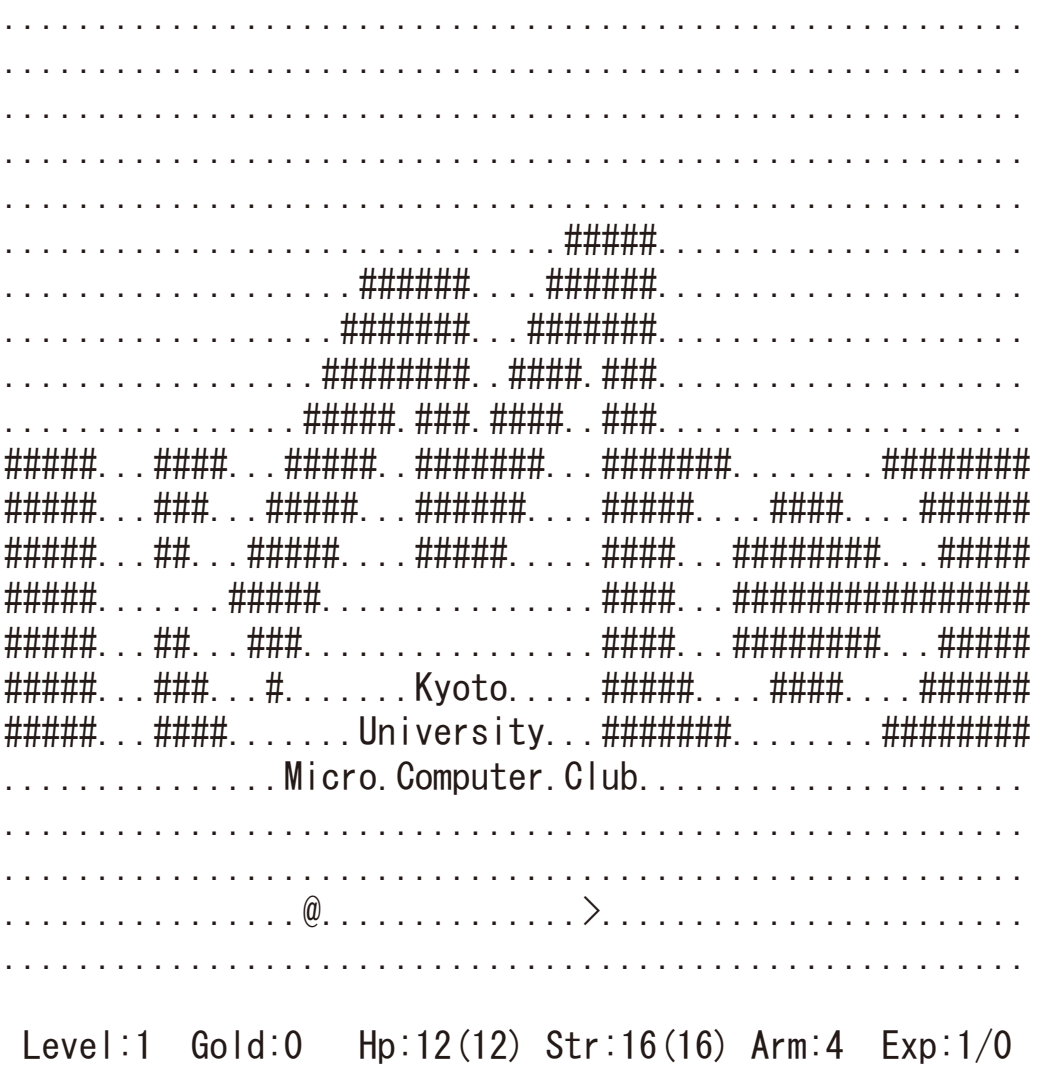
Teach Yourself KMC

独習KMC vol.6

京大マイコンクラブ 著



はじめに

ごあいさつ

こんにちは。京大マイコンクラブ 6 回生の Moko です。『独習 KMC vol.6』を手にとっていただきありがとうございます。

『独習 KMC』は、3 年前、私が編集長として創刊しました。というのもウン年前にも部誌が作られていたことを知り、「いいなあ、やってみたい」と言ったところ「じゃあおまえ編集長ね」となったのがきっかけです。第 1 号はただのコピー本だったのですが、2 代目編集長の seikichi 君によって複数の TeX ファイルから 1 つの PDF を生成するシステムが作られ、レイアウト作業が事実上不要となり、部誌作成が格段に楽になりました。

さらに印刷所に製本が依頼されるなど、どんどん良くなっていき、複数人から成る編集委員会が作られ、前回発刊した『vol.5』では、なんと、表紙がカラーというではありませんか！ ここしばらく朝から晩まで研究室にいるためそんなことも知らなかった元編集長、気づけば修了も近づき、ははぁ KMC はこうやって発展していくのだなあ……とぼんやりした感想を抱いていたところ、編集長の jf712 君に巻頭言を書かないかとお誘いいただきました。ありがとうございます。

振り返って

6 年前、同期が殆ど情報系の中、所属は農学部・得意な分野はグラフィック・おまけに女子部員……と、マイナーな存在として入部しました（そういえば当時は Mac を使うこともマイナーでした）。必然的に丸 3 年間、部内のグラフィック関連の作業のほとんどを担当するなかで、必要に駆られてバージョン管理システムの使い方・トンネリング・UNIX の操作・ドット絵・イラスト・デザイン・グッズ製作などを学びました。またイラスト勉強会では、“ デッサンの狂いなど、絵の改善点を論理的にわかりやすく説明する TA ” という困難な役割を仰せつかり、ずいぶんコミュニケーション力がついたように思います。

学部と KMC という 2 つの世界を同時に見てきた中で思うのは、このサークルは本当にさっぱりとしていて、非合理的なことを嫌い、好奇心が旺盛で、遊び好き、学ぶことが好きな人の集まりだということです。ですから女子部員だからといって特別扱いもありませんでした。女性のコミュニケーションでは特に重要視される気遣いや同調が苦手な私にとって、そしてコンピュータが好きと言うだけで友達に壁を作られていた私にとって、こうやって各自がそれぞれの方向を向いて好きなことに取り組んでいる KMC は、いつでも救われる場所でした。

最近のこと

最近 KMC の入部員数が急激に増えています。6 年前の私達 32 代は 10 人であるのに対し、37 代では 40 人を超えました。また 36 代では女子部員が 1 人、37 代では 2 人入部してくれました。それに伴い勉強会の内容はイラスト・DTM 関連も加わりぐっと多彩になりました。また度重なる掃除と改装によって、部室はとても清潔になり、長年形骸化していた台所がついに稼働しはじめるなど、雰囲気が大きく変化しました。いつのまにか部室を美化してくださった妖精さんに感謝いたします。

反面、人数が膨れ上がったせいで、部員間の連絡がうまくいかないことが問題になりつつあります。昔のように、少数の部員を全員 IRC に join させて、文字上で連絡を取り合い議論をするというのもなかなか難しくなってきました。……とここまで難しい顔で書いていたのですが、IRC を利用して部員ひとりひとりに ping を送れるシステムがすでに稼働しているとのことで、実に KMC らしい解決策が取られたものです。

最後に

このサークルはやる気次第で、大学の授業かそれ以上に多様なことを学ぶことのできる場所です。それこそがこのサークルの魅力であり、人が増えようとシステムが変化しようと、この点だけはきっと変わらず在り続けると思っています。こう書くと何かお堅いようですが、（私の）実情は IRC で奇声を上げたりトルエンかけるぞといいながら絵を描いたり、ssh したり、アニメを観ながら酒を飲んだりだべったりしているだけでした。文字にしてみたら非常にアレな感じになってしまいました。どうしよう。うーん。

えー、みんながそうやって、自由に遊んだり勉強したりした成果の詰まった『独習 KMC vol.6』楽しんでいただければ幸いです。

京大マイコンクラブ 6 回生 Moko

目次

部員求む！

最近の KMC

lastcat

はじめに

こんにちは、あるいはこんばんは。初めまして、もしくはお久しぶりです。KMC 第 36 代会長の lastcat です。と言っても、この記事が皆さんの手に届くころには私は第 36 代代表になっているでしょう。ちなみに代表は「カイチョウダッタヒト」と読みます [*1]。この記事の執筆がどうやら代表としての第一歩らしいので、力強く踏み出したいと思います。方向性が合っているかはわかりません。

8 月〜9 月（〜夏休み〜）

コミケが終わった私は、インターンシップという名の出稼ぎに出かけていました。今年は KMC にいくつかの企業さんからお誘いのお話が来たので、金銭に飢えた大勢の部員が野に放たれました。みんなそれぞれの思い出と給料を持ちかえったようです。京都に在留している部員は、立命館コンピュータクラブさんが主催した関西情報系学生団体交流会 (KC3) に参加しました。有意義な交流ができたようで、どこにいても KMC 部員は輝くという事実が世に広く知らしめられましたね。コミケが終わった私は、インターンシップという名の出稼ぎに出かけていました。今年は KMC にいくつかの企業さんからお誘いのお話が来たので、金銭に飢えた部員が大勢野に放たれました。みんなそれぞれの思い出と給料を持ちかえっていたようです。京都に在留している部員は立命館コンピュータクラブさんが主宰した関西情報系学生団体交流会 (KC3) に参加していました。有意義な交流ができたようで、どこにいても KMC 部員は輝くという事実が世に広く知らしめられましたね。

10 月〜11 月

新学期が始まると同時に KMC は祭りの気配に色めき立ちます。京都大学の学祭（以下 NF）で自作ゲームを楽しんでもらうために色々なプロジェクトが動き出します。この頃は部員はまだ元気です。そして 11 月。紅葉が色づき始めるころ、一部の部員の顔は青く染まり始めます。今年は一部

[*1] KMC では会長が翌年の代表を務めるシステムになっています。

の界隈で「進捗どうですか」というおよそ誰も幸せにならない言葉がブームになったせいで、例年より互いにかける重圧がすごかったように思います。この部誌には一部のゲームの開発記録記事もありますので、そちらを読んでいただければどんなノリで開発しているかが伝わるかもしれません。部員の粉骨砕身の努力の甲斐あって、今年も NF の KMC ブースは盛況でした。
~~間違って作りすぎてしまった~~ゲーム CD もほとんどさばくことができ、また準備・後始末においても大きな問題が起きることなく祭りを終えることができました。NF の各役職には 1 回生が就くことになっているのですが、初めてのサークル的タスクも先達の力を頼ったり互いに進捗をつつきあったりしながらよく進めてくれていたように思います。こうして人はだんだんと成長していくのですね。

12 月

一つの祭りが終わり、打ち上げたと思ったらもう年末の祭りが目の前に迫っています。新年代の役職も内定し、徐々に世代も移り変わっていきます。今年もいろいろな出来事があり、そのたびにそれぞれ骨を折った部員もいますが、いい糧にしてくれたと信じたいです。会長としてのこの 1 年を振り返った時、決していい会長ではなかったなぁと思いますが、どうにか次の世代につなぐことができそうです。これからは徐々に後ろから支えていくような役柄になっていけたらうれしいですね。
京都の厳しい寒さにたえるべく部室のぬくぬくこたつに転がりながら

KMC 第 36 代会長：lastcat

勉強会

DTM 練習会

okabi and maple

ごきげんようおひさしぶり。初めましての人は初めまして。KMC2 回生、DTM 担当の okabi と maple です。DTM とは「Desk Top Music」の略で、パソコンで音楽を作ることです。
前期に引き続き、DTM 練習会の皆は元気に活動しています。今期は、前期のような基本的な内容ではなく、一から作曲しています。11 月にあった学祭で出した CD に曲を入れたりもしました。最近ではガチ回と称して各々の曲を辛口に評価しあう回を設けました。気を遣わない意見というのは物を作るうえでは重要ですね。KMC のテーマ作りたいねーとか言いながら日々精進しています。またお耳にかかりましょう。

Coq 勉強会

asi1024

この勉強会では Coq という論理証明の正当性の根拠を導きだす手助けをしてくれるプログラミング言語の勉強をのんびりと行っています．勉強会では *Software Foundations* という教科書を用いており，文中の練習問題をみんなで解いて勉強を進めています．練習問題では，実際に様々な数学的対象や関数などを定義したり，与えられた命題を証明したり，あるいは参加者が示した命題をみんなで証明したりしています．
参加者は 4 人です．私以外の 3 人は理学部（もしくは理学研究科）の人たちで，毎週恐ろしい思いをしていますが，それでも 3 人に負けないようにこれからも頑張っていこうと思います．現在は Poly の章を読んでいます．

プログラミングコンテスト練習会

asi1024

この練習会では，プログラムの設計と実装の能力を競う「プログラミングコンテスト」に強くなることを目的として，各種アルゴリズムを学びプログラムを素早く正確に書き上げるための練習を

行っています.
まず，最初の 30 分で「アルゴリズムコンテストチャレンジブック」という本を読みアルゴリズムやプログラムの設計について学び，次にジャッジシステムを用いて 90 分ほど練習問題を解き，最後に 30 分でそれらの解説を行うという週 1 回 150 分のスケジュールで行っております.
参加者は 10 人前後です．今年は去年と大きく方針を変えてアルゴリズムを重視するようにしたため，例年と比べかなりハードな内容を行っていますが，それでも参加者の皆さんは諦めずに練習を重ね実力をつけてきてくれていて，プロジェクト担当者としては嬉しい限りです.
最近は，実際に公で開催されているプログラミングコンテストに参加したりもしています．これからも参加者の皆さんと一緒に楽しく練習を続けていけたらいいなと思っています.

ラテン語勉強会

dama

この勉強会は，京都大学の全学共通科目「ラテン語 A」（前期）及び「ラテン語 B」（後期）の予習を協力して行うことを主な目的として，『新ラテン文法』を読みながら練習問題を解いていく勉強会です.
そもそも，何故このような勉強会が発足したのかというと，予習量が異常に多いラテン語の講義をわざわざ取るような奇特な KMC 部員が，同じ年度に 3 人もいたからです．具体的にどのくらい予習が大変かというと，なんとなく練習問題を解いているだけでも，講義 1 回あたり 3〜5 時間かかるくらい大変なのです.
さて，ラテン語についてよく知らない方も多いと思いますので，ここで一度ラテン語について簡単に説明しておきましょう.
歴史的な説明をすると，ラテン語は元々イタリア半島中部のラティウム地方でラテン人に用いられていた少数言語でしたが，古代ローマ帝国の公用語になったことにより，地中海全域を含む広範囲で用いられることとなりました．イタリア語・ルーマニア語・スペイン語・フランス語・ポルトガル語などの所謂ロマンス諸語は，このラテン語の方言のようなものです．文法・語彙においては，ドイツ語・オランダ語・英語などのゲルマン諸語にも多大な影響を与えています．また，ローマ字はラテン・アルファベットとも言うように，元々ラテン語で用いられていたものです.
しかし，今ではラテン語は死語だとよく言われます．確かに，ラテン語を公用語としているのはバチカン市国のみであり，しかも用いられるのは公式会見くらいで，日常会話ではイタリア語が用いられているようです．ですが，欧米諸国では「古典」としての教育が行われていますし，知識人層の一部では根強い人気があり，ラテン語の新聞・SNS・ニュース番組なども存在しています.
また，学問的世界においては，それ以上に広く，権威ある言語として用いられています．特に，生物の学名がラテン語風に綴られているのは有名な話でしょう．どうやら，日常会話でほとんど用いられない故に，文法などの変化がほぼ起きない，中立的であるということが，現在でも用いられている主な要因のようです.
言語的な特徴として最も目立つのは，語の活用が多いことでしょう．例えば，一般的な動詞は，人

称・法・態・時制に対して活用するので，1 つの動詞に 100 以上の活用が存在します．名詞でさえ，7 つの格を持つ上に男女中性の別があるため，数十通りの活用をします．そのため，覚えることが尋常でないほどに多いわけですが，その代わりとして，主語を省略できたり，語順をほぼ自由に変えたりできることで，詩歌を作りやすかったりするのです．また，発音がほぼローマ字読みであるため，音読すること自体は日本人にとっても容易です．
さて，「結局勉強会はどうだったんだ」と言いますと，前期の「ラテン語 A」については，3 人ともほぼ毎回勉強会に来て，無事単位を取ることが出来ました．しかし，後期になると，1 人はギブアップし，さらにもう 1 人が他の講義と時間が被って登録できず，講義を受けるのは私 1 人となってしまいました．こうして，ラテン語勉強会は前期だけで終わりか，と一度は思ったわけですが，時間が被った方の部員の「単位取れないけどラテン語勉強したい」というヤバい発言により，ラテン語勉強会はどうにか命を繋いだのでした．
最近は，若干グダグダしつつもなんとか 2 人で続けています．後期も最後まで頑張って勉強して，~~私が単位を取れたら~~ラテン語の素養を高められたら良いと思っています．

Linux 勉強会

wacky

こんにちは。KMC 3 回生の wacky です。KMC 3 回生と書きましたが、学部 2 回生の時に KMC に入ったので、大学では 4 回生です。今年の夏休みに Linux 勉強会を開いたので、それについて記事を書くことにしました。

勉強会の目的

今回の勉強会の表向きの目的は、Linux を扱える部員を増やすことです。KMC では、様々な活動に Linux が使われており、Linux に関わらずに KMC でやっていくのは不可能と言っていいでしょう。しかし、自分の PC に何らかの Linux をインストールしたことがあり、Linux 上で自由に作業が行える部員は案外少ないものです。そこで、Linux を扱える部員を増やすべく、今回の勉強会を開くことにしました。

ディストリビューションの選定

まずはじめに言います。私の好きなディストリビューションは Arch Linux です。どれくらい好きかと言うと、自分のデスクトップとノート PC とさくら VPS（要するに全て）に Arch がインストールしてあるくらい好きです。
今回の勉強会でも Arch を使いたかったのですが、参加者のほとんどが Linux に触れたことのない 1 回生だったので、断念しました。Linux に触れたことのない人に勧めるディストリビューションとして Ubuntu は有名ですが、あれはインストールした時点で色々なものが入りすぎていて、初心者が Linux について勉強するのには不向きです。そこで、Debian を使うことにしました。

勉強会の内容

部室のサーバにログインして、ディレクトリの移動、ファイルの編集など基本的な操作を一通り練習した後、仮想マシンに Debian をインストールして、環境構築したり、ツールの使い方を覚えたりする、という流れをとりました。Debian をインストールする時には、「追加ソフトウェアのインストール」で全てのチェックを外すのがポイントです。Debian をインストールした後は、sudo の設定をしたり、X と fluxbox をインストールして GUI な環境を構築したり、git や make の使い方を学習したり、/home を別のディスクイメージに移してマウントしたりしていました。

勉強会を終えて

勉強会で扱った内容はどれも地味な内容ばかりで、やっておもしろい、というものではなかったのですが、1 回生のみなさんはほとんど誰も脱落することなく、きちんとついて来てくれました。これだけの知識があれば、いきなり Arch をインストールしろと言われても、ArchWiki を見ながらインストールすることができるでしょう。今度機会があったら ArchInstallButtle を KMC で開いてみたいです。
ところで、この勉強会の本当の目的は何だったのでしょう？

終焉の C++

hatsusato

KMC 2 回生の hatsusato です。ここでは「終焉の C++」という勉強会について報告します。おどろおどろしい名前ですが、単に「初めての〜」といったよくあるシリーズに対抗しただけの名前なので、特に他意はありません。
この勉強会は、入門書を読んだだけでは使いこなすのが難しい C++ という言語をよりうまく扱えるように、C++ の参考書を読んでいこうというものです。この勉強会の前に『独習 C++』[*1] を読む勉強会をしたので、この勉強会では C++ の入門書は既に読んでいるものと仮定して進めました。そうでもしないと C++ の勉強はなかなか先へ進みませんからね。
私たちはまず、C++ のバイブルと言っても過言ではない『Effective C++』[*2] を読みました。『Effective C++』は、これを読まずして C++ のコードを書いてはいけないといってもいいくらい重要な本です。皆で『Effective C++』の同じところを一緒に読みながら、私が時々解説を加えたり、新しい規格の C++11 での変更点などに言及したりしながら進めました。6 月末頃から少しずつ読み進めて、NF [*3] が始まる前に無事全てを読み終わりました。参加者が少なかったり、参加

[*1]『独習 C++ 第 4 版』. ハーバート・シルト. トップスタジオ訳. 翔泳社. 2010. 978-4798119762.
[*2]『Effective C++ 第 3 版』. スコット・メイヤーズ. 小林健一郎訳. ピアソン・エデュケーション. 2006. 978-4894714519.
[*3]京都大学の学祭、11 月祭 (November Festival) のこと。多分他のページのどこかに説明がある。

者が遅刻したり [*4] する回が多くなってしまったのが少し残念ですが、いずれにせよ、これで晴れて私たちは C++ を書く権利を得たわけです。

この勉強会はまだまだ続きます。C++ の良質な参考書はまだまだたくさんあるからです。参加者の希望を調べたところ、C++11 の勉強をしたい人が多かったので、次に読む本は『C++ ポケットリファレンス』[*5] に決まりました。この本は、日本語でありながら C++11 について言及していて、しかも PDF 版が購入できるということで採用しました。この本から C++11 に関する項目だけを抜き出して読んでいこうと考えています。『C++ ポケットリファレンス』も読み終えたら、その次は *Exceptional C++* [*6] あたりを読みたいなあ。

[*4] 主に遅刻したのは私です……。

[*5] 『C++ ポケットリファレンス』. 高橋晶, 安藤敏彦, 一戸優介, 楠田真矢, 道化師, 湯朝剛介. 技術評論社. 2013. 978-4774157153.

[*6] *Exceptional C++*. Herb Sutter. Addison-Wesley Professional. 1999. 978-0201615623.

