# Penerapan Algoritma Fisher-Yates dan *Linear Congruential Generator* dalam *Shuffling Playlist* Musik

Dewantoro Triatmojo - 13522011[1]
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
[1]*13522011@std.stei.itb.ac.id*

***Abstract*—Algoritma Fisher-Yates dikenal sebagai salah satu metode efektif untuk mengacak urutan elemen-elemen dalam suatu himpunan dengan probabilitas yang merata. *Linear Congruential Generator* (LCG) merupakan suatu metode dalam teori bilangan yang sering digunakan untuk menghasilkan deret angka bersifat *pseudorandom*. *Shuffling playlist* musik merupakan sebuah fitur yang sering ditawarkan oleh berbagai aplikasi pemain musik untuk memainkan musik secara acak. Penerapan algoritma Fisher-Yates dan LCG dalam *shuffling playlist* musik dapat memberikan pengalaman mendengarkan musik yang lebih dinamis dan tidak membosankan bagi pendengar. Makalah ini akan membahas langkah-langkah implementasi Algoritma Fisher-Yates dan LCG dalam mengacak lagu dalam sebuah *playlist* musik.**

***Keywords*—Algoritma, Fisher-Yates, *Linear Congruential Generator*, Musik, *Playlist, Random, Shuffle*.**

## I. Pendahuluan

Dalam era digital ini, pengembangan teknologi semakin luas terutama dalam hal pemutaran musik. Dalam dunia informatika sendiri sudah ada banyak aplikasi pemutaran musik terkenal yang dikembangi seperti Spotify, Apple Music, Youtube Music, dan masih banyak lagi.

Salah satu fitur yang terdapat dalam aplikasi tersebut adalah *playlist* musik. Sebuah *playlist* musik merupakan kumpulan berbagai musik yang telah dipilih dan disusun secara tertentu oleh pengguna. *Playlist* musik memberikan pengalaman mendengarkan musik sesuai dengan preferensi pendengar. Namun, mendengarkan musik dengan urutan yang tetap berulang kali dapat membuat pendengar bosan. Oleh karena itu, dibutuhkan sebuah fitur *shuffle* atau acak agar urutan lagu yang dimainkan berubah.

Salah satu pendekatan yang dapat digunakan untuk menyelesaikan persoalan ini adalah dengan menggunakan Algoritma Fisher-Yates dan *Linear Congruential Generator* (LCG). Pada makalah ini, dibahas bagaimana implementasi Algoritma Fisher-Yates dan LCG dapat digunakan untuk menyelesaikan masalah *shuffling* lagu dalam sebuah *playlist* musik.

## II. Landasan Teori

### A. Relasi Rekurens

Relasi rekurens adalah sebuah persamaan yang menyatakan suku ke $n$ suatu barisan bilangan menggunakan gabungan satu atau lebih suku-suku sebelumnya [1]. Relasi rekurens memiliki dua bagian, yaitu [1]:

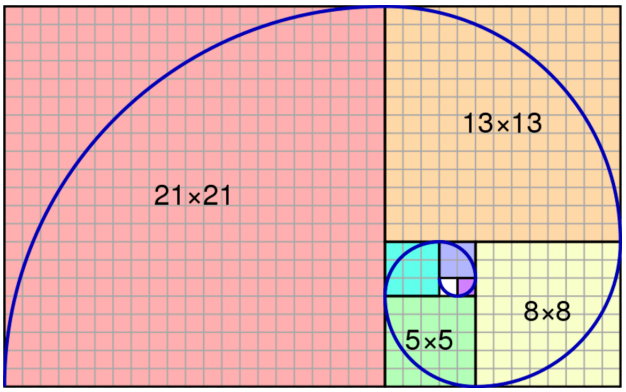
a. Basis, yaitu nilai awal dari relasi rekurens yang berguna untuk menghentikan persamaan rekurens.
b. Rekurens, yaitu bagian yang mendefinisikan dengan suku-suku sebelumnya.

