2022年度 プログラミングIII 第8,9,10,11回 レポート

学籍番号 33114098
野上恵吾

2022年12月23日

1 演習課題

1.1 題名

今回私が作ろうとしたのは markdown parser と markdown to html ライブラリである。markdown parser を作ろうと決めた理由は、文字列処理が苦手な c 言語であえて文字列処理をすることで c 言語の不自由さを体感したかったからだ。まず当初の実装内容について説明する。最初は markdown を html に変換するには

1. markdown
2. abstract syntax tree
3. html

と変換するつもりであった。先に markdown to ast の説明をする。

2 markdown to ast の説明

markdown to abstract syntax tree(以下 ast) 変換と ast to html 変換のうち markdown to ast 変換について説明する。図 1 は文”今日の昼ごはんは**ラーメン**です。”の一例である。文章を以下の 3 トークンに分けている。

1. 太字以前のテキスト
2. 太字部分
3. 太字以降のテキスト

さらに 2 の太字部分のテキストでは太字トークンの子としてテキストトークンを持つようにしている。つまり太字トークン内にテキストを持っていないのだ。これには理由がある。例えば図 2 のように太字トークンの中に斜体テキストがある場合を考えてみてほしい。これを実装したいとき、太字トークン内にテキストを持つようにすると斜体が適用される範囲を表すのにややこしくなる。そのため太字トークンや斜体トークンなどの特殊トークンとテキストトークンをわけている。

これをソースコード紛いのものに直すと疑似ソースコード 1 ようである。簡単に説明すると、正規表現によって見出しや箇条書きの記号を探しに行く。もしそれらを見つけたら、その箇条書きのテキストに太字などがあるかもしれないため、マッチした部分のテキストに対してトークンがあるか調べに行く。そして正規表現にしなかったテキストに特殊記法がないか調べるため、元の文章からマッチした部分を消去し、トークン化すべきものがないか調べに行く。ちなみにこれは再帰関数であり、深さ優先探索である。”文章をトークン化して ast に保存する関数”の中でさらに”文章をトークン化して ast に保存する関数”を使っているからである。

ソースコード 1: makrdown to ast のアルゴリズム

```
1 function 文章をトークン化してast に保存(){
2   // トークンを格納するスタック配列
```

```
3   Token[] ast = [];
4
5   // "# hello"や "- goodbye"など
6   正規表現で最も外側部分でmarkdown の特殊記法に合致するものがないか調べる();
7   while(文章が空でないなら){
8     if(マッチするものがある){
9       //e.g. "## 今日の**昼ごはん**"の"**昼ごはん**"を検知しにいく
10      中身の文章に対して文章をトークン化してast に保存();
11
12      //
13      マッチしたテキストをもとの文章から消去 ();
14      マッチしなかったテキストに特殊記法がないか調べる ();
15    } else {
16      マッチしたテキストをトークン化 ();
17    }
18  }
19 }
```

このソースコードの 17 行目で、マッチしたテキストをもとの文章から消去 () とあり、c 言語でこの処理をうまく実装する方法が思いつかなかった。J 私は初めのころ、UNIX の sed コマンドや Javascript の String クラスの replace メソッドのような便利な文字列置換の方法があると思っていた。だが c 言語の標準ライブラリ string.h にあるのは strcpy 関数や strlen 関数などであり、置換として使えそうなのは文字列探索の strpbrk 関数くらいであった。

私が頭を悩ませたのは関数の少なさだけではない。一番の関門はメモリ管理だ。1 つ問題提起をしたい。置換後の文字列を格納する変数にどれだけメモリを割けばよいのだろうか。例えば”赤巻紙 青巻紙 ... 黒巻紙”のうちの”巻紙”を適当な文字に変えたいとすると、一体どれだけのメモリを確保すればよいのだろうか。私にはベストな方法が思いつかなかった。どうしてもメモリの過剰確保が起きそうだったため、再帰的な探索はやめて、一階層だけ探索する markdown parser を実装した。

これが当初の予定であり、置換関数を実装できなかったためここで挫折した。実装していないが ast to html の仕組みについてもアイデアだけ説明する。

2.1 ast to html の説明

文章を ast に直したあと、html に変換する手続きについて説明する。概していうと、文章として後ろのトークンから順に、上にリフトアップしながら HTML に変換する。”今日のお昼ご飯は**ラーメン**です。”の例でいうと図 3 から図 7 までの流れのように HTML に変換する予定であった。

