KSS PC BOOK 2022

2021-05-xx 版　　KSS PC Club 発行

目次

はじめに

このたびは本書をお手に取っていただきありがとうございます。本書は令和 4 年度けやき祭のためにパソコン部の有志によって作成された部誌です。

古河中等教育学校パソコン部 (通称：KSS PC Club) では部員それぞれが主に好きなことをし、競技プログラミングや Web 開発、ゲーム開発など様々なことに取り組んでいますが、本書では部員の興味を持っていることや語りたいことなどについて記事を自由に執筆してもらい、一冊の本に仕上げました。

新型コロナウイルスの影響により、文化祭でパソコン部が作品等を展示するのは 2019 年以来の 3 年ぶりであり、久しぶりの作品展示の機会であるのでぜひ特別棟2階奥のパソコン室にもお立ち寄りください。

部誌の制作については今年度からの初の試みであり、自分の言葉で物事を発信する場を設けるという意義があります。このような発信活動の場のひとつである本書を通して、読者のみなさまにも創作・技術に触れる楽しさ、好きなことに接する楽しさを感じてもらえると幸いです。

部長 細島涼雅

お問い合わせ先

本書に関するお問い合わせは部員まで。

免責事項

本書に記載された内容は、情報の提供のみを目的としています。したがって、本書を用いた開発、製作、運用は、必ずご自身の責任及び判断の上で行ってください。これらの情報による開発等の結果について、著者はいかなる責任も負いません。

この部誌を支える技術

Ryoga.exe

はじめに

こんにちは、5 期生の Ryoga.exe です。この部誌は今年からの初の試みということもあり折角なので (？) 今回はこの部誌を支える技術についてお話しようかと思います。この部誌については GitHub 上で管理されているため、そのリポジトリも合わせてご覧ください。 [1]

Vivliostyle について

本書は HTML & CSS で組版ができる `Vivliostyle` [2] の `Create Book` [3] というものを使って書かれています。

▲図1: Vivliostyle のロゴ

Vivliostyle は CSS 組版というWeb標準技術をベースにした、自動組版システムのオープンソースプロジェクトです。

CSS組版はHTMLとCSSをベースにしているので、普段から HTML/CSS を扱っている人にとってはVivliostyle のプロダクトが比較的手軽に感じるでしょう。

Vivliostyle を選択した理由

まず、複数人で書くことが前提となっていたので原稿等を共有する必要があります。また、見た目などもある程度統一させたいです。そのため、Word 等のソフトウェアは真っ先に候補から外れました。

---

[1] https://github.com/kss-pc-club/book-2022

[2] https://vivliostyle.org/ja/

[3] https://docs.vivliostyle.org/#/ja/create-book

そして、本を書く上で新たに多くの記法などを覚えるのは大変かつ、脳のリソースの無駄になります。そこで、Markdown で書けて、スタイルをコードで整えることができ、GitHub 上で管理できるといったことが必要でした。

そのため、Vivliostyle を選択しました。(日本語のドキュメントが充実しているという理由もありました。)

VFM について

Create Book がサポートする Markdown 方言は、Vivliostyle Flavored Markdown (VFM) です。詳細は公式ドキュメント [4] を参照してください。

今回はこの VFM でサポートされている記法について、ざっくりとご紹介します。

```
---
title: セロ弾きのゴーシュ
---

# セロ弾きのゴーシュ

**宮沢賢治**[^1]

ゴーシュは町の活動写真館でセロを弾く係りでした。
けれどもあんまり上手でないという評判でした。
上手でないどころではなく実は仲間の楽手のなかではいちばん下手でしたから、
いつでも楽長にいじめられるのでした。
ひるすぎみんなは楽屋に円くならんで今度の町の音楽会へ出す
第六{交響曲|こうきょうきょく}の練習をしていました。

![挿絵]('image/fig334.png'){width=300}

[^1]:
https://ja.wikipedia.org/wiki/%E5%AE%AE%E6%B2%A2%E8%B3%A2%E6%B2%BB
```

基本的には GitHub でサポートされている Markdown の方言である GFM (GitHub Flavored Markdown) の記法は使えます。しかし、上のサンプルにはなにか見慣れない記法がありますね。それぞれ説明します。

[4] https://vivliostyle.github.io/vfm/#/ja/vfm

ルビ

`{親文字|ヨミ}` とすることでルビが振れます。便利(べんり)。~~悔(ミ)やむと書いてミライ(ミライ)と読ませることもできますね！~~

脚注 (後注)