Contoh sederhana relasi rekurens yang sudah sering dijumpai adalah barisan Fibonacci. Basis dari relasi rekurens barisan fibonacci adalah

$$F_0 = 0$$
$$F_1 = 1$$

dan persamaan rekurens (untuk $n > 1$) adalah

$$F_n = F_{n-1} + F_{n-2}$$




**Gambar 1.** Visualisasi Pola Barisan Fibonacci
Sumber: https://cdn.elearningindustry.com/wp-content/uploads/2017/09/dd7620fc0fceffcce27b5d11c5c01a96-768x476.png

### B. Linear Congruential Generator (LCG)

*Linear Congruential Generator* (LCG) adalah sebuah algoritma yang menghasilkan bilangan acak semu [2].

Algoritma ini merupakan salah satu algoritma penghasil bilangan acak semu paling tua dan umum diketahui. LCG didefinisikan menggunakan relasi rekurens [2]

$$X_{n+1} = (aX_n + c) \bmod m$$

Dimana $X$ adalah bilangan acak semu, dan

$m, 0 < m$ adalah modulus,
$a, 0 < a < m$ adalah pengali,
$c, 0 \leq c < m$ adalah peningkat,
$X_0, 0 \leq X_0 < m$ adalah *seed* atau nilai awal

Merupakan konstanta bilangan bulat yang mendefinisikan penghasil bilangan acak semu tersebut. Dari rumus di atas, dapat dilihat bahwa nilai yang dihasilkan oleh LCG dipengaruhi oleh konstanta $a, c, m$ dan juga nilai awal atau *seed* ($X_0$).

Algoritma ini sangat mudah untuk diimplementasikan. Namun, algoritma ini memiliki beberapa kekurangan. Pemilihan nilai konstanta dan nilai awal sangat mempengaruhi hasil barisan baik dari segi periode maupun keacakan [3]. Contohnya andaikan $a = 1$ dan $m = 1$, maka akan dihasilkan $X_{n+1} = (X_n + 1) \bmod m$ yang jelas hasilnya tidak *random* sehingga ini buruk untuk pembangkit bilangan acak. Artinya, untuk menghasilkan barisan yang *random* dengan LCG, harus ditentukan nilai konstanta dan *seed* yang tepat. Umumnya, nilai $a$ dan $m$ yang dipakai sangat besar. Sebagai contoh, glibc (dipakai oleh GCC) nilai $a = 1103515245$, $c = 12345$, dan $m = 2^{31}$ [4].

### *C. Algoritma Fisher-Yates*

Algoritma Fisher-Yates adalah sebuah algoritma yang digunakan untuk mengacak sebuah barisan yang hingga secara adil. Algoritma ini pertama kali dideskripsikan pada tahun 1938 oleh Ronald Fisher dan Frank Yates [5]. Algoritma ini sering juga disebut sebagai Knuth *Shuffle* karena dipopulerkan oleh Donald Knuth pada bukunya yang berjudul *"The Art of Computer Programming"* [6].

Algoritma Fisher-Yates menghasilkan hasil acakan yang tidak bias karena semua permutasi memiliki probabilitas yang sama. Untuk sebuah barisan dengan panjang $n$, algoritma ini bekerja dengan mengiterasikan elemen-elemennya dari elemen ke $n - 1$ sampai elemen ke 1 [7]. Untuk iterasi elemen ke $i$, tentukan sebuah nilai $j$ acak yang berada di rentang $0 \leq j \leq i$ [7]. Setelah itu tukarkan posisi elemen ke $i$ dan elemen ke $j$ [7]. Karena algoritma ini hanya melakukan *looping* satu kali pada $n - 1$ elemen, maka algoritma ini memiliki kompleksitas waktu sebesar $O(n)$ [8].

| Range | Roll | Scratch | Result |
|---|---|---|---|
| | | 12345678 | |
| **1-8** | 4 | 123**8**567 | 4 |
| **1-7** | 3 | 12**7**856 | 34 |
| **1-6** | 1 | **6**2785 | 134 |
| **1-5** | 2 | 6**5**78 | 2134 |
| **1-4** | 4 | 657 | 82134 |
| **1-3** | 1 | 75 | 682134 |
| **1-2** | 1 | **5** | 7682134 |
| | | | 57682134 |

**Gambar 2.** Contoh Visualisasi Fisher-Yates
Sumber: https://www.researchgate.net/publication/327597003/figure/tbl1/AS:670022447951885@1536757208251/Fisher-Yates-Shuffle-Algorhytm.png

Dalam implementasi Algoritma Fisher-Yates harus berhati-hati karena sifat tidak bias dari algoritma ini bisa saja hilang. Hal ini terutama terjadi saat penentuan bilangan acak pada rentang $0 \leq j \leq i$ menggunakan LCG [9]. Untuk meminimalisir bias dalam mengacak barisan, harus dipilih konstanta dan nilai awal pada persamaan LCG yang menghasilkan *state* yang sangat besar dibandingkan jumlah permutasi barisan.

