

総称選択(generic selection)の話(C言語)

cattus412

概要

総称選択の文法、C11からC17で修正された仕様の話と使用例を一つ。

総称選択の文法

C17での総称選択の文法です、規格を引用しつつ簡単にまとめます。

構文と制約

```
generic-selection:  
    _Generic ( controlling-expression , generic-assoc-list )  
  
generic-assoc-list:  
    generic-association  
    generic-assoc-list , generic-association  
  
generic-association:  
    type-name : expression  
    default : expression
```

ただし、

- **default**をもつgeneric-associationは二つ以上存在してはいけない
- 互いにtype-nameが互換(compatible)な型(type)をもつgeneric-associationが存在してはいけない
- **default**をもつgeneric-associationが存在しない場合、controlling-expressionの型と互換な型をtype-nameにもつgeneric-associationが存在しなければならない
- controlling-expression、expressionはコンマ演算子(comma operator)を除く任意の式
- type-nameは可変修飾(variably modified type)でない任意の完全オブジェクト型(complete object type)
- controlling-expressionの型に対して左辺値変換(lvalue conversion)、配列型からポインタ型への変換⁽¹⁾、関数型(function type)をもつ関数指示子から関数ポインタ型への変換⁽²⁾がされる

以下は規格該当部(N2716 6.5.1.1 Generic selection - Constraints)の引用。

A generic selection shall have no more than one **default** generic association. The type name in a generic association shall specify a complete object type other than a variably modified type. No two generic associations in the same generic selection shall specify compatible types. The type of the controlling expression is the type of the expression as if it had undergone an lvalue conversion, array to pointer conversion, or function to pointer conversion. That type shall be compatible with at most one of the types named in the generic association list. If a generic selection has no **default** generic association, its controlling expression shall have type compatible with exactly one of the types named in its generic association list.

意味規則

- *controlling-expression*の型と互換な型を*type-name*にもつ*generic-association*が存在した場合、総称選択の結果の式はその*generic-association*の*expression*となる
- *controlling-expression*の型と互換な型を*type-name*にもつ*generic-association*が存在しない場合、総称選択の結果の式は**default**をもつ*generic-association*の*expression*となる
- *controlling-expression*及び選択されなかつた*generic-association*の*expression*は評価されない
- 総称選択の値(value)、値カテゴリ⁽³⁾、型はその結果の式の値、値カテゴリ、型となる

以下は規格該当部(N2716 6.5.1.1 Generic selection - Semantics)の引用。

The controlling expression of a generic selection is not evaluated. If a generic selection has a generic association with a type name that is compatible with the type of the controlling expression, then the result expression of the generic selection is the expression in that generic association. Otherwise, the result expression of the generic selection is the expression in the default generic association. None of the expressions from any other generic association of the generic selection is evaluated.

The type and value of a generic selection are identical to those of its result expression. It is an lvalue, a function designator, or a void expression if its result expression is, respectively, an lvalue, a function designator, or a void expression.

サンプルコード

```
puts(_Generic((int){1}, int : "is int", char : "is char", default :
"default")); //-> is int
puts(_Generic((int){1}, int : "is int", char : "is char")); //-> is
int
```

```

puts(_Generic((long){1}, int : "is int", char : "is char", default :
"default")); //-> default

void foo(void){
    puts("foo");
}

puts(_Generic(foo, void (*)(void) : "is pointer to function
returning void", default : "default"));
//^ is pointer to function returning void

typedef int hogehoge;
puts(_Generic((hogehoge){1}, int : "is int", char : "is char")); //-
> is int

typedef void (*fuga)(void);
puts(_Generic(foo, fuga : "is pointer to function returning void",
default : "default"));
//^ is pointer to function returning void

```

`typedef`⁴による型エイリアス(`typedef name`)も当然使用できます。

C11からC17への修正点

C11からC17へは大きな言語機能の変更、追加はなかったものの様々な修正がされました。総称選択もそれらの内の一つです。

C11とC17の規格

どちらも6.5.1.1 Generic selection - Constraintsを引用。

C11(N1570)

A generic selection shall have no more than one **default** generic association. The type name in a generic association shall specify a complete object type other than a variably modified type. No two generic associations in the same generic selection shall specify compatible types. The controlling expression of a generic selection shall have type compatible with at most one of the types named in its generic association list. If a generic selection has no **default** generic association, its controlling expression shall have type compatible with exactly one of the types named in its generic association list.

C17(N2716)

A generic selection shall have no more than one **default** generic association. The type name in a generic association shall specify a complete object type other than a variably modified type. No two generic associations in the same generic selection shall specify compatible types. The type of the controlling expression is the type of the expression as if it had undergone an lvalue conversion, array to pointer conversion, or function to pointer conversion. That type shall be compatible with at most one of the types named in the generic association list. If a generic selection has no **default** generic association, its controlling expression shall have type compatible with exactly one of the types named in its generic association list.

変更点

変更点はC17には*controlling-expression*の型の左辺値変換、配列型からポインタ型への変換、関数型から関数ポインタ型への変換についての言及が追加されています。C11にはこれらについての言及はありませんでした。

実は、6.3 Conversionsに記載されている型変換を総称選択の*controlling-expression*にも適用するかどうかで解釈のゆらぎが起こっていたのです。実際に、GCCやClangなどの著名なコンパイラ間でも実装が異なっていました。この欠陥についてのレポートであるDR 481では以下のようない例が挙げられています。

```
char const* a = _Generic("bla", char*: "blu");           //  
clang error  
char const* b = _Generic("bla", char[4]: "blu");          //  
gcc error  
char const* c = _Generic((int const){ 0 }, int: "blu");    //  
clang error  
char const* d = _Generic((int const){ 0 }, int const: "blu"); //  
gcc error  
char const* e = _Generic(+ (int const){ 0 }, int: "blu");   //  
both ok  
char const* f = _Generic(+ (int const){ 0 }, int const: "blu"); //  
both error
```

このようなコンパイラ間での実装の差を避けるために条件演算子(conditional operator)のオペランドにすることで型変換を強制する`_Generic(1 ? (X) : (X), ...)`のような奇妙なテクニックまでもが登場しました。

議論の後、結果的にC17では*controlling-expression*の型に左辺値変換、配列型からポインタ型への変換、関数型から関数ポインタ型への変換をすることとなりました。

使用例と解説

総称選択は関数形式マクロ(function-like macro)と組み合わせて下記のようにC++などの関数オーバーロードと表記上似たようなことを可能にするために使用されることがあります。C99で追加されたtgmath.hに含まれる型総称数学関数(Type-generic math)も似たように実装できます。

```
#define foo(ARG) _Generic((ARG), int : foo_int, long : foo_long,  
char : foo_char)(ARG)  
  
foo((int){0}) //-> call f_int  
foo((long){0}) //-> call f_long
```

(*f_int*, *f_long*, *f_char*は全て関数指示子とします。)

この例では上記はCプリプロセッサ(preprocessor)により以下のように展開されます

```
_Generic(((int){0}), int : foo_int, long : foo_long, char :  
foo_char)((int){0})  
_Generic(((long){0}), int : foo_int, long : foo_long, char :  
foo_char)((long){0})
```

これを見ると総称選択がCプリプロセッサのようにソースファイルを書き換えることで呼ばれる関数を分けているように見えるかもしれません、そんなことはありません。まず、上の例では*f_int*が、下の例では*f_long*が総称選択の結果の式となります。

総称選択の後ろにある(*ARG*)は関数呼び出し式

```
expression ( argument-list )
```

の()の中に*ARG*を展開したものとなります。

このとき、*expression*は*void*又は配列型以外の完全オブジェクト型を返す関数へのポインタでなければなりません。(5

この例においての総称選択の結果の式はどちらも関数指示子(`f_int`と`f_long`)であるため、関数型を持ちます。しかし、これらは関数ポインタ型へと型変換される⁽⁶⁾のでこのコードで総称選択で選択された関数指示子が指す関数を呼び出すことが出来ます。

注釈と小話

1. **array of type** から **pointer to type** へ(N2176 6.3.2.1 Lvalues, arrays, and function designators - 3)
2. **function returning type** から **pointer to function returning type** へ(N2176 6.3.2.1 Lvalues, arrays, and function designators - 4)
3. lvalue,rvalue(non-lvalue object),function designatorのうちいずれか
4. **typedef**の小話

typedefは**typedef int hoge**のように使われることが多いからか、

```
typedef type-name alias;
```

のようなRustの**type**、C++の**using**のに似た構文をしていると思われていることがあります、配列型や関数ポインタ型の型エイリアスの宣言を見ればわかるようにこれは誤解です。

正しくは、**typedef**は文法的には記憶域クラス指定子(storage-class specifier)にあたり、記憶域クラス指定子が**typedef**の宣言子(declarator)は識別子(identifier)をその指定された型の型エイリアスとして定義するという仕組みです。

以下は規格該当部(N2716 6.7.8 Type definitions - Semantics)の引用。

In a declaration whose storage-class specifier is **typedef**, each declarator defines an identifier to be a **typedef** name that denotes the type specified for the identifier in the way described in 6.7.6. Any array size expressions associated with variable length array declarators are evaluated each time the declaration of the **typedef** name is reached in the order of execution. A **typedef** declaration does not introduce a new type, only a synonym for the type so specified.