3 どのような技術を使ったか

ライブラリはすべて標準のものを使った。めずらしいのは正規表現を行う regex.h のみである。

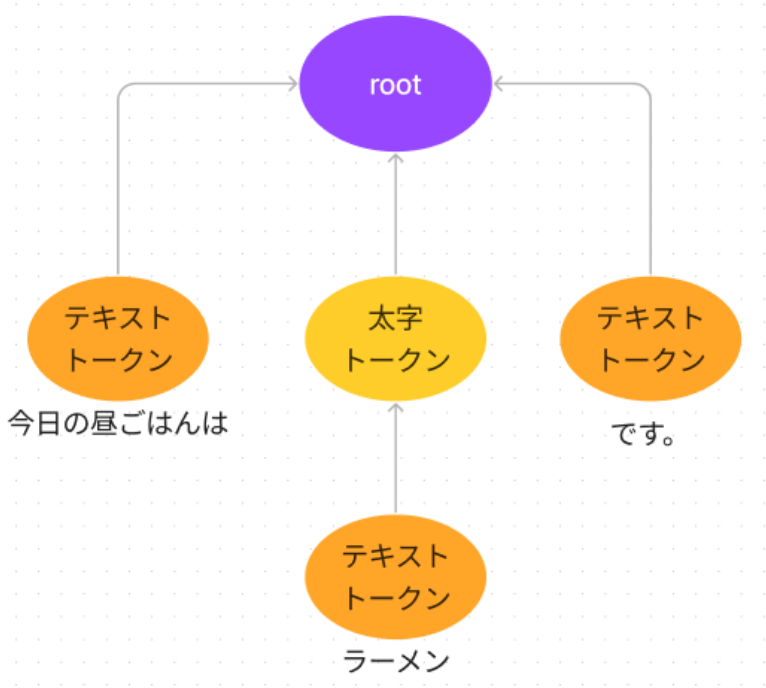

図 1: abstract syntax tree の例

4 実行例

今回の成果から説明する。markdown に機能はたくさんあるが、実際にできたのは以下の 2 つの記法だけである。

- 正規表現/^# (.+)\n$/が入力されたら h1 タグに変換
- それ以外は p タグに変換

これだけを実装しようとしたが後述するエラーが発生し、p タグ変換が機能しちない。私も最初こそはリスト、画像、code、太字などすべて実装しようと息巻いていたが c 言語の文字列操作の弱さによって断念した。それと単純に私のコーディング力が足りなかった。