音楽もやってます

Calc=Calc 開発記

若林 秀也 (hideya)

初めに

この記事は、この部誌と同じくコミックマーケット 85 で頒布するゲーム CD に収録されているゲーム「Calc=Calc」の制作秘話……という程のものでもない話をテキトーにだらだらと書き綴るものです。「読むことで何かが得られる記事にしよう」などとは全く考えずに私の気の向くままに書いておりますので、知人のどうでもいい酒飲み話に付き合うような感覚で読んで頂ければと思います。

Calc=Calc のゲーム内容についても説明しておきましょう [*1]。数字と符号 (+ と −) の並びが問題として表示されるので、その数字や符号を上下に分配し、上下で 2 つの式を作ります。上下の式で計算結果が等しくなると得点が入るので、制限時間内に高得点を目指すというゲームです。

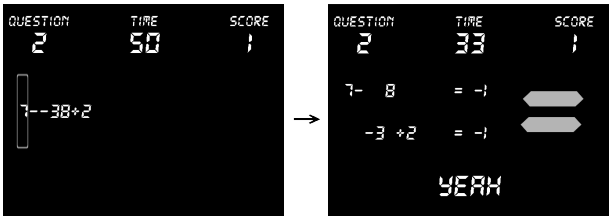


図 1　問題・回答例

見た目も中身もシンプルな印象のするゲームですが、開発には紆余曲折がありました。現在の形になるまでに実に 8 ヶ月もの月日を重ねており、途中で一度開発環境の変更に伴う作り直しもしています。それでは、8 ヶ月間の笑いあり涙ありバグフィックスありの思い出話に、今しばらくお付き合い下さい。

[*1] ゲーム CD を持っている方は、そちらのマニュアルを見るかぜひ実際にプレイされたし。

アイデア構築（2013 年 3 月上旬）

このゲームは、最初は KMC 内でのゲームコンテスト「KMC3 分ゲーコンテスト [*2]」に向けて制作されました。その回のコンテストは「しき (漢字は自由)」というテーマが与えられ、2013 年 3 月下旬に行われる春合宿 [*3] で各自の作品を発表するという形で行われました。
私は KMC 的活動ではゲーム開発くらいしかできる事がないので、ヒャッハー活躍の場だーと言わんばかりにコンテストへの参戦を心に決めました。そうなると、次にすることはどんなゲームを作るか考えることです。なにぶん解釈に幅があるテーマであるので、ユニットを「指揮」して戦うゲームや「四季」になぞらえた 4 つのモードが移り変わるブロック崩しなど様々なアイデアが私の頭に浮かびました。その中の一つとして、「数式」をモチーフとしたパズルゲームのアイデアがあった訳です。数字を使ったパズルゲームとして BLOCKSUM[*4] という素晴らしい作品がありますが、このゲームが頭の中にあったので当初のアイデアは BLOCKSUM と同様に……というか半分パクリのようなものであり、平面に敷き詰められた数字のブロックを使ってあれやこれやするというものでした。この時点から「等式を作る」というアイデアはあったので、数字に混じって = や + といったブロックがあり、= の左右で同じ計算結果になるような並びが完成したらその部分のブロックが消えて得点が入る、というようなルールを考えていました。他にも数字のブロックを 2 つ繋げると 2 ケタとして解釈される、など完成品にも取り入れられている要素のアイデアは既に生まれていたのですが、この時点では制作に踏み切るほどアイデアを固められず、日々思考を繰り返していたわけです。同様の事を「数式」以外のアイデアに対しても行っていたので、大学の講義の内容が右耳から左耳へと通り抜けていったのも無理はない話です。今思うと実に有意義な日々でした。
私は数あるアイデアの中でも「数式（等式）」を気に入っており、このアイデアに関しては特に頻繁に思考していました。数字が書かれたブロックで「倉庫番 [*5]」をやって等式を作る、というようなキメラじみたアイデア達が生まれては投げ捨てられていく中で、ある時ふっと現在の形へと繋がる「結果が等しい 2 つの式が一纏まりになっているので、それを分解して元の 2 つの式に復元する」というアイデアを思いついたわけです。セレンディピティ [*6] 発動の瞬間ですね。これならサックリと遊べてかついい感じに頭を悩ませられそうだし、等式や数字を 2 つ繋げて 2 ケタの数にするという使いたかったアイデアが入っている。それに問題の自動生成というのも（ゲームルール的にも実装の難易度的にも）出来そうで面白いじゃないかと思い、コンテストにはこれで行こうと決めました。
そうと決まればさぁ実装だ……といきたかったのですが、残念ながら大学生には定期試験という名の試練があります。日頃の講義時間をゲームデザインの構築に充て、試験期間にも欠かさずダンジョンに潜り鍛錬を怠らなかった私は追試という名のペナルティエリアを攻略する羽目になり、

[*2]かつて web 上で行われていた某ゲームコンテストを真似て開催されたもの。ピンと来た方もきっと居るだろう。
[*3]インターネットもロクに繋がらない山奥の施設に 4 日ほど篭り、部員が講座や会議などを行う KMC 恒例行事。
[*4]`http://infotech.rim.zenno.info/products/blocksum/ja/` で無料公開中。
[*5]人間を動かし荷物を押して格納場所へと片付ける、歴史のあるパズルゲーム。
[*6]思わぬものを発見する能力。ここでは「下手な鉄砲も数打ちゃ当たる」くらいの意味。

まったくどこのイメージファイト [*7] だよとボヤきつつも留年はしたくないので必死に勉強し、追試が終わった後は春合宿で行う講座の準備をし……と次々襲い掛かるタスクをちぎっては投げちぎっては投げ、さぁ今度こそゲームが作れるぞとなった時には既に**春合宿初日の出発 5 時間前**となっていました。直前まで講座の発表練習をしていて疲れているにも関わらず、私は直ちに開発に取り掛かったのです。ただし、寝落ちするのが怖いので集合場所の部室に行ってから。

Ruby 時代の思い出（2013 年 3 月中旬）

ようやくゲーム制作に取り掛かる段階となったわけですが、発表会まで残された時間は僅か 3 日程度。当然部員の講座などはちゃんと聴かなければならないので、時間をフルにゲーム開発に使えるわけではありません。開発開始前からデスマーチが確定しているこの状況に、私のテンションも初めからクライマックスでした。バスに乗り込む直前までコーディングをする私。施設に到着するなり小池スタイル [*8] でコーディングをする私。KMC 部員かくあるべき、という姿をこの時の私は体現していたように思います。気のせいですか。そうですね。
さて今回の状況、急ごしらえとなる事が目に見えています。手早くゲームを完成させるため、開発環境として（当時）日頃から使っていた C#+XNA [*9] ではなく、ラピッドゲームコーディング祭り [*10] で用いられている Ruby+MyGame[*11] を選択しました。小規模なゲーム開発に向いている、という点ではこの選択は良かったのですが、なにせ私は日頃 Ruby を使いません。覚えている文法はほんの僅かです。そして合宿の施設ではインターネットも使えないためリファレンスをググることも出来ず、結果として日頃から Ruby を使っている possum さんに
~~リファレンス代わりになってもらい~~質問に答えてもらいながら開発を進めることになりました。
possum さんにはこの場を借りてお礼申し上げます。
そのようにして、合宿の空き時間を縫って開発を進めていきました。実装が上手くいかず諦めようと途中で思ったものの、執念で開発を続行したりもしました。私と同じくコンテストに向けて横で必死でゲームを作っている tsutcho さんの姿に共闘意識のような物を感じ、密かに支えられたりもしました。変数の頭に次々と付加される$[*12]、メインループ中の switch 文で実装されるシーン遷移。手段を選ぶ余裕が無い必死な状況の中、私が感じているものはまさしく「生きる喜び」でした。
――そしてコンテスト約 15 分前、めでたくゲームは完成したのです。
ゲームとして体を成した段階で、自分でもこのゲームは十分な面白さがあると感じていました。その感触は間違っておらず、コンテストでは部員の皆に楽しく遊んでもらえました。素っ気ないデザインで、アンドゥなんて気の利いた機能は無く、タイトルすら決まっていない。そんな状態でも他人に楽しんでもらえたというのは、とても大きな手応えです。皆から好意的なフィードバックを頂

[*7] アイレム制作のシューティングゲーム。演習ステージでの成績が悪いと、実戦よりも難しい補習ステージを攻略しなければならない。
[*8] 立った状態でノート PC を太ももと手首で支えながらコーディング等を行う姿勢。
[*9] 現在、Microsoft によるサポートは終了している。
[*10] KMC の新勧イベント。上回生のサポートの元、初心者の新入生に簡単なゲームを 1 日で 1 本完成させる。
[*11] `http://dgames.jp/ja/projects/mygame/`
[*12] Ruby におけるグローバル変数。

図 2　簡素なゲーム画面

いた事もあり、今回の「ラフスケッチ」のような形ではなくもっとしっかりとした「清書」をしてこのゲームを世の中に出してやりたい、という思いを抱きつつ、私の春合宿は満足と共に終わったのです。

Unity 触り始め（2013 年 1 月頃〜3 月下旬）

話が変わりまして、近年 Unity[*13] というゲームエンジンが流行しております。私は流行に敏感な人間というわけではなく、むしろ時代の流れに取り残される事が多いような人間なのですが、この Unity は 3D が簡単に扱えてしかもブラウザゲームの制作にも対応しているといった点が気になったので、チュートリアルをこなしてみるなどして軽く Unity の勉強をしていました。公式のチュートリアル「はじめての Unity[*14]」は第 2 回以降がまだ公開されていなかった [*15] ので、4Gamer.net に掲載されていた入門記事 [*16] の内容をひと通りやってみた私は Unity の強いパワーを感じ、Unity を使ってゲーム制作をしてみたいと思いました。
しかしこの Unity、今まで使ってきた XNA などのゲーム制作環境とはかなり性質が違います。これまで私が経験してきたゲーム開発が絵を出すのにも音を鳴らすのにもプログラミングを介する必要があったのに対し、Unity は世界を構築するにあたってプログラミングが全然必要とされません。Unity Editor 上でマウスをポチポチするだけで、オブジェクトの生成や当たり判定の設定、リソースの読み込みが出来てしまうのです。新しい技術を身につける為にはとにかく実践だということで Unity でゲームを 1 本作ってみようと試みましたが、この文化的障壁に阻まれて開発が全然進まないままやる気も減衰していき、結局そのゲームは作りかけのままエターナルな闇へと飲まれていったのです。安らかに眠れ。
そうこうしているうちに前節に書いたように大学の定期試験が始まり、その次は春合宿の準備に追われ、しばらく Unity どころではありませんでした。しかしこのままでは終わりたくない、いつかもう一度 Unity 開発に取り組みたいという想いは残っていました。

[*13] http://japan.unity3d.com/
[*14] http://japan.unity3d.com/developer/document/tutorial/my-first-unity/
[*15] 残念ながら、2013 年 12 月 8 日現在でも第 2 回以降は「近日公開」のままである。
[*16] http://www.4gamer.net/games/032/G003263/20111210004/

Unity への移植-前編（2013 年 3 月下旬〜8 月末）

春合宿を終えた私は、激しい戦いの疲れを癒やすためにコタツに埋まる日々を過ごしていました。大してやるべき事もなく、たまたまその時は特にやりたいゲームも無かった私はふと思い立ったようにデスクトップの整理を始め、そしてかつて闇に飲まれた Unity プロジェクトのフォルダに目を止めました。私はとても気移りしやすい性格で、生理学の勉強をしていたはずがいつの間にか前日作ったホルモン炒めの味をより良くするための調味料バランスについて考えている、というような事は日常茶飯事です。その時も目に止まった Unity のアイコンをきっかけとして、デスクトップ整理を放り出して Unity との思い出を回想し始めたのです。4Gamer.net のチュートリアル記事ページが広告のせいでやたら重かったこと、コーディングせずとも世界が構築できる異世界の感覚、そしてあの時作ろうとしていたゲームの内容……思考のランダムウォークを経て、私はある事に気づきます。あの時のゲーム制作が失敗したのは、ゲームデザイン等を含めたゲーム制作と Unity 慣れを同時に行おうとしていたからではないだろうか？　既に完成図のビジョンが見えているゲームの制作であれば Unity 慣れすることに専念でき、無事ゲームの完成までこぎつけられるのではないだろうか？

そこで、先日春合宿で作ったゲームを Unity で移植するという発想が生まれました。一度作成したことのあるゲームであれば、完成形が見えているためにゲームの方針で迷うことも少なく Unity の習得に集中できるだろう。ついでにあのゲームを「清書」して世に出せる。ブラウザで遊びたい、という声もあったがそれも Unity で実現できるではないか。このアイデアは素晴らしい物に思え、思い立った私は即実行へ移すことにしたのです。2013 年 3 月 25 日、プロジェクト [*17] “ thyphy [*18] ” が静かに始動しました。

Unity 向けの git の設定を整え、Unity の基本的な要素について理解を深めていく毎日が続きました。Unity でのコーディングせずに世界を構築する感覚が気に食わなかった私は、今までのようなコーディングありきのゲーム開発を Unity 上で半ば無理やり行っていました。それは今になって思い返してみると中二病のようなものであり、今まさに呻いています。あーうー。そんなこんなで Unity による開発が少しづつ進んでいくうちに、私は 4 回生になりました。

どの学部でも、大学の 4 回生というものは研究室配属というイベントがあるもののように思います。私も多分にもれず、薬学部のどこぞの研究室へと配属されました。いわゆるラボ畜と言われるほどに厳しい研究室というわけではありませんが、やはり新しい環境に慣れるというものは精神力がすり減るものであり、その研究室の活動内容がさして興味の持てないものであれば尚更です。私は日々をやりすごすのに精一杯となり、thyphy に取り掛かる気力を捻出できずにいました。なにぶん Unity での開発を始めたばかりなので問題に対する解決法をすぐに導く力が無く、作業を一つ終わらせるにも結構な時間と労力がかかります。そのため開発に掛けたカロリーに対して達成感・爽快感が乏しく、どうにも楽しさが無かったのです。開発を始めては壁に当たって疲れてやる気を

[*17] 勉強会やゲーム開発などの KMC の活動は、基本的にプロジェクトという単位で取り扱われる。
[*18] ゲームプロジェクトの名称はゲームタイトルと一緒にはならない事が多い。

失い、1 ヶ月ほど経ってふと思い出したように開発を始め、前回当たった問題を解決するもののまた新たな壁に阻まれて開発が停止する、そんなサイクルが 8 月頃まで続きました。
大学生の 8 月といえば夏休みであるはずですが、私の所属する研究室では夏休みがお盆休み 3 日程度と自由な休みが 7 日程度しかありません。お盆は実家で過ごしたのですが、実家ではやることがあまりないために比較的 thyphy の開発が捗りました。大画面テレビを利用したデュアルディスプレイによる開発はとても快適でした。飼い猫がキーボードに飛び乗り、Unity Editor のよく分からない機能が発動してリカバリーするのに小一時間かかったのはいい思い出です。またこの記事の本筋とは関係ない話になりますが、残りの 7 日の休みは東京にある某ゲーム会社のインターンに参加していました。というのもこの頃は薬学部での活動内容に嫌気がさしており、薬学系の仕事に就くよりもゲーム会社に就職する方が自分は幸せではないだろうかという疑念に駆られていたためです。まだ現時点では将来の希望進路は確定していませんが、ゲーム開発の現場を見ることが出来たのは良い経験だったと思います。山手線で毎日通勤するのはもう勘弁ですが。
8 月にゲーム開発方面の機運に乗ったのか、それとも Unity の開発に多少慣れてきたのか、9 月に入って学校に通う日々が再び始まっても比較的コンスタントに開発を続ける事ができました。そして 9 月半ばに、個人的には KMC 一大イベントであるコーディング合宿が行われました。コーディング合宿とは毎年夏休みに行われる恒例行事であり、琵琶湖近辺の合宿施設で部屋を借り、2 泊 3 日でただひたすらコーディングを行うというストイックな合宿です。娯楽といえば琵琶湖に石を投げ込みに行くくらいしか無い環境で連日コーディングにいそしむのはとても中毒性があり [*19]、私は 1 回生の頃になんとなく参加して以降その魅力に取り憑かれ、毎年参加 [*20] しています。今年も無事参加することができた私は、琵琶湖に満ちる魔力の助けもあってかかつて春合宿で発表した状態と同程度までゲームを一気に作成できました。
エフェクトを付けるなどして見た目をより綺麗にする、問題生成のアルゴリズムをより良くするなど手を加えたい部分はまだまだありました。ですが、かつての完成形に追いついてしまったということで良くも悪くも「安定した状態」になってしまい、気持ちにも一区切りが付いてしまったために再び開発意欲は低下していきました。コタツから布団に移動するのが難しいように、安定した状態から抜け出すのには（より安定な状態がその先にあったとしても）大きなエネルギーが必要であるということは皆さんも承知のことだと思います。せっかく Unity を使っているのでキレイなエフェクトや 3D 演出をバンバン使いたいと思っていたのですが、エフェクトや演出を付けるにしてもどのようなデザインに纏めるかのビジョンが見えておらず、問題生成のアルゴリズムも具体的な改善方針がパッと思いつかない現状ではこれ以上開発を進めることは無理だと思った私は、しばらく開発を停止することにしました。

[*19] 激しく個人差がある。
[*20] 原則全員参加の春合宿とは違い、コーディング合宿は任意参加。

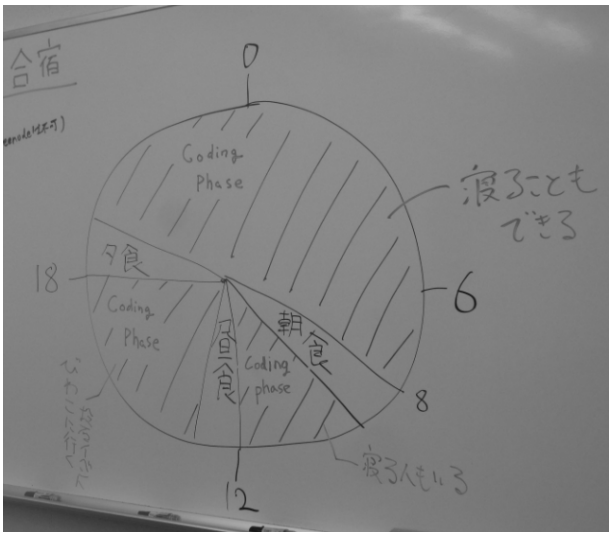


図 3　食事とコーディングフェイズしか無いストイックな合宿のスケジュール

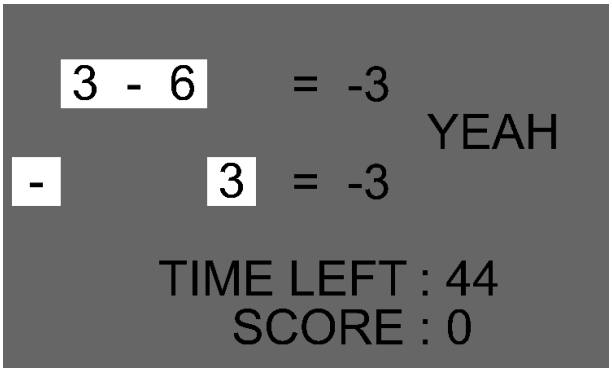


図 4　コーディング合宿時点での画面。このパネルがくるんくるんする予定だった

インターバル（9 月下旬〜10 月下旬）

創作活動において、自分の中にあるものを output する手段を獲得する事は大事です。ゲーム制作ならばゲームプログラミングや RPG ツクール [*21] などの取り扱いスキルであり、イラスト制作ならばペイントツールの使い方やデッサン力というものがそれにあたるでしょうか。ですがそれと同等か或いはそれ以上に、output したいものを自分の中に生み出すための input をしておくことも非常に大事です。面白いゲームを遊んだことの無い人に面白いゲームは作れません。絵や音楽、小説や映画にも同様の事が言えるでしょう。創作に対する意欲を維持するという観点からも、input

[*21] エンターブレインより発売されている、RPG を制作するためのツールソフト。

の十分な摂取はクリエイターにとっては欠かせないものなのです。
さて、私が開発を停止する理由となったのは、開発の行き先が見えなくなったからです。言い換えると、このゲームでどのような表現を行おうか、というものが自分の中に構築できていなかったからです。そのために、しばらくは開発を停止して input に専念し、output すべきものを構築するパワーを蓄えようと考えたのです。
ということで、1 ヶ月ほどの間はまったりとダンジョンに潜ったり、突き指で来院した患者を医療ミスで死なせてしまったり、女魔導師と一緒にダンジョンに落っこちたりしていました。おかげで新しく作りたいゲームのアイデアが浮かんだり、かつて考えていたゲームをより良くする新たなアイデアが見つかったりしました。しかし困った事に、thyphy に関するアイデアはちっとも現れません。そのうち制作に着手したい「積みゲー」が増えていく一方で、目の前のゲームが全然終わらないという困った状況になってしまったのです。10 月も終わりかけ、京都にそろそろ冬が来る [*22] かという頃でした。

Unity への移植-後編（10 月末〜11 月下旬）

京都大学では毎年 11 月下旬に学生祭 [*23] が行われ、様々なサークルが教室を使用して展示を行ったり、体育会系の若者がタコ焼きを売り歩いたりします。KMC も毎年教室での展示を行い、制作したゲームをお客さんに遊んでもらうのをメインに様々な活動の成果を発表しています。先も述べたとおり私は KMC 的活動ではゲーム開発くらいしかできる事がないので、やはりヒャッハー活躍の場だーと言わんばかりに毎年学祭向けにゲームを制作して出展していました。今年も同様にヒャッハーしたいと思っていたのですが、研究室の活動があることなどを考えると学祭のために新たにゲームプロジェクトを立ててそれを完成に導けるほどのカロリーを捻出できるとは思えませんでした。そのため、今年は年度初めからゆっくりと開発していた thyphy を出展しようと考えていたのです。例年私は夏休み頃から学祭向けのゲーム開発を始めていたので、年度初めから開発していた今年は余裕だろう。そう思っていた時期が私にもありました。
前節で述べたように、10 月も下旬になって学祭まであと 1 ヶ月を切ってもなお、このゲームの完成形がどのようなデザインになるかというビジョンはさっぱり見えていませんでした。そうこうしているうちに締め切りは近づいていき、凝ったエフェクトや BGM を作成する時間はどんどん無くなっていきます。今からデザインが考えついてもそれをちゃんと実現する時間が無いのではないだろうか、という焦りに追われてきた頃になってようやく、現在の形に通じる「セグメント表示を用いた、電卓のようにシンプルなコンピュータ的デザイン」というアイデアが浮かびました。
しかし、このアイデアはある意味で逃げでした。ゲーム画面を見ると分かる通り、キレイなエフェクトも 3D 演出も登場しないばかりか黒背景に文字が表示されているだけという時代錯誤なレベルに簡素なグラフィック。お前はなぜコレを Unity で開発したのだ、とあちこちから言われそう [*24]

[*22] ある部員が「京都の四季は春・梅雨・夏・冬」と言うほどに、京都の秋は短い。
[*23] November Festival を略して NF と呼ばれる。
[*24] 実際何度か言われたし、自分でも言った。

なほどに Unity の特色を生かしていません。せいぜいマルチプラットフォームやブラウザ向けのビルドができる程度です。とはいえ、ここまで簡素なものにしたお陰で締め切りまでに無事「完成品」とする事が出来たのも事実です。

図 5　最新鋭のゲームエンジンを使用しているとは思えない画面

そしてインターバルの間に頭が冷えたのか、なんでもプログラミングで済ませようというこれまでの自分の考えを捨て、Unity の特色を活かしたマウスをポチポチする開発手法を試していました。その結果、今までの苦労は何だったのかと言わんばかりに開発がスムーズに進み、ゲームタイトルも無事決定、身長は伸びませんでしたが Unity の生産性の高さを身を持って実感することができました。時間が足りなければ問題生成アルゴリズムの改善を諦めるつもりでしたが、無事により良いアルゴリズムを実装することができました。その変更が原因で問題生成時に低確率 *25 でゲームがフリーズするというバグが締め切り前日に発覚したりもしましたが、この程度はよくある事です。むしろ私は「そう来なくっちゃ」「盛り上がって参りました」「この風、この肌触りこそ締め切りよ」などと一人ハイテンションになっていたのですが、おそらく締め切り前で頭が軽くおかしくなっていたのでしょう。良い子は真似しないでね。結局、1 時間程度かけて無事に特定の条件下で無限ループに陥っていることを看破し、それを修正することに成功しました。いやぁ、バグを潰した後のコーヒーは格別ですね。

また、HTML マニュアルの作成 *26 や例会等での連絡という部分において jf712 君の協力を頂けたお陰で、想定していたよりも細かな部分の調整まで実現させる事が出来ました。この場を借りて再度感謝の意を表したいと思います。今度エルラ *27 でも行こうか。

結果として学祭では中々多くのお客さんに楽しんでもらえました。想像していた通り万人受けはしなかったものの、ある程度考えるのが好きそうな人には受けが良かったように思います。コレは良

*25 1/100 程度。確率がもっと低ければ、気付かずにリリースしてしまっていた可能性が高い。
*26 恥ずかしながら、私は KMC4 回生にもなって HTML がロクに書けない。
*27 KMC 御用達のメキシコ料理屋「エル・ラティーノ」のこと。気さくなおっちゃんがあなたを待つ。

く出来ていると何度も褒めて下さった方がいたり、結果表示の際に表示される英語の短文 *28 に外人さんがノリよく反応して下さったりしたようで、ゲーム開発者冥利に尽きるというものでした。ヒャッハー。

学祭以降、そしてこれから（11 月下旬〜）

学祭を終えて一段落し、再びプロジェクトは安定した状態になりました。さて、KMC での次なる発表の場は、冬のコミックマーケット。そこで頒布されるであろうゲーム CD と、この記事が掲載される部誌の準備が部員の次なる課題です。例年では学祭で発表されたゲームが多少のバージョンアップを経てゲーム CD へと収録されます。thyphy もバージョンアップさせたい点が少しあるのですが、学校で共用試験 *29 対策の講義が始まり、またこの記事を書くのにも忙しくて未だ着手できていない状況です。冬コミ版では「VER 1.2」としてリリースできるよう頑張りたいと思っていますが、もしそれ以前のバージョンが CD に収録されていたら「あぁ、駄目だったんだな」と笑い飛ばしてください。
また、学祭直前に Unity4.3 がリリースされ、待望の 2D ゲームツールキットが使用可能になりました。こちらの方もぜひ使ってみて、そしてできればまたゲームを作って何らかの機会に皆さんに発表できればなぁなどと思っています。来年も学校やらなんだかんだで忙しくなりそうなのですが、まぁなんとかなるでしょう。
それでは、hideya の次回作にご期待下さい。

*28 正解したら「COOL」、不正解ならば「TRY AGAIN」等といった文が表示される。
*29 薬学部 5 回生の実習を受けるために通らなければいけない試験。2013 年度の京都大学薬学部では 1 月上旬に本番が行われる。

CTF 戦記 スペイン編

tyage

CTF (Capture The Flag) というコンテストをご存知でしょうか。これはコンピュータセキュリティに関する技術や知識で競うコンテストです。具体的にはバイナリ解析・パケット解析・フォレンジック・Web セキュリティ・暗号・トリビアなどの知識を使って競技します。セキュリティの知識を利用したプログラミングコンテストのようなものを想像してもらうとわかりやすいかと思います。最近は日本でも普及しつつあり各地でコンテストが開かれているためご存知の方もいるかもしれません。
ここではスペインで行われた No cON Name Facebook CTF[*1]（以下、Facebook CTF）での私と私のチームの体験談を述べていきます。

Facebook CTF is ...?