5. (N2716 6.5.2.2 Function calls - Constraints)
6. 注釈 2 の型変換が行われる

静的な型と黒魔術に浸ろう ~型に嵌 まった人生でもいいじゃない~

H2 にこなのにふわわあ

導入: 型ってなに

プログラミングにおけるデータ型（データがた、 data type）あるいは単に型（かた、 type）は、値の種類を示し分類分けするラベルである。データタイプともいう。

データ型 - Wikipedia

プログラミングでは数も文字も扱う必要があるんですが、コンピュータの中では1と0の羅列でしかないので工夫なしには区別が効きません…ここで値が何を意味するのかを示すしくみが型というわけです

前提: プログラミング言語には大きく分けて2種類ある

- 静的型付け: コードを書いている段階で型が決まる。
- 動的型付け: 実行しているときに型が決まる。

比較項目	静的型付け	動的型付け
コーディング中の手間	大: 型のことを常に気にする	小: 特になし
型システムの複雑さ	莫: 間が深い	微: そんなものは存在しない
コンパイル時間	長: 型システムのチェック	短: そのまま動かすだけ

いかにも動的のほうが良いような比較項目を並べました(印象操作)が、今回推すのは静的のほう。何が良いのかじっくり解説していこうではありませんか…

静的型付きのメリット

1. ミスを避けられる

プログラムが間違ったデータを処理しようとするとどうなるかというと、当然エラーが出る(もしくはバグる)んですね。これをコードを書いているときに発見できるんですね。もちろん実行するまで気付かないこともあります。発見しやすくなります。それから、単純にコードにミスがある場合に気付くやすくなります。

2. リファレンスを見なくても生きていける

ライブラリを使用するときなど、どの順番で何を与えれば動くのかといった情報が型で与えられるので、わざわざリファレンスを読みにいく手間が省けます。プログラマは調べる時間が9割なのでけっこうありがたいのです。

~~そろそろ執筆モチベが消えてきたのでこのあたりで終わりにします。調べる~~
とても深い世界なので興味がわいたら是非しらべてみてください…

おまけ: 黒魔術

型システムはその機能の強さ故想定されていない使われかたをすることがあります。そういうことをする変態(私含む)がわりとたくさんいます。最近、[ts-sql](#) というコンパイル時にSQL文を解析して実行するものが出現して話題になりました。半分遊びのようなものなのですが…

はじめて。高1のDOLLYです。
部誌には初参加の幽霊部員ですが、
以後宜しくお願ひします。
さて、今回は最近観賞した映画を紹介します。
大体はアマプラかHuluで観れるので、
気になった作品は
観賞していただけたら、と思います。
全体的に文章は短くしているので、
暇な時にでもご一読下さい。

本レビュー(感想)集において、
基本的に文中では、
ストーリーに深く触れることなく、
各作品の紹介を試みている。
これは筆者の
「各作品をなるべく楽しんで観賞してもらいたい」
という思いによるものであり、
筆者は各紹介作品に関して、
最低一度は観賞を行っていることを、
レビュー本文前に記しておく。

題:トータル・リコール(旧)
「失った記憶を求めて、火星に行く男の話」
徹頭徹尾「大胆」な作品。
小道具は大体爆発する、と考えてよいだろう。
シュワちゃんのアクションも相まって、
制作当時のある種「大胆」な未来予想図を
五感をもって堪能できる。
今はない「大胆」さが魅力的な一本になっている。

題:トータル・リコール(新)
変装道具は爆発しない。これが本作の全て。
「大胆」さを尽く今風な「スタイリッシュ」に置換した謎リメイク。
それでいて前作オマージュが半端にされるから
設定と絵面がかみ合わないという最悪な有様。
映画版大人のガリガリ君。
資本主義に疑問を投げかけざるを得ない一作。

題:サマータイムマシン・ブルース
劇団「ヨーロッパ企画」が原作を務めたSF映画。
「壊れたクーラーのリモコンを、タイムマシンで昨日から押借する」
というしようもない状況から、
頭がショートしそうな高濃度のSFが展開される様は圧巻の一言。
藤子F不二雄の短編の様な空気感が心地いい一本である。

題:トゥルーマン・ショー
人生を見世物にされた男の話。
所謂「ファミリーYoutuber」を想起した。

丁寧な演出の中で描かれるは成長と自由の獲得。
それらを小気味良いコメディ調の演出が包む一作。
SNS全盛の時代、プライバシーと自己顯示欲を
天秤にかける今だからこそ、改めて視聴してみては。

題:悪の教典

伊藤英明主演の自称サスペンス系スプラッタ。
教師が真夜中の学校で生徒を皆殺しにする話。
惨殺中も確認できる主人公の几帳面さや、
くどいほど流れる名曲" Mack the knife "が
妙な統一感を演出している。
故に続編を匂わせるEDと
その間流れるエグザイルは蛇足でしかないが。

題:シャークネード

「巨大竜巻に乗って無数のサメが街に降り注ぐ」という
フォークト=カンプフ検査みたいな一作。
いわゆる「サメ映画」ではあるものの、
アクションは爽快、ストーリーも「家族との復縁」と
分り易いので比較的楽しく鑑賞できる。
少々CGがちゃちっいが、そんくらいしか言ふことがない。

題:レイザーヘッド

カルト映画の金字塔。デヴィッド・リンチ監督作。
当時リンチが体験した「若くして父になった苦悩」を、白黒の画面と環境音が包み
込む。
多くの描写が抽象的で理解に苦しむ物でありながら、
飽きも許さずにのめりこませる本作は正しく「映像体験」。
109分間の酩酊を、ご堪能あれ。

題:メッセージ

~ 宇宙NPOと円形文字の秘密 ~
長寿アニメの劇場版みたいなサブタイを勝手に付けたが、
大体はこんな感じの話。
最大の持ち味は全編通した構成の巧さ。
メリハリのメリでも退屈しないし、伏線はしっかり回収される。
もし全編観ても理解に困ったら、「円形文字の秘密」を思い出してみよう。

題:太陽を盗んだ男

日本産カルト映画の代表格。
「個人が原爆を作り政府を脅す」
という過激な筋書きながら、
主演の沢田研二のコミカルな演技や
シユールな演出により、意外と抵抗なく楽しめる。
総じて7,80年代的なエネルギーッシュさの目立つ作品である。
今の映画に飽き飽きしたら是非視聴してみては。

題:デビルマン

同名漫画の実写化にして、
邦画におけるサタンポジション。
大根演技と制作陣上層の怠慢、
ツツコミ所ばかりの内容は目に余るが、

各場面の描写は妙に丁寧。
それにより生じた白昼夢じみた雰囲気には、
作品の出来を超えた確かな凄みが確認できた。
この作品は見てから笑い飛ばすべき一本だ。

題:パプリカ
今敏監督のアニメ作品。
夢と夢が混ざり合い、
現実と区別がつかなくなる話。
手描きのセルアニメで描写される悪夢は圧巻で、
まるでコンピューター・グラフィックスなんだ、それが！
全体的にカルト映画一歩手前の
熱狂的な支持を受けている作品なので、
正直実際に観ての合う合わないが大きい。

題:レディ・プレイヤー1
ゲームオーバーでアカウント削除という
狂いバランスのVRゲームで宝探しをする話。
ガンダム、シャイニング、果てはゴジラと
ハチャメチャなクロスオーバーが楽しめる。
画の圧と演出、CGは間違いない一級品。
しかし相対的にストーリ部分が負け気味な感触。
個人情報は守ろう。

題:ターミネーター
未来から暗殺者が来る話。
カースタンドをはじめとしたアクションも見所だが、
どちらかといえば全編通してホラー要素が強めな。
しかし、仕方のないことではあるものの、
公開から四半世紀は経過した作品のため、
劇中の未来と我々の現在に乖離が生じている。
気にならないなら視聴し徳。

blender で実用的でないことをしよう！

onsen_manjuuu

1. はじめに

こんにちは、高校一年の onsen_manjuuu です。普段は競技プログラミングをしています。部誌を書くということで競技プログラミングのことを書けるといいのですが、残念なことに書くことが思いつかないので blender と呼ばれるソフトウェアの話をしようと思います。この原稿で blender を知らない方には blender に興味を持ってもらえばいいと思います。blender に元から詳しい方にはこの記事から実用的な技を生み出していただければ嬉しく思います。ところで、僕が知っている blender と最近の blender には乖離があるため頓珍漢なことが書かれていたりするかもしれないので、有識者の方々はそのような箇所を発見したら教えていただけると幸いです。

