第∞版

LOCAL Students

やばい同人誌執筆合宿

LOCAL Students
LOCAL学生部

LOCAL学生部 著
訳

知らねー

うっひょい　　ちくうぇいと　　あわあわ　　けんつ　　さわだ

Jumpaku

2018 年 1 月 6 日

はじめに

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam

vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

目次

LaTeX の乱数生成アルゴリズムを改善する

第 1 章 うっひょい

1.1 乱数生成コマンド

LaTeX のパッケージに固定小数演算ができる FP パッケージがあります.

1.1.1 固定小数演算

TeX,LaTeX は共に整数の演算は可能ですが, 小数点を含む計算は行うことができません (寸法を除く). 固定小数点の演算をすることを目的としてあるのが FP パッケージです.

1.1.2 乱数を出力する FPrandom コマンド

fp パッケージの中には上記以外にも 0 ～ 1 の範囲の疑似乱数を出力する FPrandom というコマンドがあります. FPseed でシード値を決めてから FPrandom を使って変数に乱数を格納します.

```
1 \FPseed = 156
2 \FPrandom{\result}
3 \FPprint{\result}
```

1.2 FPrandom の乱数生成アルゴリズムを調べる

1.2.1 目的

今回は FPrandom に使われている疑似乱数がいわゆる多くの問題点がある昔のアルゴリズムかどうかを調べる為に行います.

1.2.2　疑似乱数アルゴリズムの問題点

現在，世に出回っている疑似乱数アルゴリズムは様々あります．これは今まで研究されてきたアルゴリズムに何かしらの問題があるからです．偏りが出たり，パターンがあったり様々です．また，疑似乱数には周期があります．一定の数の乱数を出力したらまた，最初から同じ疑似乱数列を出力し始めます．現在は計算機のスペックも高くなりより複雑なシミュレーションを行えるようになりました．その為，用いられる疑似乱数の数が従来のアルゴリズムでは周期を上回る危険性もありました．周期を伸ばすことも新しい疑似乱数アルゴリズムを開発する理由の一つです．

1.2.3　ソースを読む

それでは fp パッケージのソースを読んでいきます．しかし，実際には fp パッケージ本体が内部的に呼び出している fp-random パッケージのソースを読んでいきます．

コメントに正解が書いてあった

22 行目から FPrandom の定義が始まります．その直後，コメントが長く続いています．コメントには次のようなことが書かれていました．Algorithm reproduce from a very old Fortran program (unknown origin!) どうやら大昔の Fortran の疑似乱数アルゴリズムを TEX に起こしたものが FPrandom の正体みたいですね．これはいろいろ問題点がありそうですね．更にその下のコメントには Fortran で書かれた疑似乱数アルゴリズムらしきものがあります．これを読んでいけばどんなアルゴリズムが使われているか分かりますがそれではつまらないのでコメントを抜けた後の TEX で書かれた疑似乱数アルゴリズムの方を見ていきましょう．

TEX のマクロで実装された疑似乱数アルゴリズム

```
1 \ifnum\FPseed=0%
2     \FPseed=123456789%
3     \FP@debug{random: seed value undefined! We will used
      ↪  \the\FPseed.%
4       Define it if you want to generate a different sequence of
        ↪  random numbers.}%
5 \else%
6     \FP@debug{random: seed value used: \the\FPseed}%
```

```
\fi%
```

これはシードを指定してるかどうかを判定してしてない場合 123456789 をシードとするというマクロですね．次見ていきます．

```
\FP@xia=\FPseed%
\divide\FP@xia by 127773%
\FP@xib=\FP@xia%
\multiply\FP@xib by 127773%
\advance\FP@xib by -\FPseed%
\FP@xib=-\FP@xib%
\multiply\FP@xia by 2836%
\FPseed=\FP@xib%
\multiply\FPseed by 16807%
\advance\FPseed by -\FP@xia%
```

このアルゴリズムのコアとなる乱数の計算ですね．TeX で書いてる為ごちゃごちゃしていますが数式で表すと以下のようになります．

$$s_{n+1} = \frac{S_n}{127773} \times 2836 + 16807 S_n \bmod 127773$$

s_n が n 番目に出力した乱数で s_{n+1} は $n+1$ 番目に出力する乱数です．つまり漸化式になっています．線形合同法の漸化式と似ていますね．線形合同法は C 言語の rand 関数に採用されているアルゴリズムですが多くの問題点が発見されていて現在は非推奨となっています．線形合同法では $16807 S_n$ の部分が定数ですが，それが乱数を含む形になっています．線形合同法の式は以下のとおりです．

$$S_{n+1} = A \times S_n + B \bmod M \quad \text{(A,B,M は定数)}$$

... 構造が非常に似ていますね．

```
 \ifnum\FPseed>0%
  \else%
      \advance\FPseed by 2147483647%
  \fi%
  \FPdiv\FP@tmpa{\the\FPseed}{2147483647}%
```

```
6   \global\let\FP@tmp\FP@tmpa%
7   \global\FPseed=\FPseed%
```