## III. IMPLEMENTASI

Program dibuat menggunakan bahasa *Go* menggunakan modul `fmt` untuk melakukan *input/output* dan `math` untuk melakukan perhitungan matematika.

```
package main

import (
    "fmt"
    "math"
)
```

**Gambar 3.** Kode Impor Modul Program Utama
Sumber: Dokumentasi Pribadi

### *A. Linear Congruential Generator (LCG)*

Implementasi *Linear Congruential Generator* (LCG) untuk pembangkit bilangan acak semu akan menggunakan nilai konstanta yang digunakan oleh glibc yaitu $a = 1103515245$, $c = 12345$, dan $m = 2^{31}$ [4]. Nilai *seed* yang dipakai adalah $X_0 = 312309232$.

```go
// Konstanta untuk LCG
// Gunakan data LCG glibc
const LCG_a = 1103515245
const LCG_c = 12345
const LCG_m = 1 << 31

// Menyimpan state dari nilai x
// Inisialisasi seed
var LCG_x = 312309232

// Algoritma LCG
// I.S. LCG_x adalah nilai sebelumnya
// F.S. LCG_x adalah nilai selanjutnya
func LCG(a, c, m int) {
    // Hitung rumus LCG, perbarui nilai x
    LCG_x = (a*LCG_x + c) % m
}
```

**Gambar 4.** Kode Program Fungsi *Linear Congruential Generator* (LCG)
Sumber: Dokumentasi Pribadi

Untuk memastikan fungsi LCG bekerja dengan baik, dilakukan uji coba keluaran hasil 15 LCG pertama.

```go
func main() {
    // 15 LCG Pertama
    fmt.Println("15 LCG Pertama:")
    for i := 0; i < 15; i++ {
        LCG(LCG_a, LCG_c, LCG_m)
        fmt.Println(LCG_x)
    }
}
```

**Gambar 5.** Kode Program Utama Uji Fungsi *Linear Congruential Generator* (LCG)
Sumber: Dokumentasi Pribadi

```
15 LCG Pertama:
245008233
826529262
153406095
335282460
1112224037
177955578
1672315051
1029051144
396540577
1588376774
96566407
2126337972
814702557
506419026
573528611
```

**Gambar 6.** 15 Hasil Pertama Uji Fungsi *Linear Congrunetial Generator* (LCG)
Sumber: Dokumentasi Pribadi

### *B. Normalisasi Hasil Linear Congruential Generator (LCG)*

Dari hasil gambar 6, nilai yang dihasilkan fungsi LCG masih sangat besar, namun dibutuhkan nilai acak pada rentang tertentu. Maka dari itu, akan dibuat fungsi normalisasi dari hasil LCG yang menghasilkan nilai acak pada rentang [min, max].

```go
// Hasilkan nilai acak pada rentang [min, max] dengan LCG
// I.S. LCG_x adalah nilai sebelumnya
// F.S. LCG_x adalah nilai selanjutnya
// dan mengembalikan nilai yang sudah dinormalisasi
func normalizedLCG(a, c, m, min, max int) int {
    // Perbarui nilai LCG
    LCG(a, c, m)

    // Dapatkan nilai hasil LCG yang sudah dinormalisasi
    floatRange := float64(max - min + 1)
    floatLCG := float64(LCG_x)
    floatM1 := float64(m)
    lcg_normalized :=
        min + int(math.Floor(floatRange*floatLCG/floatM1))

    return lcg_normalized
}
```

