```
/* Forward declaration */
static int hugetlb_acct_memory(struct hstate *h, long delta);
static void hugetlb_vma_lock_free(struct vm_area_struct *vma);
static void hugetlb_vma_lock_alloc(struct vm_area_struct *vma);
static void hugetlb_vma_unlock_write_free(struct vm_area_struct *vma);
static void hugetlb_unshare_pmds(struct vm_area_struct *vma,
                unsigned long start, unsigned long end);

static inline bool subpool_is_free(struct hugepage_subpool *spool)
{
        if (spool->count)
                return false;
        if (spool->max_hpages != -1)
                return spool->used_hpages == 0;
        if (spool->min_hpages != -1)
                return spool->rsv_hpages == spool->min_hpages;

        return true;
}

static inline void unlock_or_release_subpool(struct hugepage_subpool *spool,
                                             unsigned long irq_flags)
{
        spin_unlock_irqrestore(&spool->lock, irq_flags);

        /* If no pages are used, and no other handles to the subpool
         * remain, give up any reservations based on minimum size and
         * free the subpool */
        if (subpool_is_free(spool)) {
                if (spool->min_hpages != -1)
                        hugetlb_acct_memory(spool->hstate,
                                            -spool->min_hpages);
                kfree(spool);
        }
}

struct hugepage_subpool *hugepage_new_subpool(struct hstate *h, long max_hpages,
                                              long min_hpages)
{
        struct hugepage_subpool *spool;

        spool = kzalloc(sizeof(*spool), GFP_KERNEL);
        if (!spool)
                return NULL;

        spin_lock_init(&spool->lock);
        spool->count = 1;
        spool->max_hpages = max_hpages;
        spool->hstate = h;
        spool->min_hpages = min_hpages;

        if (min_hpages != -1 && hugetlb_acct_memory(h, min_hpages)) {
                kfree(spool);
                return NULL;
        }
        spool->rsv_hpages = min_hpages;

        return spool;
}

void hugepage_put_subpool(struct hugepage_subpool *spool)
{
        unsigned long flags;

        spin_lock_irqsave(&spool->lock, flags);
        BUG_ON(!spool->count);
        spool->count--;
        unlock_or_release_subpool(spool, flags);
}
```

はじめに

まずは、3年生の皆さん、ご卒業おめでとうございます。私自身は一年生のため、関わる機会があまりありませんでしたが、さまざまな活動を精力的になさっていたと聞いています。これからの人生を存分に楽しみ、夢に向かって頑張っていってください！

今回は予餞会号ということで4人の部員が記事を書きました。どの記事も魅力的な内容になっていますので、どうぞ最後まで楽しんでお読みいただければ幸いです。

目次

EDMのすゝめ

Loy3r

「最近、PCに触ってない」と思っていたら、予餞会号の季節が近づいてきてしまいました。この Word を開く前に Spotify で NCS(NoCopyrightSounds)を聞いていなかったらこの記事はできていないでしょう。

EDM(Electronic Dance Music)は、電子音でできた音楽のことで、クラブで流れる曲のイメージです。この記事では「すゝめ」とある通り、自分が EDM を推す理由、推しの曲を記していきます。

結論を先に述べておくと、自分は EDM(あとは Minecraft の PvP)で英語を学んできました。

1. EDM を推す理由

EDM のいいところ、それは英語が学べるところです。出身国にかかわらず、大体の EDM の作曲者(以後 EDMer)は英語の歌詞の曲を作ります。

DJ の曲といえば、歌詞のない音楽を思い浮かべる人も多いかもしれないですが、有名な EDMer である Avicii に見て取れるように、ドロップ(サビ)まで歌詞あって、ドロップで歌詞をなくし一気にアゲるパターンが多いです。

実は結構身近で聞いていたりします。先述した NCS は登録者が 2200 万人の Youtube チャンネルを持っていて、MonsterCat Uncaged は登録者 775 万人です。実は一回だけ NCS に投稿されている曲が昼休みに放送で流れていました。

Remix も盛んです。ただし日本と違い、Remix というのは cover とは異なります。Nightcore と呼ばれる、速さを変えたりピッチ変えたりして投稿するのも Remix の一種でしょう。