先ほど CTF はプログラミングコンテストに近いと述べたのですが、CTF はチーム戦であることが多く、また競技方法が大きく分けて 2 種類であるという特徴があります。1 つ目の種類が Jeopardy と呼ばれるクイズ形式のもので、主催者が出題する問題の答えを回答して得点を得る形式です。2 つ目の種類が Attack-Defence（攻防戦）形式です。各チームが配布されたプログラムを起動してサービスを運営しており、他チームのサービスに攻撃しパスワードを得ると得点になります。同時に自チームへの攻撃をできないよう、パッチをあてる等して防御します。各チームで領土を奪い合う戦争ゲームのようなものを想像してもらうと分かりやすいかと思います。
今回紹介する Facebook CTF では、予選は前者の Jeopardy、決勝は後者の Attack-Defence に近いものとなっているのですが、どちらも少し特殊な形式となっていました（詳しくは後ほど述べることとします）。

[*1] http://noconname.org/files/CTF_NocONName_2013_ENG.pdf

Quals

2013/9/28 7:00AM–9/30 0:00AM（日本時間）の期間で予選 (No cON Name Facebook CTF Quals 2013[*2]) が開かれました。予選はオンラインで開催されるため、世界中のチームが参加することになります。私は EpsilonDelta というチーム [*3] に所属しており、チームの 3 人で 8:30 ごろから問題を解き始めました。この時点で開始時間から既に 1 時間半が経過していますが、CTF の参加時間としてはさほど遅くありません。これは CTF は開催期間が 2・3 日であることが多く Jeopardy 形式の場合は特に参加が遅れても不利になりにくいためです。
そんなこんなでいつも通り予選を始めたのですが、参加者の間にはざわ... ざわ... とした空気が漂っていました。Jeopardy 形式の CTF に参加したことのある方はよく理解できると思うのですが、この形式の CTF は「チーム登録をする→コンテストサイトにログインする→問題を解く→問題の答えを回答する→自チームのポイントが増える」という流れであることが大半です。しかしこの予選では「問題を解く→ writeup[*4] 書く→それを運営にメールで送る」という流れになっていました。そのため「どうやってチーム登録するんだ？」「どこに問題の答えを回答するんだ？」と混乱する参加者が多かったようです。実際、私のチームもコンテストの説明を詳しく読んでいなかったためにメールを送るのに少々手間取りました。CTF では IRC で運営や他の参加者と連絡を取ることが一般的なのですが、それがなかったのも混乱を招いた一因かなと思います。
ちなみに予選の問題はたった 3 問しかなく、その日参加した 3 人でそれぞれ 1 問ずつ解いて writeup を提出するまでに 1 時間もかかりませんでした [*5]。1 問目は JavaScript のコードを読んでパスワードを生成する問題、2 問目は apk ファイル内のバラバラになった QR コードを読み取る問題、3 問目は 64bit elf 形式のファイルを disassemble する問題でした。
後から気づいたのですが、この予選は writeup の精度と提出までの早さを基準に順位付けをするらしく、遅くから参加したチームには不利になる仕組みになっていました。結果私のチームは 13 位でしたが、決勝戦を辞退したチームが出たために決勝に進む 12 チームに入ることが出来ました。

Before the Finals

さて、決勝に進めることが分かったのはよいのですが、そこからが一番大変でした。
決勝進出が決まったのが 10/7 のことで、その日のうちに返事をしろというメールが送られてきました。決勝はスペイン・バルセロナで 11/1 に開催されると分かったのですが、残り 1 ヶ月もない状態で準備ができるのか、旅費等の費用はどうするのか [*6] など問題は山積みです。チームから 4 人 (@tyage, @hiromu1996, @potetisensei, @nk0t) が行くことになったのですが、最低でも 1 人

[*2] http://ctf.noconname.org/
[*3] このチームは私立灘中高の生徒 3 人と、私を含むその他数名で構成されています。
[*4] 自分がその問題をどのように解いたかを書いたものです。CTF ではコンテスト終了後に解いた問題の writeup を書く文化があります。
[*5] http://bit.ly/ICdu01
[*6] Facebook からは賞金は出るが旅費等の援助はありませんでした。

12 万ほどかかる旅費を出すのはなかなか大変です。
辞退することも視野に含めて検討していたところ、@syuu1228 さんがカンパを募りましょうと音頭を取ってくださいました [*7]。その結果、カンパページ [*8] が作られ、あっという間に 4 人分の旅費をまかなえるだけの金額が集まりました。深夜にも関わらず、数時間足らずの出来事だったので本当に驚きました。改めてありがとうございました。
そこからはパスポートを取りに行ったり、旅行会社に連絡して日程を組んだり、コンテスト運営側と連絡を取ったり、必要機材を買ったり [*9]、情報収集したりと非常に忙しかったのですが、無事に決勝までに間に合わせることができました。参加するメンバーが中学 2 年生・高校 2 年生・高校 3 年生・大学 2 年生（私）と全員未成年であるという不安要素もあったのですが、灘高校の教師の方が付き添いしてくれることで安心して旅立つことができました。
決勝は 1 日だけなのですが、1 日だけで帰るのは非常にもったいないということで、移動を含めて 1 週間ほどの予定を組みました。今回は決勝のみを書きますが、その他の日程の様子はブログ [*10] や Flickr[*11] で見ることができます。

Finals

CTF 決勝の開催期間は午前 10 時から午後 6 時までの 8 時間と短期決戦です。そのため、チーム内ではそこまで分量が多くないだろうと予想をしていました。
また決勝の形式は Attack-Defence であると告げられていたため、DEFCON CTF などでよく使用される、各チームでサービスを起動する形式（冒頭で述べたものと同じ）を想定して心の準備をしていました。具体的には、4 人のうち 2 人が運営から配布されるプログラムの脆弱性を探して他のチームに攻撃をする一方で、残り 2 人はネットワークを監視したり SLA[*12] 対策をするというような作業分担を決めて、決勝当日を迎えました。
決勝の会場はバルセロナの中心街から少し離れたところにある CosmoCaixa という科学博物館でした。CTF は No cON Name というセキュリティカンファレンス内のイベントとして開催されており、CTF 参加者以外にもセキュリティ関係者が集まっており平日の博物館の一角は普段とは異なる雰囲気が漂っていたように思います。
受付にいた運営に挨拶をした後に、会場に入ると既にいくつかのチームが準備を始めており、端の方では Facebook セキュリティチームが最後のセットアップをしているところでした。全 11 チームが揃い、今回の大会の説明と簡単なチーム紹介などが終わったところで、ついに決勝が始まりました。

[*7] https://twitter.com/syuu1228/status/387113908063838208
[*8] http://syuu1228.github.io/hiromu1996ctf.html
[*9] 決勝戦は Attack-Defence 戦のため、ネットワークスイッチなどの購入も検討したのですが今回は必要ありませんでした。そのためコンテストには各自のパソコンと周辺機器のみで望みました。
[*10] http://epsilondelta.hatenablog.jp
[*11] http://www.flickr.com/photos/tyage/sets/72157637147617233/
[*12] Service Level Agreement のことで、要はサービスがちゃんと起動しているかどうかということです。CTF 運営側は一定時間ごとにサービスが動いているかどうかを確認し、確認できなければそのチームを減点します。

大会の説明の際に初めて知らされたのですが、今回の試合内容は以下の通りでした。

- 試合はボードゲーム Risk から発想を得て作られている。
- 各問題には国の名前がついており、その問題の得点を得ると会場のスクリーンに写っている世界地図上で問題の国がチームの色で塗られる。
- 問題には 2 種類があり、1 つは Base 形式でもう 1 つは Flag 形式である。
- Base 形式は Jeopardy 形式と同じで、問題が 1 つ与えられ正しい答えを入力すると問題ごとに個別の得点が与えられる。
- Flag 形式は Attack-Defence 形式に似ているが、問題につき 1 つのサーバー上で（各チームがサービスを運営するわけではない）攻撃と防御をする。
- 問題で用意されたサーバー内の「/tmp/SCORE_POINTS」にチーム名（私達の場合は EpsilonDelta）を書きこむと、5 秒につき 1 点が入る。
- ただし 5 秒ごとの確認の間に他のチームによって上書きされたり消されたりすると得点は入らない。
- 問題サーバー上での他チームの妨害は許可されているが、他チームへの直接攻撃や会場の Wi-Fi に攻撃、DDoS や flooding や spoofing といったコンテストを維持するのが困難になる攻撃は許可されていない。

想定していた Attack-Defence とは異なり、各チームが互いのサーバーに攻撃するのではなく、1 つのサーバー上でみなが攻防戦を繰り広げるという試合形式になっていました。SLA 対策やサービスにパッチを当てるといったことは行う必要がないため、戦い方も想定と大きく変わることとなります。
私のチームではこれを SECCON 形式 [*13] と勝手に呼んでおり、この方式での CTF では一度負けているため、あまり嬉しくない形式でした。
また少ないと思っていた問題は 17 問も与えられ、8 時間では普通無理だろ！ と会場内で声が聞こえてきました。
実は決勝前日に運営に挨拶に行っていたのですが「予選よりずっと難しくなるから覚悟しておけよ」「明日の決勝はちょっと驚くと思うよ」といった内容のことを言われており [*14]、運営の思惑通りチーム全員が驚くことになったのでした。
しかし決勝が始まってしまったのでとにかく問題に取り組まねばなりません。とりあえず全問題の傾向分析から始めました。表 1 の様な表で管理していました（例として 17 問中 5 つを表示しています）。
とりあえず解けそうな問題から手をつけていた結果、India という Flag 形式の問題が簡単だったので一番初めに解くことができました。India で用意された Web ページには CVE-2012-1823[*15] の脆弱性があり、それを利用して外部から任意の PHP コードが実行できるようになっていました。

[*13] 日本で開催された SECCON 2012 全国大会で使われた形式で、今回と同じく 1 つのサーバー上で攻防戦を行うものでした。
[*14] さすがに問題傾向や試合の詳細は教えてもらえなかったのですが。
[*15] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-1823

表 1　問題管理表

Country	IP	Type	Category
USA	192.168.69.5	Flag	Pcap
China	192.168.69.15	Flag	Web
Russia	192.168.69.25	Flag	?
Australia	192.168.69.35	Base, 500pts	ELF Reversing
Canada	192.168.69.45	Base, 1200pts	ELF Reversing

CVE-2012-1823 は有名な脆弱性で、CGI 上で動作する PHP にて等号を含まない query が送られてきた場合に、query をコマンドラインの引数にしてしまうというものです [*16]。Facebook もこの脆弱性を使った採用ページを作ったため、話題になったことがありました [*17]。
例えば、PHP の実行時引数に-s オプションを渡すとそのファイルのソースコードを表示することができます。そのためこの脆弱性のあるページ上で `http://example.com/hoge.php?-s` とすることで、hoge.php のソースコードを表示することが可能になります。
今回は、-d オプションを使って php.ini ディレクティブの設定を行います。以下の 2 つのオプションを設定することで、php://input から送られてくるファイルをサーバー上で実行してもらうことができます。

- allow_url_include=On
- auto_prepend_file=php://input

ここまでくれば後は送信するファイルに、/tmp/SCORE_POINTS に自分のチーム名を書き込むスクリプトを書くだけです。
このようにして India を攻略し、その後は他の問題に取り組んでいたのですが、India が攻略されたことが会場のスクリーンで大きく映しだされていたために、他のチームも India の攻略を開始しました。1,2 時間ほどして/tmp/SCORE_POINTS に他のチームの名前が書かれ、私のチーム名は消されてしまいました。
私達も無限ループを利用して自分のチーム名を書き込むスクリプトを実行するなどして対抗したのですが、すぐに他のチームにプロセスが殺されてしまい、イタチごっこが始まっていました。他のチームが India で得点するたびに、ニヤニヤと笑みを浮かべながらこちらを向いてくるのが印象的でした。
その後も数チームが入り乱れてそんなことを繰り返しているうちに、India のサーバーがダウンしてしまいました。運営からは「India は死んでしまったから、もうアクセスせずに他の問題に取り組んでくれ」と言われてしまいました。
他の問題も 6 問ほど解いたのですが、解けたのは Base 形式のものがほとんどでなかなか Flag 形

[*16] `http://blog.tokumaru.org/2012/05/php-cgi-remote-scripting-cve-2012-1823.html`
[*17] `http://facebook.com/?-s` 今は動作していません。

式の問題を解くことが出来ませんでした。China（python のオブジェクトコードを base64 にして送ると実行されるという問題だったと記憶しています）の Flag 形式の問題を解いた際には、既に多くのチームが入り乱れており、新参者には入りづらい土地にになっていました。運営も「China is new India!」みたいなことを言っていたように思います。
そんな感じで Flag の問題をうまく得点できないでいるうちに、他のチームは Flag の問題による得点を多く積み重ね、初めは 1 位だった私のチームの順位も落ちていき、最終的には 6 位で終了しました。

After the Finals

大会後、運営の Facebook チームからいくつかアドバイスをいただきました。
「君達のチームは Defence ができれば、入賞出来たよね。」「チーム内で攻防戦をして Defence とペネトレーションテストを勉強すれば iptables とか SELinux とかわかるようになるよ。」「fork 爆弾（プロセスを大量に生成して、他のプロセス生成を妨害する攻撃）をうまく使えば、他のチームへの Defence になるのを覚えておくといいよ。」といった内容でした。
大会参加者の中で一番若かったこともあってか、他のチームからもいくつか声をかけてもらったり、大会公式ページにも取り上げてもらえました。[*18][*19]。
そういえば会場で行われていた No cON Name のカンファレンスを覗いてみたのですが、スライドも話者もスペイン語で、半分も理解出来なかったためあまり長居せずに退出してしまいました。内容は面白そうなものがいくつかあったので少し残念です。

End

今回のように CTF はオンラインで開催されるものもオフラインで開催されるものも様々にあるのですが、それらの多くは CTFtime[*20] というサイトで確認することができます。
CTF 参加チームの多くがこのサイトに登録しており、各コンテストで獲得した点数によりその年のランキングが変動する仕組みになっています。CTFtime に登録されているものだけでも平均して 1 ヶ月に 2・3 回のペースでコンテストが開催されているので気軽に参加してみてください。
CTFtime に登録されるコンテストは、国際コンテストなので英語であることがほとんどです。日本では SECCON[*21] というコンテストが開催されており、今年は全国各地で開催されています。今年の地方開催の予選は終了してしまいましたが、2014 年 1 月末にあるオンライン大会はまだ参加登録できるはず [*22] なのでそちらも参加してみてはいかがでしょうか。

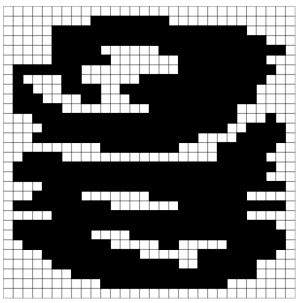
[*18] https://www.facebook.com/photo.php?fbid=180559925479396
[*19] https://www.facebook.com/officialctf/posts/180435942158461
[*20] https://ctftime.org/
[*21] http://www.seccon.jp/
[*22] 12/9 執筆当時には参加登録期限が公開されていません。

計算機による代数的位相幾何

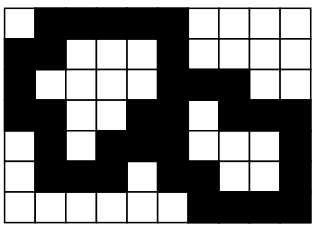
大林

穴の数を数える

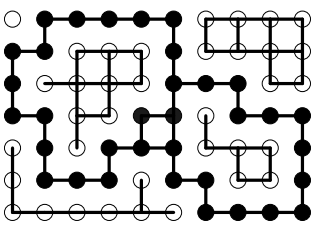
図 1(a) のようなビットマップ画像があったとしましょう。この黒い領域の個数、黒い領域に空いた穴の個数を計算機によって数える、という問題を考えます。この図では見ればわかるように黒い領域が 2 個、穴の個数は 3 個です。これをいかにして計算機に計算させればよいでしょうか。
解法の 1 つとしてグラフの連結成分の問題に置き換える方法が考えられます。簡単のため図 1(b) のような画像に対してこの問題を考えることにしましょう。これに対し、白色の画素をグラフの白丸の頂点とし、白色の画像が隣接しているときにはその 2 つをつないだグラフを考えます (図 1(c))。黒色の画素に対しても同様のグラフを作ります。このグラフの連結成分を数えれば求める結果が得られます。白の連結成分の個数は外枠と繋がっている部分は捨てる必要があることに注意しましょう。



(a) 黒い部分が 2 つで穴が 3 つある画像



(b) 穴が 2 つある画像



(c) グラフ化したもの

図 1　2 次元画像の穴の数

3 次元の穴

さて、2 次元はできたので次は 3 次元の問題を考えてみましょう。3 次元の場合はピクセルデータではなくボクセルデータを使います。3 次元で考えるにあたっては、まず「3 次元空間での穴とは

何か？」という問題を考える必要があります。
図 2(a) のような筒状の物体を考えてみましょう。確かにこれには穴が 1 つあいていると言えそうです。次に図 2(b) のような風船を考えてみましょう。するとこれは中に空気が入っているわけですから、何か穴のようなものが 1 つありそうだ、と言えます。穴というよりも空洞というべきかもしれません。
この 2 つの「穴」「空洞」はちょっと考えると別物であることがわかります。「穴」のほうは外側から内側を見ることができます。図 2(c) のような曲がったパイプだと内側を見るのは難しいですが、胃カメラとか内視鏡のようなものを持ってくれば内側も見えます。一方「空洞」のほうはそういうことができません。風船に小さな穴でも開けないと内側が見えません。また別の見方として「穴」は紐を通すことができますが、「空洞」は紐を通すことができない、という違いがあります。
図 2(d) のような浮き輪を考えるともっと複雑になります。まず真ん中に 1 つ「穴」があいています。そして浮き輪の空気の入っているところに「空洞」があります。ただこの空洞はさきほどの空洞と少し違い紐をぐるっと通すことができます (図 2(d) の破線)。つまり浮き輪の空気の入っている部分は「穴」と「空洞」の複合体ではないか、と思えてくるわけです。
ここまでで、3 次元で穴の数を数えるにあたっては次のような課題があることがわかりました。

- 考えている穴なるものは「穴」と「空洞」の 2 種類あって区別する必要がありそう
- 浮き輪の例のように「穴」であり「空洞」であるようなものがある

つまり穴や空洞といった概念をちゃんと定義する必要があります。

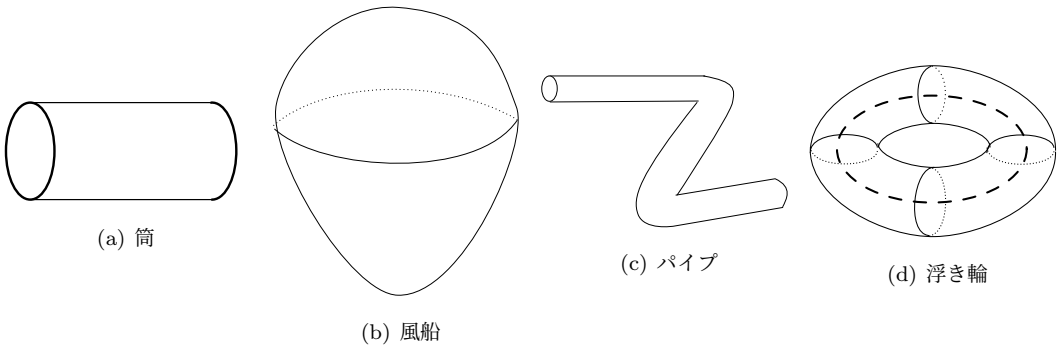


図 2　3 次元の物体の穴

ホモロジー

ここで数学の出番です。幾何学の一分野ではこのような「穴」や「空洞」はホモロジー論と呼ばれる分野で取り扱われます。詳しい理論を知りたい人は参考文献を見て勉強してもらうことにして、簡単に説明しましょう。ホモロジー論においてここでいう「穴」や「空洞」はベクトル空間のよう

なもので [*1] 表されます。そしてそのベクトル空間の次元が穴の個数です。このベクトル空間は

$$H_0(M) : \text{図形の連結成分}$$
$$H_1(M) : \text{穴}$$
$$H_2(M) : \text{空洞}$$

と書くことにします。この M は対象となっている物体のことです。そしてこの次元がそれぞれの個数を表わしています。例えば

$$\begin{array}{lll} \dim(H_0(\text{図 1(a)})) = 2, & \dim(H_1(\text{図 1(a)})) = 3, & \dim(H_2(\text{図 1(a)})) = 0 \\ \dim(H_0(\text{筒})) = 1, & \dim(H_1(\text{筒})) = 1, & \dim(H_2(\text{筒})) = 0 \\ \dim(H_0(\text{風船})) = 1, & \dim(H_1(\text{風船})) = 0, & \dim(H_2(\text{風船})) = 1 \\ \dim(H_0(\text{浮き輪})) = 1, & \dim(H_1(\text{浮き輪})) = 2, & \dim(H_2(\text{浮き輪})) = 1 \end{array}$$

となります。平面に空洞はないので $\dim(H_2(\text{図 1}(a))) = 0$ と定義されます。浮き輪では、「穴」の数に相当するものが 2 に、「空洞」の数に相当するものが 1 になっています。内側の空洞が「穴」と「空洞」の両方の性質を持っているのではないか、というさきほどの話に対応しています [*2]。もっと大きな k に対しても H_k が定義されますが、この場合はすべて 0 になります。4 次元の図形などを考えると、「『穴』でも『空洞』でもないけどそれに似た何か」が表れてくるため、それを表すためにより大きな k に対しても H_k が定義されています。

計算機によるホモロジーの計算

ある一定の条件の下、ホモロジーは計算機によって計算できます [*3]。これは計算ホモロジーと呼ばれています。計算のためのソフトウェアはいくつかあるのですが、ここでは Vidit Nanda が開発した Perseus [*4] を使ってみましょう。

まずこの URL のページの Source Code, Executables and Usage をクリックして Here のところからソースの zip ファイルをダウンロードできます。Windows 用のバイナリなども同じところから取得できるので、そちらを使ってもよいでしょう。コンパイルする場合は、これを適当なディレクトリに展開して、

```
g++ -fpermissive -O2 Pers.cpp -o perseus
```

などとしてコンパイルできます。

[*1] $\mathbb{Z}$ 係数加群、つまりアーベル群です。

[*2] もちろんそんなにいいかげんな話ではなく、ちゃんと数学的に根拠があってこの値になっています。

[*3] ホモロジーを勉強したことのある人に対する説明: 単体ホモロジーは単体複体で表現された図形に対して定義されます。そして単体複体は有限個の単体の組み合わせとして実現されます。各単体の境界作用素 (∂) が定義されれば各単体を基底とする自由アーベル群によってホモロジー群が定義できます。代数的には $\mathbb{Z}$ 係数の有限次元線形代数で、境界作用素 ($\mathbb{Z}$ 線形写像) の像やカーネルは計算機で計算できるので、ホモロジー群の次元が計算できるわけです。単体を立方体に取り替えて同じことができます (Cubical homology といいます)。

[*4] `http://www.math.rutgers.edu/~vidit/perseus/index.html`

とりあえず例として 図 1(a) の 30x30 の画像を計算してみましょう。データファイル [*5] をダウンロードしてください。データは

```
2
30
30
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
-1 -1 -1 -1 -1 -1 -1 -1 -1  1 ...
-1 -1 -1 -1 -1  1  1  1  1  1 ...
-1 -1 -1 -1 -1  1  1  1  1  1 ...
-1 -1 -1 -1 -1  1  1  1  1  1 ...
```

という形式で、最初の 3 行は図形の次元、X 座標方向のピクセル数、Y 座標方向のピクセル数を表し、それ以降は 1 でピクセルが存在することを、-1 で存在しないことを表現しています。

```
./perseus cubtop bitmap.txt
```

で計算します。いくつかファイルが出力されますが、ここでは output_betti.txt を見てください。

```
Frame [1]: 2 3
```

と出力されているはずです。この 2 と 3 が $\dim(H_0)$ と $\dim(H_1)$ です。連結成分が 2 個、穴が 3 つあることが計算できています。
次に 3 次元の筒 [*6]、風船 [*7]、浮き輪 [*8] の 3 つのデータについて perseus を使いましょう。3 次元を 1 と-1 で埋めていくとファイルが大きすぎるので別のフォーマットを使います。cylinder.txt を見ると

```
3
18 44 10 1
18 44 11 1
18 44 12 1
18 44 13 1
18 44 14 1
    :
```

というデータになっています。最初の 3 は次元です。それ以降の行はボクセルの x,y,z 座標です [*9]。最後の 1 はここではとりあえずすべて 1 にしておけば OK です。このファイルに対し

[*5] http://www.kmc.gr.jp/~ohai/kmcpress2013winter/bitmap.txt
[*6] http://www.kmc.gr.jp/~ohai/kmcpress2013winter/cylinder.txt
[*7] http://www.kmc.gr.jp/~ohai/kmcpress2013winter/balloon.txt
[*8] http://www.kmc.gr.jp/~ohai/kmcpress2013winter/torus.txt
[*9] これらのデータは gnuplot で `splot "cylinder.txt"` などとすることで 3 次元に表示できます。本当にこの

```
./perseus scubtop cylinder.txt
```

として、output_betti.txt を見ると

```
Frame [1]: 1 1
```

と出力されます。balloon.txt, torus.txt ではそれぞれ

```
Frame [1]: 1 0 1
```

と

```
Frame [1]: 1 2 1
```

と出力されます。上で説明した $\dim(H_0), \dim(H_1), \dim(H_2)$ に一致しています。このようにして、図形の穴や空洞の個数を計算機で計算できます。

最後に

3 次元の穴の個数を数えるのが何の役に立つのか、というのを最後にちょっと考えましょう。軽石のような多孔質の材料やスポンジの穴を数えるというのはどうでしょう。穴や空洞の数や大きさはこれらの材料の機能に重要な役割を果たしていると考えられます。実は穴や空洞の数を数えるだけでなく、その大きさを「なんとなく」測ることもできるので [*10]、これを使って新材料の性能評価ができるかもしれません。現在は何に使えるかを模索している段階です。他にも金属ガラスや蛋白質の構造解析のような材料科学、画像処理、機械学習などへの応用が試みられています。

参考文献

平岡裕章. タンパク質構造とトポロジー:パーシステントホモロジー群入門. 共立出版, 東京, 2013.