**Gambar 7.** Kode Program Fungsi Normalisasi *Linear Congruential Generator* (LCG)
Sumber: Dokumentasi Pribadi

Untuk memastikan fungsi LCG yang telah dinormalisasi bekerja dengan baik, dilakukan uji coba keluaran 15 LCG pertama yang telah dinormalisasi pada rentang [1, 20].

```go
func main() {
    // 15 LCG Pertama yang sudah dinormalisasi
    fmt.Println("15 LCG Pertama yang sudah dinormalisasi:")
    fmt.Println("Dengan range [1, 20]")
    for i := 0; i < 15; i++ {
        fmt.Println(normalizedLCG(LCG_a, LCG_c, LCG_m, 1, 20))
    }
}
```

**Gambar 8.** Kode Program Uji Fungsi *Linear Congruential Generator* (LCG) pada rentang [1, 20]
Sumber: Dokumentasi Pribadi

```
15 LCG Pertama yang sudah dinormalisasi:
Dengan range [1, 20]
3
8
2
4
11
2
16
10
4
15
1
20
8
5
6
```

**Gambar 9.** 15 Hasil Pertama Uji Fungsi Normalisasi *Linear*

*Congruential Generator* (LCG) pada rentang [1, 20]
Sumber: Dokumentasi Pribadi

### *C. Algoritma Fisher-Yates*

Untuk mengacak lagu dalam sebuah *playlist*, akan digunakan Algoritma Fisher-Yates. Untuk kasus sederhana, akan diuji menggunakan bilangan bulat terlebih dahulu.

```go
// Algorithm Fisher-Yates
// I.S. array yang belum diacak
// F.S. array yang sudah diacak dengan algoritma Fisher-Yates
func shuffle[T string | int](arr []T) []T {
    // Dapatkan panjang array
    n := len(arr)

    // Lakukan perulangan dari n - 1 sampai 1
    for i := n - 1; i >= 1; i-- {
        // Dapatkan nilai acak j dari rentang [0, i]
        j := normalizedLCG(LCG_a, LCG_c, LCG_m, 0, i)

        // Tukar posisi pada array
        arr[j], arr[i] = arr[i], arr[j]
    }

    return arr
}
```

**Gambar 10.** Kode Program Fungsi Acak Menggunakan Algoritma Fisher-Yates
Sumber: Dokumentasi Pribadi

Diperlukan implementasi juga sebuah fungsi untuk mencetak elemen-elemen di dalam sebuah *array*.

```go
// Cetak musik
// I.S. m terdefinisi
// F.S. m dicetak ke layar
func printArr[T string | int](m []T) {
    for _, v := range m {
        // Cetak
        fmt.Println(v)
    }
}
```

**Gambar 11.** Kode Program Fungsi Mencetak *Array*
Sumber: Dokumentasi Pribadi

Untuk memastikan fungsi *shuffle* dan fungsi *print array* bekerja dengan baik, akan dilakukan uji coba keluaran *array* yang berisi angka dari 1 sampai 15.

```go
// // Test menggunakan angka
var testNumber = []int{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}

func main() {
    fmt.Println("========================================")
    fmt.Println("Before shuffle:")
    printArr(testNumber)
    fmt.Println("========================================")
    fmt.Println("After shuffle:")
    printArr(shuffle(testNumber))
    fmt.Println("========================================")
}
```

**Gambar 12.** Kode Program Utama Uji Fungsi Acak Menggunakan Algoritma Fisher-Yates
Sumber: Dokumentasi Pribadi

```
==========
Before shuffle:
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
==========
After shuffle:
3
11
7
9
10
5
12
4
8
13
14
15
1
6
2
==========
```

**Gambar 13.** Hasil Uji Fungsi Acak Menggunakan Algoritma Fisher-Yates
Sumber: Dokumentasi Pribadi

### *D. Pengacakan Lagu dalam Album*

Saat pertama kali mendengarkan musik, pengguna akan mendengarkan hasil acakan pertama. Setelah hasil acakan pertama sudah selesai didengarkan, maka pengguna akan mendengarkan hasil acakan dari acakan pertama. Untuk $n > 1$, setelah hasil acakan ke $n$ sudah selesai, pengguna akan mendengarkan hasil acakan dari acakan ke $n - 1$.

