

計算機科学基礎実験
第一回レポート
Rust によるプログラミング演習

情報科学科
Rai

2024 年 5 月 14 日

1. 課題内容

1.1. 課題 1: Rust でハローワールドを出力する

Rust のプログラミング言語を使用して、コンソールに「Hello, World!」と出力するプログラムを作成してください。

プログラムの実行結果は コード 1 のようになることを確認してください。

コード 1

```
$ cargo run
Compiling hello-world v0.1.0 (/home/r4ai/Projects/hello-world)
Finished dev [unoptimized + debuginfo] target(s) in 0.25s
    Running `target/debug/hello-world`
Hello, World!
```

1.2. 課題 2: Rust でFizzBuzz を実装する

Rust のプログラミング言語を使用して、1から 100 までの整数を順番に出力するプログラムを作成してください。

ただし、以下の条件を満たすようにしてください。

- ・ 3 の倍数の場合は「Fizz」を出力する
- ・ 5 の倍数の場合は「Buzz」を出力する
- ・ 3 の倍数かつ 5 の倍数の場合は「FizzBuzz」を出力する

プログラムの実行結果は以下のようになることを確認してください。

```
$ cargo run
Compiling fizzbuzz v0.1.0 (/home/r4ai/Projects/fizzbuzz)
Finished dev [unoptimized + debuginfo] target(s) in 0.25s
    Running `target/debug/fizzbuzz`
1
2
Fizz
4
Buzz
Fizz
7
...
... (中略)
94
Buzz
Fizz
```

97
98
Fizz
Buzz

1.3. 課題 3: Rust でクイックソートを実装する

Rust のプログラミング言語を使用して、クイックソートを実装してください。

クイックソートは、与えられたいくつかの要素を、所定の順序にしたがって並び替えるソートアルゴリズムです。クイックソートには次の特徴があります(要素の個数を N とします)。

- ・部分問題を再帰的に解くアルゴリズムである(分割統治法)
- ・最悪時の計算量は $O(N^2)$
- ・最良時と平均時の計算量は $O(N \log N)$

以下に示すアルゴリズムは、サイズ N の配列 A を昇順に並び替えるクイックソートの動作を表したものです [1]。

$A[X](X = \lfloor \frac{N}{2} \rfloor)$ を軸としてソートを行う。

1. 空の配列 L, R を用意し、次の操作を $i = 0, 1, \dots, N - 1$ について行う。
 1. $i = X$ ならば何も行わない。
 2. $i \neq X$ かつ $A[i] < A[X]$ ならば $A[i]$ を L の末尾に追加する。
 3. $i \neq X$ かつ $A[i] \geq A[X]$ ならば $A[i]$ を R の末尾に追加する。
2. L, R をクイックソートを用いて再帰的にソートする。空配列の場合は何も変わらない。
3. L の要素、 $A[X], R$ の要素をこの順につなげてできる配列を出力する。

プログラムの実行結果は以下のようになることを確認してください。

```
$ cargo run
Compiling quicksort v0.1.0 (/home/r4ai/Projects/quicksort)
Finished dev [unoptimized + debuginfo] target(s) in 0.25s
  Running `target/debug/quicksort`
```

1
2
3
4
5
6
7
8
9
10

2. アルゴリズム

2.1. 課題 1: Rust でハローワールドを出力する

以下に、ハローワールドを出力するアルゴリズムを示す。

1. 「Hello, World!」を出力する。

2.2. 課題 2: Rust でFizzBuzzを実装する

以下に、FizzBuzzを実装するアルゴリズムを示す。

1. $i = 1, 2, \dots, 100$ について、次の操作を行う。
 1. i が3の倍数かつ5の倍数ならば「FizzBuzz」を出力する。
 2. i が3の倍数ならば「Fizz」を出力する。
 3. i が5の倍数ならば「Buzz」を出力する。
 4. いずれでもない場合は i を出力する。