2. blender ってなーに

先ほど「blender、blender」と連呼しましたが blender とは何でしょう、ここで説明するよりも「blender」とググってみる方がわかりやすいのですが、簡単にいえば blender とは高性能で無料の 3DCG ソフトウェアです。オタクの皆さんのが好きであろう MMD のモデルを作ったり VRChat のモデルを作ったりできるらしいです、すごいですね。そんなすごい blender ですが本稿では blender の特に役に立たない使用法について紹介します(役に立つ使用法についてはさんざん説明されているので)。

3. bpy を使って何かをする

bpyをご存じでしょうか、 bpy とは blender の操作ができる Python スクリプトです。 bpy を使うと退屈な大量コピーなどの操作が簡単にできるようになりますがあまり blender ユーザーに知られてない気がします(個人の感想)、この章では bpy の使い方について紹介したいのですが、実際にはここから紹介するようなことを行う場合には Unity などを使った方がよさそうな気がします。(僕は Unity が扱えないでよくわからない)

さて、 bpy の話です。 bpy を使うとアルゴリズムの動き可視化などができます、 いいですね。

例えば Space Colonization という自然な木を生成するアルゴリズムがあります、 右の写真は Space Colonization のアルゴリズムをもとに bpy で作った木ですが、 この木を手作業で作るのはかなり大変だと思います(Space Colonization で木を生成する Add-on があるというのは秘密です)しかし bpy を使うと幹からの距離と枝の始点と終点をもってループを回すとこのような木が簡単に作れます、 便利ですね。 でもこういうのは Houdini なり Unity なり使った方がいいはずです。 ということで役に立たない bpy の使い方の紹介でした。

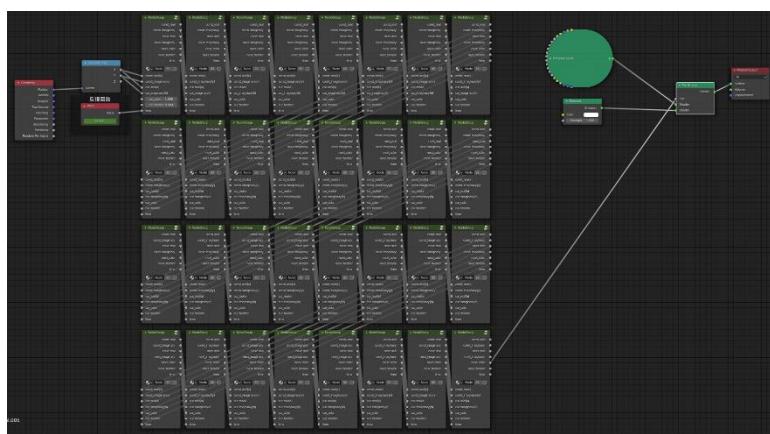


4. マテリアルノードを使って何かする

blender のマテリアルノードはとても便利で効果的です、 マテリアルノードを使うことで物質に自然な質感や奇妙な質感を与えることができますがここでは物質に質感を与える本来の機能とは違ったマテリアルノードの使用法を紹介しようと思います。

フラクタル図形を描画する

マテリアルノードを使ってマンデルブロ集合を描画することを考えましょう。 減化式をそのままマテリアルノードで表せばいいのですが、 残念ながらマテリアルノードではループが回せないので図 4.1 のように気合でループを回します、 その結果として図 4.2 のようなものを作ることができます。



←図 4.1

↓図 4.2



図 4.2 だけを見るとこの方法での図形の描画は普通にテクスチャを貼るのと大差がないように感じますが、図 4.1 のマテリアルノードの反復回数と書いてあるイテレーションを回すと(解像度の関係で読めないかもしれません)が青いノードの下です)図形の細かさを変えるような動画を作ることができます。

しかしこのマテリアルはかなり重いので実用にはあまり向いていません、悲しいですね。(そういえば図 4.2 は図形を球に投影している関係で端の方がゆがんでいますね)

レイトレーシングもどきをする

blender のマテリアルノードはシーン内のカメラから手に入れた情報を扱うことができます。この情報を使って平面のポリゴンに球を投影したりすることができますが「レイと球の交差判定」等でググって出てきた内容をそのままマテリアルノードで表現するだけなので詳しいやり方は省略します。

5. おわりに

ここまで読んでいただきありがとうございます。かなりニッチな内容になってしまったことを書きながら反省していました。次はアニメに出てくるパソコン研究会の描かれ方でもまとめてみたいですね。何はともかく blender には特に意味もない実用的でない使い方があることを感じていただければ幸いです。

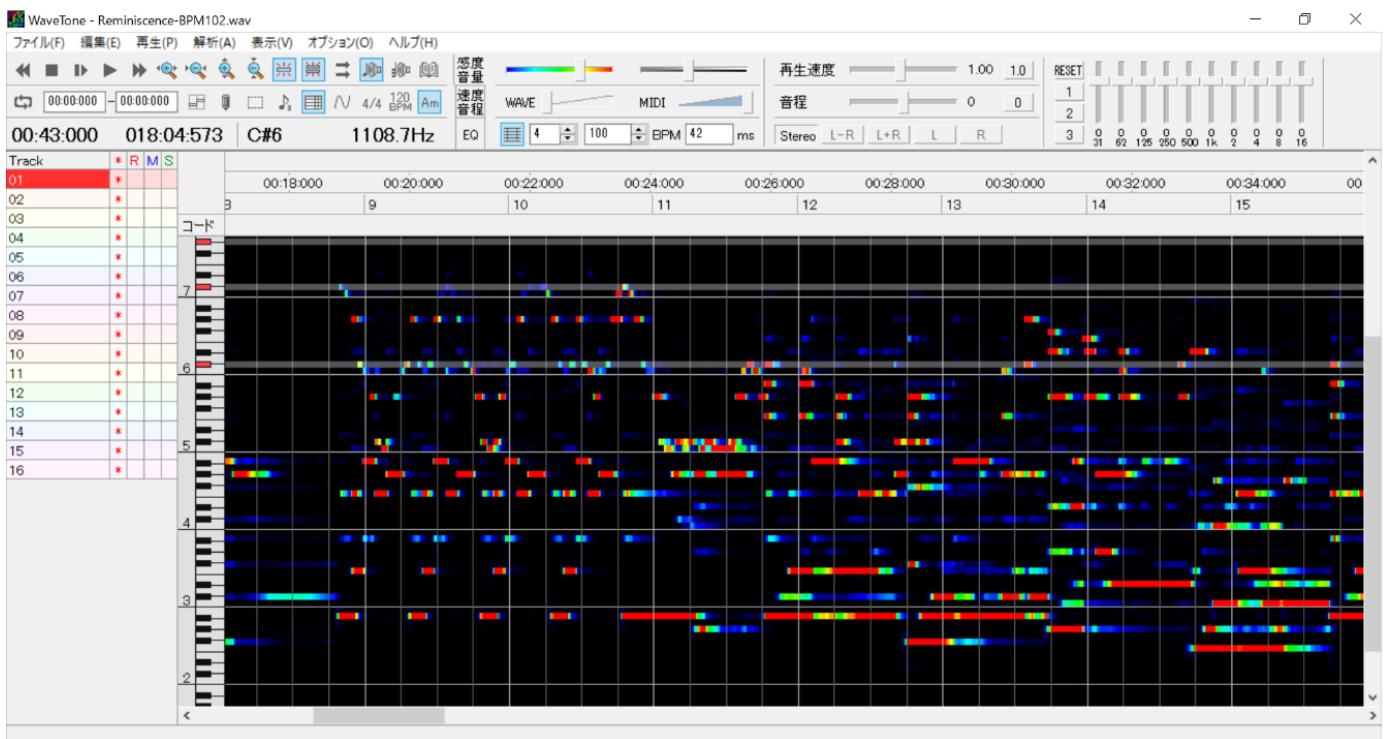
最強耳コピソフト WaveTone

H1 水上

◎WaveTone とは

WaveTone とは、Ackie Sound 氏製作の超便利な耳コピ・採譜支援ソフトです。このソフトを使えば、絶対音感が無くとも耳コピができるようになります。

具体的には、耳コピしたい曲を読み込ませることで下のような画面がでてきて、縦軸に音の高さ、横軸が時間、赤いほど音が大きい、といった感じでどこでどの音が鳴っているかというのが分かるようになります。例として部員の H1 高橋が作った曲を使います。



このように聴覚的にしか音の情報を得られなかったのが視覚的にも分かるようになるので、耳コピが半端なく楽になります（耳以外も使うのに耳コピというのも変ですが）。

しかし実際には演奏されていない音（※）もたくさん表示されるので、表示されている音全部書き出せば採譜完了という事にはなりません（ここが WaveTone の難しい所！）。しかしこれから紹介する様々な機能と音楽理論の知識を駆使すれば、多分どんな曲だって耳コピできるようになります（ここが WaveTone の凄い所！）。これからはその便利機能たちをどのように使うのか説明していきます。