Sebelumnya telah diuji pengacakan angka. Sekarang akan diuji pengacakan lagu dalam sebuah album. Album yang akan dipakai untuk uji coba adalah album *"Close To You"* (1970) oleh Carpenters.

```go
// Album Close To You (1970) - Carpenters
// Urutan nomor track pada album
var closeToYou = []string{
    "1. We've Only Just Begun - Carpenters - Close To You (1970)",
    "2. Love is Surrender - Carpenters - Close To You (1970)",
    "3. Maybe It's You - Carpenters - Close To You (1970)",
    "4. Reason to Believe - Carpenters - Close To You (1970)",
    "5. Help! - Carpenters - Close To You (1970)",
    "6. (They Long to Be) Close to You - Carpenters - Close To You (1970)",
    "7. Baby It's You - Carpenters - Close To You (1970)",
    "8. I'll Never Fall in Love Again - Carpenters - Close To You (1970)",
    "9. Crescent Noon - Carpenters - Close To You (1970)",
    "10. Mr. Guder - Carpenters - Close To You (1970)",
    "11. I Kept on Loving You - Carpenters - Close To You (1970)",
    "12. Another Song - Carpenters - Close To You (1970)",
}
```

**Gambar 14.** *Array* Berisi Lagu-Lagu pada Album *"Close To You"* (1970) oleh Carpenters.
Sumber: Dokumentasi Pribadi

Untuk memastikan fungsi *shuffle* bekerja dengan baik dalam pengacakan lagu dalam suatu album, akan diuji untuk $n = 3$ yaitu 3 kali acakan.

```go
func main() {
    // Inisialisasi shuffled
    shuffled := []string{}

    fmt.Println("========================================")
    fmt.Println("Before shuffle:")
    printArr(closeToYou)
    fmt.Println("========================================")
    fmt.Println("After shuffle: #1")
    shuffled = shuffle(closeToYou)
    printArr(shuffled)
    fmt.Println("========================================")
    fmt.Println("After shuffle: #2")
    shuffled = shuffle(shuffled)
    printArr(shuffled)
    fmt.Println("========================================")
    fmt.Println("After shuffle: #3")
    shuffled = shuffle(shuffled)
    printArr(shuffled)
    fmt.Println("========================================")
}
```

**Gambar 15.** Kode Program Utama Pengacakan 3 Kali Album *"Close To You"* (1970) oleh Carpenters.
Sumber: Dokumentasi Pribadi

```
========================================
Before shuffle:
1. We've Only Just Begun - Carpenters - Close To You (1970)
2. Love is Surrender - Carpenters - Close To You (1970)
3. Maybe It's You - Carpenters - Close To You (1970)
4. Reason to Believe - Carpenters - Close To You (1970)
5. Help! - Carpenters - Close To You (1970)
6. (They Long to Be) Close to You - Carpenters - Close To You (1970)
7. Baby It's You - Carpenters - Close To You (1970)
8. I'll Never Fall in Love Again - Carpenters - Close To You (1970)
9. Crescent Noon - Carpenters - Close To You (1970)
10. Mr. Guder - Carpenters - Close To You (1970)
11. I Kept on Loving You - Carpenters - Close To You (1970)
12. Another Song - Carpenters - Close To You (1970)
========================================
After shuffle: #1
9. Crescent Noon - Carpenters - Close To You (1970)
4. Reason to Believe - Carpenters - Close To You (1970)
6. (They Long to Be) Close to You - Carpenters - Close To You (1970)
7. Baby It's You - Carpenters - Close To You (1970)
3. Maybe It's You - Carpenters - Close To You (1970)
8. I'll Never Fall in Love Again - Carpenters - Close To You (1970)
10. Mr. Guder - Carpenters - Close To You (1970)
11. I Kept on Loving You - Carpenters - Close To You (1970)
12. Another Song - Carpenters - Close To You (1970)
1. We've Only Just Begun - Carpenters - Close To You (1970)
5. Help! - Carpenters - Close To You (1970)
2. Love is Surrender - Carpenters - Close To You (1970)
========================================
After shuffle: #2
10. Mr. Guder - Carpenters - Close To You (1970)
9. Crescent Noon - Carpenters - Close To You (1970)
7. Baby It's You - Carpenters - Close To You (1970)
5. Help! - Carpenters - Close To You (1970)
12. Another Song - Carpenters - Close To You (1970)
11. I Kept on Loving You - Carpenters - Close To You (1970)
8. I'll Never Fall in Love Again - Carpenters - Close To You (1970)
4. Reason to Believe - Carpenters - Close To You (1970)
1. We've Only Just Begun - Carpenters - Close To You (1970)
6. (They Long to Be) Close to You - Carpenters - Close To You (1970)
3. Maybe It's You - Carpenters - Close To You (1970)
2. Love is Surrender - Carpenters - Close To You (1970)
========================================
After shuffle: #3
2. Love is Surrender - Carpenters - Close To You (1970)
4. Reason to Believe - Carpenters - Close To You (1970)
12. Another Song - Carpenters - Close To You (1970)
6. (They Long to Be) Close to You - Carpenters - Close To You (1970)
1. We've Only Just Begun - Carpenters - Close To You (1970)
3. Maybe It's You - Carpenters - Close To You (1970)
5. Help! - Carpenters - Close To You (1970)
7. Baby It's You - Carpenters - Close To You (1970)
10. Mr. Guder - Carpenters - Close To You (1970)
8. I'll Never Fall in Love Again - Carpenters - Close To You (1970)
11. I Kept on Loving You - Carpenters - Close To You (1970)
9. Crescent Noon - Carpenters - Close To You (1970)
========================================
```