2.3. 課題 3: Rust でクイックソートを実装する

以下に、クイックソートを実装するアルゴリズムを示す。

1. $A[X] (X = \lfloor \frac{N}{2} \rfloor)$ を軸としてソートを行う。
2. 空の配列 L, R を用意し、次の操作を $i = 0, 1, \dots, N - 1$ について行う。
 1. $i = X$ ならば何も行わない。
 2. $i \neq X$ かつ $A[i] < A[X]$ ならば $A[i]$ を L の末尾に追加する。
 3. $i \neq X$ かつ $A[i] \geq A[X]$ ならば $A[i]$ を R の末尾に追加する。
3. L, R をクイックソートを用いて再帰的にソートする。空配列の場合は何も変わらない。
4. L の要素、 $A[X], R$ の要素をこの順につなげてできる配列を出力する。

3. プログラム

3.1. 課題 1: Rust でハローワールドを出力する

節 2.1 を Rust で実装したプログラムを コード 2 に示す。

コード 2: helloworld.rs

```
1 fn main() {
```

```
2     println!("Hello, World!");
3 }
```

3.2. 課題 2: Rust で FizzBuzz を実装する

節 2.2 を Rust で実装したプログラムを コード 3 に示す。

コード 3: fizzbuzz.rs

```
1 fn main() {
2     for i in 1..=100 {
3         if i % 15 == 0 {
4             println!("FizzBuzz");
5         } else if i % 3 == 0 {
6             println!("Fizz");
7         } else if i % 5 == 0 {
8             println!("Buzz");
9         } else {
10            println!("{}", i);
11        }
12    }
13 }
```

3.3. 課題 3: Rust でクイックソートを実装する

節 2.3 を Rust で実装したプログラムを コード 4 に示す。

コード 4: quicksort.rs

```
1 fn quicksort<T: Ord + Copy>(a: &mut [T]) {
2     if a.len() ≤ 1 {
3         return;
4     }
5
6     let pivot = a.len() / 2;
7     let mut l = Vec::new();
8     let mut r = Vec::new();
9
10    for i in 0..a.len() {
11        if i == pivot {
12            continue;
13        }
14
15        if a[i] < a[pivot] {
16            l.push(a[i]);
17        } else {
18            r.push(a[i]);
19        }
20    }
21
22    quicksort(&mut l);
23    quicksort(&mut r);
24
25    a[0..pivot].copy_from_slice(&l);
26    a[pivot..l.len() + pivot].copy_from_slice(&r);
27 }
```

```
19         }
20     }
21
22     quicksort(&mut l);
23     quicksort(&mut r);
24
25     for i in 0..l.len() {
26         a[i] = l[i];
27     }
28     a[l.len()] = a[pivot];
29     for i in 0..r.len() {
30         a[l.len() + 1 + i] = r[i];
31     }
32 }
33
34 fn main() {
35     let mut a = [6, 3, 8, 2, 9, 1, 4, 7, 5, 10];
36     quicksort(&mut a);
37     for i in 0..a.len() {
38         println!("{}", a[i]);
39     }
40 }
```

4. 実行結果

4.1. 課題 1: Rust でハローワールドを出力する

コード 2 を実行した結果を 図 1 に示す。



```
~/.ls/c/p/t/s/source ✘ + ▾
~/.local/share/chezmoi/private_dot_config/typst/sample/source on ↵ main via 🐀 v1.74.0
> ./helloworld
Hello, World!
~/.local/share/chezmoi/private_dot_config/typst/sample/source on ↵ main via 🐀 v1.74.0
>
```

図 1: helloworld.rs の実行結果

4.2. 課題 2: Rust で FizzBuzz を実装する

コード 3 を実行した結果を 図 2 に示す。なお、出力結果が長すぎるため、一部のみを抜粋している。

```
~/.local/share/chezmoi/private_dot_config/typst/sample/source on ✘ main via 🐣 v1.74.0
> rustc fizzbuzz.rs
~/local/share/chezmoi/private_dot_config/typst/sample/source on ✘ main via 🐣 v1.74.0
> ./fizzbuzz
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
```

