プログラミング演習 2
期末レポート

出題日: 2019 年 6 月 19 日
締切日: 2019 年 7 月 31 日

1 概要

本演習では，外部からの入力データを計算機で扱える内部形式に変換して格納し，それらを操作する方法について学習する．具体的には，標準入力から与えられる名簿の CSV データを C 言語の構造体の配列に格納し，それらのデータに対し表示，追加などの処理を行うプログラムを作成する．また，これらの機能は個別の関数で作成することとし必要に応じて関数を追加できるようにする．

与えられたプログラムの基本仕様と要件，および，本レポートにおける実装の概要を以下に述べる．

1. 基本仕様
   (a) 標準入力から「ID, 氏名，年月日，住所，備考」からなるコンマ区切り形式 (CSV 形式) の名簿データを受け付けて，それらをメモリ中に登録する機能を持つ．CSV 形式の例を以下に示す．

```
46,The Bridge,1845-11-2,14 Seafield,SEN Unit2.0 Open
27,Bower School,1908-1-19,Bowermadden Bower,Caithness 25 2.6 Open
24,Canisbay School,1928-7-5,Canisbay Wick,56 3.5 Open
21,Castletown School,1913-11-4,Castletown Thurso,137 8.5 Open
:
```

   (b) 標準入力から%で始まるコマンドを受け付けて，登録してあるデータを表示したり整列したりする機能を持つ．実装するコマンドを表 1 に示す．
2. 要件
   (a) 名簿データは配列などを用いて少なくとも 10000 件のデータを登録できるようにする．今回のプログラムでは，構造体 `struct profile` の配列 `profile_data_store[]` を宣言して，10000 件のデータを格納できるようにする．
   (b) 名簿データは構造体 `struct person` および構造体 `struct date` を利用して，構造を持ったデータとしてプログラム中に定義して利用する．実装すべきデータ構造は表 2 である．
   (c) コマンドの実行結果出力に不必要なエラーや警告は標準エラー出力に出力すること．

また，本レポートでは以下の考察課題について考察をおこなった．

表 1: 実装するコマンド

コマンド	意味	備考
%Q	終了 (Quit)	
%C	登録件数の表示 (Check)	
%P n	CSV の先頭から n 件表示 (Print)	n=0:全件表示，n<0:後ろから-n 件表示
%CP n,m	CSV の n 件目から m 件目までを表示	
%R file	file から読込み (Read)	file:読み込むファイル名
%W file	file への読み出し (Write)	file:ファイル名
%F word	検索結果を表示 (Find)	%P と同じ形式で表示
%S n	CSV の n 番目の項目で整列 (Sort)	表示はしない
%D n	n 番目の項目を消去	
%BW file	file へのバイナリ形式での書き出し	
%BR file	バイナリ形式である file からの読み込み	

表 2: 名簿データ

ID	学校名	設立年月日	所在地	備考データ
32 bit 整数	70 bytes	struct date	70 bytes	任意長

1. 不足機能についての考察を行った．不足している機能として，現段階では誤ったデータを入力した場合に修正する方法が無かったため，入力済みのデータを改変・削除する関数が必要であると考えた．また，ソート後に中央付近にあるデータを探すために指定した範囲のデータを表示する機能が必要であると考えた．
2. エラー処理についての考察をおこなった．例えば，CSV 形式の入力に対して 5 つより多い・又は 5 つ未満の入力が行われたり年月日に int-int-int 以外の入力が行われた場合にエラーメッセージを表示するようにした．また，入力件数が 0 件である状況で%P コマンドを実行した場合に表示できない旨を出力するようにした．また，%W，%R，%BW，%BR コマンド実行時に不適切なファイル名が指定されていたり，データ登録件数が 0 件であるときに%W，%BW コマンドの実行を行った場合にエラーメッセージを出力するようにした．

2 プログラムの作成方針

プログラムをおおよそ以下の部分から構成することにした．それぞれについて作成方針を立てる．

1. 必要なデータ構造の宣言部（2.1 節）
2. 標準入力から得た CSV データの解析部（2.2 節）
3. 構文解析したデータの内部形式への変換部（2.3 節）
4. 各種コマンド実現部（2.4 節）