文章内に `[^n]` を置きその文の後の箇所で `[^n]: hoge` とすると脚注をつけることができます。ちなみに、`@vivliostyle/theme-techbook` のテーマパッケージを使用している場合は `<span class="footnote">hoge</span>` とすると本文 (ページ) の下に注記をつけることができます。

余談ですが、脚注をつける記法は実は GFM で同様にサポートされていたりします。

Sass を使う

この節では Sass と一緒にビルド/プレビューする方法をご紹介します。CSS を使ってスタイルを当てるのですが CSS よりも Sass の方がいろいろと楽だったりします。

Create Book でプレビューするとき、直接 .scss ファイルを読み込めないため、スタイルを調整しながらプレビューするといったことが困難です。

これを解決するために、`npm-run-all` を使います。`run-p` というコマンドがあり、パラレル実行ができるためこれを活用します。npm や yarn でインストールして

```
$ yarn add --dev npm-run-all
```

package.json の script に以下を追加します。

```
"start": "run-p preview watch:scss",
"preview": "vivliostyle preview",
"watch:scss": "sass --watch scss:css"
```

これで `yarn start` とすると SCSS のビルドとプレビューが同時にできます。便利。

GitHub Actions でビルドする

この部誌は GitHub 上で管理されており、各部員がブランチを切り、PR を出す…といった感じで書かれています。

しかし、現状どのような見た目になっているかを確認できない部員もいたりします。そのため、GitHub Actions を活用してビルドし、自動で `publish` ブランチに push されるような仕組みにすることにしました。以下のワークフローを作成するとできます。

.github/workflows/build.yml

```yaml
name: Build
on:
  workflow_dispatch:

jobs:
  build:
    name: Build
    runs-on: ubuntu-latest
    env:
      TZ: Asia/Tokyo
      GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
    steps:
      - name: Checkout
        uses: actions/checkout@v2.3.4
        with:
          ref: main
      - name: Setup Node
        uses: actions/setup-node@v2.1.5
        with:
          node-version: '14.16.0'
      - name: Get yarn cache directory path
        id: yarn-cache-dir-path
        run: echo "::set-output name=dir::$(yarn cache dir)"
      - name: Cache deps
        uses: actions/cache@v2.1.6
        with:
          path: ${{ steps.yarn-cache-dir-path.outputs.dir }}
          key: ${{ runner.os }}-yarn-${{ hashFiles('yarn.lock') }}
          restore-keys: |
            ${{ runner.os }}-yarn-
      - name: Install deps
        run: yarn install --frozen-lockfile
      - name: Install ghostscript
        run: |
          sudo apt-get -yqq install libgbm1 ghostscript
          sudo apt install poppler-utils poppler-data
      - name: Build
        run: yarn build
      - name: Build Press-Ready
        run: yarn press-ready
      - name: Deploy
        uses: s0/git-publish-subdir-action@develop
```

```
        env:
          REPO: self
          BRANCH: publish
          FOLDER: public
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

`workflow_dispatch` ではなく main ブランチ等への push をトリガーにしたほうが楽かもしれないですね。

この部誌をビルドする

前述したとおりこの部誌は GitHub のリポジトリで管理されているのでそれを clone してきましょう。

```
$ git clone git@github.com:kss-pc-club/book-2022.git
```

依存関係をインストールし、`yarn build` とするとビルドができます。