図 2: fizzbuzz.rs の実行結果

4.3. 課題 3: Rust でクイックソートを実装する

コード 4 を実行した結果を 図 3 に示す。

```
~/.local/share/chezmoi/private_dot_config/typst/sample/source on ✘ main via 🐣 v1.74.0
> rustc quicksort.rs
~/local/share/chezmoi/private_dot_config/typst/sample/source on ✘ main via 🐣 v1.74.0
> ./quicksort
1
2
3
4
5
6
7
8
6
10
```

図 3: quicksort.rs の実行結果

5. 考察

5.1. 課題 1: Rust でハローワールドを出力する

Rust のプログラミング言語を使用して、コンソールに「Hello, World!」と出力するプログラムを作成した。

5.2. 課題 2: Rust で FizzBuzz を実装する

Rust のプログラミング言語を使用して、1 から 100 までの整数を順番に出力するプログラムを作成した。

5.3. 課題 3: Rust でクイックソートを実装する

Rust のプログラミング言語を使用して、クイックソートを実装した。クイックソートの計算量は、最悪時は式 (1)、最良時と平均時は式 (2) である。

$$O(N^2) \quad (1)$$

$$O(N \log N) \quad (2)$$

よって、クイックソートはバブルソートや挿入ソートと比較して、計算量が少ないアルゴリズムであると言える。また、クイックソートは並列化が容易であるため、並列計算に適しており、さらなる高速化が期待できる。

6. おまけ

6.1. 有名分布

6.1.1. 正規分布

X を確率変数、 $\mu \in \mathbb{R}, \sigma > 0$ とする。 X の確率密度関数が、

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

となるとき、 X は平均 μ 、分散 σ^2 の正規分布に従うといい、 $X \sim N(\mu, \sigma^2)$ と表す。

6.1.1.1. 積率母関数の導出

$$\begin{aligned}
E[e^{tX}] &= \int_{-\infty}^{\infty} e^{tx} f(x) \, dx \\
&= \int_{-\infty}^{\infty} e^{tx} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] dx \\
&= \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2} + tx\right] dx \\
&= \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2} + tx\right] dx \\
&= \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} \exp\left[-\frac{(x-(\mu+\sigma^2t))^2}{2\sigma^2} + \mu t + \frac{\sigma^2 t^2}{2}\right] dx \\
&= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[\mu t + \frac{\sigma^2 t^2}{2}\right] \int_{-\infty}^{\infty} \exp\left[-\frac{(x-(\mu+\sigma^2t))^2}{2\sigma^2}\right] dx
\end{aligned}$$

ここで、 $y = \frac{x - (\mu + \sigma^2 t)}{\sqrt{2\sigma^2}}$ とおくと、 $dy = \frac{1}{\sqrt{2\sigma^2}} dx$ であり、

$$\begin{aligned}
E[e^{tX}] &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[\mu t + \frac{\sigma^2 t^2}{2}\right] \sqrt{2\sigma^2} \int_{-\infty}^{\infty} e^{-y^2} dy \\
&= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[\mu t + \frac{\sigma^2 t^2}{2}\right] \sqrt{2\pi\sigma^2} \quad \left(\because \int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi} \right) \\
&= \exp\left[\mu t + \frac{\sigma^2 t^2}{2}\right]
\end{aligned}$$

を得る。

よって、正規分布の積率母関数は、

$$E[e^{tX}] = \exp\left[\mu t + \frac{\sigma^2 t^2}{2}\right]$$

である。

6.1.1.2. 期待値と分散の導出

積率母関数より、正規分布の期待値は、

$$\begin{aligned}
E[X] &= \frac{d}{dt} E[e^{tX}] \Big|_{t=0} \\
&= \frac{d}{dt} \exp \left[\mu t + \frac{\sigma^2 t^2}{2} \right] \Big|_{t=0} \\
&= (\mu + \sigma^2 t) \exp \left[\mu t + \frac{\sigma^2 t^2}{2} \right] \Big|_{t=0} \\
&= \mu
\end{aligned} \tag{3}$$