データが筒や浮き輪を表しているのか気になる方はこのようにして確認すればよいでしょう。

[*10] persistent homology というアイデアを用います。Perseus の 3 次元入力での各行の最後の 1 や、出力で無視したファイルは実はこれに関連しています。

C++11 コア言語編

hatsusato

KMC 2 回生の hatsusato です。この記事では C++11 で新しく加わったコア言語の新機能をひと通り解説してみようと思います。突然何を言うかと思うかもしれませんが、私は実は春合宿 [*1] で同様のプレゼンをしていて、これは焼き直し [*2] なのです。スライドと違って部誌は紙幅がたくさんあるので、のびのびいこうと思います。

C++11 の新しい言語機能解説

おしながき

1. `auto`
2. `nullptr`
3. 初期化リスト
4. 統一された初期化構文
5. 範囲 `for`
6. 右辺値参照とムーブセマンティクス
7. 参照修飾子
8. `decltype` と後置戻り値構文
9. ラムダ
10. 可変長テンプレート
11. 強く型付けされた列挙型
12. コンパイル時定数式
13. 非静的メンバの初期化
14. 関数の `default`・`delete` 指定
15. 委譲コンストラクタ
16. 継承コンストラクタ
17. `static_assert`
18. `noexcept`
19. Unicode 文字列リテラル
20. raw 文字列リテラル
21. ユーザ定義リテラル
22. `explicit` 変換演算子
23. エイリアステンプレート
24. デフォルトテンプレート引数
25. 局所型や無名型のテンプレート引数渡し
26. `extern` テンプレート宣言
27. 拡張 `sizeof` と拡張 `friend` 宣言
28. 縮小変換の禁止
29. 無制限共用体
30. アトリビュート
31. C99 の機能
32. その他の機能

[*1] KMC のビッグイベントの一つ。部員総出で山篭りしてなんやかやする。
[*2] ここ (`http://www.slideshare.net/KMC_JP/more-c11`) とほとんど同じ。

auto

今まで長らく、auto キーワードはほとんど無意味な記憶クラス指定子 [*3] でした。この度、auto から記憶クラス指定子としての機能を奪い、後方互換性を犠牲にして、新たに型推論という強力な機能を得ました。
通常の変数の初期化において、変数の型名として auto を用いると、コンパイラが右辺の初期化式の型で auto を自動で置き換えてくれます。例えば、3 行目の Init 関数がテンプレートやオーバーロードを駆使していて、引数の型に応じて戻り値の型が様々に変化するようなものだったりすると、z の型を何にすればいいのか考えるのが困難になることがあります。そんなときでも、auto を使えば z の型をコンパイラが考えてくれるので、z を簡単に初期化できます。また、4 行目のような表現できない型の変数や、6 行目のような長い型名の変数でも、簡単に初期化できます。その他、auto キーワードは後置戻り値構文（後述）などで、型のプレースホルダーとしても活躍しています。

```
1 auto x = 42;  // xはint型
2 auto y = 0.0f;  // yはfloat型
3 auto z = Init(x, y);  // zはInit(x, y)の戻り値の型
4 auto f = [](){};  // fはラムダ式の型
5 const std::vector<std::string> v = InitVectorString();
6 for (auto it = begin(v); it != end(v); ++it) {
7   // itの型はstd::vector<std::string>::const_iterator
8 }
```

nullptr

今まで長らく、ヌルポインタとして NULL [*4] や 0 が使われてきました。ポインタ演算の文脈においては 0 をヌルポインタとして扱う、という特別ルールを運用することで何とかしていました。しかし、0 は整数型であるため、サンプルコードのように 0 が整数としてもポインタとしても解釈できてしまってオーバーロードが曖昧になる問題に対処できません。
C++11 では新しくヌルポインタを表す std::nullptr_t 型のキーワード、nullptr が導入されました。std::nullptr_t 型は任意のポインタ型のみへ暗黙変換できるので、従来のような 0 のポインタへの流用に付随する問題が発生しません。これからはヌルポインタとして nullptr を使うべきです。

```
1 void f(int) { std::cout << "int" << std::endl; }
```

[*3] 自動変数を意味する記憶クラス指定子だった。デフォルトで変数は自動変数になるので auto をつける意味はない。C 言語の基になった B 言語との後方互換性のために存在していた。
[*4] 「#define NULL 0」などとなっていることが多い。

```
2 void f(char*) { std::cout << "char*" << std::endl; }
3 //f(NULL); コンパイルエラー：曖昧なオーバーロード
4 f(nullptr);  // void f(char*)が呼び出される
```

初期化リスト

標準ライブラリに initializer_list クラスの入った<initializer_list>ヘッダが新しく加わりました。C 言語での配列や構造体 [*5] の初期化と同じ要領で、{ } の間に T 型の要素を並べたものを initializer_list<T>型のオブジェクトとして扱うことができるようになります。また、標準コンテナが initializer_list を引数に取るコンストラクタを持つようになりました。initializer_list を上手に用いて便利に初期化しましょう。

```
1  // vを具体的な要素で初期化
2  std::vector<double> v = {0, 1, 2.71, 3.14};
3  // initializer_listは普通値渡しで使う
4  void f(std::initializer_list<int> l) {
5    for (const auto& elem : l) {
6      std::cout << elem << std::endl;
7    }
8  }
9  // 関数の引数に渡す
10 f({0, 12, 345, 6789});
11 f({});  // 空のinitializer_list
```

統一された初期化構文

これまで、初期化構文には関数形式のコンストラクタ呼び出しや、上述の配列や構造体の初期化リスト形式がありました。C++11 からはコンストラクタ呼び出しを { } を用いて行うことができるようになりました。これにより、全ての初期化を { } で統一的に表現できるようになります。この形式は上述の initializer_list とも非常に親和性が高いです。初期化の際にはこの形式を積極的に使うと良いと思います。ただし、コンストラクタオーバーロードに initializer_list を受け取るコンストラクタがあった場合、オーバーロード解決は initializer_list が優先されるので注意しましょう。

```
1 class Point {
2  public:
3   Point(int x, int y) : point_{x, y} {}
```

[*5] ここでの配列や構造体のことを集成体型 (aggregate) という。

```
 private:
  std::pair<int, int> point_;
};
// 以下のいずれもwell-formed
Point p1 = Point{1, 2};
Point p2 = {1, 2};
Point p3{1, 2};
Point* p = new Point{1, 2};
```

範囲 for

ループを書くときによく現れる、コンテナ内の全要素を走査するコードを簡単に書くことのできる構文糖衣が追加されました。使い勝手はサンプルコードのとおりです。要素の型は const auto&で受けるのが普通です。コンテナの内容を変更したいなら auto&で受けて使いましょう。

```
for (const auto& x : {1, 1, 2, 3, 5, 8, 13, 21, 34, 55}) {
  std::cout << x << std::endl;
}
std::string str{"abcd"};
for (auto& c : str) {
  c += 4;
}
std::cout << str << std::endl;  // efgh
```

右辺値参照とムーブセマンティクス

C++11 から、新たに右辺値参照という概念が加わりました。そもそも、左辺値とは代入文の左辺に来られるような名前のあるオブジェクトのことで、右辺値とは、一時オブジェクトやリテラルのような無名オブジェクトのことです [*6]。右辺値参照とは、右辺値のみを指すことができる参照のことで、&&を用いて表します。右辺値参照によって束縛されたオブジェクトは一時的なオブジェクトなので、そのオブジェクトの寿命を気にせずに扱うことができます。これにより、ムーブという、コピーとは異なる新たな概念を表現できるようになります。

ムーブは、関数の引数などを右辺値参照で受け取り、その引数を破壊的に [*7] 利用することで実現されます。例えば、15 行目のムーブコンストラクタでは、引数の情報は*this によって奪われています。この機構を導入することで、内部で暗黙的に行われていたコピーの省略による一層の効率

[*6] このあたりの記述は厳密ではない。厳密な定義が知りたければ rvalue、xvalue、prvalue あたりを検索するとよい。

[*7] もちろん必ず破壊的に利用しなければいけないわけではない。破壊的に利用することができるということ。

化が期待できます。「ムーブセマンティクス」と呼ばれるこの機構を利用すると、意味的にコピーが適切でないクラスもうまく表現することができます。
STL がこの右辺値参照に対応しているので、私たち一般ユーザはただ STL を利用するだけで、ユーザコードを一切変更することなくこの効率化の恩恵を受けることができます。

```
class String {
 public:
  String(const char* src)
      : len_{std::strlen(src)},
        str_{nullptr} {
    str_ = new char[len_ + 1];
    std::copy_n(src, len_ + 1, str_);
  }
  ~String() {
    delete[] str_;
  }
  // 委譲コンストラクタを用いている
  String(const String& src) : String{src.str_} {}
  // ムーブコンストラクタ
  String(String&& src) : len_{src.len_}, str_{src.str_} {
    src.len_ = 0;
    src.str_ = nullptr;
  }
  // ムーブ代入演算子
  String& operator=(String&& src) {
    len_ = src.len_;
    str_ = src.str_;
    src.len_ = 0;
    src.str_ = nullptr;
    return *this;
  }
 private:
  size_t len_;
  char* str_;
};
```

参照修飾子

上述の右辺値参照の導入に伴って、クラスオブジェクトが左辺値の場合と右辺値の場合とでメンバ関数をオーバーロードする需要が生まれました。それを実現するのが参照修飾子です。メンバ関数宣言の const・volatile 指定の後、例外指定の前に&または&&を置くことで指定します。*this が左辺値のときは&をつけた方に、右辺値のときは&&をつけた方にオーバーロード解決されます。参照修飾子は従来通り省略できますが、省略したメンバ関数と指定したメンバ関数とでオーバーロードはできません。ちなみに、const &&は意味がない [*8] のでオーバーロードの delete 指定（後述）に用いるぐらいの用途しかありません。

```
1  struct Example {
2    Example() = default;
3    void f() & { std::cout << "lvalue" << std::endl; }
4    void f() const & { std::cout << "const lvalue" << std::endl; }
5    void f() && { std::cout << "rvalue" << std::endl; }
6    // void f() const && { std::cout << "nonsense" << std::endl; }
7  };
8  Example g() { return Example{}; }  // 戻り値は右辺値になる
9  Example e;
10 const Example ce;
11 e.f();  // lvalue
12 ce.f();  // const lvalue
13 g().f();  // rvalue
```

decltype と後置戻り値構文

コンパイル時に式の型を取得できる decltype 演算子がキーワードに追加されました。sizeof 演算子と同じ要領で用いることができます。

関数の戻り値を後置する記法が導入されました。9 行目の mul2 の要領で表現します。この記法の導入により、特に一部のテンプレート関数において困難だった、戻り値の型の表現が容易になりました。どれだけ簡単になるかはサンプルコードを見れば一目瞭然です。また、この後置構文はラムダ式での戻り値の表現でも用いられています。

```
1  // 後置戻り値構文を使わない場合
2  template <class T, class U>
3  decltype((*static_cast<T*>(nullptr)) *
```

[*8] &&は破壊的変更ができることが最大にして唯一の利点。const をつけて変更なしに扱いたいのなら const &で十分。

```
4           (*static_cast<U*>(nullptr))) mul1(T x, U y) {
5   return x * y;
6 }
7 // 後置戻り値構文を使う場合
8 template <class T, class U>
9 auto mul2(T x, U y) -> decltype(x * y) {
10  return x * y;
11 }
```

ラムダ

ラムダ式と呼ばれる無名関数オブジェクトを簡便に作成する方法が導入されました。ラムダ式の基本形は `[]() -> 戻り値型 {}` となります。

`[]` をキャプチャといい、この中にラムダ式の作られたスコープから見える変数を並べることで、その変数をラムダ式（が表現する無名関数オブジェクト）に取り込むことができます。変数名をそのまま並べるとコピーキャプチャ、変数名の前に`&`をつけるとリファレンスキャプチャとなります。コピーキャプチャは元の変数のコピーを、リファレンスキャプチャは元の変数への参照を無名関数オブジェクトのメンバとして取り込みます。そのため、リファレンスキャプチャの場合は参照先の変数の寿命を意識する必要があります。

`()` は仮引数リストです。仮引数リストは省略することができ、その場合引数なしの関数になります。

戻り値型の部分に戻り値の型を指定します。`-> 戻り値型` は省略することができます。戻り値の型を省略し、かつ関数の定義が `return` 文 1 文のみの場合、戻り値の型はコンパイラが推論してくれます。戻り値の型を省略していて、かつ関数の定義が `return` 文 1 文のみではない場合、戻り値の型は void とみなされます。そのため、関数定義に複数の文を書いていて、かつ値を返す場合、必ず戻り値の型を指定する必要があります。

`{}` には関数定義を書きます。普通の関数と同様に書くことができます。

ラムダ式を用いると、その場一度きりの関数オブジェクトの型に名前をつけて名前空間を汚すということがないので、非常に便利です。

```
1 std::vector<int> v(10);
2 std::iota(begin(v), end(v), 1);
3 int sum = 0;
4 std::for_each(begin(v), end(v), [&sum](int x) { sum += x; });
5 std::cout << sum << std::endl;  // 55
```

可変長テンプレート

可変個の引数を取るテンプレートを作成する方法が導入されました。これを用いると、タプル [*9] や型安全な可変長引数関数を作成できるようになります。

9 行目のように、テンプレート仮引数の前にパック演算子... をつけたものをテンプレート仮引数パックといい、0 個以上の任意個のテンプレート実引数の列を表します。非型テンプレート仮引数やテンプレートテンプレート仮引数も同様にパックにすることができます。可変長引数関数を作る場合は、10 行目のように関数仮引数の前にパック演算子... をつけた関数仮引数パックを用いて仮引数を宣言します。14 行目のように関数仮引数パックの後にアンパック演算子... をつけることで、カンマで区切られた仮引数の列に展開することができます。テンプレート仮引数パックの場合も同様に、後にアンパック演算子をつけることでカンマで区切られた型の列に展開することができます。

また、仮引数パックに対して用いることで仮引数の個数を返す sizeof... 演算子が新たに導入されました。

```
1  void print(const char* s) {
2    while(s && *s) {
3      if (*s == '%' && *++s != '%') {
4        throw std::runtime_error("missing arguments");
5      }
6      std::cout << *s++;
7    }
8  }
9  template <typename T, typename... Args>
10 void print(const char* s, T value, Args... args) {
11   while (s && *s) {
12     if (*s == '%' && *++s != '%') {
13       std::cout << value;
14       return print(++s, args...);
15     }
16     std::cout << *s++;
17   }
18   throw std::runtime_error("extra arguments");
19 }
20 print("Hello C++%d\n", 11);  // Hello C++11
```

[*9] 標準ライブラリに std::tuple として実装されている。

強く型付けされた列挙型

これまでの enum は、型安全でなく、前方宣言できず、外部の名前空間を侵していました。C++11 からは enum class という強く型付けされた enum が導入され、列挙型を型安全に扱うことができるようになりました。

enum class は 1 行目のように前方宣言できます。enum class には 2 行目のように内部型という基底となる整数型を指定できます。3 行目のように省略するとデフォルトで内部型は int 型になります。前方宣言の内部型を省略すると内部型は int 型であるものとして解釈されます。enum class にはスコープがあるので、4 行目のように必ずスコープ解決する必要があります。enum class の列挙子は整数型に暗黙変換されないので、5 行目のように明示的に型変換する必要があります。

これまでの enum は明示的にスコープ解決できませんでしたが、C++11 からは明示的なスコープ解決が許容されるようになりました。

```
1 enum class TrafficLight : int;  // 前方宣言
2 enum class Color : char { red, green, blue, }; // charベース
3 enum class TrafficLight { red, yellow, green }; // intベース
4 TrafficLight tl = TrafficLight::red;  // 必ずスコープを指定する
5 char c = static_cast<char>(Color::red);  // 明示的な型変換が必要
6 enum Number { ZERO, ONE, TWO, THREE };  // 古い形式
7 int n = ONE;  // 以前はスコープを指定できなかった
8 n = Number::TWO;  // C++11からはスコープを明示することもできる
```

コンパイル時定数式

constexpr キーワードが新しく導入され、コンパイル時定数やコンパイル時に実行される関数を表現できるようになりました。

constexpr 指定された変数は、コンパイル時定数式で初期化することでコンパイル時定数として使うことができます。constexpr 指定された関数の定義には typedef 宣言や using 宣言などを除くと、return 文 1 文しか書くことができません。constexpr 関数に渡された全ての実引数がコンパイル時定数のとき、その関数はコンパイル時に実行され、戻り値はコンパイル時定数になります。constexpr 関数に渡された実引数にコンパイル時定数でないものがあったとき、コンパイルエラーにはならず、その関数は実行時に実行されます。

また、コンストラクタに constexpr 指定をすることで、コンパイル時定数の型として使えるリテラル型を定義することができます。このとき、そのリテラル型のメンバは全てリテラル型である必要があり、すべてのメンバを constexpr コンストラクタの初期化リスト内で初期化する必要があ

ります[*10]。

```
enum class Flags : unsigned int {
  good = 0, fail = 1, bad = 2, eof = 4
};
constexpr Flags operator|(Flags f1, Flags f2) {
  return static_cast<Flags>(static_cast<unsigned int>(f1) |
                            static_cast<unsigned int>(f2));
}
switch (flag) {
  case Flags::bad:
    // Do Something
    break;
  case Flags::eof:
    // Do Something
    break;
  // このoperator|はコンパイル時に計算される
  case Flags::bad | Flags::eof:
    // Do Something
    break;
  default:
    // Do Something
    break;
}
```

非静的メンバの初期化

これまでは、クラス内で初期化できるメンバは static const な整数型に限られていました。C++11 からは 7、8 行目のように、任意のメンバの宣言の場所にデフォルトの初期値を与えられるようになりました。既にデフォルトの初期値が与えられているメンバをコンストラクタで初期化すると、デフォルトの初期値が上書きされます。コンストラクタで初期化しなければ、デフォルトの初期値で初期化されます。

```
class Example {
 public:
  Example() {}  // str_ == "default" && d_ == 3.14
  Example(double src) : d_{src} {}  // str_ == "default"
```

[*10] ここで述べている条件はリテラル型の要件の一部であって、全てではない。詳細は各自調べられたし。

```
5   Example(const std::string& src) : str_{src} {}  // d_ == 3.14
6  private:
7   std::string str_{"default"};
8   double d_{3.14};
9 };
```

関数の default・delete 指定

これまでは、コピーの禁止を表現するときには、コピーコンストラクタとコピー代入演算子を private にするイディオムを用いたりする必要がありました。C++11 からは 3、4 行目のように関数を明示的に delete 指定することで、その関数の使用の禁止を表現することができます。他にも、7 行目のように関数オーバーロードの一部を delete 指定することで、オーバーロードを制限するといったこともできます。また、コンパイラによるデフォルトの定義が存在する特殊メンバ関数 [*11] には明示的に default 指定をすることができます。default 指定された特殊メンバ関数の定義にはコンパイラが暗黙に生成する定義が用いられます。

```
1 struct Uncopyable {
2   Uncopyable() = default;  // 明示的にデフォルトコンストラクト
3   Uncopyable(const Uncopyable&) = delete;  // コピー禁止
4   Uncopyable& operator=(const Uncopyable&) = delete; // 代入禁止
5 };
6 template <typename T>
7 void f(T) = delete;  // fにはint型以外の型の変数を渡せない
8 void f(int) {}  // fにはint型の変数のみを渡せる
```

委譲コンストラクタ

6 行目のように、コンストラクタの初期化リストにおいて別のコンストラクタを呼び出せるようになりました。これにより、たくさんのコンストラクタオーバーロードがあっても、冗長性の少ないコードを書くのが簡単になりました。

```
1 class Month {
2  public:
3   Month(int m) : month_{m} {
4     assert(0 < m && m <= 12);
```

[*11] デフォルトコンストラクタ・コピーコンストラクタ・コピー代入演算子・ムーブコンストラクタ・ムーブ代入演算子・デストラクタの 6 種類のメンバ関数のこと。これらにはいずれにもコンパイラが暗黙に生成するデフォルトの定義が存在する。

```
5   }
6   Month(const std::string& src) : Month{std::stoi(src)} {}
7  private:
8   int month_;
9 };
```

継承コンストラクタ

これまでは、基底クラスのメンバ関数が派生クラスのメンバ関数のオーバーロードによって隠蔽されてしまうときのために、using 宣言を用いて基底クラスのメンバ関数を派生クラスのスコープに取り込む機能が存在しました。C++11 からは、それに加えて基底クラスのコンストラクタも派生クラスに取り込むことができるようになりました。3 行目のように書くと、派生クラスの初期化は何もせず、基底クラスのコンストラクタをそのまま呼び出すだけのコードを取り込んだことになります。派生クラスにメンバがあるときは、上述の非静的メンバの初期化を用いて初期化する必要があります。

```
1 class Derived : public Base {
2  public:
3   using Base::Base;  // Baseのコンストラクタを利用する
4  private:
5   int x_{0};
6 };
```

static_assert

assert マクロのコンパイル時版である、static_assert キーワードが新しく追加されました。第一引数に定数式、第二引数に文字列リテラルを受け取ります。コンパイル時に第一引数を評価して bool に変換し、true なら何もせず、false ならコンパイルエラーとなって、第二引数の文字列がコンパイルエラーメッセージとして出力されます。この static_assert を用いると、constexpr やテンプレートメタプログラミングなどのコンパイル時計算のデバッグが比較的容易になります。

```
1 template <class T, class U>
2 struct S {
3   T first;
4   U second;
5 };
6 template <class T, class U>
7 void f() {
```

```
8   static_assert(sizeof(S<T, U>) == sizeof(T) + sizeof(U),
9                 "padding exists in S");
10  }
```

noexcept

これまでは、関数の例外指定には throw を用いましたが、C++11 から throw 例外指定は非推奨になりました [*12]。代わりに新しく noexcept 演算子がキーワードに追加されました。noexcept 演算子は、引数にとった式が例外を投げうる式を含む場合 false を、含まない場合 true を返します。

noexcept 演算子の引数に bool 型の定数式を渡したものを関数宣言に用いることで、例外指定をすることができます。noexcept 演算子に渡された定数式が true のときは例外を投げないことを、false のときは例外を投げうることをその関数に指定します。よって、特定の式が例外を投げうるかどうかに依存して、関数が例外を投げうるかどうかが変わるような例外指定をするときは、2 行目のように noexcept を 2 つ組み合わせて用いることで表現します。また、5 行目のように noexcept 演算子の引数を省略した場合、noexcept(true) を表します。例外によって例外指定が noexcept(true) な関数から抜けだすと、実行時に std::terminate が呼び出され強制終了します。

```
1  void swap(Example& lhs, Example& rhs)
2      noexcept(noexcept(lhs.swap(rhs))) {
3    lhs.swap(rhs);
4  }
5  int twice(int val) noexcept {
6    return 2 * val;
7  }
```

Unicode 文字列リテラル

これまで、ワイド文字（列）リテラルを表現するプレフィックスとして L が存在しました。

C++11 からは、UTF-8、UTF-16、UTF-32 文字（列）リテラルを表現するために、u8、u、U の 3 つのプレフィックスが追加されました。この 3 つのプレフィックスを文字（列）リテラルの先頭につけることで、それぞれ char 型、char16_t 型、char32_t 型の、UTF-8、UTF-16、UTF-32 文字（列）を表現できるようになりました [*13]。