※実際に鳴らしている音よりも高い所で一緒に鳴り響く音（整数次倍音）とノイズ的なやつ（非整数次倍音）。合わせて「倍音」と言う。

◎使い方

ニコニコ大百科の WaveTone のページがやたら丁寧だったので、本気で学びたい方はそういうのを参考にしましょう。ここでは WaveTone にはどんな機能があるか、どんな感じで WaveTone を使っているか、雰囲気だけ紹介することにします。

○導入

「Wave Tone」で調べれば Ackie Sound 氏のサイトが出てくるのでそこでダウンロードしてください。

○基本的な操作

耳コピしたい曲の音声ファイルを WaveTone にドラッグ&ドロップするか [ファイル→開く] から選ぶかすると曲を解析できます。そうするとさっきの画面が出てくるので、下三つの機能とコード等の音楽理論の知識を使いつつ採譜していくのが基本的な操作となります。



1.感度&コントラスト

感度…画面にどの基準より大きい音を表示するか決められる。

コントラスト…表示されている中での音量の差（赤いほど大きい）をどのくらい誇張するか決める。

2.解析するチャンネル

ステレオ音源（今自分たちが聞いている音声はほとんどこれ）は左と右の二つの音声（チャンネル）を合わせることで立体感を持たせている。これを片方ずつ聞いたり、どちらか片方でしか鳴っていないものを取り出して聞いたりできる。

3.イコライザ

特定の周波数（音の高さ）の音の大きさを変えられるもの。これを使う事で聞こえづらい音を聞きやすくできる。

ここから紹介するのはそれを楽にする便利機能たちです。

○便利機能

1.解析

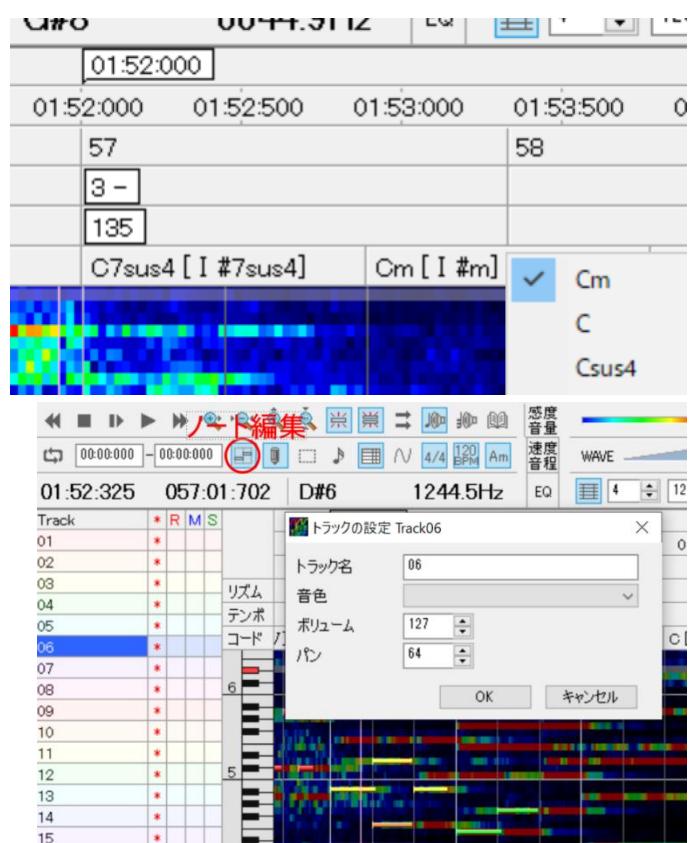
左上にある [解析] からテンポとキーとコードを解析でき、[表示] から～～トラックを表示することで右のような画面になります。曲の途中で BPM・拍子は変更できます。コードも必要に応じてこちらで調整しましょう。これらは採譜の大きな助けになるし、作曲などする方はその参考にもなると思います。

2.ノート編集

ノート編集のボタンを押すことでノート編集モードになります、簡単な DTM っぽい事ができます。音色や音量を調節できますがあくまでメモに使える程度の機能しかありません。ここで作成したノートを MIDI ファイルに書き出すこともできます。

◎おわりに

WaveTone の最強さが伝わりましたでしょうか。自分は音 MAD を作りはじめたときにこのソフトの存在を知りましたが、このソフトは便利すぎて音 MAD にばっかり使われるのは勿体ないと思います。リミックスや音 MAD などを制作する予定は無くとも、好きな曲を耳コピする作業はきっと楽しいので是非皆さんさわってみてください。



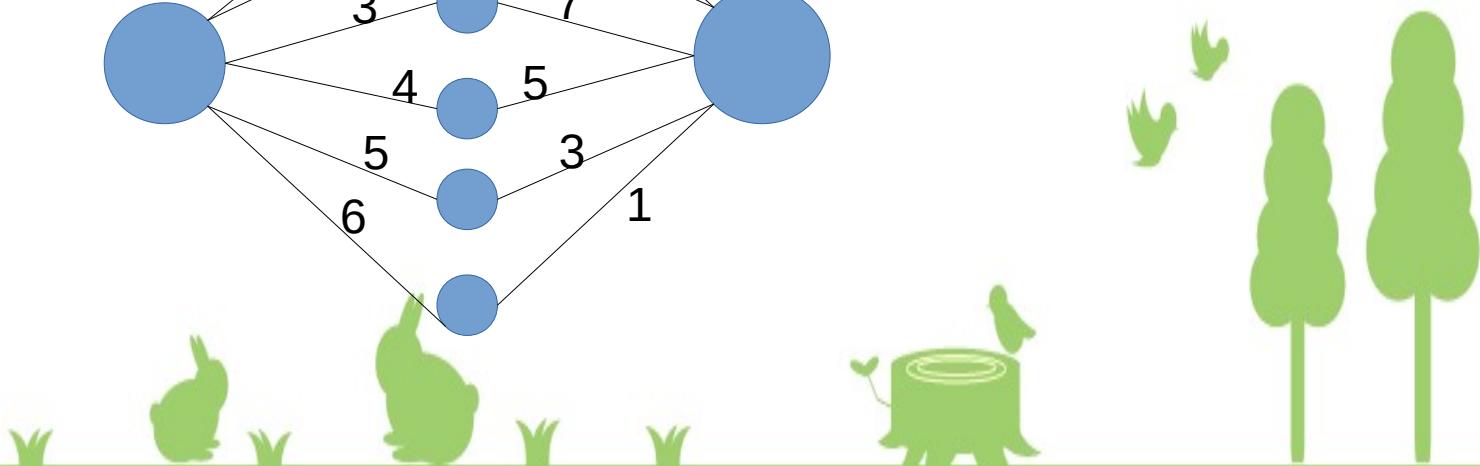
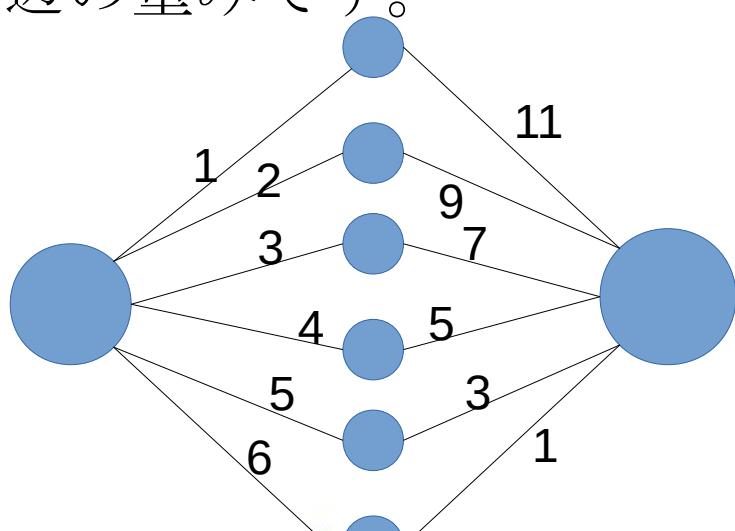
間違ったdijkstra法