である。

また、正規分布の2次積率は、

$$\begin{aligned}
E[X^2] &= \frac{d^2}{dt^2} E[e^{tX}] \Big|_{t=0} \\
&= \frac{d}{dt} (\mu + \sigma^2 t) \exp \left[\mu t + \frac{\sigma^2 t^2}{2} \right] \Big|_{t=0} \\
&= \sigma^2 \exp \left[\mu t + \frac{\sigma^2 t^2}{2} \right] + (\mu + \sigma^2 t)^2 \exp \left[\mu t + \frac{\sigma^2 t^2}{2} \right] \Big|_{t=0} \\
&= \sigma^2 + \mu^2
\end{aligned} \tag{4}$$

である。よって、式(3)と式(4)より正規分布の分散は、

$$\begin{aligned}
V[X] &= E[X^2] - E[X]^2 \\
&= \sigma^2 + \mu^2 - \mu^2 \\
&= \sigma^2
\end{aligned}$$

である。

6.2. Callout

6.2.1. デフォルト Callout

① Note

これはノートです。

② Warning

これは警告です。

” Quote

これは引用です。

ং Rocket

これはロケットです。

○ ToDo

これは TODO です。

① Note

与えられた種類の Callout が存在しない場合、note として扱われます。

タイトル付き Callout :

① コラム

C 言語は、1972 年に AT&T ベル研究所で開発されたプログラミング言語である。

② Deno の注意点

Node.js と完璧な互換性を持つわけではない。

6.2.2. カスタム Callout

独自の Callout も定義できます：

```
#import "@local/jsreport:0.1.0": callout, create-callout

#create-callout(
  "spark",
  (
    "Spark",
    image.decode("<svg width=\"15\" height=\"15\" viewBox=\"0 0 15 15\" fill=\"none\""
      xmlns=\"http://www.w3.org/2000/svg\"><path d=\"M8.69667 0.0403541C8.90859 0.131038 9.03106
      0.354857 8.99316 0.582235L8.0902 6.00001H12.5C12.6893 6.00001 12.8625 6.10701 12.9472
      6.27641C13.0319 6.4458 13.0136 6.6485 12.8999 6.80001L6.89997 14.8C6.76167 14.9844 6.51521
      15.0503 6.30328 14.9597C6.09135 14.869 5.96888 14.6452 6.00678 14.4178L6.90974
      9H2.49999C2.31061 9 2.13748 8.893 2.05278 8.72361C1.96809 8.55422 1.98636 8.35151 2.09999
      8.2L8.09997 0.200038C8.23828 0.0156255 8.48474 -0.0503301 8.69667 0.0403541ZM3.49999
      8.00001H7.49997C7.64695 8.00001 7.78648 8.06467 7.88148 8.17682C7.97648 8.28896 8.01733 8.43723
      7.99317 8.5822L7.33027 12.5596L11.5 7.00001H7.49997C7.353 7.00001 7.21347 6.93534 7.11846
      6.8232C7.02346 6.71105 6.98261 6.56279 7.00678 6.41781L7.66968 2.44042L3.49999 8.00001Z"
      fill="currentColor" fill-rule="evenodd" clip-rule="evenodd"></path></svg>")
  )
)

#callout("spark")[
  Sparking!!
]
```

↳ Spark

Sparking!!

6.3. コードブロック

fizzbuzz.rs

```
1 fn main() {
2     for i in 1..=100 {
3         if i % 15 == 0 {
4             println!("FizzBuzz");
5         } else if i % 3 == 0 {
6             println!("Fizz");
7         } else if i % 5 == 0 {
8             println!("Buzz");
9         } else {
10            println!("{}", i);
```

```
11      }
12    }
13 }
```

6.4. 参考文献テスト

ハリー・ポッターと不死鳥の騎士団 [2] は、J・K・ローリングが 2003 年に発表した、小説『ハリー・ポッター』シリーズの第 5 巻である。

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliquam quaerat voluptatem. Ut enim aequo doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere. [3]

参考文献

- [1] アルゴ式, 「Q4. クイックソート」. [Online]. 入手先: <https://algo-method.com/tasks/442>
- [2] J. K. Rowling, Harry Potter and the Order of the Phoenix, vol. 5. 2003.
- [3] Ishkur, 「Ishkur's Guide to Electronic Music」. [Online]. 入手先: <http://www.techno.org/electronic-music-guide/>