また、\u の後に 4 桁、\U の後に 8 桁の 16 進数を指定することで、ユニバーサル文字名を指定で

[*12] 例外指定子としての throw は非推奨となったが、例外を投げる throw 式での役割は健在である。

[*13] ただし、u8 プレフィックスは文字リテラルの先頭につけることができない。

きるようになりました。

```
1 wchar_t wc = L'あ';
2 char16_t u16c = u'\u3042';  // \u3042 == あ
3 char32_t u32c = U'\U00003042';
4 const char* u8s = u8"UTF-8文字列";
5 const wchar_t* ws = L"ワイド文字列";
6 const char16_t* u16s = u"UTF-16文字列";
7 const char32_t* u32s = U"UTF-32文字列";
```

raw 文字列リテラル

バックスラッシュがエスケープシーケンスプレフィックスとして解釈されない文字列リテラルである、raw 文字列を表現できるようになりました。文字列リテラルの先頭に R プレフィックスをつけると、"delimiter(と)delimiter"に挟まれた部分が raw 文字列としてそのまま解釈されます。ただし、delimiter は 0 文字以上 16 文字以下の他の文字列で置き換えることができます。また、R プレフィックスの前には L、u8、u、U プレフィックスも指定できます。

```
1 const char* rs = R"(/\w\\\w/)";  // /\w\\\w/
2 const char* u8rs = u8R"(("raw\str"))";  // ("raw\str")
3 const wchar_t* wrs = LR"+("(raw\str)")+";  // "(raw\str)"
4 const char16_t* u16rs = uR"***(("))***";  // (")
5 const char32_t* u32rs = UR"delimiter(("))delimiter";  // (")
```

ユーザ定義リテラル

リテラルの末尾につけることでリテラルをユーザ定義の方法で変換することができる、ユーザ定義リテラルを定義できるようになりました。ユーザ定義リテラル演算子 operator"" _**識別子** () をオーバーロードすることで実現します。このとき、ユーザ定義リテラルの識別子は必ずアンダースコア 1 つから始まる必要があります [*14]。

数値型に対するユーザ定義リテラルでは、リテラルの情報を raw リテラルまたは cooked リテラルで受け取ります。raw リテラルは、const char*型または template <char...>テンプレート仮引数パックを介して受け取ります。cooked リテラルは、整数型の場合 unsigned long long int 型、浮動小数点数型の場合 long double 型で受け取ります。ユーザ定義リテラルでは負数を受け取ることができないので、単項マイナス演算子はユーザ定義リテラルの戻り値に適用されます。

[*14]通常、アンダースコア 1 つから始まる名前は、グローバル名前空間において予約されているが、ユーザ定義リテラルの識別子は例外となっている。また、アンダースコア 1 つで始まりその次が大文字の名前などといった予約された名前は当然避ける必要がある。

文字型に対するユーザ定義リテラルでは、リテラルの情報を char 型で受け取ります。リテラルがワイド文字、UTF-16 文字、UTF-32 文字の場合は、それぞれ wchar_t 型、char16_t 型、char32_t 型で受け取ります。
文字列型に対するユーザ定義リテラルでは、リテラルの情報を (const char*, size_t) で受け取ります。第一引数は NULL 終端された文字列、第二引数には NULL 終端文字を除いた文字数が入ります。リテラルがワイド文字列、UTF-16 文字列、UTF-32 文字列の場合は、const char*の char の部分がそれぞれ wchar_t、char16_t、char32_t に置き換わります。

```
template <char... Chars>
std::string operator"" _to_string() {
  std::string ret;
  for (const auto& c : std::initializer_list<char>{Chars...}) {
    ret.push_back(c);
  }
  return ret;
}
std::string operator"" _s(const char* str, size_t) {
  return std::string{str};
}
constexpr std::complex<double> operator"" _i(long double d) {
  return {0.0, static_cast<double>(d)};
}
std::string str1 = 12345_to_string;  // "12345"
std::string str2 = "std::string"_s.substr(0,3);  // "std"
// std::complex<double>{2.0, 3.0}
std::complex<double> c = 2.0 + 3.0_i;
```

explicit 変換演算子

これまでは explicit キーワードはコンストラクタにしか指定できませんでしたが、変換演算子にも指定できるようになりました。

```
template <class T>
class SmartPtr {
 public:
  explicit SmartPtr(T* src) : ptr_{src} {}
  ~SmartPtr() { delete ptr_; }
  explicit operator bool() {
    return ptr_ != nullptr;
```

```
8    }
9   private:
10   T* ptr_;
11 };
```

エイリアステンプレート

2 行目のように、using キーワードを用いて、テンプレート引数を部分的に束縛したテンプレートのエイリアスを書くことができるようになりました。また、5 行目のように、テンプレートのエイリアスとしてだけではなく typedef の代わりとしても using を使うことができるようになりました。

```
1 template <class T>
2 using MyVec = std::vector<T, MyAllocator<T>>;
3 MyVec<int> v;  // std::vector<int, MyAllocator<int>>型
4 typedef int(*FUNCPTR_OLD)();  // 従来の宣言方法
5 using FUNCPTR = int(*)();  // これからの宣言方法
```

デフォルトテンプレート引数

これまでデフォルトテンプレート引数はクラステンプレートにしか指定できませんでしたが、テンプレート関数にも指定できるようになりました。

```
1 template <class T, class U = double>
2 void f(T t = 0, U u = 0);
3 void g() {
4   f(1, 'c');  // f<int,char>(1, 'c')
5   f(1);  // f<int,double>(1, 0)
6   f<int>();  // f<int,double>(0, 0)
7   f<int, char>();  // f<int, char>(0, 0)
8 }
```

局所型や無名型のテンプレート引数渡し

これまではテンプレート引数として受け取れなかった局所型や無名型を、テンプレート引数として受け取れるようになりました。

```
1 template <class T>
2 void f(T) {}
```

```
enum {e};
struct {} noname;
int main() {
  struct Local {} local;
  f(e);
  f(noname);
  f(local);
}
```

extern テンプレート宣言

これまでは翻訳単位ごとにテンプレートがインスタンス化されることで、コンパイル時間とオブジェクトサイズが大きくなりがちでした。C++11 からは、extern キーワードをつけてテンプレート宣言することで、テンプレートのインスタンス化を抑制できるようになりました。例えば、サンプルコードのように 3 つの翻訳単位、one.cpp、two.cpp、three.cpp があるとします。これらのいずれでも std::vector<int>は明示的に宣言されていて、利用することができます。C++11 以前ではこのとき、各翻訳単位において 1 つずつ std::vector<int>の実体が存在していました。しかし、C++11 以降では、two.cpp と three.cpp のように extern テンプレート宣言をすることで、std::vector<int>のインスタンス化を抑制することができます。これにより、two.cpp と three.cpp のコンパイル時間とオブジェクトサイズを削減することができます。two.cpp と three.cpp における std::vector<int>の実体は、あとでリンク時に one.cpp からもたらされます。

extern テンプレート宣言を利用するときは、いずれか 1 つの翻訳単位においてのみでテンプレートの明示的インスタンス化が行われたほうが効率が良くなります。そのため、どの翻訳単位でインスタンス化をするかを注意深く考える必要があります。

```
// one.cpp
template class std::vector<int>;
// テンプレートの明示的インスタンス化

// two.cpp
extern template class std::vector<int>;
// テンプレートのインスタンス化を抑制

// three.cpp
extern template class std::vector<int>;
// テンプレートのインスタンス化を抑制
```

拡張 sizeof と拡張 friend 宣言

これまでは sizeof 演算子でメンバ変数のサイズを取得するためには、そのメンバ変数が static であるか、実体を持っている必要がありました。C++11 からはインスタンス化をすることなくメンバ変数のサイズを取得できるようになりました。
これまでは friend 宣言にテンプレート引数や typedef 名を指定できませんでしたが、C++11 からはできるようになりました。

```
1  template <class T>
2  struct Example {
3    int hoge;
4    friend T;
5  };
6  struct Foo {};
7  // メンバ変数のサイズを取得
8  constexpr size_t hogesize = sizeof(Example<int>::hoge);
9  Example<Foo> ef;  // Fooは Example<Foo>の friend
10 Example<int> ei;  // friend intは無視される
```

縮小変換の禁止

統一された初期化構文 { } を用いて初期化 [*15] する際に、縮小変換が起こるとコンパイルエラーになります。整数型と浮動小数点数型との間の暗黙変換は縮小変換です。整数型同士、浮動小数点型同士の間の変換でも、変換先の型が変換元の型の値を全て表現できないとき、縮小変換となります。しかし、変換元が定数式で、その値が変換先の型で表現可能 [*16] なときは、縮小変換には当てはまりません。

```
1  double d{3.14};
2  // int i1{3.14}, i2{d}; コンパイルエラー：doubleからintへの変換
3  // コンパイルエラー：intからdoubleへの変換
4  // std::vector<double> vi = {i1, i2, d};
5  char c1{42};
6  // char c2{12345}; コンパイルエラー：char型で表現できない初期値
```

[*15] ここでの初期化は initializer_list や集成体型 (配列や構造体) の初期化も含む。
[*16] 変換先の型が浮動小数点数型の場合、定数式の値を正確に表現できる必要はない。

無制限共用体

これまでは、ユーザ定義のコンストラクタやデストラクタを持つクラスは共用体のメンバになることができませんでしたが、C++11 からはそういった制限が撤廃され、static でない参照を除く全てのオブジェクトが共用体のメンバになることができるようになりました。
共用体のメンバがユーザ定義の特殊メンバ関数を持つ場合、それらに対応する共用体の特殊メンバ関数が暗黙に delete されます。その共用体がクラス内の無名共用体の場合は、その無名共用体をメンバに持つクラスの対応する特殊メンバ関数が暗黙に delete されます。共用体のメンバの内、1 つだけなら非静的メンバの初期化を用いて直接デフォルト初期値を指定することができます。

```
1  class Example {
2    enum class Tag { number, text };
3    public:
4    Example(int src) : type_{Tag::number}, i_{src} {}
5    Example(const std::string& src) : type_{Tag::text}, s_{src} {}
6    Example(const Example& rhs) : type_{Tag::number}, i_{0} {
7      *this = rhs;
8    }
9    ~Example() {
10     if (type_ == Tag::text) {
11       s_.~basic_string();
12     }
13   }
14   Example& operator=(const Example& rhs) {
15     if (type_ == Tag::text) {
16       if (rhs.type_ == Tag::text) {
17         s_ = rhs.s_;
18         return *this;
19       } else {
20         s_.~basic_string();
21       }
22     }
23     type_ = rhs.type_;
24     switch (type_) {
25       case Tag::number:
26         i_ = rhs.i_;
27         break;
```

```
28        case Tag::text:
29          new(&s_) std::string{rhs.s_};
30          break;
31      }
32      return *this;
33    }
34   private:
35    Tag type_{Tag::number};
36    union {
37      int i_{0};
38      std::string s_;
39    };
40  };
```

アトリビュート

これまではコンパイラごとに存在した属性を指定する独自拡張に対して、統一的な構文が規格に明記されました。任意の追加属性を [[ ]] で囲んで指定します。
変数やクラスのアラインメントを指定する alignas 演算子と、型のアラインメント要求を返す alignof 演算子が追加されました。alignas は渡された定数式の値でアラインメントを要求します。alignof には型名を渡すことで、その型のアラインメント要求を返します。alignas に型名を渡した場合は、alignas(alignof(**型名**)) と同じ事になります。
オーバーライドを明示する override と、オーバーライドの禁止とクラスからの派生を禁止する final が導入されました。override 指定された関数が基底クラスのメンバ関数をオーバーライドしていないとき、コンパイルエラーになります。final 指定された関数をオーバーライドする関数があるとき、コンパイルエラーになります。final 指定されたクラスを継承するクラスがあるとき、コンパイルエラーになります。この override、final の 2 つは文脈キーワードと呼ばれ、仮想関数やクラスへの指定子として使われる文脈でのみキーワードとして扱われ、それ以外の文脈では通常の識別子として使うことができます。

```
1   struct Base {
2   virtual void f() const;
3   [[noreturn]] virtual void g() final;  // 値を返さない
4  };
5  struct Derived : public Base {
6    void f() const override;  // オーバーライドを明示
7    //void g() override; final関数はオーバーライドできない
8  };
```

```
9  // overrideとfinalは識別子として使える
10 alignas(float) unsigned char override[sizeof(float)];
11 constexpr int final = alignof(int);
```

C99 の機能

C99 に入った機能の一部が C++11 にも導入されました。

- long long int
- __func__
- __STDC_HOSTED__マクロ
- _Pragma 演算子
- 可変長引数マクロ
- 空のマクロ引数を許容
- etc.

少なくとも 64bit の整数を格納することができる long long int 型が導入されました。そのマクロが展開される場所の関数名に置き換わる__func__マクロが導入されました。その他いくつかの機能が導入されました。詳細はここでは省略します。

その他の機能

__cplusplus マクロ

__cplusplus マクロの値が 199711L より大きい値に変更されました。具体的には 201103L としているコンパイラが多いようです。

二重閉じ山カッコ

テンプレートにおいて、閉じ山カッコが 2 つ並ぶと右シフト演算子に間違って解釈されてしまう問題が解決されました。

thread_local

各スレッドごとに独立したグローバル変数・静的変数を作ることができる thread_local 記憶クラス指定子が導入されました。

Plain Old Data

POD (Plain Old Data) の定義が整理され、より多くの型が POD に属するようになりました。

Substitution Failure Is Not An Error

SFINAE (Substitution Failure Is Not An Error) の仕様が変更されました。これまでの規格では SFINAE になる場合を列挙していましたが、C++11 からは SFINAE にならない場合を全て列挙し、それ以外は全て SFINAE となるように変更されました。

inline namespace

透過的な名前空間を作成できるようになりました。inline 名前空間スコープの中の名前は、その名前空間の外側のスコープでも使うことができるようになりました。これにより、ライブラリ提供者が、異なるバージョンのライブラリをより柔軟に提供できるようになります。

メモリモデル

C++11 では、マルチスレッドをサポートするメモリモデルが定義されました。これまではプログラム中にただ 1 つの実行スレッドしか存在しない場合に対しての規定しかありませんでした。C++11 では複数の実行スレッドの存在を許容し、それに伴いメモリモデルの定義が見直されました。

ガベッジコレクション

将来のガベッジコレクションの実装を容易にするための仕様が追加されました。ガベッジコレクタの扱うポインタの性質について、いくつかの概念が導入されました。

おわりに

いかがだったでしょうか。ここまで読んだあなたは C++11 をひと通り知ったと言っていいのではないのでしょうか。ただ、本記事ではあくまで各機能を概説するに過ぎないので、より詳細な説明は各自でネットで拾うなり、本を買うなりしてください。
また、タイトルに「コア言語編」とあることから推測されるように、「ライブラリ編」も予定しています。「ライブラリ編」では標準ライブラリの C++11 での変更について述べようと思っています。こちらも「コア言語編」と同様に、まずは次の春合宿で発表するスライド [*17] を作り、その後その焼き直しを部誌に載せようと考えています。乞うご期待。

参考文献

C++11: Syntax and Feature
http://ezoeryou.github.io/cpp-book/C++11-Syntax-and-Feature.xhtml

[*17] これも春合宿での発表後に SlideShare (http://www.slideshare.net/KMC_JP/) にアップロード予定です。

ゆうやけこやけ的物語　ぐうちく衛星 前篇

fuddy

注意書き

この物語は KMC 部員 6 名が集まり、『ゆうやけこやけ』というテーブルトークロールプレイングゲーム (TRPG) のシステムを用いて行われたセッションに基づいたものになっています。
また、この物語は実際のセッションにおけるプレイヤーキャラクターたちの行動に、後付的な解釈や設定が大幅に付け加えられて作られたものです。実際のセッション内容とは、展開を除き、大きく異なります。
そして、この物語は『ゆうやけこやけ』ルールブックでは推奨されていないようなものになっています。ご留意ください。

恵笑市について

どこかにある不思議な場所、恵笑市(けいえむし)。
そこには不思議な力を持つ動物「変化」や不思議な存在「物の怪」、そして土地を守護している「土地神」、名状しがたい「神話生物」が仲良く共存しています。彼らは人間の姿に化け、時に人間にいたずらし、時に人間を助けて日々を暮しています。
そんな恵笑市は 3 つの町からできています。
海に面した商人町、京代町(きょうだいちょう)。
歴史が残る城下町、舞今町(まいこんちょう)。
そして、自然が多く残る田舎町、鞍部町(くらぶちょう)。
今からするのは、鞍部町に関するお話です。

物語の初めに　～ぐうちくと仲間たち

もしもあなたが鞍部町に引っ越して来たら、西の空に緑色の星が輝いていることにきっと気が付くでしょう。おそらくあなたが住んでいる場所からはこの星は見えないでしょうし、恵笑市内でも鞍部町以外の場所では見ることができません。
鞍部町に住んでいる人はみんなこの星のことを知っています。そして、人々はこの名状しがたい緑色に輝く星のことを「ぐうちく衛星」と呼んでいます。
このぐうちく衛星は、つい 1 カ月ほど前（嬉々 25 年 9 月現在。現代日本人が知らないどこかの時空での話）突如として空に現れました。ですが、町の人々はみんな何故このぐうちく衛星が現れたかを知っているので不思議がったりはしません。ここで出会ったのも何かの縁です。あなたにも、「ぐうちく衛星」が現れたいきさつをお教えしましょう。
まず、「ぐうちく」について話さなければなりません。「ぐうちく」というのは鞍部町の旧布団村地区（以下布団村）のあたりに住んでいた狸の変化の名前です。ぐうちくは布団村の西の方にある布団山というところをねぐらにしていました。
布団山にはカエンタケなどの危ないキノコやオニフスベのようなとても大きなキノコが生えており、また茂みにはアオダイショウやマムシが潜んでいて、人間はめったに立ち寄りません。そんな魔境に住むぐうちくの心が清浄なわけもなく（偏見）、ぐうちくは、ぐうの音も出ない畜生なのでした。
ぐうちくは人間に化けることができました（変化と呼ばれる存在は人間に化けることができる）。人間に化けたぐうちくはサッカーボールを抱えたパンチパーマのブラジル人の坊主の姿をしていました（鞍部町にはブラジル人移民が多く、ブラジル人街が形成されているほどなので、ぐうちくの姿は街中ではそこまで目立ちません）。
ぐうちくには 2 人の友達がいました。「きいち」という布団山の北の斜面に住んでいる狐の変化と「しろ」という布団山のふもとに住んでいる見た目が怖い犬の変化です。人間に化けたきいちは、いまどき見かけないようなまるで昔話に出てくる子供のような恰好をした、10 歳くらいの男の子の姿をしています。人間に化けたしろは、犬歯が鋭くひげが生えた白目がちな 5 歳くらいの男の子の姿をしています（どんな 5 歳ですか）。2 匹ともいたずら好きで、ぐうちくと気が合い、よく一緒に行動していました。
きいち、しろ、そしてぐうちくの 3 匹は、しばしば村に降りて悪さを行っていました。きいちは布団村の民家に忍び込み、何かを盗ることを趣味にしていました。しろは収穫されたお米や野菜を食い荒らすことを趣味にしていました。そして、ぐうちくは田畑で作業しているおじいさんの腰を折ることを趣味にしていました。
普通の動物がこのような害を人間になすと猟友会が出張りそうなものですが、3 匹にその心配はありませんでした。なぜならば、この 3 匹は変化なので、人間を出し抜いていたずらすることなどは朝飯前だったからです。人間は 3 匹が行った「いたずら」の原因のしっぽすら掴むことができませんでした。
ですから、布団村に住む人々は原因不明の怪現象に悩まされることになりました。留守にすると冷

蔵庫の中身が空になる、汗水を流して作った農作物はどんなに対策をとっても食い荒らされる、おじいちゃんが腰を折られて町病院に入院する等々。
そのようなことを受けて、布団村は呪われていると考える人たちが現れ、余裕がある人は土地を売って、布団村の隣にある蛇真村や、町のほかの地域に移り住みました。そして村にはあまり裕福ではない人だけが残りました。あまり裕福ではない彼らは先祖代々受け継いだ田畑を放棄することなどはけっしてできず、おびえて暮らす以外にすべはありませんでした。布団村に残った人々はどんどん貧しくなり、数年前に突如現れた宗教団体「主父に暮らす会」にすがって暮らすようになりましたがそれはまた別の話。
とにかく、いたずら好きのぐうちく達は自由気ままに生活していました。そんなぐうちく達にも、頭が上がらない存在が二匹いました。
一匹は、布団山のふもとにある社に祭られている、名状しがたい狐のようなもの「簀斗羅（すとら）」です（明らかに狐じゃないのだが、簀斗羅自身が狐だと言い張っているため狐とする）。
簀斗羅はこどもっぽくていたずら好きです。油揚げを献上しない人間に対して、蛇を投げつけることを趣味にしています。ですが根はいい子で、本当に困っている人や子供に対しては優しく振る舞います。ぐうちくの度が過ぎた行動を諫めることもしばしばあります。ぐうちくは簀斗羅の名状しがたいやばさを動物的直感で嗅ぎ取り、はむかうようなことは行いません。それどころか、自主的に貢物として油揚げを贈ることさえあります（もちろん盗品）。人間に化けた簀斗羅は名状しがたい 6 歳幼女のような姿をしています（頭にアホ毛が生え、目や服の隙間は闇のように暗い）。
そしてぐうちくが逆らえないもう一匹は、約 900 年生きている烏、「バルムンク＝フェザリオン」です（昔、船に紛れて英国からやってきたらしい。以下バルさん）。バルさんは布団村と川を挟んで接している蛇真村（じゃしんむら）にある山、蛇真山に住んでいます。その羽を使って町中を飛び回っているので、町のいろんな地域に住む存在と顔なじみです。ぐうちくもその顔なじみの一人です。
バルさんはとてもお人好しで道徳的な変化です。お人好しなバルさんは人の世話を焼くのが好きです。困っている人がいると放っておけません。また、道徳的なバルさんは不正や度を過ぎたいたずらが大嫌いです。以前、ぐうちくはおじいさんの腰を折ろうとしていたところをバルさんに見られ、ひどい目に遭いました。それ以来、ぐうちくはバルさんに逆らうことはできませんでしたし、きいちやしろも同じようなことがあり、バルさんには逆らえません（でもいたずらはやめない）。
そんなバルさんはおしゃべりが大好きです。自分が体験したことは何でもすぐ人に話してしまいます。ついこの間もザリガニのような天狗と一緒に宇宙に遊びに行った話などをしたり顔で吹聴していました（バルさんの話を聞いて宇宙飛行士を目指している子供がいるとかいないとか）。バルさんに秘密を教えることは、町の人みんなに秘密を教えるのと同じようなものです。ですからバルさんに内緒話をする人はいません。人間に化けたバルさんは羽のようなリボンをつけた幼女の姿をしています（※バルさんは雄です）。
以上がぐうちくとその周りの変化についてのお話です。なんとなくぐうちくがどんな存在か分かったのではないでしょうか。
いままでの話は前置きです。これからが「ぐうちく衛星」の物語です。

簀斗羅の社にて

あれは 1 カ月ほど前、つまり 8 月のことでした。青々とした空には入道雲がふんぞり返り、夏という季節を報せていました。生命力あふれる季節を喜ぶ植物が鞍部町のそこらじゅうで見られました。布団村にある布団山も濃い緑をした木々でびっしりと覆われていました。世間は夏休みの真っただ中ですが、布団山に入るような命知らずな子供は鞍部町にはもちろんいないでしょうし、いたとしても来年の夏にその子はこの世にいないでしょう。生命力があふれた夏の布団山はまさしく魔境で、生きとし生けるものがぎらぎらと獲物がくるのを待ち構えているのです。
そんな布団山のふもとには古ぼけた社があります。その社は鳥居こそあるものの、あまり広くなく、4 畳半ほどの境内に人家に置かれている仏壇程の大きさの木でできた祠があるだけのものです。社の境内の石畳は雑草や苔がのび放題のびていて、あまり手入れがされていないことがわかります。しかし祠は不思議なことに荒れておらず、堂々と気品を持ってその場にたたずんでいます。祠の前にある小さな賽銭箱にも埃ひとつついていません。
その社にはその日、生命力を感じさせないぐうたらな影が 3 つだらしなさげにありました。ぐうちく、きいち、しろの 3 匹です。3 匹はみんな、動物の姿で祠の陰でたたずんでいました。どうやら 3 匹は夏の暑さにへばり、神社に涼みにやってきていたようです。ちょうどそのときはお昼時で、空のてっぺんではお日様が、我が世の夏を謳歌していました。社の境内には、無情にも昼間の太陽の日差しが差し込み、蒸し暑い空気が漂っていました。さらに悪いことにはミンミン、ジージー、ワシャワシャと蝉の声がうるさく響いていました。
そんな中で狸のぐうちくは立派な腹を突き出して横たわりながら唸っていました。
「うぅ……暑いし……うるさい……お日様も蝉も……爆発しろ……」
ぐうちくの泣き言がじめじめした空気に溶けました。
ぐうちくの左右には、犬のしろと狐のきいちが座っています。2 匹ともぐうちくほどはへばっている様子もなく、むしろぐうちくを心配そうに見つめています。
「大丈夫かよ、ぐうちく」
しろが不機嫌そうに鼻を鳴らしながらそういいました。不機嫌そうではありますが、しろはいつも不機嫌そうに見えるので、特に何かに苛立っているわけではありません。
「ぐうちくはほんとにだらしないな」
ぐうちくよりも年上であるきいちは、年上風を吹かせつつ、しっぽをパタパタさせ、風を起こしていました。
ぐうちくを中心にした 3 匹は何をするでもなく、いたずらに夏の昼が早く過ぎ去るのを待っていました。
そんなぐうたらな空気を打ち破るように唐突に祠の扉が開きました。そこから、機械油のような鼻につく臭気が漂う、黒ずんだ玉虫色の、ところどころ泡立った液体のような個体のような、この世のものであるかも怪しいどろどろの物質が、ところてんのようににゅるりと賽銭箱の上に落ちました。そして、そのどろどろは賽銭箱の中身を物色するように数秒うごめくと、賽銭箱からうごめき