Avicii (1989-2018)
スウェーデン出身
有名な曲は
・Wake me up
・The Nights
・Waiting for Love

ぜひ聞いてください

2. 推しの EDM

せっかくなので、NCS と Monstercat、Nightcore、あと個人のチャンネルで出されている曲から選んできました。

2.1 NCS

NCS は 2015 年の黄金期を過ぎてからここ最近、再生数も低迷しています。
さらに NCS で「Fade」や「Spectre」などの曲を出した Alan Walker が個人チャンネルに移動。今個人チャンネル 4290 万人登録者がいます。

そんな NCS の推し。
ただし曲ではなく EDMer。

NIVIRO です。

NCS を聞いている人の中ではそこそこ知られています。
NIVIRO 推しの理由はドロップ。
NIVIRO は House 系(ドンドンと一定間隔でなってる系)と bass（重厚な音系)。ジャンル分けは知らなくても OK です。

曲の推しは「The edge」
Nightcore もおすすめです。

Alan Walker(1997-)
ノルウェー出身
有名な曲は
・Faded
・The Spectre
・Alone

2.2 MonsterCat

再生回数はこちらも不調
重厚な音が好きな方は NCS よりこっちを漁る方がいい曲が見つかるかもしれません。私は House 派なので Monstercat はあんまり。

推しの曲は
「Everything Black」
-Unlike Pluto

ドロップ最強すぎます。以上。これ以上語ることもないです。

2.3 その他

EDMer はあちこちに曲を出しています。

今回推し曲を作った Vicetone も Monstercat から投稿した Nevada が人気を出したと思ったら、WaveMusic からも Nevada を投稿しています。
Vicetone の曲も大体好きですが、今回はこの曲。

「Angels」

普通にカラオケに欲しいです。ドロップはもちろんのこと、その前の歌詞がある部分も、手拍子の使い方が神ってます。

2.4 NightCore

NightCore は Syrex を推しています。何せ歌詞が大体ついているんです。(イラスト可愛い)

推し曲は「Ghosts」
検索かけるときは Nightcore Ghosts と入力してください。

3. まとめ

EDM はすごいんです。

4.あとがき

受験期お疲れさまでした。今回明るく音楽テーマの話でした。PC 室来る機会ってそこまで多くなかったのではないのでしょうか。そんな学校の端っこの今 7 人しかいない部活ですが 110 号出せました。

最後になりましたが、卒業おめでとうございます。

県船アカウントのデータを残したい！

著：バター

3 年生の皆さん、ご卒業おめでとうございます。思い出いっぱいの県船から飛び立つ準備の最中かと思います。ここで忘れていただきたくないのが「県船アカウントのデータの引継ぎ」です。3 年間ともに歩んできた学校用アカウントも卒業と同時に無効化されます。アクセスできなくなってしまう前にきちんと引き継いで心残りをなくしましょう。

引継ぎ時は移行先アカウントを用意する必要があります。すでに持っている中から 1 つ決めるか、新しく作成してください。画面の UI の関係上、基本的にはパソコンで作業してください。また、シークレットウィンドウを活用してログインセッションを分離すると、同時並行で作業でき、事故も少ないのでお勧めです。なお、下に載っていないサービスは未調査か、手動でのコピーが必要です。

各サービスの移行方法

Google コンタクト

Google コンタクトにアクセスし、連絡先をファイルに書き出します。県船アカウントで画面左の「エクスポート」を開き、エクスポート対象を選んでから「Google CSV」形式でエクスポートを実行します。

ダウンロードが完了したら、移行先アカウントで画面左の「インポート」を選び、先ほどダウンロードしたファイルを指定して処理を実行します。自動で「インポート：（日付）」というラベルが挿入されますが、不必要であれば削除しましょう。

Gmail

移行先アカウントからの操作で完結します。移行先アカウントで Gmail を開き、設定＞すべての設定を表示＞アカウントとインポート＞メッセージと連絡先のインポートまで進みます。県船アカウントのアドレスを入力した後、指示に従っ

て県船アカウントにログインして情報の取得を許可します。あとは、インポートするものを選べば自動でメールがコピーされます。ここで Google コンタクトのデータをコピーすることも可能です。また、以後 30 日間転送し続けるように設定することもできます（1 日 1 回まとめて転送されるのでタイムラグが生じます）。すべて完了したら画面は閉じて結構です。

