Beginner Contest 280)	1941	1096	1061	+4
2022-11-12(土) 22:40	大和証券プログラミングコンテスト2022 Autumn (AtCoder Beginner Contest 277)	2101	1040	1057	-2
2022-11-05(土) 22:40	AtCoder Beginner Contest 276	1724	1146	1059	+10
2022-10-22(土) 22:40	キーエンスプログラミングコンテスト2022（AtCoder Beginner Contest 274）	3091	760	1049	-28

図 3.2　パフォの色をつけるやつ

1500 (3) 108:25	100 2:01	200 4:36	300 9:29	400 22:29	500 (3) 93:25	-	-	-	1373	1138 → 1164 (+26)

図 3.3　レートの予測をしてみたり

ナビゲーターがでたり (1)、AC 状況のプログレスバーを出したり (2)、AC 速報が降ってきたりします (3)。

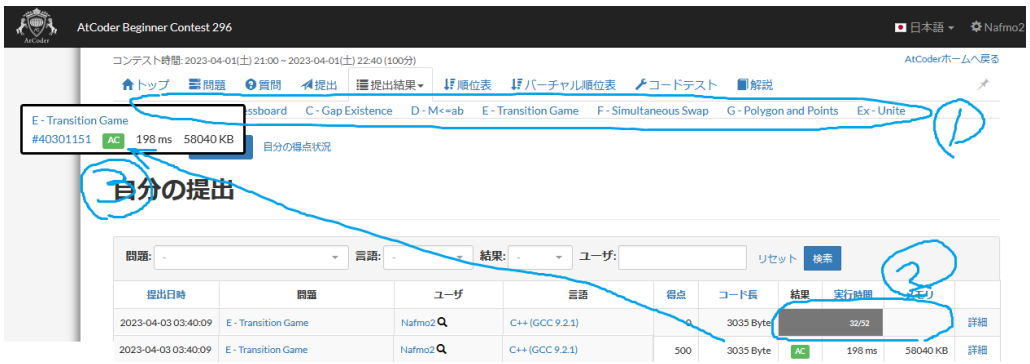

図 3.4　いろんな機能

3.5.2　atcoder-tool

類似ツールに oj とか atcoder-cli とかがありますね。詳しくは AtCoder Clans とかで調べると良さそう、僕も調べた（ぇ）

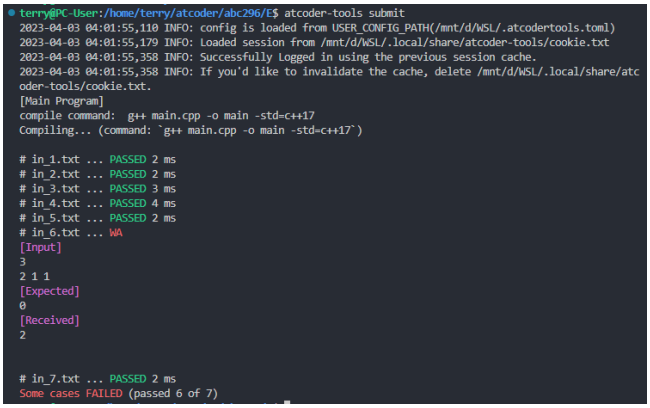

図 3.5　自動でテストケースと自作ケース通してくれて、自作ケース間違って提出止められてる人の図

3.5.3　その他

あとはうしさんのサイト `ei1333.github.io` や、スニペット (uf って押すと UnionFind をタブキーで出せるやつ) とかを駆使しています。知らない人は是非調べてみてね。

3.6　終わりに

あのですね、時間なくて落としそうです。助けてください (締め切り当日 3:26)(今 4:13)[*2]

MMA では競プロは活発になりつつあるので人がほしいです、是非お待ちしています。部室で話しながらワイワイ競プロの話をしたり解いたりしましょう！

未経験者こそ競プロが苦じゃなければ続けることで、ゲームやアプリを作る基礎体力づくりが自動でできるので強烈におすすめしておきます。初心者の方は一緒に僕とプログラミング頑張りませんか？？？

[*2] 短期間ゲーム制作は落としました。みんなにプレーできるゲームを見せられなくてごめんね

百萬石 2023 -春- © 電気通信大学 MMA

2023 年 4 月 6 日　初版第一刷発行　【本書の無断転載を禁ず】

著　者　gae, Udon, ryota2357, terry
表　紙　Udon
編集者　Udon
発行者　電気通信大学 MMA
発行所　電気通信大学 MMA 部室
〒182−8585 東京都調布市調布ヶ丘 1−5−1 サークル会館 208 号室
https://www.mma.club.uec.ac.jp/
印刷所　電気通信大学 MMA 部室
製本所　電気通信大学 MMA 部室

電 気 通 信 大 学