roman_trans ver1.0

【作成者】67140390 平木場 風太
【制作日】2016 年 6 月 15 日
【連絡先】hm14039@student.miyazaki-u.ac.jp
【動作環境】OS X El Capitan(10.11.5)
【開発環境】gcc 4.2.1

1 概要

このソフトウェアはアラビア数字をローマ数字に変換するソフトウェアです。

2 使い方

コマンドラインから実行してください。
プログラム名の後に、引数を与えるか与えないかで挙動が異なります。

2.1 引数を与えない場合

プログラム実行時に引数を与えない場合、キーボードから入力された数字を変換して出力します。終了コード'q' が入力されるまで繰り返します。

動作：

1. コマンドライン上で roman_trans を実行する。
2. 画面の支持に従い、変換したいアラビア数字をキーボードから入力する。
3. 変換されたローマ数字が標準出力される。(異常な値が入力された場合、エラーを出力する。)
4. **??**,**??**を繰り返す。('q' を入力したら終了)

例：入力値 17

```
$ ./roman_trans
---------------------------------------
アラビア数字を入力してください (q で終了)
17
XVII
---------------------------------------
アラビア数字を入力してください (q で終了)
q
```

2.2 引数を与える場合

プログラム実行時に引数を与える場合、与えた引数を 1 つずつ変換し出力します。全ての引数を処理するまで繰り返します。

動作：

1. コマンドライン上で roman_trans を実行する。
2. 引数として与えたアラビア数字を 1 つずつローマ数字に変換し、それぞれ標準出力される。(異常な値が入力された場合、エラーメッセージを出力する。)
3. 終了

例：入力値 17,350,1298

```
$ ./roman_trans 17 350 1298
XVII
CCCL
MCCXCVIII
```

3 注意事項

- ローマ数字をアラビア数字に変換することはできません。(これから搭載していきたい)
- 数字のリストが記述されたテキストファイル等を引数として渡したり、パイプで複数の数字を渡したりして、それらを一括で変換して出力させる機能はありません。あくまで、引数として一つ一つ手入力する必要があります。(これから搭載していきたい)
- ローマ数字は一般的な表記法として、1 から 3999 の範囲の数字にしか対応していない [?] ので、それ以外の数字を入力した場合、エラーを出力します。
- 数字以外の文字を入力した場合、エラーを出力します。
- C 言語で作られているため、コマンドラインからしか実行できません。

参考文献

[1] ローマ数字は奥が深い・・・3999 までは！ http://plaza.rakuten.co.jp/higashiindo/diary/201207200000/

付録 [ソースコード]

Listing 1　roman_trans.v1.0.c

```
1  /**
2   * のヘッダファイルです。Roman_num_trans
3   * Roman_num_trans.h
4   **/
5
6  #ifndef _ROMAN_NUM_TRANS_H_
7  #define _ROMAN_NUM_TRANS_H_
8
9  //プリプロセッサ
10 #include <stdio.h>
11 #include <stdlib.h>
12 #include <string.h>
13 #include <unistd.h>
14
15 #define BUFSIZE 128 //文字列の最大値
16 #define MAX 3999 //ローマ数字の最大値
17 #define MIN 1 //ローマ数字の最小値
18
19 #define ER_NO_INT "ERR数字以外が入力されました:\n"
20 #define ER_REMITES_OVER "ERRローマ数字はまでの数字しか扱えません:1~3999\n"
21
22 //の第一引数にプリプロセッサを適用するための設定 scanf
23 #define STR(s) #s
24 #define XSTR(s) STR(s)
25
26 //構造体宣言
27 struct roman_arabic { //ローマ数字とアラビア数字の変換表の構造体
28     char roman[3]; /* ローマ数字*/
29     int arabic; /* アラビア数字*/
30 };
31
32 //プロトタイプ宣言
33 int rcheck(char* str);
34 void rtrans_print(int num, struct roman_arabic* a);
35
36 #endif //_ROMAN_NUM_TRANS_H_
```

付録 [その他]

★制作スケジュール

理想：6/7(月)〜6/9(木) プログラム作成、6/10(木)〜6/14(火) マニュアル作成

現実：6/12(日)〜6/13(月) プログラム作成、6 14(火)〜6/15(水) マニュアル作成

結局スケジュール通りに作業を進めることができなかった。製作開始がひどく遅れてしまった。実際テストも控えており、プログラミング演習の課題もあったが、そもそも 5 月時点で始めておけば良かった話である。プログラム自体は簡単であったため何とか納期には間に合ったが、マニュアル作成時に、ああしたいこうしたいといったアイデアが次々と浮かび、遅く始めた自分を呪った。実際の仕事だった場合、納期に間に合わないことは許されない上に、余裕があるとソフトウェアの品質を上げることもできるので、余裕を持ったスケジュールを計画・実行していきたい。

★工夫した箇所

プログラム

- 入力する値が数字以外だった場合の例外処理を実装した。
- 今回はソフトウェア制作ということで、普段ならしない、scanf 文でのバッファオーバーフロー (バッファオーバーラン) を防ぐための方法を時間をかけて調べて、何度もテストした。結果うまく動いた。
- ソフトウェア実行時に引数を与えることで、余計なメッセージを出さずに効率よく変換が行えるようにした。これによりシェルスクリプトに組み込むことも用意となった。(ただし、複数の数字をパイプで渡すことができなかった。今後の課題となった。)

マニュアル

- 普段なら MSWord を使うが、今回は LaTeX で文書を作った。自分は Word より LaTeX の方が性に合ってると思った。
- マニュアルを書くということで、フリーソフトにお馴染みの Readme.txt を複数参考にした。
- ソースコードを文書内に取り込んだ (listings というパッケージを使っており、行番号が振られたり、インデントがずれなかったりするだけでなく、ソースファイルに変更があった時、.tex ファイルをいじらなくても、タイプセットすれば文書内のソースコードが更新されるというメリットもある) から楽だった。