ながら離れ、石畳の上でとぐろしました。とぐろしたどろどろは金色に怪しく瞬いたと思うと、狐の姿に変わりました。
「今日もあんまり入っておらんでござる」
狐のような姿をした、この社の主である簀斗羅は 10 円玉 3 枚を右前脚で弄びながら拗ねたような顔をしました。簀斗羅の後ろには、でっぷりと肥えた山羊のような体から樹木のような枝が何本も生えているようにみえる巨大な影が蠢いています。
簀斗羅は、祠に寄り添ってだらしなく口を開けているぐうちくたちを見つけたようで、賽銭の少ないことに対する若干の不満をぶつけようとぐうちくたちに這いよりました。
「こりゃ、おぬしら。若いのになんじゃそのざまは」
そういう簀斗羅の口調はあきれの色が入ったもので、この風景に慣れてしまっているようにも思われました。
ぐうちくは寝返りを打って簀斗羅のほうを向きました。ほかの 2 匹は首だけ動かして簀斗羅のほうを見ました。3 匹の様子も、またか、といった様子でした。
「だって簀斗羅さま。暑いものは暑いんだから仕方ないじゃん。文句ならお日様に行ってくれよ」
「おれはぐうちくの世話をしてただけだし！」
「そもそも狐の 15 歳って若くないでしょ！」
3 匹は口々に自分の言い分を簀斗羅に投げつけました。
「あぁ！　うるさいでござるうるさいでござる！　わしに口答えするんじゃないでござる！」
それに加えて簀斗羅までが騒ぎ立てました。蝉に負けないようなうるささが境内にひろがりました。これもいつもよく見られた光景です。ちなみにここで溜まった憤りやいら立ちが布団村の人々の不幸を引き起こす原因の一つとなっています。
このうるささを聞きつけたのかどうかは定かではありませんが、そこに一匹の黒い烏が飛んできました。バルさんです。バルさんは勢いを緩めながら赤い鳥居を潜り抜け、てかてかと光る賽銭箱の縁に華麗に着地しました。しかし 4 匹はそれに気づくこともなく、お互い罵り合っていました。
「なんだお前ら。喧嘩か？」
バルさんはしわがれた声でゆっくりとつぶやきました。そのしわがれた、それでもよく聞こえる、声を聞いて、うるさかった 4 匹はピタッと言葉を止めました。そしてぎこちない動きで首を動かし、声のした方向を見ました。そこでは黒い烏がじっと 4 匹を凝視していました。
「喧嘩か？」
バルさんはもう一度ゆっくりと問いました。
「「「「違います」」」」
生物としても変化としても大先輩であるバルさん（893 歳）の修羅場をどれだけ見てきたのか計り知れない眼に睨まれた 4 匹には、仲良く振る舞うしか方法はありませんでした。
「おいらたちは疑いようもなく仲良しです」
「おれたちは完璧で幸福な仲良しです」
「仲良しは我々の義務です」
「そうでござる。仲良しは義務でござるからな」

口々に仲良しであることを主張しました。肩を組んで仲良しであると示しました。
「そうか、ならいいんだ」
バルさんがそれ以上追及しないということがわかり、4 匹は胸をなでおろします。
「ところで……」
バルさんの口調はゆっくりとしたままでした。そのことに 4 匹の背筋はまたこわばりました。バルさんの口調がゆっくりなときは何かを疑ったり、探ったりするときだからです。
「ぐうちく。昨日布団村で『また』おじいさんの腰が折れたそうだが？」
バルさんの黒々とした目がぐうちくを見据えました。返答次第では飛びかからんと言わんばかりに羽をぴくぴくさせていました。
「えっ、そうなんですか。初耳です！」
そんなバルさんの様子におびえながらも、ぐうちくは本当に初耳であるかのように驚いたふりをしました。
「むっ、そうなのか。私はてっきりまたお前の仕業かと思ったのだが」
バルさんは少し早い口調でそういいました。そして意外そうに目をぱちくりさせていました。
「ちがいますよ〜おいらはもう悪いことはしないって決めたんすよ」
ぐうちくはさも自分が正しいかのように堂々としていました。
「そうだったのか、すまない」
そして、お人好しのバルさんはぐうちくの口から出まかせを信じてしまいました。無実のものを疑った自分を恥じるようにうつむきました。
「いえいえ、誰にだって間違いはあるものですよ。顔を上げてください」
ぐうちくは紳士的なドヤ顔を浮かべながらそういいました。
もちろん昨日おじいさんの腰を折ったのはぐうちくの仕業です。根っからの悪者であるぐうちくがいたずらをやめることなどありえないのです。
「あっ、おれも昨日スイカを食い荒らしてなんかないっす！」
「ぼくも昨日駄菓子屋のアイスなんて盗んでないですよ！」
「わしも腰が折れたおじいさんに蛇を投げつけることなどしておらんでござる！」
ほかのみんなも自分は悪いことをしていないと言い立てます。
「そうか、みんないい子になったんだな……誤解して済まなかった」
バルさんは、恥ずかしそうに上目遣いではあるが、満足そうにカァと一鳴きしました。みんなはそっとほっと胸を撫で下ろしました。同時に、罪悪感を覚えるものもいました。
「と、ところでバルムンク殿、いったい何の用事でござろうか？　バルムンク殿がここを訪ねるなんて珍しいことでござる」
簀斗羅が恥じている様子のバルさんに話しかけました。おそらく話題を変えて自分の罪悪感を消し去るためだったのでしょう。また、実際バルさんは鞍部町のあちこちに行っているため、この神社には滅多に訪れません。ぐうちくのいたずら程度でわざわざ神社に来たこともありません（基本的に現行犯成敗）。用事を聞かれたバルさんは俯くのを止め、ふと考えた後で、きらきらした眼で簀斗羅のほうを見つめました。

「聞きたいか？」
言葉の調子は明るく、先ほどまでの恥は感じられませんでした。いや、むしろワクワクしているように思えました。まずい、とぐうちくは思いました。お人好しのバルさんのことだから何か厄介ごとを抱えているに違いないと感じたからです。
「バルさん、おいら用事が……」
「まぁ、聞け。実はな、布団村診療所の先生と話していた時の話なんだが……」
ぐうちくは厄介ごとを避けるために逃げようとしましたが、時すでに遅し。バルさんのお話は始まってしまったのです。蝉の声が響く暑い境内にしわがれ声が延々と続きました。

瑪瑙（めのう）の森のはなし

おしゃべり好きなバルさんの話は 1 時間にも及びましたが、内容はそんなに複雑なものではありませんでした。まとめると以下のようになります。
診療所の先生曰く、
「最近、布団村で腰が折れるおじいさんが多い」
「どうにかして腰が折れにくくなるようにしたい」
「古い文献によると、鞍部町の南西にある『瑪瑙の森（めのうのもり）』に、煎じて飲めば骨が強くなる薬草が生えているらしい」
「誰か採りに行ってくれる人がいないかな（ちらっちらっ）」
バルさん曰く、
「私が行きましょう！」
人のいいバルさんは困っている人を放ってはおけないのでした。
お日様が少し西に寄った頃、バルさんのとりとめのない話には区切りがつきました。
「というわけで、みんなで瑪瑙の森に、薬草採りかたがた遊びに行こうじゃないか」
バルさんは、一通り話して満足したのか、羽をバサバサさせながらそう言いました。ぐうちく、きいち、しろの 3 匹は心ここにあらずといった感じで、ぽけーっとしています。暑さと長話にあてられたのでしょう。
「ふむ、おもしろそうじゃからわしは行くでござる」
唯一、簀斗羅は瑪瑙の森に行くことに興味を持ったようでした。
「たしか瑪瑙の森というときれいな植物がたくさん生えていて、遠くから見ると宝石のように輝いて見える不思議な森でござるな。話を聞いたことがあるでござる」
どうやら簀斗羅は瑪瑙の森のことを少し知っているようでした。
「そうだ、ただあんまり人間はあそこの森に近寄りたくないらしいんだ。理由は知らないがな」
バルさんが診療所の先生から聞いたことを追加しました。
「そうでござるか。どうして近寄らないんでござろうか？」
簀斗羅は首をかしげますが、理由はわかりません。
「さぁてな。あぁ、そういえばちょっと前に瑪瑙の森の開発計画の話を聞いたな。あれは結局どう

なったのだろう」
バルさんも首をかしげ、関係がありそうなことに思い至りますが、簀斗羅同様理由はわかりません。
バルさんの言うちょっと前とは、30 年前のことです。当時、瑪瑙の森周辺を開発するという計画がありました。それは、景気の良さに乗っかったどこぞの成金が、余った金でテーマパークを作るというものでした。しかし、瑪瑙の森で罰当たりなことをすると、祟りに遭うということが鞍部町の人間の間ではまことしやかにささやかれていました。そういうこともあって多くの人々はその計画を聞き、森の神様の祟りがあると恐れましたが、成金はそんなことは気にせず、森に重機を入れたのでした。そして計画は頓挫しました。
原因は定かではありませんが、噂によると工事を請け負っていた会社がいくつも夜逃げしたとか蒸発したとかなんとか。そんなことがあり、人々はやはり瑪瑙の森は危険な場所であると考え、禁制地にしたのでした。
「まぁ、なんにせよ。行くことには変わらないのだからここでぐだぐだ話していても仕方がない。いくぞ」
バルさんはそういって、賽銭箱の上で羽を数回バサバサさせてから飛び上がり、社の境内に体で円を描きました。
「いまから行くでござるか？　　それは楽しみでござる」
簀斗羅は生来の好奇心から、瑪瑙の森に行くことに乗り気のようでした。浮かれたように体表を波立たせます。それに伴って異常なまでに大きい簀斗羅の影もざわざわと動きます。
「やだよ、めんどくさい」
「そうだよ、おれたちがついていく必要なんかないじゃん」
「バルさんが頼まれたことなんだからバルさんが一人で片づければいいじゃないですか」
ぐうちく、しろ、きいちの 3 匹は、暑さでしんどいのか、行くことを嫌がっているようでした。
そういう 3 匹を説得するためか、バルさんはぐうちくの頭の上に降り立ちました。鋭い爪がぐうちくの頭の上に突きささっています。
「あいたたたた」
「お前たちも村人には迷惑をかけることもあるだろう。たまには役に立つことでもしたらどうだ？」
痛がるぐうちくを気にすることもなく、バルさんはみんなを言いくるめようと試みました。
「そうでござる、ぐうちく。おぬしらもたまには人の役に立てるようなことをするべきでござる」
簀斗羅もバルさんに便乗してぐうちくに言いつけました。2 つの頭が上がらない存在から意見されては流石のぐうちくも行かざるを得ません。
「わかったから早く頭の上からどいてくれ！」
ぐうちくは叫びながら手で頭を払いました。バルさんはすぐに飛びのきました。そしてぐうちくは頭をさすります。幸い血は出ていませんが、深く爪痕が残っています。
「決まりだな、行くぞ」
バルさんが出発の音頭をとります。
「おー！」
それに続いて簀斗羅が声をあげます。冒険を前にして期待に胸をふくらましているのでしょう。

「おー……」
「おー……」
ぐうちくとしろが力なく声を出します。抵抗するのもめんどくさいのでしょう。
「ぼ、ぼくは急用を思い出したから」
きいちは逃げ出そうと試みました。この後、昨日腰を折ったおじいさんの家を荒らしに行くつもりだったのです。
「あっ？　きいち？　もちろん一緒に来てくれるよな？」
「きいちも一緒に行くでござる」
しかし、バルさんと簀斗羅に回り込まれてしまいました。こうなってはどうしようもありません。
「い、行きます！　行かせてください。なんでもしますから」
きいちは観念したように両手をあげ、懇願しました。こうして、5 匹は瑪瑙の森に遊びに行くのでした。社には蝉の声だけが残りました。

瑪瑙の森にて

瑪瑙の森は布団山から 3km ほど南に行ったところにあります。
5 匹は瑪瑙の森まで、動物の姿のままで、途中休みを入れつつ向かい、お日様が少し下がってきたころに森の北端に到着しました。その場所は空き地になっており、立ち入り禁止とかかれた看板や放置された角材、空のポリタンク、蔦が絡まって放置された小型の重機などが置かれていました。世間は夏休みであり、なおかつ住宅地からそこまで離れた場所にあるわけでもないにもかかわらず、その空地には人の気配が全くなく、物寂しい感じがしました。砂利が多い地面には草がまばらに生えていました。
空き地から森の中を覗くときらきらと緑色に輝く植物や赤色の美しい花などがたくさん生えているようでした。そして、人間の子供程度なら通れるであろう幅の獣道が森の奥に向かって続いていました。5 匹はそこから森の中に入ることにしました。
バルさんを先頭にして、簀斗羅、きいち、しろ、ぐうちくの順番で森に入りました。
森の中は夏真っ盛りな布団山とは異なり、涼しさで溢れていました。夏という季節であるにもかかわらず、蝉の声が一切せず、時々鳥の声と風が通る音がするくらいでした。植物の根が蜘蛛の巣のようにはびこり、気を抜くと躓いてしまうほどでした。ところどころ太くて白い糸のようなものが木の間にかかっていて、異様な雰囲気が漂っていました。そして、あちこちから花の甘くさわやかな香りが強く漂っていました。人間が立ち寄れば、『異界』とはまさにこのような場所だと感じるかもしれません。
「なんだか気味の悪い場所だな……」
しろがおっかなびっくりあたりを見回し、不安そうにきいちのほうに体を寄せました。
「なんだよ、しろ。歩きにくいじゃんか」
きいちはそういいながらも、しろを振り払うことなく歩いていました。ぐうちくは押し黙り、あたりを注意深く見回していました。何か感じるものでもあったのでしょうか。バルさんと簀斗羅はの

んきに歌などを口ずさみながら、特に警戒することもなく進みました。
しばらく歩いていると、バルさんは急に立ち止まり、声をあげました。
「なんだ、ここは！」
その様子は戸惑いと、わくわくに満ち溢れたものでした。
バルさんが立ち止ったのは、不思議な場所を見つけたからです。
そこは、広場のように開けた場所でした。その広場の中心には、石造りの祠がありました。石と言っても縞模様の入ったきれいな緑色の石です。それは、どれくらい古いものかはわかりませんが、長年雨風にさらされたようで、つるつるに磨かれて全く角がありません。祠の前には、ひどく汚れてくすんだ金色をしている、文旦ほどの大きさの鈴が吊るしてありました。鈴には白色の縄がついており、それを引けば鈴が鳴らせるように見えました。そしてその広場は、円を描くように、祠を中心として、木々が等しい距離だけ生えていませんでした。もしも空からその場所を見ることができるならば、完全なる円形に開けた空間が見えるでしょう。広さは小さな公園程で、たとえば小学生 6 人が鬼ごっこなどをするには十分な広さがありました。地面にはさまざまな色をした大きな瑪瑙が放射状に何重か並べてあり、蜘蛛の巣のような模様を地面に描いていました。地面の、瑪瑙が置かれていない場所は、短い草で覆われていて、時折風がその草を撫でていきました。
「ほう、ここは社でござるか。ずいぶんと古いものでござるな」
簀斗羅はそう言って、ぴょんっとその場所に足を踏み入れました。簀斗羅の後ろで簀斗羅の影はぐねぐねとなにかを探すかのように動いていました。神様として何か感じるものがあるのでしょう。
「は〜」
きいちは感嘆の声をあげました。おそらくそこにあるものは 15 年の狐生で初めて見る類のものだったのでしょう。
「おい、帰ろうぜ。なんか出てきそうじゃあないか」
しろはおびえてきいちの後ろでぷるぷる震えていました。見た目の割にしろは臆病なのでした。
「おいらは……おいらは……」
ぐうちくは一見普通そうに見えますが、その眼は何かに怯えているようにも思えました。そんなしろとぐうちくの怯えた様子を感じ取ったきいちは、その場で宙返りをしました。宙返りをしたきいちの姿は、10 歳くらいのつぶらな瞳をした黄色人種の男の子に変わりました。和服に前掛けをつけた丁稚のような恰好をしています。「ほら、しろもぐうちくも怯えるんじゃあない」きいちは年上らしく、2 匹の緊張をほぐすために、2 匹にじゃれ付きました。
「わんわんお！　わんわんお！」
しろは緊張がほぐれたのか、単純なのか、足りないのか、元気そうに吠え始めました。
「……」
ぐうちくは黙ったまま何かを考えているようです。傍から見れば子供が動物と戯れているほんわかとした光景が広がりました。簀斗羅は、そんな光景を気にすることもなく、祠のほうに近づいて、白い縄を掴んで引っ張りました。するとシャラン……と綺麗な澄んだ音がその空間に響きました。くすんだ鈴からは想像もつかないようなきれいな音です。
「だれかおらんでござるか！」

簀斗羅は、祠の主に対する呼びかけを行いました。普段まがりなりにも神様のようなことをやっているので、他の社に来たならば神様に挨拶をするのが礼儀と考えたのでしょう。やり方は礼儀知らずですが。社のほうから返事はなく、風の音と、園児と犬の声がするだけです。
「これだけ古い社だ。もう神様はいないのかもしれないな」
バルさんは祠の周りをぐるりとまわりながらそういいました。
すると、その言葉に反応するように、いるよー、という声が元来た道のほうから風の音と共にしました。
強い風が広場を駆け抜けました。
(つづく)

お詫び

今回はここまでとなります。
中途半端であると思われる方もいらっしゃるとは思いますが、悲しいことに作業中にデータの後ろ半分が消滅してしまい、書き直さなければならない状態となってしまいました。
おそらく次回の部誌までには書き直せるとは思います。ここまで読み進めて頂いた方には申し訳ありませんが、ぐうちく衛星の由来はまだお教えすることができません。ご容赦ください。

電気工事士の居る部室

@l_possum

プロローグ

この部屋の全ての柱にコンセントを！

僕たちの部室は築数十年の木造 2 階建てアパートにあります。計算機系サークルらしく、部員の持ち込む端末が多い時には十数台、これに加えてサーバ群が部室に収まっているのですが、古いアパートということもあり設置されているコンセントの数は少なく、テーブルタップと 3 口タップが大活躍していました。

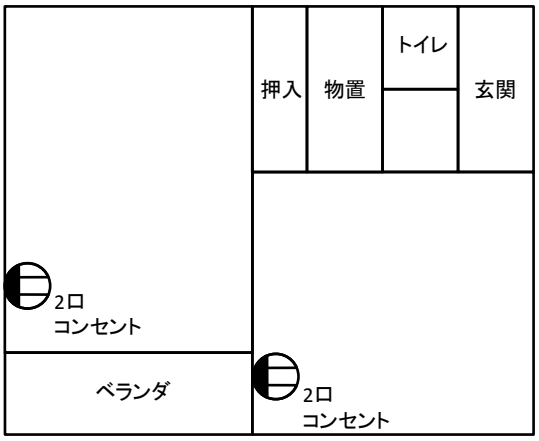


図 1　元のコンセント配置（エアコン用や台所用を除く）。各部屋に 1 つずつしかない。

そこで上記の目標の下に立ち上がったのが僕と香澄さん [*1] の 2 人でした。ここではコンセント増設を目指す僕ら 2 人の歩みをお話ししようと思います。
コンセントの増設、ひいては部屋の中の電気配線をいじるためには第二種電気工事士（通称電工二種）という国家資格を取得する必要があります。受験するためには受験料およそ 1 万円を支払わな

[*1] 修士 2 回生。僕の 1 つ上。KMC には 3 回生の時に入会。

ければならないので僕は申込期限のぎりぎりまで迷っていたのですが、香澄さんが申し込んだというのを聞いて 4 月 6 日、思い切って申し込みました。
電工二種の試験は上期と下期で年 2 回開かれ、それぞれに筆記試験と技能試験があります。僕たちは上期で申し込んだので、6 月頭の筆記試験を突破し、さらに 7 月末の技能試験に挑む日程となりました。

4 月 13 日 金曜日

申し込みから 1 週間後の 4 月 13 日金曜日、筆記試験と技能試験の問題集を購入した僕たちは、筆記試験の過去問を試験 2 回分ずつ解く勉強会を始めました。
筆記試験は 4 択問題が 50 問のマークシート式で、電気回路基礎、電気工事関連の法令・基準、配線・工事方法、配線図などの内容が出題されます。
大学受験を離れて 4 年以上が経過する僕ら修士 2 人にとって、原理を考えずひたすら暗記しろと言ってくる問題と正答はつらいものがあり、高校生の時に受けておくべきだったとこぼしながらの勉強会となりました。
たとえば、図記号とその示す物の対応をおぼえる必要があるのですが、図記号にあるアルファベットは何の略なのかが解説に示されていることはほとんどありませんでした。
記号の具体例としてスイッチを挙げましょう。スイッチは黒い丸で示され、何も文字が付されていなければ 1 つの回路を ON-OFF するだけのスイッチとなります（図 2 上段)。図 2 の下段左のように「3」を付すと 3 路スイッチを示します。

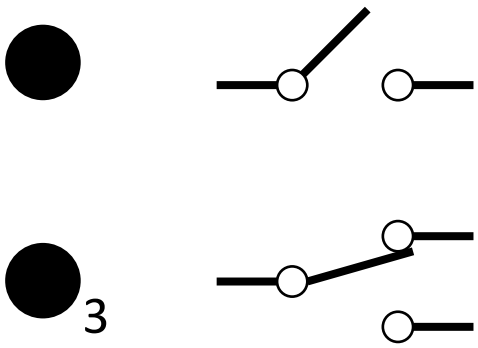


図 2　スイッチの図記号。左の図記号が右の回路図記号に対応する。

これはまだ良い方で、図 3 に示す種類になると英和の推理が必要になってきます。防雨形スイッチの WP、自動点滅器 *2 の A、リモコンスイッチの R、プルスイッチの P は対訳がぱっと思い浮か

*2明るさによって ON-OFF を切り替えるスイッチ。

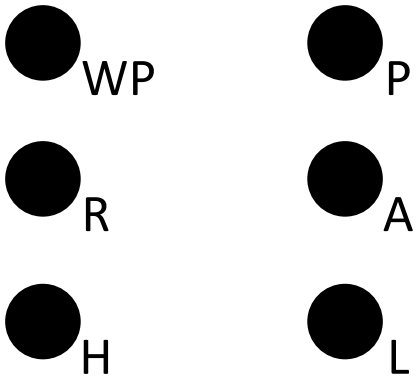


図 3　その他のスイッチの図記号

ぶので良いのですが、確認表示灯内蔵スイッチ [*3] の L と位置表示灯内蔵スイッチ [*4] の H は結局最後までわかりませんでした。今もわかりません。
この他にも、「露出配線は図面で点線で書いて隠蔽配線は実線で書く」「電線管に付された数字が偶数なら内径を、奇数なら外径を示す」「1 種も無しに突然登場する 2 種金属製可撓電線管」「ケーブルの英字略称の由来」など、何故なのかわからない不思議がたくさんありました。余計な情報を与えると混乱を招きかねないという方針なのかもしれません。
勉強会は第 1 回目以降おおよそ 1 週間おきに開きました。リングスリーブ [*5] 最小必要個数計算問題に苦戦し、屋内配線図に登場した民家に住みにくいだろうと文句を付け、この家にもシャンデリアを付けるのかと笑いながら、本番までに試験 9 回分の過去問を解きました。試験は 60 点合格なのでちょうど学校の単位認定と同じです。僕は全 9 回中 80 点以上を取ることは一度もなくて良止まりでしたが、香澄さんは 2 回は優を取っていました。さすがです。
そんなこんなで試験本番を迎えましたが、あまり覚えていることはありません。覚えている事と言えば、我々の予想通り民家の配線図にシャンデリアが登場したことくらいです。試験後に自己採点もしなかったので、何点だったかもわかりません。可なら取れるだろうと信じながら会場を後にしました。
受験者はおじさんばかりかと思っていたのですが、案外若い人や女性もいるものなんですね。若い人については工業高校生なのかもしれません。試験会場の出口で香澄さんを待っていたら、高校生らしい男の子と女の子が待ち合わせをしているのを見かけて、電気工事に理解のある彼女がいたらと考えましたがそもそも僕には望むべくもないことでした。あきらめます。念のため付け加えておくと残念ながら僕にとって香澄さんはそういう女性ではありません。