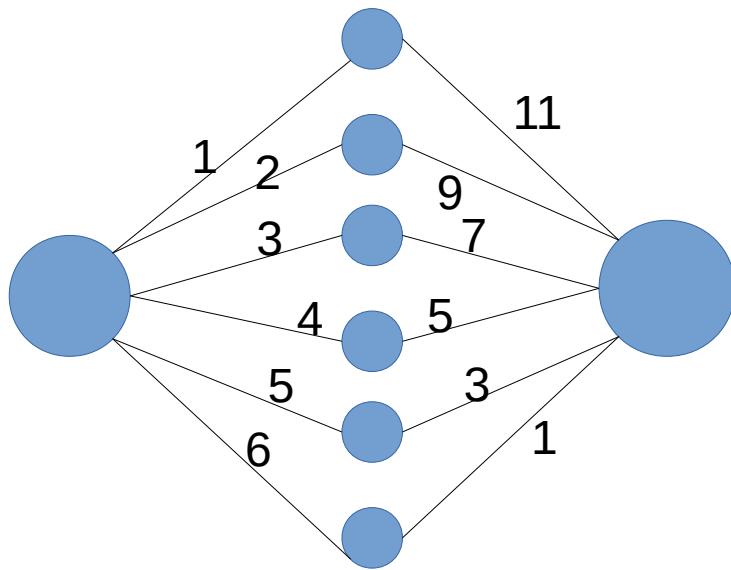
QCFium

dijkstra法は非負重み付きグラフの最短経路を求めるアルゴリズムとして有名です。その中でもC++における`std::priority_queue`（ヒープ）を使用する亜種は、頂点数N辺数Mのグラフの一頂点からの最短経路を $O((M + N)\log(N))$ で求めることができます。

しかし実装のミスによってこれが $\Omega(N^2)$ になってしまことがあります。（ Ω は簡単に言えば Ω の中身の定数倍以上かかるという意味）

下のグラフを見てください。辺についてる数がその辺の重みです。



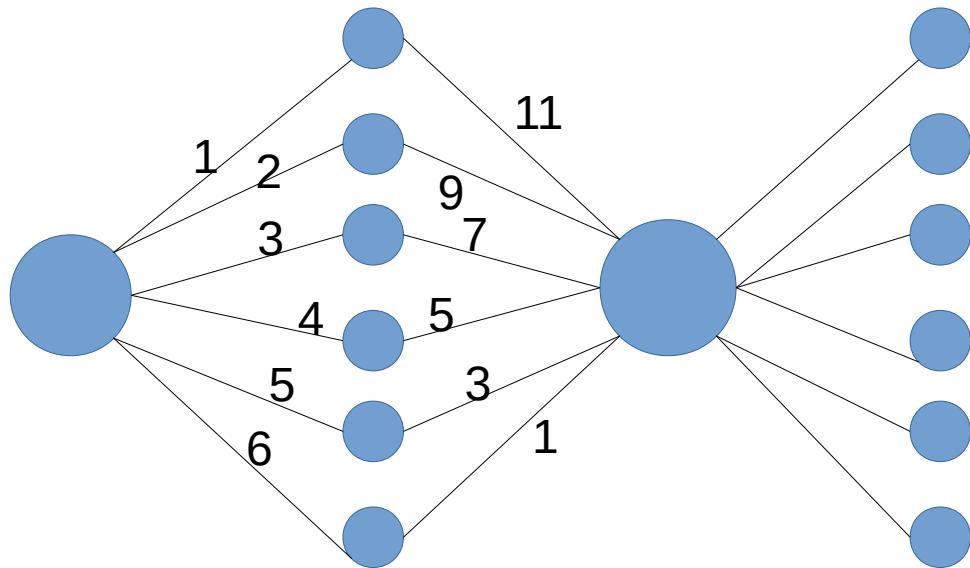


このグラフで一番左の頂点を始点として正しいdijkstraを実行するとどうなるでしょうか？

まず真ん中の6つの頂点が全てヒープにpushされます。始点からの距離が短い順にpopされるのでこれらは上の頂点からpopされていきます。popされると一番右の頂点が見られますが、一番右の頂点への距離は上から順に**12, 11, 10, ...**と減っていくのでpopされる度に一番右の頂点の距離は更新され、一番右の頂点がヒープにpushされます。一般には $\Theta(N)$ 回一番右の頂点がpushされます。



さて、ここからが問題です。下のグラフのように一番右の頂点から更にたくさん辺が生えていたらどうなるでしょうか



`priority_queue`から`pop`する度に右側の $\Theta(N)$ 個の頂点を見ていると $\Theta(N^2)$ になってしまいます。これが間違ったダイクストラです。

このような意図的に作られたグラフ以外ではなかなか遅くならないので問題が発覚しないことが多いですが、競技プログラミングでは嘘解法ということになる可能性があります。

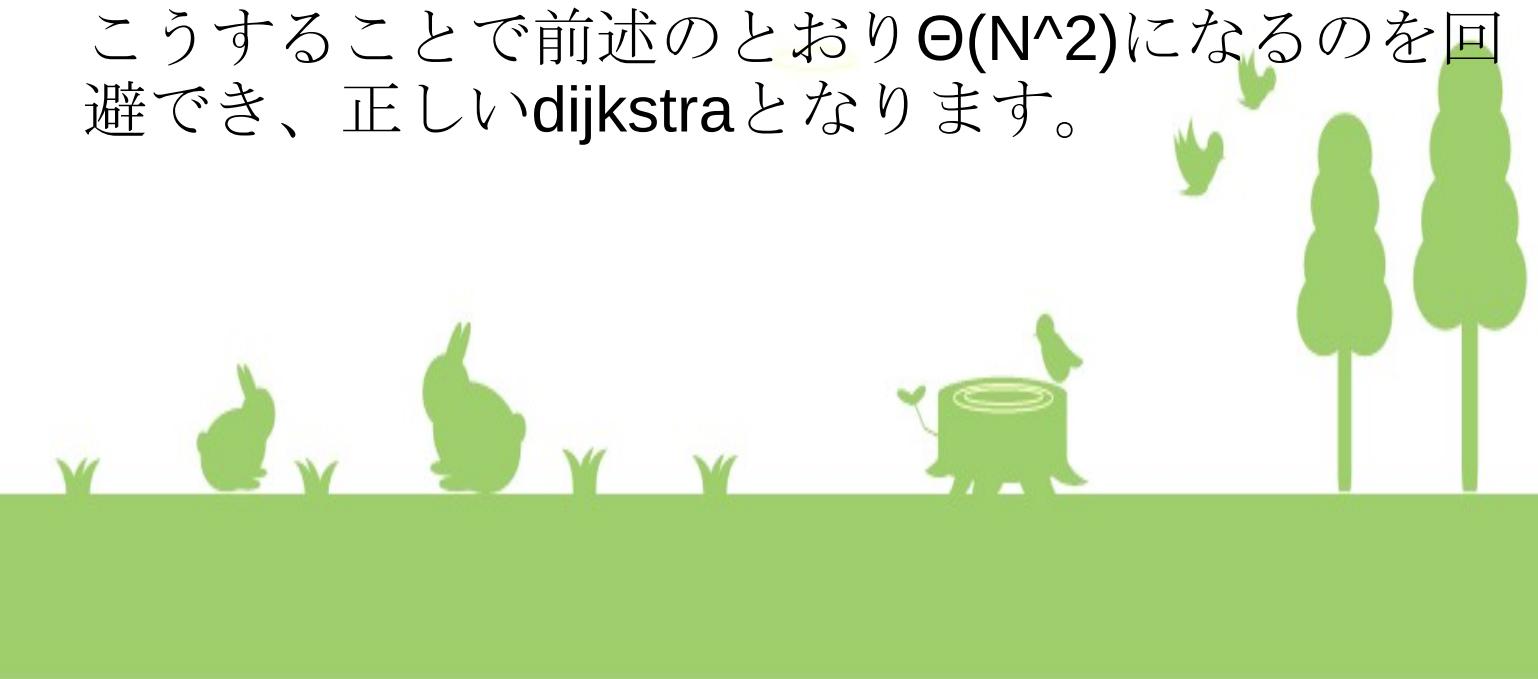


しかし正しいダイクストラは確かに $O((N + M) \log(N))$ です。何が悪かったのでしょうか？

少なくともC++の**std::priority_queue**では中に入っている要素の変更ができないので、先のようなグラフで一つの頂点が $\Theta(N)$ 回pushされるのは避けようがありません。 $\Theta(N)$ 回pushされる頂点があったとしても、pushがおきるのは辺一つにつき一回のみなので $O(M)$ となっていてこれは問題ではなさそうです。

問題は、popした後その頂点につながっている辺を全部調べる部分の計算量が抑えられていないことです。解決する方法として例えば一つの頂点につき一回までしか隣接辺を調べないでよければ調べる回数が合計で $O(M)$ 回になるというものがあります。一つの頂点から $\Theta(N)$ 回popされるからといってその毎回隣接辺を調べる必要はありません。一番最初にpopされた時点でその頂点の距離は確定していてそれ以上減らないのでそのタイミングで一回隣接辺を見れば後は見なくてもよいのです。

こうすることで前述のとおり $\Theta(N^2)$ になるのを回避でき、正しいdijkstraとなります。



余談としてscipyというpythonのライブラリには
scipy.sparse.csgraph.dijkstraという名前でdijkstra
法の実装があります。ドキュメントに計算量保証
は書かれていないので「間違った」というのはい
い過ぎですが、この記事が書かれている現在同じ
問題を抱えています。