成果物について説明する。例えば以下のような markdown ファイルの入力があるとき出力のようなものが出力される。

入力:

```
# java
# c
# node
```

出力:

```
<h1>java</h1>
<h1>c</h1>
<h1>node</h1>
```

ここの問題は以下である。

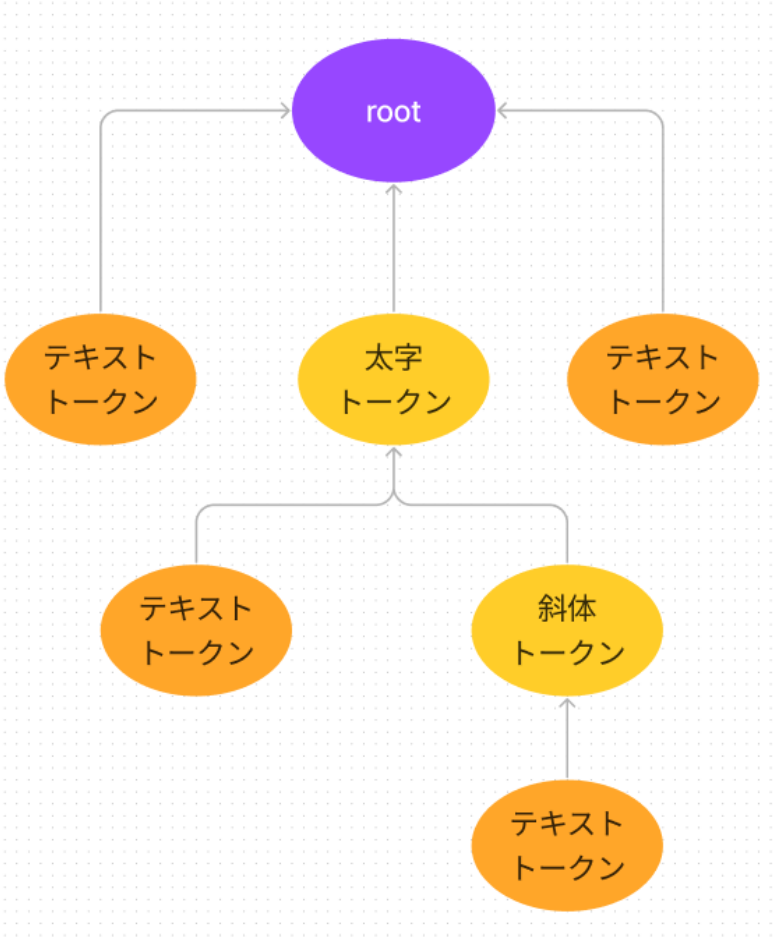

図 2: 斜体トークンが太字トークン内にある場合

- 入力を 4 行以上にすると Segmentation Error が発生する
- #をつけないと Segmentation Error が起こる

これは malloc で確保したメモリを開放していないこと、もしくは確保したメモリの領域が少なかったためだと推測している。時間不足により断念した。

5　ファイル構造の説明

課題では c 言語ファイルは task8.c の 1 つのみであったが、関数やヘッダを 1 つのファイルにまとめるのは著しく可読性に欠けるためファイルを分割した。そのためその説明をする。

ベースディレクトリには次のファイル、フォルダーがある。

- include/
- lib/
- markdown/

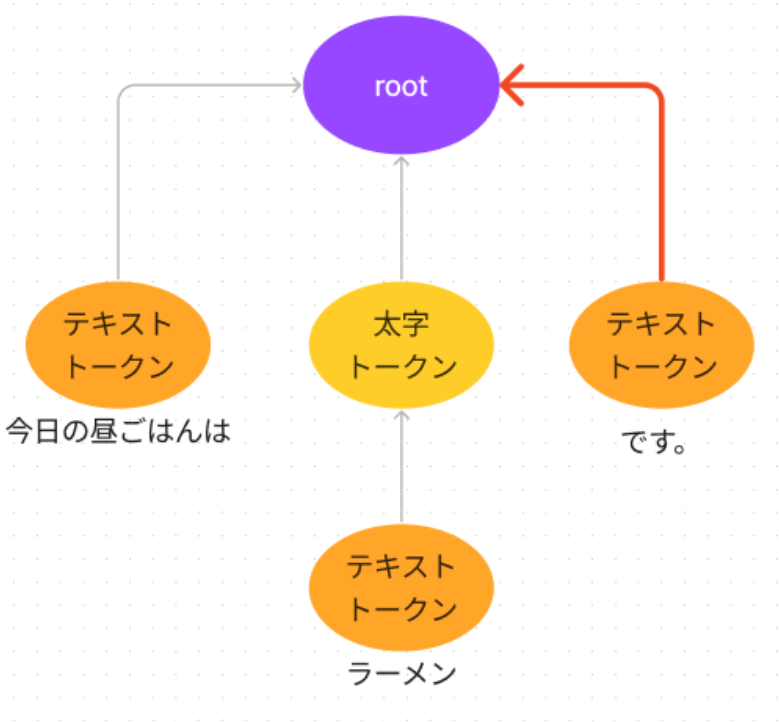

図 3: ast から html に直す例 1

- in.md
- result.html
- task8
- task8.c

ファイルから説明する。task8.c が main 関数を持つインデックスファイルである。この task8.c ファイルで入力の in.md ファイルから中身を取得し、parse をした後に generate() で html のタグツリーを result.html に出力している。

task8 は実行ファイルである。次のようにコマンドをたたくことで非常に限定的であるが markdown が html に変換される。

```
# コンパイル
gcc task8.c ./markdown/lexer.c ./markdown/parser.c ./lib/regexp.c ./lib/func.c -o task8
# 実行
./task8 in.md
```

include ディレクトリはヘッダファイルを集めたものである。このヘッダファイルにはプロトタイプ宣言、マクロ宣言が入っている。多重インクルードしないように ifndef キーワードを使う必要があることを初めて知った。

lib ディレクトリには正規表現プログラム (regexp.c) とリストプログラムの本体 (list.c) が入っている。私は知らなかったのだが、c 言語では正規表現の実装にも多大なる苦労が必要だった。