[*3] スイッチが ON の時に内蔵ランプが点灯するスイッチ。
[*4] スイッチが OFF の時に内蔵ランプが点灯するスイッチ。
[*5] 電線同士を圧着接続するための金属円筒。長さ 10mm 直径 5mm 程度の大きさ。

6 月 3 日 日曜日

筆記試験が終わればすぐさま技能試験の練習です。技能試験は配線図と資材が与えられ、40 分以内に資材を使って配線図通りに配線を行う形式です。評価は減点方式で、電気的に致命的な欠陥か施工上重大な欠陥がひとつでもあれば不合格、施工上軽微な欠陥が 3 個以上あっても不合格となります。なお、道具は受験者が持ち込む必要があります。
当日出題される問題は予め公表された 13 問の内の 1 問なので、13 問全てを練習することにしました。VVF ケーブルの被覆すら剥いたこともない僕らにとって 1 回周りの練習だけでは不安なので、それ以上の練習回数を 7 月末までに確保しようと思うとほぼ毎日練習しなければならない計算になりました。筆記試験の合格通知を待つ猶予などありません。6 月 3 日日曜日の第 1 回以降、毎朝候補問題を解く朝練が始まりました。朝練の開始時刻は 7 時とされていましたが守られたことはほぼ無く、毎回ログを「寝坊したせいで」と始めるのが僕の日課となりました。
技能試験の作業の大まかな流れは、

1. 単線図から複線図を起こし、ケーブルの切断寸法を決め、電線の接続方法を確認する。
2. ケーブルを切断し、被覆を剥く。スイッチやコンセントなどの器具に接続する。
3. ケーブル同士を接続する。

のようになります（順序は前後することもあり、もちろん作業前の材料の数量確認や最後の配線チェックも行います)。
単線図、複線図というのはどちらも配線図なのですが、単線図が複数の電線・ケーブル [*6] を 1 本の線で表すのに対して、複線図は電線 1 本 1 本を区別して表記します（図 4)。試験で与えられるのは単線図なので、実際に工事するには電線ごとの接続がわかる複線図を起こす必要があります。ここで間違えると配線間違いによる不合格ルート一直線です。
複線図が書き終わればいよいよ切る剥く曲げるねじ込むの作業です。僕らはホーザンの発売している電気工事士試験用の工具セット [*7] を買って練習・試験に臨んだのですが、VVF ストリッパ [*8] を握っている時間が一番長かったように思います。被覆を剥く工具としては電工ナイフもあるのですが、ストリッパですべてなんとかなると香澄さんから助言をもらって以降、まったくと言って良いほど使いませんでした。文明の利器ですね。
ケーブル切断、被覆剥きと並んで重要な作業がケーブル同士の接続です。接続方法はリングスリーブを使う場合と差し込み型コネクタを使う場合の 2 通りがあり、いずれの方法も毎回出題されます。差し込み型コネクタは被覆を剥いた電線を差し込むだけで終わりです。一方リングスリーブは、接続する電線の露出した心線を揃えてリングスリーブに差し込み、そのままの状態でリングスリーブを押しつぶすという力作業が必要になります。リングスリーブは大スリーブ中スリーブ小ス

[*6] 電線は 1 本の金属の心線を絶縁体の被覆で覆ったもので、ケーブルは電線をまとめてさらに被覆で覆ったものです。
[*7] ホーザン S-18 電気工事士技能試験セット
[*8] ストリッパはケーブルや電線の被覆を剥く工具。VVF はケーブルの一種。

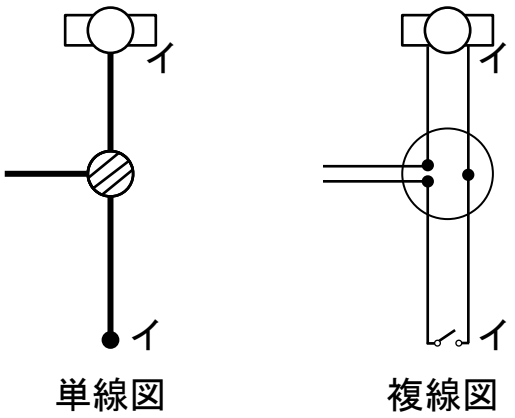


図 4　蛍光灯とそのスイッチの配線図。実際には電線の太さや種類なども書き込まれる。真ん中の記号は電線同士を接続するための器具（ジョイントボックス）。

リーブと大きさがあるのですが、中スリーブをつぶすのにはそれなりの握力が必要で、朝練のメニューにテニスボール握りの導入も検討されました。
3 路スイッチ、4 路スイッチを含む複線図起こしに泣いた僕も、握力が足りなくて中スリーブでの圧着が厳しいと言っていた香澄さんも 13 問全問一通りともう少しを練習したところで本番に臨むことになりました。20 問分ほど練習を重ねたことで、13 問の中でも 1 問でしか出てこなかったような配線箇所が出てこなければ合格するだろうという自信が持てるほどになりました。
結局危惧していたような配線箇所が出題されることはなく、2 人とも十分な余裕を持って作業を終えることができました。香澄さんは時間が余りすぎて試験官のおじいさんとおしゃべりしていたそうです。
9 月頭に合格通知を受け取り、都道府県から交付された免状を手に入れた 10 月頭、ようやく僕たちは屋内電気配線を取り扱っても良い資格を手にしました。

11 月 24 日 土曜日

もともとコンセントの増設を目指して資格取得をしたのですが、10 月頃から電気工事に関わる別のプロジェクトが動き出していました。あるゲーム会社に部員が行ったつながりで、デスクトップマシンを十数台供与して頂けることになり、これらのマシンをプライベートクラウド用サーバに充てればどうかという話になっていたのです。ここのところ部員数は増加の一途をたどっているので、部屋の中に机やラックを置くのは避け、押入にクラウド用サーバを配置することとなりました。電気工事士がいるのですから押入の中にコンセントを増設すれば完璧です。
しかしながら、プライベートクラウド計画を例会で提案したところ、それらのサーバの稼働によって電気代が大きく上がるのではないか、との指摘を受け、結局今年度分はサークルの会計が経費として電気代を支払うがそれ以降はこの限りでない、と決まってしまいました。そこで、来年度以降

は有志からのカンパで電気代をまかなうべく、電気代算定のために電力量計が押入に設置されることとなりました。
以上のような経緯があって迎えた京都大学 11 月祭 [*9] 中の部室掃除 [*10] で、片付けと掃除が一段落した 11 月 24 日土曜日、コンセント増設工事がはじまりました。
回り縁や柱、長押を沿わせる程度の工事は難なく終わったのですが、ここでひとつ問題が浮上してきました。配線の経路選定に関わる問題で、鴨居の下をくぐる露出ケーブル工事を行うか、壁に穴を開けて一部電線管工事を行うかで意見が分かれました。前者の方が簡単な工事ではあるのですが、少し経路が大回りになるのと見た目があまり良くないという欠点がある一方で、後者は壁を破壊するという「賃貸の一室でそれは大丈夫なのか？」と（僕は）思う選択肢でした。
結局、配線の美学と振動ドリルに魅了された僕たちは穿孔を敢行し、無事コンセント増設工事、押入電源工事を完了することができました。
ここまで来ればあとはサーバのセットアップの領域なので、単なる電気工事士たる僕は手をひき、以降の作業はサーバ管理者ののなさん [*11] に引き継ぎました。

エピローグ

プライベートクラウド計画自体は 2013 年秋から始まっており、当時の部員がインターンに行ったことでつながりのできた C 社からもらったマシン 4 台を投入して始動したそうです。

> 「香澄さん、一緒に押入に入れることになったこの 4 台のサーバはなんで軍艦の名前なんですかね」
> 「どうせ『艦これ』から取ったんでしょ」

このことについてあとからのなさんに聞いたらその通りで、当時ぽっすむさん [*12] とのなさんが命名したそうです。のなさんは、これで第一水雷戦隊がつくれる！ と言っていました。よくわかりません。
ともあれ、今まで机の上にあったサーバ 4 台（「第六駆逐隊」と呼ばれています）が押入に行ったことで部室の使える領域が広がったので、来年の新勧期には余裕ができました。これまでは第二部室の 2 部屋と、今回電気工事をした第三部室の狭い部屋 2 つという構成でしたが、第三部室の 1 室が広々と使えるようになったことは嬉しい限りです。第三部室は第二部室の上の部屋なので階段を上る必要があるのは多少手間ですが。
悩みの種は電気代ですが、こればかりはどうしようもありません。来年 ITER[*13] が稼働するそうですが、早く核融合発電が実現すると良いですね。生きている内に見られることを祈りましょう。
これで僕の話は終わりです。それでは皆様よいお年をお迎えください。

[*9] 毎年 11 月末に開催される京都大学の文化祭。
[*10] 11 月祭会場にマシンを持って行き部室が空っぽになるため、このタイミングで大掃除をします。
[*11] 修士 2 回生。僕の 1 つ上。
[*12] 第二部室の電気工事を手がけた人。僕より 5 つくらい上。
[*13] 国際熱核融合実験炉。商用核融合発電の 1 個手前の実験炉。2019 年稼働予定。

最近のインターネットの最悪さについて

pastak

図 1　要するにこういうはなしです。

はじめに

はじめまして、こんばんは、こんにちはおはようございます、KMC1 回生の pastak こと Pasta-K です。皆さん元気にお過ごしですか？　僕は先輩から不快な気持ちになる GIF アニメがリプライで送られてきて気分が最悪です。[*1]。最悪繋がりで、この部誌のこのパートでは、僕が個人的に最近というか大学に入ってからインターネットが最悪になったと感じるという話を書きます。もっと具体的に言うと、大学に入ってから以下に当てはまるようなついったーユーザーが身の回りに増えてきて気分が害されて最悪だっていう話をします。端的にいうと「愚痴」です。なので、以下に当てはまる人は読まずに飛ばしてください。

- 自称ツイ廃だ。

[*1] `http://pastak-diary.hatenadiary.com/entry/2013/12/03/012632` 参照

- インターネットで変な顔文字をコピペすることに生きがいを感じる。
- Twitter をやっていて、且つ京都大学情報学科の 1 回生だ。
- 「アルファツイッタラー」という言葉を発したことがある。

もうこのリストを書いただけで気分が悪くなってきました。まだ序章なのにやる気が皆無です。『ゆゆ式』を無限に見たくなってきたけど、頑張って書くので、読んだ人は Twitter で@pastak まで応援コメントをよろしくお願いします。では、短い間ですがお付き合いくださいませ。

いんたーねっとさいこう！！！！！！

本題に入る前に、これまでずっとことある度に僕は「インターネット最高！！！！！！！」特に「Twitter 最高！！！！！！！」と言ってきたことを紹介しておきます。

「ホムペ」？「リアル」？ こわ……ついったー最高！！！

高校 1 年生の時にインターネット上のコミュニティである「きゅーいち世代」[*2] の関西忘年会をしたのですが、その時にした LT の資料 [*3] を引き合いにしてみたいと思います。この頃はちょうど、「リアル」とかいうケータイ向けサービスが中高生の間ですごく流行っていたのだけど、僕はこの頃から Twitter を使っていた [*4] から全く知らなくて周りに聞いて実際の大多数の「普通の高校生」はどういう感じでインターネットをしてるのかなということをまとめた LT でした。実際に色々と話を聞いていると、SNS と現実の紐付きが思いの外根深くて、それが自分にとってはとても怖くて資料の最後がこんな感じでした（図 2)。

僕の大好きな Twitter であって欲しい。リアルからの逃避先としての安心できる場所であってほしい

昨年の第 15 回文芸フリマで頒布された『Project: AMNIS VOL.1』[*5] に『インターネットに押し寄せる「リアル」の波』というタイトルで寄稿させてもらったのですが、その中でも現実世界的な繋がりから Twitter に流入し、その現実世界的な狭い繋がりの感覚のまま WWW の広さに無自覚であるが故のやり取りを強制され、「逃避先としての Twitter」が機能しなくなるというような話を書かせてもらいました。
この『AMNIS』に書いた文章がより最悪な感じになって今まさに僕の目の前にはあると感じている訳なのですが、次のセクションからはつらつらとその辺に関して不快に感じていることなどを書いていこうと思います。よろしくお願いします。

[*2] http://generation1991.g.hatena.ne.jp/ここで知り合った人たちが KMC に何人かいるから世界は恐ろしい……

[*3] http://www.slideshare.net/pastak/g91lt-2748111

[*4] 正確には 2008 年の春からなので、この時には 1 年半以上使ってた。古参アピールだ〜〜〜〜〜〜

[*5] http://lamer-e.tv/amnis/

図 2　インターネットこわい

“自称キチガイ”、“自称ツイ廃”、コピペで群れる人たち. . . etc

いつの頃からか Twitter でネタツイートをコピペして発言したりそれらを RT しまくるような人たちが出てきだして、よく分からない顔文字 *6 を使い、「ふぁぼ」（お気に入り）に異常な価値を見出すような人たちが現れだしました。好みの問題なのでしょうけど、僕にはこういう行為の面白さが全然共感出来なかった。全く面白くないのに、それを面白がって RT したり投稿したりしている *7。そういう状況を見てるとめっちゃ身内っぽいコミュニケーションに感じて余計気持ち悪いなと感じていました。この頃ぐらいから「〜〜ふぁぼ Bot」「〜〜RTBot」「おもしろツイート Bot」みたいなアカウントも現れ出して、TL の居心地の悪さが加速します *8。

『AMNIS』にも書いたように、僕にとって Twitter は現実逃避の手段で、オープンで広大なインターネットに書いたり読んだりするからこそ細かいことを気にせず *9 に書いたりも出来たり、人の書いたのを適当に読み流して時間を浪費したり出来ていたわけですが、そこにさっきのような顔文字が流れてきて、しかもその面白さが全く理解出来ないというのは、非常に自己中心的な考え方だとは思いますが、自分の息抜き空間であった TL やインターネットを汚されている感覚になるわけです。

*6 NAVER まとめの『【キチツイの森】ツイッターで使える便利なコピペ・テンプレ・AA 集【Twitter】』というところにいっぱいサンプルがあったので、ピンとこない人はここを見てください `http://matome.naver.jp/odai/2136505429300763801`

*7 そしてそれをしてる人も RT されてる人も大体いつも一緒だ

*8 こういうコピペ系 Bot アカウントは RT で流れてくる度に即攻でミュートする。ブロックとか Report for Spam じゃなくてクライアントの機能でミュート。ミュートじゃないと RT で流れてきて見える

*9 気にしなさ過ぎて炎上する例もあるそうですが……

大学

特に大学に入ってからは「高度な情報戦」[*10] に参加するためには Twitter で同級生たちとフォローしあう必要があったりするわけで、最初のうちはまぁ同級生付き合いも大事だしフォローしとくかって感じなわけですが、これが最悪のはじまりだったわけです。けど、一部界隈で有名な人間がいるらしく、皆が皆そろってその人達 [*11] をネタにしたり、その人のツイートをパクったりしまくってる。ずっと一日中そんな人たちのツイートばっかりが流れえてくるようになった。まぁただの身内ネタだったら適当に見過ごせば良いんだけど、内容が最悪でそれが前述したいわゆる「キチツイ」と呼ばれる類で彼らはそういうネタを手を変え品を変え、変えと言いつつ内容もネタもなんだかんだでいつも同じで全く代わり映えしない。それで「〜さんは有名だ」「〜はふぁぼ魔だ」「〜は乞食だ」などと言い合い褒めあってる。完全に慣れ合い。慣れ合いでしか無い。それを見させられても全く理解できないし、面白くないし、一刻もはやくインターネットをやめて欲しい気持ちになる。
インターネットは人権 [*12] って言葉もあるし、インターネットは人権だから保護されないといけないし、こういうの見てると自分のインターネットをすごく荒らされてる気分になる。こういうことを人に相談すると「リムれ [*13] ば良いじゃん」と言われるし、ちょっと前の自分が人にこういうことを相談されたら同じように答えてたと思う。でも、今はそういうことだけでインターネット出来なくなっていて、リアルの人間関係とかそういうのもインターネットに関わるようになってきているし、実際 Twitter してるから同級生の情報を得られて授業の情報を取得成功するということもある中で、リムることが軽々しく出来ない現実もある。厳しい。

さいごに

つらつらと全く生産性の無い愚痴を KMC の部誌という場所に文体もグダグダで且つ締切を過ぎた状態で深夜に勢いで書いていて申し訳ない気持ちでいっぱいなので、今度関係各位への土下座謝罪を検討しているところです。それはさておき、こういう内容、ブログに書いても良いのですが、インターネットの気持ち悪いツイートをしている人たちはこういう文章に敏感で異常に反応するし、さらに「ツイ廃ですまんね」みたいなリプライまで飛んできたりして直滑降的に最悪さが増すので、部誌というクローズド [*14] で且つ彼らは見ないだろう場所に書かせてもらう機会を頂けて個人的にはスッキリしたので、感謝しています。この記事の冒頭で載せた僕の Twitter への投稿 [*15] はそんな彼らを dis ったものだったんですけど、投稿した数日後に同級生の 1 人（彼もまた「アル

[*10] 授業とか課題とか遊びの情報をキャッチする戦い
[*11] 彼ら曰く「アルファツイッタラー」と呼ばれている人種
[*12] ちょうどこれを書いてる時に部内の IRC で「インターネットは人権」という言葉が話題になり、流行のキッカケとしてこの URL が紹介されてた `http://labaq.com/archives/51684320.html`
[*13] リムる = remove する = Unfollow する = フォローを外すこと
[*14] といいつつも、実は 1 年経つと公式サイトで公開される
[*15] `https://twitter.com/pastak/status/381118540469633024`

ファツイッタラー」の 1 人）に RT された瞬間にまたたく間にお仲間内に広まったりしたということを付け加えておきます。
あんまり綺麗な話じゃないので文章にするのも難しい感じだなとは思っていたのですが、案の定今読み返してみると汚くて読みづらくて訳わからん文章になってしまった感は否めませんね。それでも、もし、もう少し聞きたいとか同じことを思ってたみたいな奇特な人がいらっしゃいましたら是非お話しましょう。寿司とかの奢りが付いていると余計に口が軽くなる可能性があります。ご検討頂ける場合は Twitter で@pastak までお声掛け頂ければと思っております。よろしくお願いします。またインターネットでお会いしましょう。ではでは。

中国東北部から極東ロシアへ渡るために必要なたった 3 つの条件

@hidesys（ひでシス）

この夏休みに、中国東北部から陸路でロシアに入国しました。その際に少し苦労があったりしたので、それを書きとめようと思います。

旅立つ前に

発端

「ロシア旅行したい、でも一人だと恐い。から付いて来いやオラ」。大学に入学する以前からの友人のお誘いです。僕は大学 1 回生の冬から長期休暇の際は必ず海外旅行をしています。その数、通算 7 回。しかし、海外へ行く際は必ず一人でした。この電話は僕にとっては青天の霹靂でしたが、たまには気の置けない友人と旅行するのもいいかなと思い、Let's Go!を出しました。そのあと、なんとか予定を合わせます。9/21-22 にハバロフスク、9/23 はシベリア超特急でウラジオストクまで移動、9/24-25 にウラジオストクを観光する予定になりました。せっかく航空運賃を払って海外へ飛ぶのに、ロシア 5 日間だけかよ。友人は医学科で多忙な身だったので仕方ありませんが、休学中 [*1] の僕は夏休みも関係なくずっと暇です。なら、隣にくっついてる僕の大好きな中国も経由していけばええやん。神。

移動計画

旅是移動 [*2]。さて、僕の方は 9/15 に中国から入国してハバロフスクの対岸にある黒龍江省撫遠まで行き、そこから船で渡ってロシアに入国することにしました。表 1 が事前に立てた行動表です。移動を地図で表すと、図 1 のようになります。中国東北って言ってますが、まぁ旧満州の辺りにな

[*1] 4 回生のころ、院試が受かり卒論も提出した後に、フランス語の単位が取れていなかったことが発覚して 1 単位留年をしました。前期に単位が揃ったので、25 年度後期は休学しています。まぁ、この時点では正確には「休学中」ではありませんが。

[*2] hidesys「旅是孤独。」 友人「Yes.」 「人生是孤独。」 「I don't agree that. You must be alone too long, so you said so.」

日付	場所	内容	（行き先）
9/15	高松	-(飛行機)->	上海
9/16	上海	-(飛行機)->	黒竜江省ハルビン
9/17	ハルビン	市内観光	
9/18	ハルビン	-(鉄道)->	撫遠
9/19	撫遠	市内観光	
9/20	撫遠	-(船)->	ハバロフスク
9/21	ハバロフスク	合流	
9/22	ハバロフスク	市内観光	
9/23	ハバロフスク	-(鉄道)->	ウラジオストク
9/24	ウラジオストク	市内観光	
9/25	ウラジオストク	-(飛行機)->	成田
9/26	新宿	-(夜行バス)->	京都

表 1　行動表

ります。

図 1　行動図

ロシアビザの発行

基本的に、他国へ入国するには事前にその国から「ウチへ入国してもいいよ」っていう許可が必要で、それはビザという形をとってパスポートへ貼り付きます。日本には相手国の領事館が設置されていることが多いので、日本国内に居る間にビザを領事館を通して発給してもらい、そのビザの付いたパスポートで入国する、という流れになるでしょう。日本はその経済的パワーと米国の核の傘の下にあるという軍事的パワーのおかげで結構な数の国がビザ免除されていて、個人旅行者としては旅行しやすい国籍の国だったりします。

旅行する際は、まず「《旅行先の国》　ビザ」でググってみて、相手国の日本領事館の Web サイト

でビザが免除されるかどうか・免除条件はなにかを調べましょう。今回合わせて訪問する中国においては日本国籍保持者は「観光目的で入国して 14 日以内に出国するのならビザ免除」なのでビザの申請をしていく必要はありませんでした。しかし、ロシアへの入国には旅行目的であってもビザを要求されます。
ロシアビザ申請に必要な書類の欄を見ると、

- 申請書類
- 申請者のパスポート原本
- 写真 1 枚
- 旅行会社が発行するバウチャーのコピー

が記載されています。申請書類やビザを貼り付けてもらうパスポート原本、写真についてはわかりますが、バウチャーとはなにか。「バウチャーには下記の情報がなければなりません」の項を見ると、

- 旅行者のデータ（氏名、生年月日、パスポート番号）
- ロシア入国日および出国日
- 観光ルート、移動手段、宿泊場所、観光プログラム
- 旅行会社の署名と印
- 支払済み証明
- ロシアの受入れ旅行会社名とその旅行レファレンス番号

となっています。実はロシア国内は自由旅行が未だ許されておらず、国内を旅行するときには必ず旅行会社を通して国内のホテル・交通手段を事前に予約し代金を払っておく必要があるのです。バウチャーは「これこれの代金を払ったのは私なので、このサービスを受けさせて下さい。」と申し出るための紙になります。これを取るのが非常にめんどくさいし、こういった規制があるとすべてがどうしても高くなる。
まぁでも規制があれば抜け道があるのが共産国の道理で、ロシア国内にはニセモノのホンモノのバウチャーを発行してくれる旅行会社が数社あります。つまり、記載されているホテルや移動手段については代金を支払っていなくてもバウチャーを発行してくれる旅行会社です。これを使います。
ロシアは共産国らしく書類主義なんですが、なんせとりあえず体裁だけ保っておけばなんでも通してくれるんですね。神。今回僕は TravelRussia という旅行会社のサイトを利用しました。
そうは言っても国内のホテルや移動手段はバウチャーを通じて事前に予約することが習わしになっていますから、わけのわからない東洋人が意味もわからんと突然ホテルのレセプションに現れてもどうにもならない可能性が大です。ホテルとシベリア鉄道は他サイトを通じて事前に予約しておきました。これで旅の準備が出来ました。

旅の記録

9/15　高松→上海

高松－上海間には春秋航空と中国の LCC[*3] が飛んでて安いですね。片道 14kJPY[*4] は神だと思う。春秋航空は貨物を一緒に運ぶことでチケットを安くしています。高松空港の旅客ターミナル横にはクロネコヤマトのトラックが並んでたりとか。なので、旅客の荷物重量制限が非常に厳しい。手荷物と受託を合わせて 15kg まで。この日は、上海から高松へ春秋航空で飛んできた中国人女性が、高松空港で来しな [*5] と同じバッグなのに機内持ち込みを拒否されて（中国→日本より日本→中国の方が厳しい）怒りまくってました。もうこんなところから中国を感じる。前のおっちゃんは「鍵とか重いものは全部ポケットに入れるんだよ。あと、手荷物って言ってもこういった手に持ってる手帳までは一緒に量られたりはしない。」って言ってたり。おもろい。僕は 8kg でした。
明日の上海→ハルビン便は朝早くの 07:05 に離陸なので、上海浦東空港で寝ることにします。上海に着いたらとりあえずファミマで水買って地下鉄で市内に出て沿岸部の美味しい料理を食べてネカフェで時間つぶして夕飯食べて地下鉄で空港へ戻ってきてファミマでパン買って食べながらベンチで寝ました。