```
$ yarn install
$ yarn build
```

`public/book.pdf` が作られます。

入稿用のデータを `yarn press-ready` というコマンドで作ることができますが、ghostscript と poppler-utils のインストールが必要なので注意してください。

おわりに

ある程度しっかりとした本を既に知っている技術で作れました。

CSS 組版で本を作りたい！Markdown で本を書きたい！...そんな方はぜひ Vivliostyle で始めてみませんか？

ICPCへのお誘い

Asa

はじめに

こんにちは、第3期生の土屋です。ネットでは「Asa (@a01sa01to)」として活動しています [1] 。2021年3月に古河中等教育学校を卒業し、現在は埼玉大学工学部情報工学科に所属しています。この度、細島部長にお声がけいただき、このような形で部誌の編集に携わることとなりました。

技術系（特にプログラミング関連）の記事がほとんどである、この部誌を読んでくださっている方の中には、「競技プログラミングについて聞いたことがある / すでに参加している」という方がいると思います。そこで、国際大学対抗プログラミングコンテスト（ICPC）に関する記事を書きたいと思います。

ICPCってなに？

国際大学対抗プログラミングコンテスト（International Collegiate Programming Contest・通称ICPC） [2] は、同じ大学の3人で1チームを作り、プログラミングの問題を解く大会です。中学生・高校生の皆さんは、「国際情報オリンピックのチーム参加版」と捉えていただけるとわかりやすいかもしれません。

▲図1: ICPCのロゴ（http://icpc.foundation/ より引用）

[1] ~~ここまでの情報で私のフルネームと誕生日がわかるらしいですよ~~

[2] https://icpc.global/（日本語版: https://icpc.iisf.or.jp/ ）

この大会には世界各国から毎年3万人以上が参加しており、日本からも様々な大学から参加しています。2021年度、私の所属する埼玉大学からも2チーム出場し、私もそのうちの1チームに参加していました [3] 。

大会の流れとしては、「国内予選→アジア地区大会→世界大会」といった流れです。情オリに似ていますね。国内予選では3時間で6-7問、アジア地区大会では5時間で10問程度の問題を解きます。情オリとは違い、得点ではなく、解いた問題数で競います。また、アジア地区大会で利用できるコンピュータもチームで1台と制限があります [4] 。そのため、1人が解いている間にほかの人がアイデアを出すなどといったチームワークが重要になります。さらに知りたい方は、ICPC日本公式団体に掲載されている「3分でわかるICPC」（ https://icpc.iisf.or.jp/acm-icpc/3min/ ）も併せてご覧ください。

▲図2: 2019年度アジア地区大会の様子（https://youtu.be/MwD254vH3Pw）

どんな問題を解くの？

「○○くらいの難易度です！」と言っても個人差があると思うので、実際に出された問題を見てみましょう。

[3] ちなみに私のチームの結果は、予選43位、アジア地区大会11位でした。もう1チームは予選104位で、アジア地区大会に進出できませんでした。

[4] 2020年度・2021年度はオンラインだったため、1人1台でした。なお国内予選では、コードではなく、自分のコンピュータで出た答えを提出します。~~時間制限は実質無限です。~~

2021年度 国内予選 A問題

https://icpc.iisf.or.jp/past-icpc/domestic2021/contest/all_ja.html#section_A より、問題文の概要を書いてみます。

4つのお椀とその中に入ったビー玉がある。1つもビー玉が入っていないお椀があることがあるが、少なくともどれか1つのお椀にはビー玉が入っていることが保証されている。

4つのお椀に対して、空でないお椀が1つだけになるまで、以下の操作を繰り返す。

1. 空でないお椀のうち、ビー玉が少ないものを選ぶ。2つ以上あるときは、一番左のお椀を選ぶ。
2. 選んだお椀以外の、空でないすべてのお椀から、選んだお椀と同じ数のビー玉を取り除く。ただし、選んだお椀のビー玉はそのままにしておく。

さて、最終的にお椀に残ったビー玉の個数はいくつ？

制約

- それぞれのビー玉の個数は、0個以上100個以下。
- 少なくとも1つは0ではない。
- データセットは100個以内。
- `0 0 0 0` は終了の合図。

サンプルの入出力を見てみましょう。

```
10 8 4 6
0 21 7 35
5 45 13 3
52 13 91 78
0 0 0 0 （←入力終了の合図）
```

```
2
7
1
13
```

さて、この問題をあなたならどう解きますか？

まず思いつくのは、単純にシミュレーションしてみることです。

具体的には、ビー玉の残ったお椀が1以下ではない限り、次の手順を繰り返します。

1. お椀を、ビー玉の個数でソートする
2. 一番少ないお椀を選び、そのビー玉の個数を他から減らす
3. ビー玉が0以下になったお椀を削除する

このような方法でも、計算量は高々 $O(T\sum a_i)$ (10^4 くらい) なので、高速です [5] 。なお、データセット数を T としています。

ans.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
  while (true) {
    vector<int> a(4);
    for (int i = 0; i < 4; ++i) cin >> a[i];

    if (a[0] + a[1] + a[2] + a[3] == 0) break;

    while (a.size() > 1) {
      sort(a.begin(), a.end());

      for (int i = 1; i < a.size(); ++i) a[i] -= a[0];

      auto itr = remove_if(a.begin(), a.end(), [](int x) {
        return x <= 0;
      });
      a.erase(itr, a.end());
    }
    cout << a[0] << endl;
  }
  return 0;
}
```