2.1 宣言部

“宣言部” は構造体や構造体に関わる変数を宣言する部分である．このレポートでは概要で示した表 2 に基づいて，以下のように宣言する．

```
int profile_data_nitems = 0
struct date {
  int y;
  int m;
  int d;
};

struct person {
  int id;
  char school[70];
  struct date birth;
  char place[70];
  char *sub;
};

struct profile profile_data_store[10000];
```

ここで，構造体 date については年月日，構造体 person については表 2 に従って各変数を宣言している．また，profile_data_nitems については構造体配列 profile_data_store に登録されているデータの件数としている．

2.2 解析部

“解析部” は標準入力から 1 行分を読み込み，適切な処理をおこなう箇所である．しかし，このままでは，処理が曖昧であるため，具体的に考える必要がある．そこで，段階的詳細化の考え方に基づいてさらなる詳細化をおこない，下記の (a) から (e) のように分割することにする．

(a) 標準入力から読むべき行が残っている間，文字の配列 char line[] に 1 行分を読み込む．

(b) line の 1 文字目が，’%’ ならば，2,3 文字目をコマンド名，4 文字目以降をその引数として，決定されたコマンドを実行する．

(c) さもなくば line を CSV とみなし’,’ を区切りとして 5 つの文字列に分割する．また，分割してできた 5 つの文字列のうち年月日の部分の文字配列を’-’ を区切り文字として 3 つの文字列に分割する．

(d) 分割したデータを適切な型に変換し構造体に代入する．

(e) 次の行を読み込む．

ここで，(b) は swhich 文で書くことでさらに詳細化することもできるが，今回は後述の関数で実装することとする．また，分割した文字列の一部は整数型として処理するため，解析部に続く変換部では構造体で宣言された型に注意する必要がある．

2.3 変換部

“変換部” は分割された CSV データを項目毎に型変換し，対応する構造体メンバに代入する部分である．メンバとして様々な型を用いているため，適切な代入の使い分けが必要となる．

文字列は関数 strcpy を用いて代入する．数値の場合，関数 atoi を用いて文字列を整数型に変換してから代入する．構造体 struct date であるメンバー birth については関数 split で分割してから代入する．

なお，構造体への代入については，アロー演算子を用いることで容易に実装することができる．例えば，"2014-10-25"のような文字列を分割し，関数 atoi によって整数型に変換しつつ格納する

という処理は，CSV 入力を整数型で代入する処理と同じ処理である．従って，区切り文字が CSV の’,’ とは異なる’-’ になること以外は同様に記述できるはずである．

また，解析部から与えられた文字列は揮発性であることにも注意する．つまり，変換部で文字列を処理する際にはポインタを参照するのではなく，関数 strcpy を使って文字列のコピーを行わなければならないことに気をつける必要がある．

2.4 各種コマンド実現部

“各種コマンド実現部” は解析部で得られたコマンドの実際の処理をおこなう部分である．このレポートでは，具体的にはコマンド%Q，%C，%P，%CP，%R，%W，%F，%S，%D，%BW，%BR を実装している．

終了（%Q）は exit で終了すればよい．登録件数の表示（%C）はグローバル変数 profile_data_nitems を printf で表示すればよい．表示（%P n）は printf で各項目毎に表示すればよい．ただし，n が負であるときは後ろから $-n$ 件を正順で表示することに注意が必要である．指定したデータの表示（%CP n,m）は%P と同様に行えばよい．データの読み込み（%R file），書き出し（%W file）はファイルポインタを用いて標準入力からの入力や標準入力への出力と同様に行えばよい．検索（%F word）は構造体に含まれる int 型の整数を sprintf 関数を用いて文字列と見なし，入力された文字列と比較すればよい．ソート（%S n）はクイックソートを用いて行う．また，バイナリ形式での入出力（%BR file，%BW file）は fwrite，fread 関数を応用して行えばよい．この実装に関する方針決定の詳細は後の 3.5 節で説明する．