Google Chrome

まずは Chrome をインストールしてください。その後、一度県船アカウントで Chrome にログインし、同期したすべてのデータを端末上に取得します。そのあと Chrome 上で県船アカウントからログアウトします。Chrome の設定から自分のアカウント名を選択し、「ログアウトして同期をオフにする」を押します。この時、端末にデータを残すか否か選択させられますが必ず残してください。

再度 Chrome を開き、今度は移行先アカウントでログインします。この時、データの取り扱いを聞かれますが、端末に残ったデータを移行先アカウントに上書き保存するよう指定してください。これで完了です。手順を間違えると正しく移行できないので注意しましょう。

Google カレンダー

移行可能なのは自身が管理者権限を持っているカレンダーのみです。Classroom のカレンダー等は移行できないのでご注意ください。また、県船アカウントでは iCal 形式の非公開 URL は発行できません。

県船アカウントの Google カレンダーにアクセスし、上部メニュー内の設定アイコンから設定＞インポート/エクスポートに移ります。その中の「エクスポート」からすべてのカレンダーをまとめて書き出すことができます。

終わったら移行先アカウントに画面を切り替え、同様に設定＞インポート/エクスポート＞インポートまで移動し、自分が今持っているカレンダーの 1 つをインポート先に指定して実行してください。現存のカレンダーとは分けて保存したい場合は設定＞カレンダーを追加＞新しいカレンダーを作成まで移動し、適当な名前でカレンダーを必要数作ってから作業を行いましょう。

Google ドライブ

県船アカウントの共有ドライブを利用します。まず、県船アカウントの Google ドライブを開き、画面左側のメニューから「共有ドライブ」を選択します。その後、画面に従って共有ドライブを適当な名前で新規作成し、共有設定から移行先アカウントを管理者として追加します。この状態で移行先アカウントの Google ドライブを再読み込み(ないしは開く)と共有ドライブのタブが出現します。ここで、県船アカウントのデータを作成した共有ドライブ内に移動すると、ファイルのオーナー権限が「千葉県立船橋高等学校」に強制的に移行します。

最後に、移行先アカウントに画面を切り替えて作成した共有ドライブ内のデータを「マイドライブ」に移動すれば、今度はファイルのオーナー権限が移行先アカウントに移行します。これで完了です。

Google Classroom

悲しいかな。Google Classroom のデータはどうやら書き出しが制限されているようです。思い出の写真や投稿があれば、個別にダウンロードしたり控えをとったりしておきましょう。また、特権管理者であれば対応可能な場合があるので、どうしてもというときは先生に相談しましょう。

他サービスのログイン情報

他のネットワークサービスで県船アカウントのメールアドレスを登録している場合や、アカウントの連携を行っている場合は再登録が必要です。

県船アカウントのメールアドレスを登録しているサービスを確実に調べ上げる手段はありませんが、Gmail に届いている登録確認メール等が参考になるかもしれません。また、現在連携しているサービスは、以下のURLにアクセスし、県船アカウントにログインすればすぐにわかるので、適宜再登録やアカウントの整理をしましょう。https://myaccount.google.com/permissions

まとめ

いかがでしたか？サービスごとの要点を抑えて作業をすれば、3 年間で蓄えたデータを取り出して手元に残すことができることがお分かりいただけたかと思います。この記事がその作業の一助となれば幸いです。

卒業に寄せて

３年生の皆さん、改めてご卒業おめでとうございます。末筆ながら、先輩方に送辞の挨拶をさせていただければと思います。月並みな言葉ですが、最後までお読みいただけると幸いです。

皆さんは、大掃除をしたことはありますか？年の瀬や引っ越し等のタイミングで一度は経験したことがあるかと思います。思い出の品を発掘しては回想に耽ってなかなか作業が進まないのは、大掃除のあるあるですよね。

県船アカウントのデータを移行する際、久しく開いていなかったファイルが沢山画面に表示されるでしょう。行事の写真、先生が出した課題、学校からのお知らせ……それらが皆さんの眼に入った瞬間、０と１の集合には到底収まらない、今までの学校生活で得たかけがえのない記憶を呼び起こされるはずです。