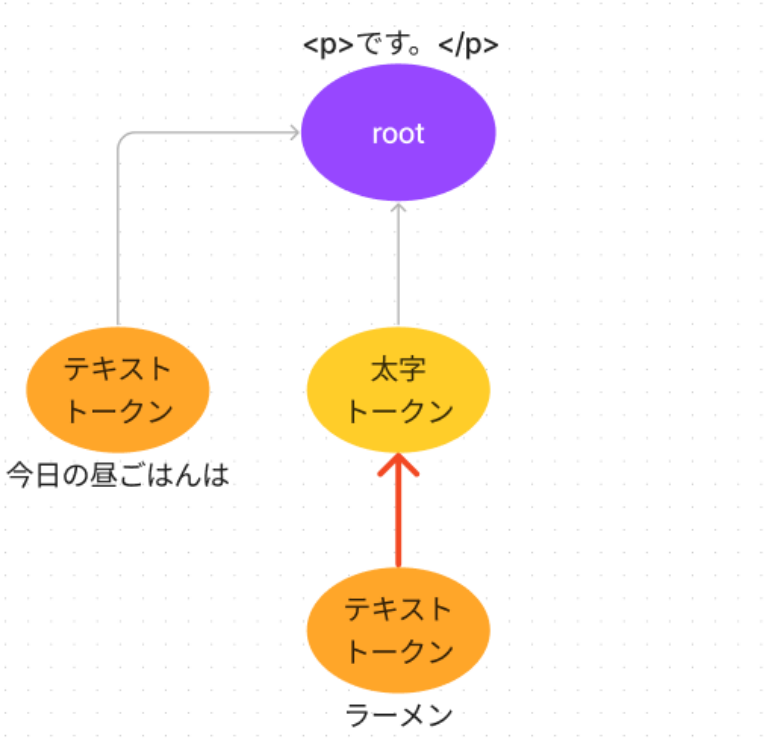

図 4: ast から html に直す例 2

markdown ディレクトリにはマークダウンの解析 (parser.c) と html の出力 (generator.c) の関数がある。

6 感想

やはり c 言語がこんなにも文字列操作に長けていないかということである。正規表現を行うのに、ほかの言語では 1 行でできることが 150 行くらいかかってしまった。これには絶句し、c 言語開発が技術的負債と言われる所以を思い知った。

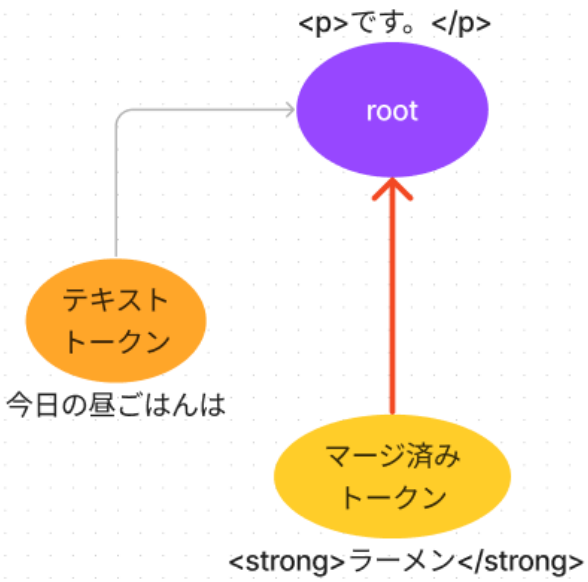

図 5: ast から html に直す例 3

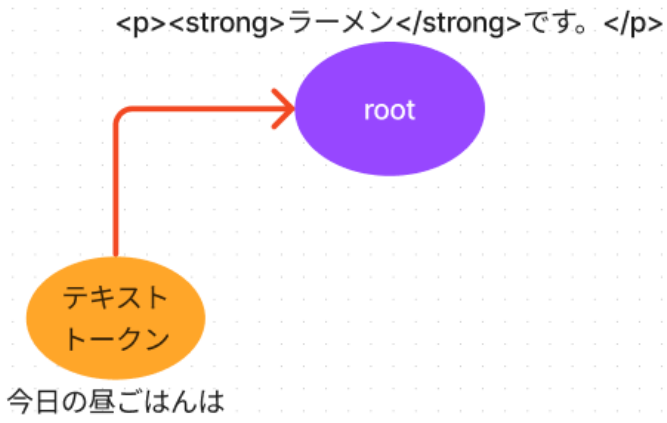

図 6: ast から html に直す例 4

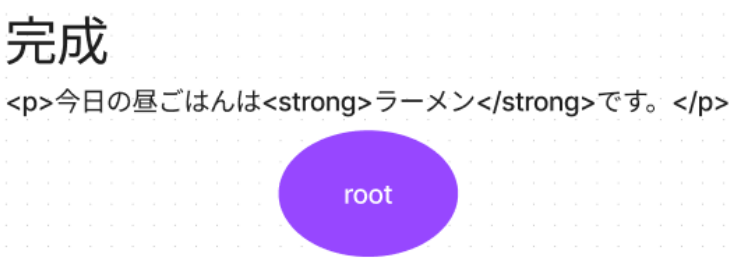

図 7: ast から html に直す例 5 最終