3 プログラムおよびその説明

プログラムリストは 8 節に添付している．プログラムは全部で 734 行からなる．以下では，前節の作成方針における分類に基づいて，プログラムの主な構造について説明する．

3.1 汎用的な関数の宣言（64 行目から 119 行目）

まず，汎用的な関数として，func_max 関数，func_min 関数をそれぞれ 64–69 行目，72–78 行目で定義し，subst() 関数を 80–91 行目で定義している．また，split() 関数を 93–109 行目，get_line 関数を 111–119 行目で定義している．

func_max 関数は与えられた 2 数のうちで大きいものを，func_min 関数は小さいものを返す．

subst は，str が指す文字列中の c1 文字を c2 に置き換える．プログラム中では，get_line で得られた文字列の改行文字を終端文字で置き換えるためや，文字列を分割する時に区切り文字終端文字に置き換えるために使用している．

split は，str が指す文字列を区切文字 sep で分割し，分割した各々の文字列を指す複数のポインタからなる配列を返す関数である．プログラム中では，CSV を’,’ で分割し，分割後の各文字列を返すのに使用されている．また，“2004-05-10” のような日付を表す文字列を ‘-’ で分割して，struct date を生成する際にも使用している．

get_line は，標準入力から 1 行読み込み line に代入する関数である．プログラム中では，使用者からの入力を受け付けるために使用している．

3.2 構造体を扱う関数（122 行目から 169 行目）

6–10 行目は構造体 struct date の宣言部である．
12–18 行目は構造体 struct profile の宣言部である．
122–169 行目は構造体 struct profile に新規データを格納する部分である．

3.3 mian 関数部（52 行目から 60 行目）

52–60 行目は，main() 関数であり，get_line 関数によって標準入力から 1 行読み込んだ値を parse_line 関数に渡し適切な処理を行う動作を繰り返す．

3.4 各種コマンドに対応する関数を呼び出す関数（171 行目から 219 行目）

171–219 行目は，%P，%S 等のコマンドを解釈して適切な関数を呼び出すための関数である．なお，各関数は cmd_から始まる名前で定義されている．

3.5 各種コマンド部（233 行目から 430 行目）

233–430 行目は，標準入力で%P，%S 等のコマンドが入力されたときに呼び出される関数群である．

cmd_quit 関数は exit 関数を用いてプログラムを終了する．

cmd_check 関数，cmd_print 関数，cmd_complex_print 関数は profile_data_nitems 等を用い表示を行う．

cmd_read 関数は get_line 関数と parse_line 関数を応用し，cmd_write 関数は save_profile 関数を呼び出して処理を行う．

cmd_find 関数は構造体配列の各メンバに対して部分一致するものを探す関数 com_str を用いて検索する．

cmd_sort 関数はクイックソートを行う quick_sort 関数を呼び出してソート行う．

cmd_del 関数は構造体を 1 つずつ前へシフトし，profile_data_nitems を 1 だけ減少させることで実現できる．

cmd_bin_write 関数と com_bin_read 関数については後述する．

3.6 各種コマンドで用いる関数（432 行目から 734 行目）

432–734 行目は，各関数で呼び出される関数群である．

output_profile 関数は指定された 1 件のプロフィールを表示する．

com_str 関数は構造体 profile_data_store の各メンバについて関数 comp_str を用いて部分一致するものがあれば検出する．

comp_str 関数は与えられた 2 つの文字列の部分一致や大小関係について，それぞれ異なる値を返す．

swap_struct 関数は指定された 2 つの構造体を入れ替える．

quick_sort 関数は基準値よりも小さい値と大きい値を両側から comp_str 関数や sum_date 関数を用いて探し，得た値を call_quick 関数に渡す．

call_quick 関数は与えられた値に応じて構造体配列の入れ替えや quick_sort 関数の呼び出しを行う．

sum_date 関数は構造体 date に含まれる 3 つの値を大小関係が比較できるように 1 つの値として返す．

save_profile 関数は output_profile 関数と同じような要領でファイルに csv 形式で出力する．

output_10_to_16 関数は fwrite 関数と sum_date 関数を用いてバイナリ形式のデータをファイルに書き出す．

read_bin 関数は fread 関数を用いてバイナリ形式のデータを読み込む．

4 プログラムの使用法

本プログラムは名簿データを管理するためのプログラムである．CSV 形式のデータと % で始まるコマンドを標準入力から受け付け，処理結果を標準出力に出力する．入力形式の詳細については，概要の節を参照のこと．