9/16　上海→ハルビン

ハルビン太平国際空港に降り立つと、寒かった。バスで市内まで出て、ハルビン駅ターミナルで市内バスに乗り換え、予約していたホステルへ行きます。ハルビンロシア国際青年旅舎ってところに泊まったんですけど、あんまりロシアへ陸路越境するための情報がなかった。情報不足はちょっと怖いので、ハルビンに泊まるのは一晩だけにして明日ハバロフスクの対岸の街撫遠まで行くことにしました。ハルビン駅で長蛇の列に並んで切符買います。

9/17　ハルビン→撫遠

東北部の中華料理は味も濃すぎず油も使いすぎずで日本人によく合うと思う。美味しい。鉄道は夕方出発なので、博物館とか行って時間を潰します。明日は満州事変の勃発日ということもあって、東北烈士記念館の訪問者ノートには「打倒日本帝国主義！　勿忘国恥！！」とか書いてあってちょっとビビる。日本大使館のサイトをチェックし、デモ・暴動・その他気を付けなければいけないことがないかどうか調べました。

[*3]格安航空会社（かくやすこうくうがいしゃ）とは、効率化の向上によって低い運航費用を実現し、低価格かつサービスが簡素化された航空輸送サービスを提供する航空会社である。ローコストキャリア、LCC とも言われる。【出典：Wikipedia】

[*4]キロ日本円。14kJPY は 1 万 4 千円のこと

[*5]「来るとき」の古語的表現。

9/18　撫遠

中国の長距離列車はよく眠れる。良い。飯屋の看板にロシア語が併記されてる。スゲー。

図 2　撫遠市内。ロシア語表記が確認できる。

実は昨日、列車のコンパートメントで一緒になった人と話をしていたんですが、「黒竜江川 *6 で 8/13 に大発水があって、僕はその影響を取材しに行く記者なんだ。」って言っててちょっと焦る。国境の川が洪水を起こしているとなると、国境が封鎖されてる可能性があったりして。う〜ん。ハルビンのホステルではこんな話一つも聞かなかったぞ。
朝に近頃できたばかりの鉄道駅 *7 に着いて、市内までのバスに乗ります。遠い。で、適当に宿を探して荷物を置いて、ホテルの人と相談。「うん。電話して聞いてあげる。……ああ。9/24 に国境が再開するって。それまで泊まっていく？」ホテルの人は宿代がもらえるかなと思ってニコニコ。僕は青い顔。完全にヤバい。どうしよう。

9/19　撫遠→饒河

9/24 まで待ってるわけにはいかないので、高速バスで別の国境へ移動します。ホテルの人と「賄賂だったら 2000 元まで出せるけど、どうにかならないか。」「賄賂でもどうにもならないよ。」「あああああ。」「饒河口岸 *8 なら開いてるってサ。」「僕は第三国人だけど、それでも通れるの？」「確認したる。……行けるって。」「ホンマか。」という話をして、撫遠より南にある饒河へ。早朝過ぎて外食が開いてなかったのでバスの中で月餅を食べます。イチゴ味で甘くて美味しい。
饒河に着いたらその足で口岸へ。タクシーの運ちゃんに「開いてるやんな??」って念を押す。「開いてる開いてる」って答えられたので、タクシーを降りて颯爽と国境の建物に足を踏み出したら、まだ乾いてないコンクリ地面に足がのめり込んだ。

*6 別名アムール川。中国東北部とロシアを隔てる。
*7 ジャムスー撫遠間の開通は 2011/12/6。Google Map には路線は載ってません。
*8 ほにゃらら口岸は中国で水を隔てて他国へ渡る国境を指す。撫遠口岸・黒河口岸など。香港と対する深湾口岸は有名。

図 3　工事中の饒河口岸

工事のおっちゃんに謝りつつドアを開けようとすると、鍵がかかってて開かない。工事のおっちゃん「明日の朝来なさい。」 ワシ「ええっ！　 今日は開いてないの!?」「開いてない。明日の朝なら通れるから。」どうしよう。早くしないと友人がハバロフスク国際空港に着いてしまう。

9/20　饒河→虎林→牡丹江

国境を工事してたおっちゃんの奥さんがやってる宿で目を覚ます。ううう。困った。間に合うんかこれ。
昨日なぜ国境が開いてなかったのかというと、それは十六夜の祝日だからだった。昨晩は外から笛の音が聴こえて来てたけど、冷たい雨が降ってるし体力もないので見に行くのを辞めておく。ここで熱を出すと大変なことになる。
「国院批准我市饒河口岸展外国人口岸」っていう記事がネットに上がってて、それによると 9/17 近日に外国人が饒河口岸を越えられるようになるとのこと。マジか、えらいタイムリーだな。
口岸へタクシーを飛ばして着く。国際バスのチケットを購入、バスに乗り込む。乗客は僕だけ。バスが国境の建物に入ると警備隊に降ろされる。おみやげを座席から持って行こうとしたら、バスの運ちゃんに「そのままでいい」って言われた。バスから降りてパスポート審査台へ。「やっとこれでロシアだ!!!! やっと、やっと!!」と思ってたら国境警備員がワラワラ集まってきて、「この国境は中国人とロシア人しか通れないよ。」と借りた携帯電話の向こうの人に言われる。バス「ブロロロロ……（僕のおみやげだけを載せてロシアへ）」。アタシは死んだ。スイーツ（笑）
ご飯食べながら悩んで、電話の人におすすめされたウラジオストク近くの綏芬河口岸から国境を超えることにする。お金が切れたのでクレカで借金。どうやって綏芬河まで行けばいいのか。地図とにらめっこをしてルートを考える。バスで虎林まで、そのあと鉄道で牡丹江まで、牡丹江からは綏芬河経由のウラジオストク直通バスを使うことに。昨日着いたバスターミナルでチケットを買ってバスに乗り込み、虎林に着いたら鉄道駅で牡丹江までの切符を購入。しっかり夕飯を食べて鉄道へ。深夜に牡丹江到着。牡丹江バスターミナルまでタクシーを飛ばしてウスリースクまでのバスチケットを購入（ウラジオストクまでの直通バスは無かった）。近くの宿を取る。

図 4　虎林→牡丹江の夜行列車

計画では今日入国するはずだったのに。

9/21　牡丹江

牡丹江は朝鮮から近いっていう地理的要因とか満州国だったときにその時日本国籍だった朝鮮人を植民したっていう歴史的要因とかがありまして朝鮮人が多い街です。バスターミナルの近くはめちゃくちゃ大きな朝鮮人街でした。夕飯のビビンバ美味しい。
あと、「饒河口岸展外国人口岸」の記事については、ロシア人向けの話じゃないかって Twitter のあさくらさん (@hirune_asakura) のご友人から情報を頂きました。と同時に、日本大使館に連絡して情報収集してみても良いのでは、とのこと。たしかにその通りでした。

9/22　牡丹江→綏芬河→ウスリースク→ウラジオストク

朝に起きてバスに乗る。朝鮮大冷麺食ってたら乗客が移動しだしたので、残してきてしまった。
バスで隣になった人と話をして仲良くなる。異国の街でそれなりに大変なことをするときは現地の人の手助けが重要。で、いよいよ綏芬河口岸へ。パスポート審査の順番が回ってきたと思ったら、別室へ連れて行かれた。パスポートを取り上げられて半時間近く待つ。あああああ。
僕のパスポートは青色の 5 年だけのやつで、それを見るのが初めてだったから真贋の区別に時間がかかったらしい。あああああ。バスへ戻ったらさっきまでにこやかに話してくれてた別の乗客に「早よせいや！」って怒鳴られる。ロシア人運転手もカンカンで、税関の検査が終わった荷物を中国人がバスの下部に入れきれなくて通路へ置いてたら、ブチ切れて蹴りまくってた。こわ。
さて、いよいよロシアへの入国なんですが、実はここでも懸念が 3 つほど。一つに、ロシアビザに入国地が書かれていないこと。「ロシア　陸路入国」でググると「ロシアへ陸路入国する際はビザに入国地が記載されていないといけません。空路で入国する際は必要ありません。」と書いてあるサイトが見つかりました。そんなんビザの申請時に書かなかったし、バウチャーにもそういうことを書く欄はなかった。もう一つの懸念は、ビザの滞在日程と実際に入国を試みる日付がズレている

こと。最後に、ビザに記載されている訪問予定地と入国する都市が食い違うこと。これらは旅程がめちゃくちゃになったのでそりゃこうなります。ここまで来て入国拒否されたらどうしよう……。で、白人の太ったおばさんにパスポートを出して、待ちます。顔をパスポートと見比べられ。うーん……。スタンプが押されて返って来ました。神。スパシーバ [*9]！
ウスリースクへ着いてとりあえずルーブルを手に入れなきゃって Cirrus[*10] で借金しようとしたんだけど、その ATM も受け付けてくれない。小さな街だから両替商も 1 件しかないのに日曜日で閉まってるし。寒いしご飯も食べられないし本当に死にそう。月餅美味しい。4 時間近く街をうろついていろんな ATM を試して、ある私銀の ATM に突っ込んだらお金が出てきて、泣きながら喜んだ。
やっと手にしたお金でウスリースクからウラジオストクまでのバスに乗り込む。本当にしんどい。ウラジオストクについたら真っ暗で、真っ暗で怖いロシアの街で違法タクシーを拾ったら「売春宿行くか？」って言われて、断ってウラジオストク駅に連れて行ってもらって、小さな駅舎の中で乗客に混じってシベ超 [*11] に乗ってくる友人を待つ。たしか 28 時に着くはず。寒い。ひもじいよぅ……。あっ　あれじゃない?!　オーイ！

終わりに

記事を書いてたら、これからどうなるんだろうっていう不安と寒いしおなか減ったっていうひもじさとあとお金を引き出せないっていうヤバさが思い出されてきて、手先が冷たくなった。

計画と実装された出費と移動図

日本国内で先に支払ったものを上、旅行中に支払ったものを下にし、かつ時系列順に並べました。表 2 を見て下さい。中国での滞在は 5 日間から 7 日間に増え、逆にロシアは 5 日間から 3 日間に減りました。中国国内でなんとか越境しようと高速バスや鉄道で動き回った分が大きく響いてますね。でも、ハバロフスクのホステル代とロシアの飯代が抜けて、合計額があまり変わってないことに驚く。
結局辿った移動図を見てみましょう。図 5 です。

最後に

中国東北部から極東ロシアへ渡るために必要なたった 3 つの条件とは???

- ロシアビザの空バウチャー

[*9] “ с п а с и б о ”。ロシア語で「ありがとう」。
[*10] Cirrus (シーラス) はマスターカードが中心となって運営しており、VISA 陣営の PLUS と並ぶ世界的な銀行のオンラインシステムである。【出典：Wikipedia】
[*11] 『シベリア超特急』（しべりあちょうとっきゅう，シベリアちょうとっきゅう）は、映画評論家・水野晴郎による映画および舞台の監督作品のシリーズ。監督名義はマイク・ミズノである。略称の『シベ超』（シベちょう）は、みうらじゅんによるもの。【出典：Wikipedia】

分類	細目	支払先	予 (JPY)	実 (JPY)
交通費	高松→上海	春秋航空	13390	13390
交通費	上海→ハルビン	春秋航空	13420	13420
交通費	ハバロフスク→ウラジオストク	Russian Trains	8000	8000
宿泊費	ハバロフスクのホステル（2 晩分）	HOSTEL WORLD	5000	200
宿泊費	ウラジオストクのホテル（2 晩分）	Expedia	8000	8000
雑費	空バウチャー発行代金	Travel Russia	1700	1700
交通費	新宿→京都	楽天バス	4000	4000
宿泊費	中国国内	@700JPY × 5	3500	5000
交通費	中国国内	@700JPY × 5	3500	17000
食費	中国国内	@700JPY × 5	3500	5000
交通費	ロシア国内	@1000JPY × 5	5000	5000
食費	ロシア国内	@2000JPY × 5	10000	5000
計			79010	85710

表 2　予算表

図 5　最終的な行動図

- 情報（収集能力）
- 地図とにらめっこをして旅程を立て直す機転

中国人は、悪気はないんだけども、かなり不正確な情報を伝えてくることが多いというのが今回の旅で身に染みました。ホンマな。はぁ。バス・鉄道がどこからどこを結んでいるか、そしてその時刻表、あと国境を第三国人が渡れるのかどうか。そういった情報って現地の人もわりと持ってない（のにデタラメに適当な事を言う）ので、とかく現地の窓口に相談が一番だと思います。では、みなさんもよい旅を。

SICP を頭に叩き込む

あとがき

■jf712　ここにはあとがきを書きます。

■@y_possum　今回は挿絵の編集担当とコラムの執筆者を兼任しました。あそこに書いたことは我々の野望です。卒業するまでにどこまでできるでしょうか。

■hideya　Calc=Calc の開発に関して書きたいことはまだまだあったのですが、文章構成力が足りなくてそこまで盛り込めませんでした。残念。
それはさておき、最近若さのピークを少し過ぎてしまった感じがあります。でもそれは、「お前はもう終わりだ」ということではなく「さっさと何かしろ」という事なのでしょう。だからまだ頑張ります。頑張りますよー。

■hatsusato　あまり仕事をできなかった校正担当です。部誌の 4 分の 1 近くを書いたのでそっちの貢献のほうが大きいですね。部誌を頑張りすぎて生活がままなってないのよくない。

■pastak　やっぱりインターネットは心地よい空間であって欲しいですね。ニホンゴムズカシイ。

■tyage　今回題材にした CTF の勉強会を KMC でも行っているのですが、まだまだ回数が少ないので来年は月に 1 回くらいのペースで行えるといいですね。それと、この 1 年間私を支えてくれた「ゆゆ式」に感謝を述べたいと思います。おっぱい。

■gire　日本初のプロコン漫画（自称）を描きました。ジャンルとしてまだまだ伸びしろがありそうなので、今後の発展に期待です。

■hidesys　uiureo の金魚のフンしてたら GifzoWin を米国企業に売ることになった話の方が KMC 部誌っぽかったかも。

■hanazuki　今回も夜中に Jenkins 氏に呼ばれて TEX のエラーを直すなどしました。あとがき執筆時点では Git リポジトリごと破壊されるようなことはなく、本誌は平和裏に完成する予定です。

■jf712　はじめての方ははじめまして、そうでない方はお久しぶりです。編集長を務めた jf712 です。
さて今回の「独習 KMC Vol.6」はいかがでしたでしょうか。この記事を書いているのが出版される前なので憶測でしか話せないのですが、おそらく「分厚いな」というのが最初の印象だと思われ

ます。前回 [*1] は記事を集めようと思ったのはいいものの、思っているまま原稿の締切日となってしまい、結局満足な数の記事を集められず、それはもう薄い薄い部誌になってしまいました。その薄さでコミケ会場で「500 円です」と言って頒布するのは非常に心苦しく（いや頒布しましたが）、次回はなんとしても胸を張って頒布できるような部誌にしよう、と思い立った結果がこの分厚さです。といっても、やったことと言って思い出せるのは、「 hogehoge さん、記事書きませんか」ぐらいなのですが。

今回記事を頼むにあたって、出来るだけ広い回生に頼むように気をつけました。最近の KMC は、毎年新勧期に数十名の部員が入部し、まぁ全員残るわけではないのですが、それでも毎世代 10 人程度は残っています。人がいなくなるよりは全然いいことではありますが、一方で世代が離れてることによる弊害 [*2] のようなものも徐々に出てきているように感じます。今一番気がかりなのは、巻頭に Moko さんが書いておられましたが、IRC に替わる連絡手段です。大体僕より上の世代の方々はサークル内連絡手段として IRC を用いています。一方で、僕から下の世代は、KMC に入る前つまり中学高校時代から Twitter などのコミュニケーションツールを使ってきて、わざわざ新しく KMC のためだけに IRC を導入するのもという感じであまり IRC を積極的には使っていない現状です。上の世代（というか僕）からしたら、サークルに入ったんだからそこらへんは郷に入っては郷に従え感覚で導入するものじゃないのかな、という気分なのですが、若い世代からしたらそういうものでもないのかな、とも思います。実際、サークル内で IRC を連絡手段としてメインに使っている人間が、世代で見るともう僅かです。これから KMC を運営していくのは年上ではなく若い彼らですから、おそらく何か新しい（もしくは既存の）連絡手段を考えてくれるのでしょう。とはいえ、なにもしないのもアレなので、IRC に ping 専用チャンネルを作りました。これも巻頭で Moko さんが書いておいでですね。

人が増えるということは、外から見れば何も問題ないですし、中から見ても毎日のようにプロジェクトやらピザパーティやら勉強会やらアニメ観るやらが立っていて、それはそれでにぎやかでいいことなのですが、一方でほころび（これを負の要素と捉えるか否かもまた考え方によるのかもしれません）が出始めているような気がします。

今回はそういった中で、とりあえず KMC の現役全世代から記事を集めて一つの部誌に集めることで、とにもかくにもいろいろな人が集まっている今の KMC を切り取って保存したつもりです。それに何の意味があるのかと問われると困りますが。

という建前で記事を無節操に集めた結果、このように分厚い部誌が出来上がりました。この本を手に取った方々が「今の KMC」を感じていただければ、幸いです。

それでは、よいお年を。

追記：12 月のインターネットと言えばアドベントカレンダー！ ということで、KMC も御多分に洩れず KMC アドベントカレンダー企画 [*3] が立ち上がり、記事が追加されました。技術的な話が多数書かれていますので、そちらも是非ご覧ください。

[*1] 前回も編集長を務めました。

[*2] これを「弊害」と呼ぶあたりがもう老害ですね。

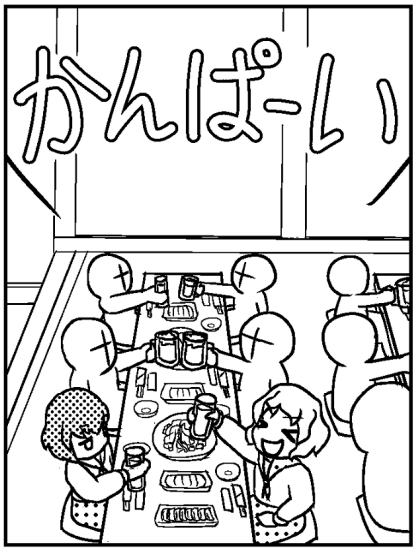
[*3] `http://www.kmc.gr.jp/advent-calendar/`

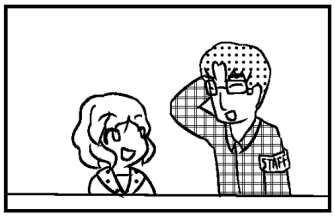

大会終了
お疲れ様でした
STAFF

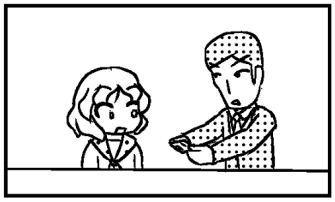

どうだった？
ぷすぷす
2完…


これから打上げに行くわよ
何ですかそれ

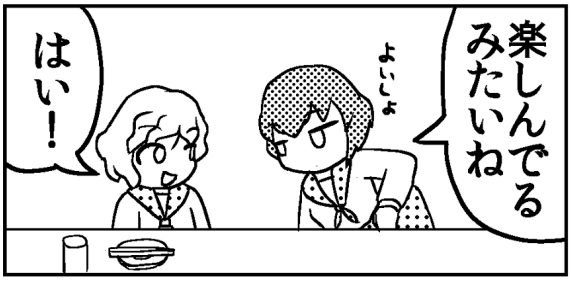

大会終了後は参加者同士の親睦を深めるために打上げが開かれるのよ
行きます！


かんぱーい


STAFF



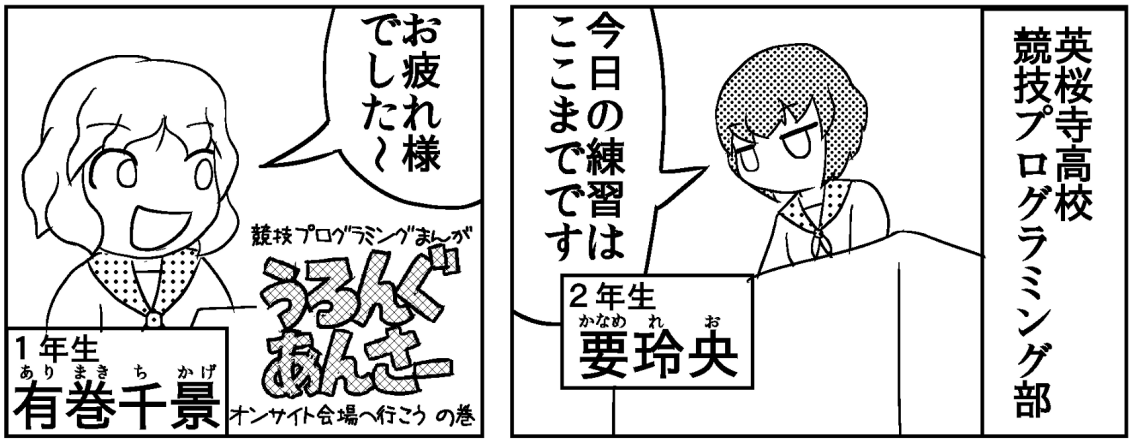
…

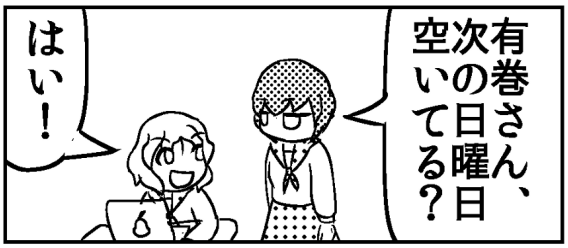

楽しんでるみたいね
よいしょ
はい！


そろそろアルハラでもしようかしら
ずいっ
え?!
私たち未成年ですよ?

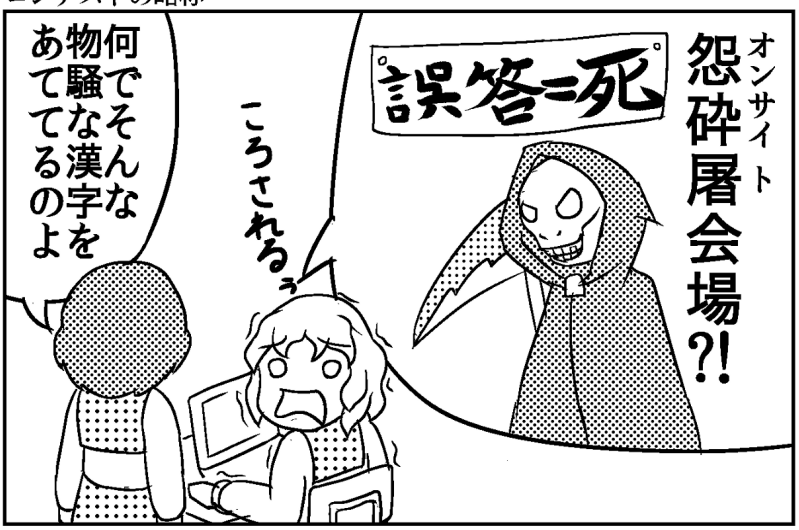

C問題は金曜日の練習会で出た問題でしょ？
はい…
D問題も…
アルゴリズム・ハラスメントでした
おしまい


日曜日
大学の教室だ〜
楽しそうね
AZUSA
だって、プロコンやってる高校生に会えるんですよ？
えっ
参加者は私達以外大学生かそれ以上だけど？
模試ってたとえが悪かった？
ぎょ
逃げるな
大会開始
開始です
STAFF
まずはA問題
どれどれ…
カタカタ
順位表
name A
1 mk5 100
2 azusa_2 100
3 kani 100
4 dr_bombay 100
5 chameleon 100
もうA問題通してる!!
ゆるしてくれ〜〜
みんな問題の文読むの早っ!!!
タイプ音
タ−ンッ





※プロコン…プログラミングコンテストの略称



独習 KMC vol.6

2013 年 12 月 31 日　初版発行

著作・発行	京大マイコンクラブ
表紙デザイン	hideya
裏表紙デザイン	crys
挿絵	koji
メールアドレス	info@kmc.gr.jp
Web	http://www.kmc.gr.jp/

落丁・乱丁の際は在庫がある限りお取り替えいたします。上記のメールアドレスまでご連絡ください。