**Gambar 16.** Hasil Keluaran Program Pengacakan 3 Kali Album *"Close To You"* (1970) oleh Carpenters.
Sumber: Dokumentasi Pribadi

### *E. Pengacakan Lagu dalam Playlist*

Sebelumnya, telah diuji pengacakan lagu dalam sebuah album sehingga semua penulis musiknya sama. Sekarang, akan diujikan untuk *playlist* lagu yang berisi berbagai penulis musik.

```
// Random playlist
// Urutan berdasarkan waktu penambahan ke playlist
var randomPlaylist = []string{
    "1. Landslide - Fleetwood Mac - Fleetwood Mac (1975)",
    "2. Breathe - Pink Floyd - The Dark Side of the Moon (1973)",
    "3. (They Long to Be) Close to You - Carpenters - Close To You (1970)",
    "4. Time - Pink Floyd - The Dark Side of the Moon (1973)",
    "5. We've Only Just Begun - Carpenters - Close To You (1970)",
    "6. Something - The Beatles - Abbey Road (1969)",
    "7. A Day in the Life - The Beatles - Sgt. Pepper's Lonely Hearts Club Band (1967)",
    "8. Here Comes the Sun - The Beatles - Abbey Road (1969)",
    "9. New Kid in Town - Eagles - Hotel California (1976)",
    "10. Dust in the Wind - Kansas - Point of Know Return (1977)",
    "11. Let It Be - The Beatles - Let It Be (1970)",
    "12. And I Love Her - The Beatles - A Hard Day's Night (1964)",
    "13. Sara - Fleetwood Mac - Tusk (1979)",
    "14. Yesterday - The Beatles - Help! (1965)",
    "15. Dreams - Fleetwood Mac - Rumours (1977)",
}
```

**Gambar 17.** *Array* Berisi Lagu-Lagu pada Sebuah *Playlist*
Sumber: Dokumentasi Pribadi

Sama seperti sebelumnya, pertama pengguna akan mendengarkan hasil acakan pertama. Untuk $n > 1$, setelah hasil acakan ke $n$ sudah selesai, pengguna akan mendengarkan hasil acakan dari acakan ke $n - 1$. Untuk kesederhanaan, akan diuji untuk $n = 3$, yaitu 3 acakan.

```
func main() {
    // Inisialisasi shuffled
    shuffled := []string{}

    fmt.Println("========================================")
    fmt.Println("Before shuffle:")
    printArr(randomPlaylist)
    fmt.Println("========================================")
    fmt.Println("After shuffle: #1")
    shuffled = shuffle(randomPlaylist)
    printArr(shuffled)
    fmt.Println("========================================")
    fmt.Println("After shuffle: #2")
    shuffled = shuffle(shuffled)
    printArr(shuffled)
    fmt.Println("========================================")
    fmt.Println("After shuffle: #3")
    shuffled = shuffle(shuffled)
    printArr(shuffled)
    fmt.Println("========================================")
}
```