プログラムは，Red Hat Linux 3.2.2-5 で動作を確認しているが，一般的な UNIX で動作することを意図している．gcc でコンパイルした後，標準入力から入力を与える．

```
% gcc -Wall -o program1 program1.c
% ./program1
```

プログラムの出力結果としては CSV データの各項目を読みやすい形式で出力する．例えば，下記のような入力に対して，

```
46,The Bridge,1845-11-2,14 Seafield,SEN Unit2.0 Open
27,Bower School,1908-1-19,Bowermadden Bower,Caithness 25 2.6 Open
%C
%P 2
%S 2
%W
%R output.csv
%S 1
%BW
%Q
```

以下のような出力を得る．

```
2
Id    : 46
Name  : The Bridge
Birth : 1845-11-2
```

```
Addr  : 14 Seafield
Com.  : SEN Unit2.0 Open

Id    : 27
Name  : Bower School
Birth : 1908-1-19
Addr  : Bowermadden Bower
Com.  : Caithness 25 2.6 Open

Saved : output:csv
Saved : bin_output.dat
```

また，output.csv，bin_output.dat にそれぞれの項目でソートされた csv データ，バイナリデータが保存されている．入力中の%C は，入力データの件数を表示することを示し，%P 2 は入力した項目を前から 2 件表示することを示している．%S 2 は 2 番目のデータでソートすることを示し，%W は，現在保持しているデータを csv 形式で保存することを示している．また，%R output.csv は output.csv というファイルから csv 形式のデータを読み込むことを示し．%S 1 は再度 1 番目のデータでソートし直すことを示し，%BW は現在のデータをバイナリデータとして出力することを示している．%Q はプログラムを終了することを示している．

5　作成過程における考察

2 節で述べた実装方針に基づいて，3 節ではその実装をおこなった．しかし，実装にあたっては実装方針の再検討が必要になる場合があった．本節では，名簿管理プログラムの作成過程において検討した内容，および，考察した内容について述べる．

5.1　split 関数についての考察

split 関数については方針通りに実装することができたが，示された例とは異なる形となった．これにより分割数以上の要素を持つ文字列に対して余計な処理を行う必要が無くなったと考えられる．

5.2　標準入力についての考察

標準入力からの不適切な CSV 入力に対して，エラーメッセージを出力しない場合%C や%P オプションが実行されたときに不都合が生じる可能性がある．なぜなら，構造体に NULL 等の値が代入されることになるからである．したがって今回は CSV 入力の分割数だけでなく，年月日に数字以外の文字が含まれている場合についてもエラーメッセージを出力し CSV 入力を構造体に代入しない処理を行うことにした．

5.3　parse_line 関数についての考察

parse_line 関数については拡張性を考慮し，標準入力の 1，2 文字目をコマンド，4 文字目以降をコマンドの引数という仕様に変更した，

5.4 cmd_find 関数についての考察

検索機能の実装において，部分一致と完全一致の 2 通りが考えられる．今回は sample.csv 等において，地名等の単語で検索きるという利点から部分一致を選択したが，オプションで部分一致と完全一致を選択できるようにしても良いと考えられる．

6 結果に関する考察

演習課題のプログラムについて仕様と要件をいずれも満たしていることをプログラムの説明および使用法における実行結果例によって示した．ここでは，概要で挙げた以下の項目について考察を述べる．

1. 不足機能についての考察
2. 既存コマンドの改良
3. コマンドの拡張
4. エラー処理についての考察
5. 新規コマンドの実装
6. テキスト形式とバイナリ形式