コンピュータが持つ記録は人間が持つ記憶のほんの一部でしかない＿＿このことは大変重要な事実を内包しています。それは、コンピュータは人々の生活を支えることは可能でも、皆さんを取り囲む世界すべてを置き換える存在にはなり得ないということです。これは、コンピュータが絶対に超えることができない限界であると私は確信しています。

３ 年間県船で培った経験はコンピュータでさえ奪えませんし、他の誰にも「コピー」されません。県船で出逢ったもの全部を武器にして、輝かしい未来へと旅立ってください。皆様のご活躍を心よりお祈りしております。

バター こと コンピュータ部部長
番場 拓海

ブレンダー、はじめました。

Flower

数独をアルゴリズムで解く

By Standard

方針

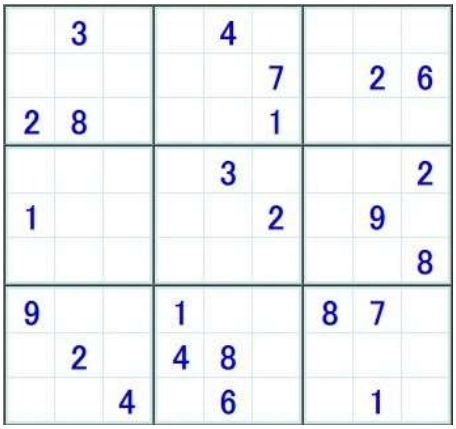


図 1

まず図 1 な問題が入力されたとして

左上の空白から順番に総当たりで数字を入れていく

左上には同じ行に 3, 4 同じ列に 1, 2, 9

同じブロックに 8 があることから 5, 6, 7 が入る。

よって、この中で一番小さい数字の 5 を入れる。

だがこんな適当な方法で最後まで行けるはずもなく

途中でどの数字も入らなくなってしまう(´・ω・`)

(図 2 この場合では列に 1,4 行に 2, 6, 7, 9

ボックスに 3, 5, 8 が既に入ってしまっている)

この場合は、直前の動作を Undo して 9 を消して次に大きい数字を入れることで処理を続ける

だがしかし、この場合は 9 より大きい数字を入れられないためまたもや 4 を消して 4 より大きい入れられる数字である 9 を入れる。

これを繰り返すことでどんな難しい問題も解ける(はず)