**Gambar 18.** Kode Program Utama Pengacakan 3 Kali Sebuah *Playlist*
Sumber: Dokumentasi Pribadi

```
========================================
Before shuffle:
1. Landslide - Fleetwood Mac - Fleetwood Mac (1975)
2. Breathe - Pink Floyd - The Dark Side of the Moon (1973)
3. (They Long to Be) Close to You - Carpenters - Close To You (1970)
4. Time - Pink Floyd - The Dark Side of the Moon (1973)
5. We've Only Just Begun - Carpenters - Close To You (1970)
6. Something - The Beatles - Abbey Road (1969)
7. A Day in the Life - The Beatles - Sgt. Pepper's Lonely Hearts Club Band (1967)
8. Here Comes the Sun - The Beatles - Abbey Road (1969)
9. New Kid in Town - Eagles - Hotel California (1976)
10. Dust in the Wind - Kansas - Point of Know Return (1977)
11. Let It Be - The Beatles - Let It Be (1970)
12. And I Love Her - The Beatles - A Hard Day's Night (1964)
13. Sara - Fleetwood Mac - Tusk (1979)
14. Yesterday - The Beatles - Help! (1965)
15. Dreams - Fleetwood Mac - Rumours (1977)
========================================
After shuffle: #1
3. (They Long to Be) Close to You - Carpenters - Close To You (1970)
11. Let It Be - The Beatles - Let It Be (1970)
7. A Day in the Life - The Beatles - Sgt. Pepper's Lonely Hearts Club Band (1967)
9. New Kid in Town - Eagles - Hotel California (1976)
10. Dust in the Wind - Kansas - Point of Know Return (1977)
5. We've Only Just Begun - Carpenters - Close To You (1970)
12. And I Love Her - The Beatles - A Hard Day's Night (1964)
4. Time - Pink Floyd - The Dark Side of the Moon (1973)
8. Here Comes the Sun - The Beatles - Abbey Road (1969)
13. Sara - Fleetwood Mac - Tusk (1979)
14. Yesterday - The Beatles - Help! (1965)
15. Dreams - Fleetwood Mac - Rumours (1977)
1. Landslide - Fleetwood Mac - Fleetwood Mac (1975)
6. Something - The Beatles - Abbey Road (1969)
2. Breathe - Pink Floyd - The Dark Side of the Moon (1973)
========================================
After shuffle: #2
15. Dreams - Fleetwood Mac - Rumours (1977)
8. Here Comes the Sun - The Beatles - Abbey Road (1969)
2. Breathe - Pink Floyd - The Dark Side of the Moon (1973)
13. Sara - Fleetwood Mac - Tusk (1979)
1. Landslide - Fleetwood Mac - Fleetwood Mac (1975)
9. New Kid in Town - Eagles - Hotel California (1976)
3. (They Long to Be) Close to You - Carpenters - Close To You (1970)
11. Let It Be - The Beatles - Let It Be (1970)
4. Time - Pink Floyd - The Dark Side of the Moon (1973)
12. And I Love Her - The Beatles - A Hard Day's Night (1964)
5. We've Only Just Begun - Carpenters - Close To You (1970)
6. Something - The Beatles - Abbey Road (1969)
14. Yesterday - The Beatles - Help! (1965)
7. A Day in the Life - The Beatles - Sgt. Pepper's Lonely Hearts Club Band (1967)
10. Dust in the Wind - Kansas - Point of Know Return (1977)
========================================
After shuffle: #3
9. New Kid in Town - Eagles - Hotel California (1976)
4. Time - Pink Floyd - The Dark Side of the Moon (1973)
12. And I Love Her - The Beatles - A Hard Day's Night (1964)
7. A Day in the Life - The Beatles - Sgt. Pepper's Lonely Hearts Club Band (1967)
3. (They Long to Be) Close to You - Carpenters - Close To You (1970)
1. Landslide - Fleetwood Mac - Fleetwood Mac (1975)
14. Yesterday - The Beatles - Help! (1965)
15. Dreams - Fleetwood Mac - Rumours (1977)
11. Let It Be - The Beatles - Let It Be (1970)
5. We've Only Just Begun - Carpenters - Close To You (1970)
13. Sara - Fleetwood Mac - Tusk (1979)
8. Here Comes the Sun - The Beatles - Abbey Road (1969)
6. Something - The Beatles - Abbey Road (1969)
2. Breathe - Pink Floyd - The Dark Side of the Moon (1973)
10. Dust in the Wind - Kansas - Point of Know Return (1977)
========================================
```