乱数の正規化を行っています．214793647 がこの乱数列における最大値 M でこの時点で $n+1$ 番目の乱数が負の値の場合は M を乱数に足します．その後 FPdiv と呼ばれる小数の割り算ができるマクロによって乱数を M で割り，乱数を $0 \sim 1$ の範囲の値に正規化しています．

ほげ

第 2 章 ちくうぇいと

ほげ

第 3 章 あわあわ

ほげ

第 4 章 けんつ

自作エディタ入門編

第 5 章 さわだ

5.1 はじめに

皆さんはちょっとしたメモやプログラミングにどんなエディタを使いますか？ Vim？それとも Emacs？まぁ色々ありますよね．人それぞれ好き嫌いがあると思います．僕は Vim を自分好みに拡張するのが好きです[*1]．でも，人には自作欲求があります．CPU，OS，言語，更に最近はキーボードなどが人気ですが，エディタも中々面白いですよ．CUI は時代遅れなんてそんなの気にしちゃいけません．ソースコードは github.com/takuzoo3868/takdit に置いてあります．

5.2 準備

外部ライブラリに依存しない事を目標としていますが，C コンパイラと`make`コマンドは準備する必要があります．`cc --version`や`make -v`でインストールされているかどうか確認できます．自身の環境にコンパイラがインストールされていなかった場合は，Google 先生に聞いてみましょう．

make によるコンパイル

解説のために本文中では`takdit`と記載しますが，自作の際は好きな名前に置き換えて下さい．`cc takdit.c -o takdit`などと打ち込めばコンパイルできます．しかし試行錯誤を繰り返すため，再コンパイルの度に同じ事をするのはあまりスマートではありません．`make`を用いることでプログラムコンパイルを少しだけ楽にしておきましょう．`Makefile`を作成し，以下の内容を記述しておきます．

[*1] そこの Emacs 教，石を投げないで！

```
takdit: takdit.c
$(CC) -o takdit takdit.c -Wall -W -pedantic -std=c99
```

この辺については，準備段階なので詳細は省きます．これで準備は

5.3 基本構成

基本となる骨格は kilo というテキストエディタを参考にします[*2]．Salvatore Sanfilippo 氏によって開発された C 言語製のエディタです．BSD 2-clause にて公開されています．紹介文に，

> Kilo is a small text editor in less than 1K lines of code (counted with cloc).

とあるように 1000 行程度なので目で追っていくもの問題ないでしょう．ちょっと厳しいなという方は`int main()`だけでも目を通す事をお勧めします．

```
int main(int argc, char **argv) {
  // takdit <ファイル名> という引数のみ実行
  if (argc != 2) {
    fprintf(stderr, "Usage: takdit <filename>\n");
    exit(1);
  }

  initEditor();                          // エディタの初期化
  editorSelectSyntaxHighlight(argv[1]); // シンタックスハイライトの適用
  editorOpen(argv[1]);                   // ファイルを開く
  enableRawMode(STDIN_FILENO);           // Raw モードの有効化
  editorSetStatusMessage(
          "HELP: Ctrl-S = save | Ctrl-Q = quit | Ctrl-F = find");
  while (1) {
    editorRefreshScreen();                 // 変更の反映
    editorProcessKeypress(STDIN_FILENO);  // キー入力待ち
  }
  return 0;
}
```

[*2] github.com/antirez/kilo

処理の流れは，

非常にコンパクトな CUI ベースのテキストエディタであり学習には最適ですが，実際の使用には難が多くあります．改善点を列挙しますと，

5.4 Build your own Editor!!!

5.4.1 シンタックスハイライト対応

ほげ

第 6 章 Jumpaku

タイトル

著者：俺
挿絵：俺
発行：2018 年 1 月 6 日
印刷：俺

落丁，乱丁の際でもお取り替えしません．

本書は著作権上の保護を受けているのかよく分かりません．本書の一部あるいは全部について，LOCAL 学生部から文書の許諾を得るよりも github のリポジトリからコピペした方が早いと思いますよ．