ICPCでは、このような単純な繰り返し処理や条件分岐などができれば、チームに貢献できます。実際、当時 AtCoder茶色 の私でも、解くことができました。

参考までに、さらに高速な方法をご紹介します。それは、最大公約数を用いる方法です。

実は、それぞれのデータセットの答えは、$a_1, \ldots, a_4$ の最大公約数になることが証明できます [6] 。

---

[5] 各手順で少なくとも1つは取り除かれるので、最大でビー玉の個数分の手順しか行われません。

[6] https://icpc.iisf.or.jp/past-icpc/domestic2021/commentaries.html#A

これを用いると、$O(T \log(\min a_i))$ で計算できます。

ans.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
  while (true) {
    int a, b, c, d;
    cin >> a >> b >> c >> d;

    if (a + b + c + d == 0) break;

    int g = a;
    g = gcd(g, b); g = gcd(g, c); g = gcd(g, d);

    cout << g << endl;
  }
  return 0;
}
```

2020年度 アジア地区大会 A問題

本当は2021年度の問題を載せようと思ったのですが、まだ公開されていませんでした...。（何かの体積を計算させる問題だったと記憶しています）

そこで、2020年度の問題を載せます！ [7]

> $N \times N \times N$ の立方体に収まるある立体を、$x-y, y-z, z-x$ 平面それぞれに射影した図が与えられます。与えられた図に適する図形が存在するか判定してください。
>
> 制約: $1 \leq N \leq 100$

解法は、$N \times N \times N$ すべてが埋まった立方体から、「削って」いき、最終的に出来上がった立体が、条件が満たしているかを確認する方法です。

アジア地区大会では、自分のコンピュータではなく、ジャッジ用コンピュータで正誤判定されるため、時間制限が定められています。この問題では2秒ですが、以上の解法は $O(N^3)$ なので、ACになります。

---

[7] https://icpc.iisf.or.jp/past-icpc/regional2020/problems-2020.pdf すべて英語です。本番では、機械翻訳は許されません（辞書はOK）。

実装は大変ですが、これも問題なく解けると思います。

ans.cpp

```
#include <bits/stdc++.h>
using namespace std;
#define rep(i, n) for (int i = 0; i < (n); ++i)

int main() {
  int n;
  cin >> n;
  vector yz(n, vector<bool>(n)), zx(n, vector<bool>(n)), xy(n,
vector<bool>(n));
  rep(i, n) {
    string s; cin >> s;
    rep(j, n) yz[j][n - i - 1] = (s[j] == '1');
  }
  rep(i, n) {
    string s; cin >> s;
    rep(j, n) zx[j][n - i - 1] = (s[j] == '1');
  }
  rep(i, n) {
    string s; cin >> s;
    rep(j, n) xy[j][n - i - 1] = (s[j] == '1');
  }

  vector ans(n, vector(n, vector<bool>(n, true)));
  rep(i, n) rep(j, n) if (!yz[i][j]) rep(k, n) ans[k][i][j] = false;
  rep(i, n) rep(j, n) if (!zx[i][j]) rep(k, n) ans[j][k][i] = false;
  rep(i, n) rep(j, n) if (!xy[i][j]) rep(k, n) ans[i][j][k] = false;

  rep(i, n) rep(j, n) {
    if (yz[i][j]) {
      bool chk = false;
      rep(k, n) if (ans[k][i][j]) chk = true;
      if (!chk) { puts("No"); return 0; }
    }
  }
  rep(i, n) rep(j, n) {
    if (zx[i][j]) {
      bool chk = false;
      rep(k, n) if (ans[j][k][i]) chk = true;
      if (!chk) { puts("No"); return 0; }
    }
  }
  rep(i, n) rep(j, n) {
    if (xy[i][j]) {
      bool chk = false;
      rep(k, n) if (ans[i][j][k]) chk = true;
```

```
      if (!chk) { puts("No"); return 0; }
    }
  }
  puts("Yes");
  return 0;
}
```

最後に

ICPCの問題を実際に見てみましたが、「こんなの解けるわけがない！」というものではなかったと思います。むしろ、少し考えたうえで、プログラミングの初歩である「繰り返し処理」「条件分岐」を使いこなすだけで、最初の問題は解けます。

競技プログラミングをするうえで、ICPCは一生に5回程度しか参加できないため、貴重な体験になることは間違いないです。そしてICPCは、プログラミング力ではなく、チームワークが鍵です。チーム戦で生まれる連帯感というのは、なかなか体験できないと思います。

この記事を通じて、少しでも興味を持っていただけたのであれば幸いです。大学に入学した際には、ぜひともICPCに参加してみてください！！！！ [8]

最後までお読みいただき、ありがとうございました！

[8] 参加資格のない方は...順位表実況や過去問などでも楽しめます！

KSS PC BOOK 2022

2022 年 5 月 x 日　初版発行
著　者　KSS PC Club
印刷所　xxxxx