以上です。



マックフライポテトと植物園

部誌書きたくない。マジで書きたくない。でも提出締め切り1時間前に過ぎたし、そろそろ書かないと担当の松○先輩に殺されそうだから書こうと思う。

実は「意味は知らないけどかっこいいパソコン用語まとめ」というテーマで意味は知らないけどかっこいいパソコン用語を集めていたりしたが、杉○先輩が「フローティングポイントエクセプション」とか言い出したあたりから普通にキモくなってきてモチベが消えてしまった。

といえばさっきから松だの杉だの名に針葉樹を冠する先輩しか登場していないが、他に○藤先輩とか竹○先輩とかいたな。なんだ？ここは植物園か？

なんだが便意を催してきたのでトイレに行ってきます。

行ってきました。お腹下してました。

マクドナルドのメニューは本当にどれも下剤でも入っているのか。特にマックフライポテトなんか、食べたらその日の夜5000兆%下痢になる。半分食中毒じゃないか。まあ人工添加物モリモリポテト美味しいから食べちゃうんだけどね。

思えば、かっこいいパソコン用語というのは人工添加物モリモリポテトのようなものだ。確かにその文面は格好良いものだが、それは決して自然由来のものではなく、単に安っぽい横文字を並べ立てただけにすぎない。

では自然由来のかっこいい言葉とはなんだろう。それを明らかにするため、我々はブラジルに飛んだ。

(しばらくの間ポルトガル語でお送りします。)

Aparentemente, há um perceptron multicamadas. Este é um termo de IA e parece que o número de camadas de neurônios de inteligência artificial é três ou mais. Como esperado, valeu a pena voar para o outro lado da terra no Brasil.

日本に帰ると、そこは自然の国だった。松、杉、そして藤に竹。四季折々の木々が私の心を潤した。今挙げたの半分常緑樹だけど。というか竹はそもそも木じゃないけど。

(そろそろ字数足りてきたので急速にオチに向かいます)

マックフライポテトは確かにうまい。しかし同時に毒でもある。真の幸せとは人工物からは得られないものなのだ。我々はこの植物園を受け継いでいかなければならないと思った。次期会長の下の名前、大樹だし。

※フィクションを含みます。

※というかほとんどフィクションです。

※ポルトガル語はgoogle翻訳なので正確ではないです。

※先輩にはちゃんと土下座しました。

Discordを使うだけの話

LINE を使う事に満足してますか... ?

こんにちは。第73回麻布学園文化祭の弊展示 **AZABU GAME CENTER** の展示責任者を務めました¹高2竹村 (Twitter: @Takeno_hito) です。新型コロナウイルスで色々潰れてしまった中、休校期間中に文化祭に関してでやってきた事をまとめようとしたのですが、あまりにも内容が増えすぎたので部誌にするのを諦めて、代わりにコミュニケーションツールの話をしたいと思います。

さて、コミュニケーションツールといえば一番最初に思い浮かぶのは LINE だと思います。東日本大震災がきっかけで開発された LINE は、今や知らない人はいません。国内でも利用者数²は8400万人以上で、大抵の人がスマホに入ってるアプリの1つでもあります。しかし、**LINE は決して使いやすい物ではありません。**

特に文化祭実行委員会みたいな数百人という大きい団体では、LINEでコミュニケーションを取るのは本当に大変です。そうでなくとも、部活のLINEグループを作るときも、大抵「全体連絡用」のグループと「各学年のグループ」、部活によっては「雑談用」「執行部」のグループもあって、個人との連絡も取らないといけなくて大変...というような人は少なくないと思います。他にも、以下のような問題があって、決して最善のアプリではないと言えます。

- いつもと違うPCでログインすると、前のPCからログアウトされてしまう。
- 携帯電話番号がないとLINEのアカウントが作れない
- 後輩だったり自治機関の役員だったりを区別しにくく、区別したければいちいち全員にニックネームを設定しないといけない。³
- 特に低学年だと、誰でもグループのメンバーを削除できるのでよく無秩序になる。
- グループのアイコンを設定しないと本当にグループの区別ができない
- 「本当に大事なお知らせ」と「普通のお知らせ」と「ただの質問」が全て同じグループで行われるので、大事なお知らせを見逃しやすい。これを回避するにはグループを大量に生成するしかない
- 誰でも好きな人をグループに招待できるので、「誰がどのグループに入ってるか」なんて分からぬし、たまに情報漏えいがおきる。

こんなに悪い点を列挙して LINE のネガキャンをしてる訳ですが、かくいう私もLINEを使ってます。だってみんなLINE使ってるし...。



こんな感じで幾つものグループに入らないといけない人も多いのでは？

¹ 他にも今年は文化祭実行委員会に入って、ウェブサイトの設計・運営をしていました: afes.info

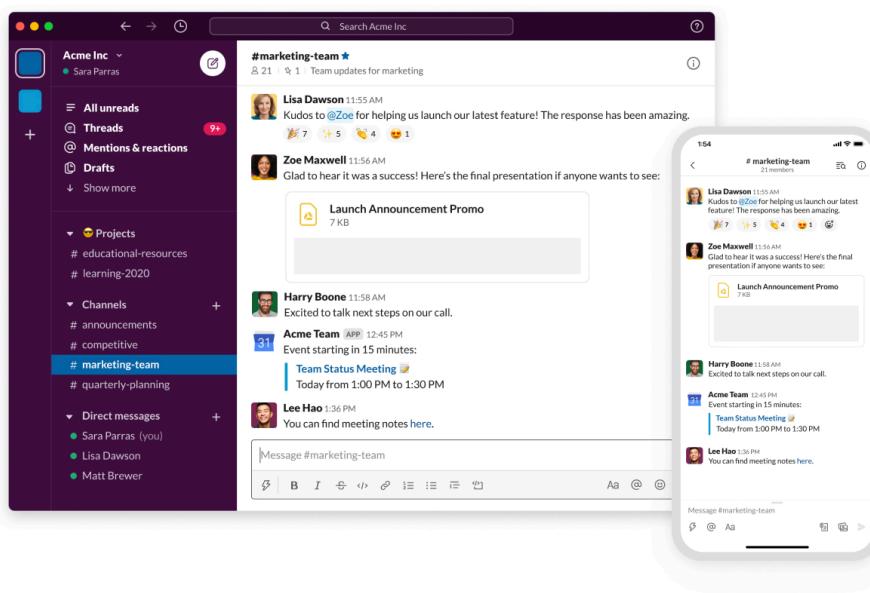
² 月間アクティブユーザー 2020年8月期 LINE Business Guide より

³ これをサボると、よく“LINEの個チャで同輩に「明日の試験の科目なんだっけー?」って聞いたつもりだったんだけど、実は先輩だった”みたいな事がよく発生します。

でも逆に言えば、実は「みんなが使っている」以外にLINEを使う理由はそんなにない…という事もあるわけです。

Slack も決して良くはない

では、LINE 以外のコミュニケーションツールってどんな物があるでしょうか。3年前、LINE が不便なものだと気づいてしまった部活の先輩たちは、部活での新たな連絡ツールを求めて旅に出ました。そして採用されたのは **Slack** というツールです。最近車内広告や CM も流れているので、知っている人も結構いると思います。



画像はSlack公式サイトよりお借りしています

当時 LINE 以外のツールを探していた先輩達にとって、Slack の長所はたくさんありました。例えば…

- 1つのグループ(Slack では“ワークスペース”)で複数のチャンネルが作れる
- グループ内の個人チャット(DM)が存在する
- LINE でいう Bot みたいなのが使いやすくて、例えば GitHub やメールの通知を Slack で確認する事ができる
- PCですごく使いやすいし1つのアカウントで複数の端末使える
- メンションが存在して、「ただの会話」と「関係のあるお知らせ・通知」を差別化できる
- グループごとに自分の名前を設定するので、誰が誰だかわかりやすい
- メッセージにリアクションが付けられる(→)

こんな感じで、(LINE と比べて) すごく便利なツールです。3年間 APCC では Slack で連絡をしてました。しかし Slack は Slack でデメリットがあります。例えば。

17:40 たけむら あーそうだ
フロンティア用も用意しないといけないらしいです
18:10 いやどうす
哭泣顔リアクション

泣き顔リアクション

- グループ内のメッセージが10000件を超えると、古いものからメッセージが削除され、検索もできなくなる
- グループ通話ができない
- デフォルトのままだと通知が飛ばない事がある
- LINE じゃないのと、APCC 以外でほぼ使わないので敷居が高い

本当は、課金すれば上2つの問題は解決できるのですが、1人あたり800円/月 払わないといけないので、部員に負担がかかって部活の敷居が高くなります。だからと言ってLINEに戻ろうとは思わなかったですが、今年の新入部員を少しでも勧誘しやすくする為に、新たなツールを使う事にしました。