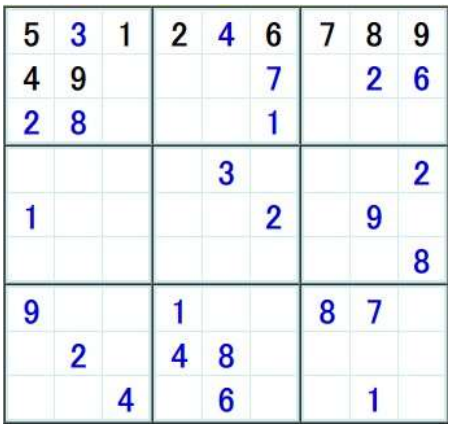


図 2

実装

ということで皆大好き我らが C++で実装していくぅ！

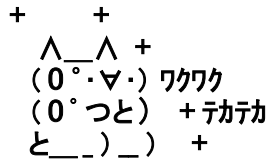
```cpp
#include <iostream>
#include <fstream>
#include <vector>
#include <cmath>
#include <set>
#include <chrono>
#include <iomanip>
using namespace std;

void print(string);
bool check(string, int);
void solve(string, int);

// 時間計測用変数 : 開始時間, 解答生成終了時間, 全探索終了時間
chrono::system_clock::time_point start, end_ans, end_search_all;

int main(int argc, char** argv) {
    string problem;
    cout << "plz input the problem" << endl; // ガバガバ英語
    cin >> problem;

    // 入力された問題を表示
    print(problem);
    cout << "¥n¥n" << flush;

    start = end_ans = chrono::system_clock::now();

    solve(problem, 0);
    cout << "finished!" << endl;

    end_search_all = chrono::system_clock::now();

    long double time_ans = static_cast<long double>(chrono::duration_cast<chrono::microseconds>(end_ans - start).count() / 1000.0);
    long double time_search_all = static_cast<long double>(chrono::duration_cast<chrono::microseconds>(end_search_all - start).count() / 1000.0);

    cout << std::fixed;
    cout  << "time_ans : " << std::setprecision(10) << time_ans << endl;
    cout << "time_search_all : " << std::setprecision(10) << time_search_all << endl;

    return 0;
}

// 文字列を整えて表示する
void print(string s) {
    if(start == end_ans) {
        end_ans = chrono::system_clock::now();
    }
```

```
48
49      for (int i = 0;i < s.length();i++) {
50          putchar(s[i]);
51          if (i % 27 == 26 && i != 80) {
52              cout << "¥n-----------¥n";
53          }
54          else if (i % 9 == 8) {
55              putchar('¥n');
56          }
57          else if (i % 3 == 2) {
58              putchar('|');
59          }
60      }
61      cout << flush;
62  }
63
64  // メインの解答生成処理
65  // 再帰を用いて全探索する
66  void solve(string s, int i) {
67      if (i == 81) {
68          print(s);
69          return;
70      }
71      else {
72          if (s[i] == '0') { // もし対象の文字が空なら当てはまる数字を探す
73              for (int j = 1;j < 10;j++) {
74                  string s_ = s.substr(0, i) + to_string(j) + s.substr(i + 1);
75                  if (check(s_, i)) { // もし一時的に追加した数字が正しいなら文字
    をーつすすめて再帰
76                      solve(s_, i + 1);
77                  }
78              }
79          }
80          else { // すでに埋まっていた場合は次に進む
81              solve(s, i + 1);
82          }
83      }
84  }
85
86  // 数字を追加した文字列が正しいかどうかを判定する関数
87  bool check(string s, int i) {
88      int temp = 0;
89      // 追加した数字の行、列、ボックス内に存在する数字の集合
90      set<char> there;
91
92      // 列チェック
93      temp = i % 9;
94      for (int i = 0;i < 9;i++) {
95          if (s[temp] >= '1' && s[temp] <= '9') {
96              if (there.count(s[temp])) {
97                  return false;
```

```
98              }
99              there.insert(s[temp]);
100         }
101         temp += 9;
102     }
103
104     // 行チェック
105     there.clear();
106     temp = i - i % 9;
107     for (int i = 0;i < 9;i++) {
108         if (s[temp] >= '1' && s[temp] <= '9') {
109             if (there.count(s[temp])) {
110                 return false;
111             }
112             there.insert(s[temp]);
113         }
114         temp++;
115     }
116
117     // ボックスチェック
118     there.clear();
119     int list[9];
120     int Nc = i % 9;
121     int Nr = static_cast<int>(round((i - Nc) / 9.0));
122     int Ncs = Nc - Nc % 3;
123     int Nrs = Nr - Nr % 3;
124     int num = Ncs + Nrs * 9;
125
126     int count = 0;
127     for (int k = 0;k < 3;k++) {
128         for (int j = 0;j < 3;j++) {
129             list[count] = num;
130             count++;
131             num++;
132         }
133         num += 6;
134     }
135
136     for (int i = 0;i < 9;i++) {
137         if (s[list[i]] >= '1' && s[list[i]] <= '9') {
138             if (there.count(s[list[i]])) {
139                 return false;
140             }
141             there.insert(s[list[i]]);
142         }
143     }
144
145     return true;
146 }
147
```

実行

という訳で図１の問題を解いていきます。



動くことを祈る。

```
C:¥tmp>second_main.exe
plz input the problem
060040000408200950200830000000009507107000030340000200006050004000000090005006000
060|040|000
408|200|950
200|830|000
-----------
000|009|507
107|000|030
340|000|200
-----------
006|050|004
000|000|090
005|006|000


563|947|128
478|261|953
291|835|476
-----------
682|319|547
157|624|839
349|578|261
-----------
916|752|384
724|183|695
835|496|712
finished!
time_ans(ms) : 24.1870000000
time_search_all(ms) : 70.9550000000
```

はい、思ったより早かったですね。結果は解答生成までに 0.024 秒全探索でも 0.070 秒で解けたのでさすが C++というところ。

今回はメモリの使用量が 1GB とかなり多くなってしまったので次回あるとしたらもっときれいなコードを書きたいです。

C.P.U. VOL.111
2022年度予餞会号

2023/03

発行　　千葉県立船橋高校 コンピュータ部

FHCC
県立船橋高校コンピュータ部