**Gambar 19.** Hasil Keluaran Program Pengacakan 3 Kali Sebuah *Playlist*.
Sumber: Dokumentasi Pribadi

## IV. SIMPULAN

Algoritma Fisher-Yates dan *Linear Congruential Generator* (LCG) dapat diimplementasikan secara efektif sebagai pengacak lagu dalam sebuah *playlist* pada aplikasi pemain musik. Kecepatan dan efisiensi kedua algoritma ini yang cepat menjadikannya pilihan yang bagus untuk memainkan musik secara acak. Dengan menerapkan kedua algoritma ini, pengguna dapat menikmati keberagaman dalam urutan lagu dengan bahagia.

## V. Ucapan Terima Kasih

Penulis berterima kasih kepada Tuhan yang Maha Esa karena telah memberikan kemudahan dalam penulisan makalah ini. Penulis juga ingin berterima kasih kepada Dr. Nur Ulva Maulidevi, S.T., M.Sc. selaku dosen dan pembimbing penulis dalam mata kuliah IF2120 Matematika Diskrit. Penulis juga ingin berterima kasih kepada Dr. Yani Widyani, S.T, M.T. dan Fitra Arifiansyah, S.Kom., M.T. selaku dosen dalam mata kuliah IF2110 Algoritma & Struktur Data. Selain itu, penulis juga ingin berterima kasih kepada keluarga serta teman-teman yang sudah mendukung penulis dalam menyelesaikan penulisan makalah ini. Terakhir, penulis juga ingin berterima kasih kepada musisi-musisi yang telah penulis sebutkan dalam makalah ini yang sudah menghibur penulis dengan lagu-lagunya dalam menyelesaikan penulisan makalah ini.

## Referensi

[1] Munir, Rinaldi. 2023. "Rekursi dan relasi rekurens (Bagian 2)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/10-Rekursi-dan-relasi-rekurens-(Bagian2)-2023.pdf (Diakses pada 12 November 2023).

[2] Munir, Rinaldi. 2023. "Teori Bilangan (Bagian 3)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/16-Teori-Bilangan-Bagian3-2023.pdf (Dikases pada 12 November 2023).

[3] L'Ecuyer, Pierre. 2017. "HISTORY OF UNIFORM RANDOM NUMBER GENERATION". https://www.iro.umontreal.ca/~lecuyer/myftp/papers/wsc17rng-history.pdf (Diakses pada 12 November 2023).

[4] https://sourceware.org/git/?p=glibc.git;a=blob;f=stdlib/random_r.c;hb=glibc-2.26#l362 (Dikases pada 12 November 2023).

[5] Fisher, Ronald A.; Yates, Frank. 1948. "Statistical tables for biological, agricultural and medical research (3rd ed.)". https://digital.library.adelaide.edu.au/dspace/bitstream/2440/10701/1/stat_tab.pdf (Diakses pada 12 November 2023).

[6] Smith, James. 2023. "Let's Do the Knuth Shuffle". https://golangprojectstructure.com/the-knuth-shuffle/ (Diakses pada 12 November 2023).

[7] Durstenfeld, R. 1964. "Algorithm 235: Random permutation". https://dl.acm.org/doi/pdf/10.1145/364520.364540#.pdf (Diakses pada 12 November 2023).

[8] Black, Paul E. 2005. "Fisher-Yates shuffle". https://xlinux.nist.gov/dads/HTML/fisherYatesShuffle.html (Diakses pada 12 November 2023).

[9] Arndt, Jörg. 2009. "Generating Random Permutations (PhD Thesis)". https://maths-people.anu.edu.au/~brent/pd/Arndt-thesis.pdf (Diakses pada 12 November 2023).

## Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 8 Desember 2022

Dewantoro Triatmojo
13522011