Discord を使う

今年の4月から **Discord** をAPCCの連絡ツールとして使用しています。

Discord は元々はゲーマー用のグループチャットサービスだったのですが、今はそんな事はなくてゲーム以外の用途でも十分に使える物になっています。

Discord のメリットはというと…

- ロールという仕組みでグループ内のメンバーにラベルを付けられる。さらには持ってるロールに合わせてメンバーの色も変えられる
- テキストチャンネルだけでなく、**VC(通話)チャンネル**も作れる
- Slackと比較すると敷居がすごい低い
- 招待リンクを作れて、リンクを送るだけで招待できる

Slackの方が優れてる点は

- Slackの方が動作が軽い(個人差あり)
- GitHubの通知に関しては優れてる

くらい…。また、”このロールがついている人しか見えないチャンネル” “このロールを持ってる人だけ他の人をサーバーに招待できる”とかかなり細かく設定ができるので、ゲームだけでなく生徒会のような機関、例えば麻布なら**文化祭実行委員会**(1学年だけで80人とか居る、もの凄く大きい機関です)とか、**サークル連合**(40の部活の代表者からなる、こっちも大きい機関)での連絡ツールにも向いてると思います。

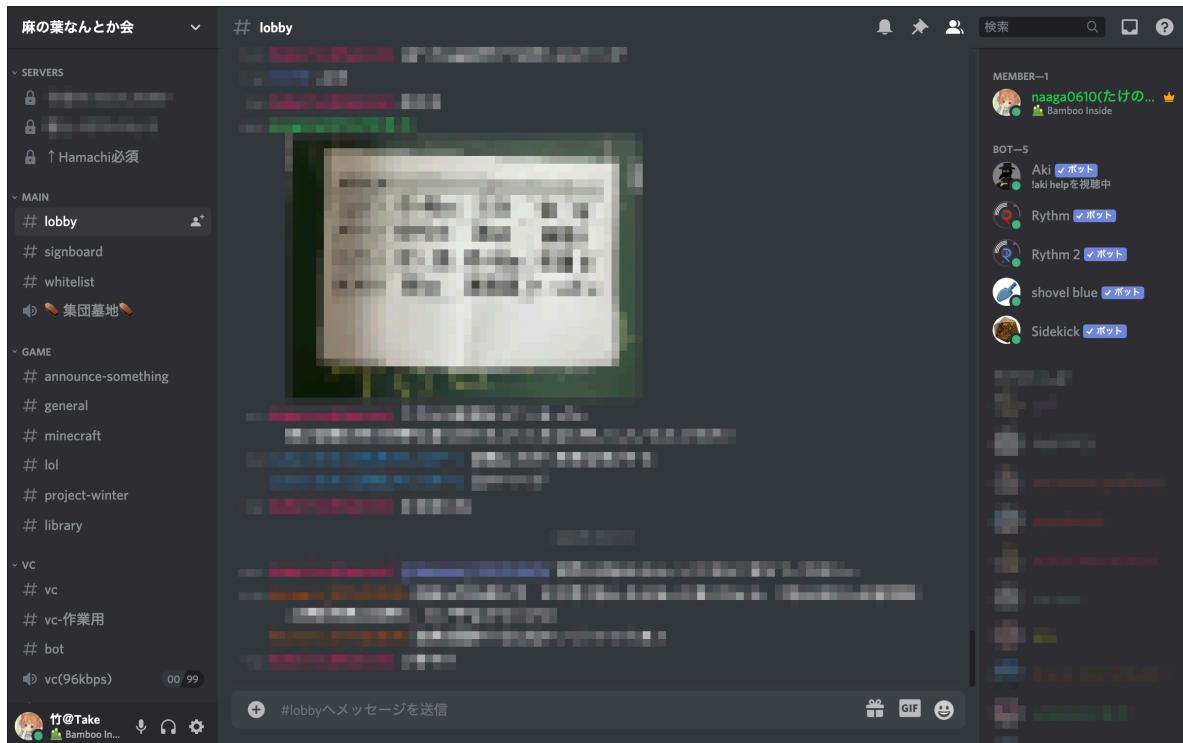
文実を例にすると、現状だとLINEには僕が参加してるだけで「高2全体」「実務調整会議(班長・局長級会議)」「一般展示責任者」「飲食展責任者」「ステージ責任者」「フロンティア責任者」「会計責任者」「総務局」「ウェブサイト班」「広報担当」と**11グループ**あって、管理がすごく大変です。僕の場合は個チャで色んな人と連絡を取るので、”この展示の責任者のLINEどれだけ？”みたいな事を何回もやるハメになります。

Discordなら、チャンネルを沢山作って、一般展示や一般局員の人は見えるチャンネルを限定して、あとはお知らせのチャンネルを作る事で、1つのグループで全部まとめて管理できる訳です。

いろんな Discord サーバー

最後に、約4年 Discord を使って来た私が、実際のサーバーのチャンネルリストを見せながら、実際に様々な用途で使われてるんですよ、っていう事を紹介して終わりたいと思います。

身内サーバー



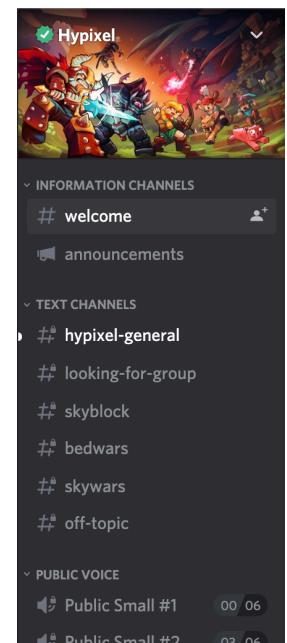
麻布生20人のサーバーです

どんな物なのか具体的にどんなのか想像つかないと思ったので、Discordのソフトの画面を持ってきました。

麻布生の身内同士でのゲームサーバーで、ゲームの種類ごとにチャンネルが分かれてたり、一番下にはVCチャンネルがあってそこで通話できたりします。モザイクを掛けてますが、発言者に色がついてるのがわかると思います。

ゲームコミュニティ

1万人以上が参加してるこのサーバーでは、通話しながら一緒にゲームしたりする事ができます。英語ができるといけないけど…



電技の入口



連絡など

全体連絡

自己紹介

id置き場

雑談とか

なんでも質問

雜談-電気技術室

MISC

competitive_pg

chokudai_005

AFES

部活

こんな感じで、ゲームだけでなく部活のチャンネルとしても使われてたりします。画像は我らがパー研のサーバーです。

下のほうに、普通の人には発言できない・みたいな設定もできます。

更に、他の人に見えない高2の一部しか見えない運営用のチャンネルも作ってます。

20:18 竹村 H2 展責 そういうや、明日から日曜除いて文化祭まで毎日部活あるよ (例会で話してたらごめん (編集済))
E 1 神 1 00 1

竹村 H2 展責 講座は多分やらないから、新入部員は次は土曜日でいいと思います。制作する方々は、部活に出るかどうかはおまかせです。高2は誰かは毎日いるはずです

このチャンネルでメッセージを送信する権限がありません。

その他

11:32 麺ヘラちゃん 小麦と肉 桃の木
Googleマップ : <https://goo.gl/maps/MtneuQneNQPReF9d9>
時間 : 月,火,木～土 : 11:00-16:00, 定休日 : 水日
新宿御苑駅近くにあるラーメン屋、新宿駅からだと20分ほど歩く距離にある。
その名前に負けず、ツルツルとした喉越しに嗜めばモッチモチ、小麦の旨み溢れる平打麺と豚の肉汁の旨味と背脂の甘みがこれでもかというほど溢れ出すつけ汁が魅力的な一杯。 (編集済)



Momo no Ki
Momo no Ki
★★★★☆ · Ramen restaurant · 1 Chome-32-4 Shinjuku



“麺ヘラどっとろぐ”サーバーです。

(人様のサーバーの画像を貼るのは本当はよくないけど、Google検索で調べたら一発で出てくるので許して！！)

ゲームも会議も全然関係ないけど、wikiみたいな感じで管理人が今まで食べてきた拉麺の中で美味しい物を大量に紹介してるサーバーです。

このあたりで紹介を終わりにしようと思います。Discordを使うと、LINEが如何に使いづらいものなのかを実感すると思います。「みんなが使ってるからLINEを使う」ではなくて、便利な新しいものを手にして、そして使ってほしいです。

全国チェーン、小規模チェーン店
日本全国
関東中心
関西中心
その他

東京都23区
新宿淡麗
新宿濃厚
高田馬場・早稲田
他新宿区
渋谷
他渋谷区
池袋
他豊島区
中野
他中野区
荻窪
他杉並区
銀座
東京駅周辺
神田・神保町
秋葉原

魔法少女になりたい！

良いよな、魔法少女って

私、都内の学校に通ういたって普通の高校一年生！別に秋葉原でアイドルやメイドをやっているわけでもないし、小動物とヤバめな契約をして魔法少女になったわけでもない、ただのJKよ。

そんな私だけど、やっぱり魔法少女なんてものには憧れちゃうわね。魔法が使えればほうきで空を飛び放題だし、魔法の力で自分を天才美少女にすることだって出来ちゃう。そんな訳で気が付いたら、その、何というか、「オタク」のようなものになっていたわ。部屋中魔法少女のフィギュアやポスターばっかりだし、コスプレからステッキから可愛いほうきまで、全部揃えちゃった！

ところで、なろう小説って知ってるかしら？あの誰でも小説を投稿できるやつね。やっぱりメルヘンなものに憧れているとああいうものにも興味が出てくるわね。それにしてもなろう小説はいつから「トラックに轢かれて異世界転生」なんて相場が決まってるのかしらね。陳腐そのものだけどちょっとワクワクしちゃう、転生、か...。私ももし転生したら魔法少女とかに...？ふふっ、冗談冗談.....

あれ...？

彼女は目覚めると、見知らぬ部屋の中にいた。

「ここはどこかしら？確か私...

嘘、もしかして...いや、そんなことあり得ないわ。大体幽霊だか転生だかなんて、おとぎ話に過ぎないんだから。意識も感覚もあるってことは、変な夢でも見ているに違いないんだから。

けど私、確かにトラックに轢かれたんだわ。痛みはなぜか感じないけど、感覚は確かに残ってる。腕や足にひどい傷はついてないかしら...」

そう呟いて、彼女は自身の手足に視線を下ろした。

「何よ、これ！？！？！？！？！？」

私の手足じゃない？？おかしいわ、転生なんてただのおとぎ話なのに。しかも私の望んだ魔法少女と全然違うじゃない！第一、私の外見、男だわ！男は生まれた瞬間から魔法少女の権利を剥奪されているのよ！あんまりじゃない…ぐすっ…」

しかし、彼女はここが何処か、それだけではなく自分が誰かすら分からぬこの状況で、めそめそしている場合ではなかった。彼女は辺りを見渡し、自身の置かれている状況を出来るだけ冷静に見つめようとした。

「どうやら私は出口の無い一室にいるっぽいわ。…ん？出口が…無い？ 大変！私、閉じ込められているのよ！密室、密室、ここから脱出…

密室といえばアレしかないじゃない、言うまでも無いわ！ …そう、“あんなことやそんなことをしないと出られない部屋” よ！もう、たまたもんじゃないわ！何かのドッキリにしてもあまりにタチが悪いわね…」

しかし、あんなことやそんなことをする“対象”はいつまでも現れなかつた。彼女は「そんな部屋、同人誌の中のお話に過ぎないわよね…」と苦笑した。そして、部屋の真ん中に1台の機械を見つけた。

「それなら出る方法を自分で探さなきゃ！ん、あそこのデカい真っ黒はパソコンかしら？持ち主には申し訳ないけど勝手に開いちゃおう！ …って、このパソコンったらパスワードロックされてないじゃない！今回は勝手に開けて好都合だから感謝だけど、どういうセキュリティ意識なのかしら、全く！勝手にツイートでもしちゃうわよ。」

彼女は誰のものかすら分からぬパソコンを弄り回していたが、その持ち主は彼女自身の外見である_言い換えれば彼女が“乗り移った”_その男だろうと確信していた。

「今は私自身がこの男の本体なんだから、このパソコンをどうしようつたって私の勝手よね？

あら、このデスクトップの背景、私の大好きな“魔法少女☆ぱー研”の娘じゃない！まさかコイツも同じ趣味を持っているなんてね…けど、別に良いわ。だって…」

そう言うと、彼女は少しだけ息を吸い込んだ。

「愛する気持ちがあれば、老若男女誰もが平等なんだみゅん！

…なんてね。」

アニメキャラが決め台詞を決めた時のごとく、彼女は愉悦げだった。

彼女が再びパソコンを弄っていると、画面にメッセージが表示された。それはメールの通知だった。他人の人間関係まで詮索してしまうのは少々気が引けるが、彼女はこの怪奇な状況をまた、少し楽しみ始めていた。

彼女は、視界に入るメッセージを読んだ。

『from:先輩

進捗どうですか？

締切は明日です。“魔法少女☆アドベンチャー”の制作、頑張ってください。』

「ふふっ、はははっ！ “進捗どうですか？” だって、笑っちゃうわ！ けど何がどうあれ、今は私がコイツなんだから、締切に間に合わせなきや…けど、一体何をすればいいって言うの？突然締切なんて言われたって、私エンジニアをやっている訳でもないのよ！？」

彼女は少し狼狽したが、目的の物はデスクトップですぐに見つけることが出来た。とりあえず彼女は、このゲームを遊んでみることにした。

「可愛いタイトル画面！結構作りこんであるのね…流石だわ。もしかすると世の中に大きく羽ばたくかもしれないゲームの製作段階を遊べるなんて、新鮮な気分ね。それにしても、これ本当に製作段階なの？ちゃんと出来上がってるじゃない。」

しかし彼女は1箇所だけ、そのゲームの白紙部分を発見した。彼女の任務は今、この部分をプログラムすることであった。幸いにも、男がパソコンの傍らに置いていた制作メモのおかげで、何をするべきかは知ることが出来た。

とはいっても彼女は、プログラミングには殆ど何の知識も無かった。

「しょうがないわね、やるしかないわ。Google 拝借させてもらうわよ。

…って、何よこの検索履歴は…/// いや、見なかったことにしておくわね。

さてと、早速調査開始だわ！それにしても私って意外と手際良いのかも？ふふっ」

数時間して、彼女は驚くべき要領でプログラミングをあらかじめ理解していた。溢れ出る関心と興味で彼女の心は止まらなくなり、彼女の任務であるゲームの製作についても自信ありげだった。

「私ってもしかして、いやもしかしなくても天才美少女に違いないわ！」

それにしてもプログラミングってすごいわね、まるで魔法みたい！何百倍何千倍も賢くて速いこのコンピューターを私が自分の手で操ることが出来るなんて夢みたいだし、それに、if 文は並行世界ハラレルワールドを生み出すことが出来るし、while 文はこの世界を何度も反復ループさせることが出来るんだもの！見た目はこんなんになってしまって、可愛いドレスもステッキもほうきも小動物も無いけど、まさに魔法少女になった気分だわ！」

彼女のプログラミング言語に対する理解の仕方はオタクとしか言えなかつたが、理解の速度は天才美少女といって過言ではなかつた。彼女はとてもわくわくしていたが、一方で迫りくる「締め切り」のことを忘れたわけではなかつた。

「けど、楽しんでいる暇は無いわ。私は締め切りに間に合わせなきやいけないし、コイツだって今まで睡眠を削って頑張って来たに違いないわ。だってこの部屋のゴミ箱、モンスター エナジーの空き缶でいっぱいなもの！」

だけど、こういうのもやっぱり魔法少女としての 1 面なのかもしれない。魔法のような現代文明だけど、それを楽しんでいるだけじゃいけない。それを使って、そうね、“進捗”を生んで行かなきやいけないんだから。魔法少女たちだってそう。可愛い衣装を着て、町のみんなに愛されているけれど、影では命を懸けて強い魔物たちと闘ってる。可愛くて楽しげな顔と、かっこよくて強い顔。そんな 2 つの顔を持っている魔法少女たちに、私は憧れていたんだわ…」

彼女は自分の「魔法少女」への想いを再認識した。そして冷蔵庫からモンスター エナジーを 1 本取り出し、推しに思いを馳せながら、最後の作業に取り掛かった。

「終わった！！！」

彼女は達成感で満ち溢れていた。そして最後のテストプレイをした。自分が手を加えたこのゲームは、やっぱり愛着があった。

「コイツよりクオリティが劣ってしまったのはもちろん承知してる。けど、頑張った自分にいっぱいお疲れ様！やっぱり眩しさを感じるわね。あれ、眩しい…画面が、いや、視界が…？」

「ここは...どこ？」

彼女は目を覚ました。一番見覚えがある場所。フィギュアやポスター、コスプレに囲まれているピンク色のベッド。

「あれ...私...？」

戻ったんだ！女の子に！おうちに！普通の JK に！」

彼女は夢の中の出来事を鮮明に覚えていた。そしてフィギュアに近づいて、みんなの頭を撫でた。

すると彼女は一瞬だけ確かにフィギュアが喋ったように感じた。彼女はその言葉を聞き逃さなかった。聞き逃すはずがなかった。

「進捗どうですか？」

APCC 部誌 「魔法少女になりたい！」 終

あとがき

ねこみみを付けると可愛さが倍増するらしいです。にゃ～ん

魔法少女になるには最小費用流を履修しないといけないっぽいので文化祭が終わったらやるつもりです。