6.1 不足機能についての考察

概要で示したように，以下のような機能を実装することでより実用的なものになると考えられる

(1) 入力済みである CSV データを改変する機能

現段階のプログラムでは，誤った入力を修正する手段が無く CSV 入力においてタイプミス等を行った場合，最初からやり直しとなる．また，後にファイルへの入出力機能を実装することからも既存のデータの改変機能は実装したほうがよいと考えられる．この機能の実装にあたって，該当データが前から何件目であるかを引数としたり，後に実装する%F コマンドで検索した結果から改変するデータを選択するようにするなどいくつかのやり方が考えられる．そのため，目的のデータの選択方法については複数用意することも視野に入れるべきであると考える．

(2) 入力済みであるデータを削除する機能

(1) と同様に，仕様の段階では誤って入力したデータを削除する機能も存在しない．したがって，複数の方法でデータを選択できるようにした上でデータを削除する機能を実装すべきであると考え実装を行った．この機能についても，検索コマンドの結果から削除できるようにしたり複数件のデータをまとめて削除できるようにするなどの改善の余地があると考える．

(3) 関数の扱い方を表示する機能

関数の呼び出しが 2 文字まで対応するようになったため，各関数の一覧を表示する機能が必要であると考える．また，その際必要に応じて各関数の引数や使用例等を表示するようにすべきであると考える．

6.2 既存コマンドの改良

現在のコマンドの問題点とその改良案について考察を行った．

(1) %W，%BW コマンドの改良 %W，%BW コマンドは実行結果を外部ファイルとして保存するためのコマンドであり，使用頻度はそれなりに高いと考えられる．したがって，ファイル名が入力されずに実行された場合に自動的にファイル名を設定することとした．実装方法は与えられた引数である文字列が終端文字のみであった場合にそれを自動的に予め設定しておいたファイル名で上書きするだけで実装できる．

(2) %P コマンドの改良 %P コマンドを用いて多くのデータを表示したときに，使用者が見ているデータが何件目であるかということは削除機能等を仕様する上で表示されていたほうが使いやすいと思われる．これは output_profile 関数に i の値を表示するようにすれば実装可能である．

6.3 コマンドの拡張

コマンドの拡張性を考慮し，2 文字のコマンドを受け付ける機能の実装を行った．

(1) 実装のために検討した内容
実装にあたりコマンド後のスペースの数について 1 個のみを受け付ける，コマンドと合わせて 2 文字にする，空白をいくらでも受け付ける等が考えられる．今回はオプションが高々2 個であるのでコマンドと合わせて 2 文字にするものを採用した．オプションを多くとるコマンドの実装が必要になれば，空白をいくらでも受け付けるように変更することも考えたい．

(2) 実装手順
コマンドの 1 文字目の実装と同様に，その次の文字を exec_command 関数の引数とした．また，拡張性を考え switch でコマンドの 1 文字目を判別した後にもう一度 switch でコマンドの 2 文字目を判別し適切な関数を呼び出すようにした．switch を用いることにより if を用いた場合と異なり後の関数の追加が容易になると考えられる．

(3) 実装結果
実際に実装した関数に関しては，後述の新規コマンドの実装 6.6 にて詳細を述べる．

6.4 エラー処理についての考察

本講義で制作した名簿管理プログラムは，CSV データが格納される構造体は 1 節で示した通り 5 つの要素から成り立っている．したがって，不適切な型の入力や不正な要素数の入力を放置するとエラーが出る恐れがある．ここでは，これらのエラーをどのようにしてエラーメッセージとして出力するかについて考察する．

6.4.1 標準入力から取得した CSV データ処理中のエラー処理

CSV データ中に，不正なデータが含まれていた場合の処理について考察する．エラーが含まれていた場合は，以下のような対処が考えられる．

(1) エラーのある場所を指摘して，無視する

この方法は，連続したデータの入力に対して不適切なデータを表示しそのデータを保存せず復帰するため，通常はこの方法が好ましい．しかし，エラーのあった状態からの復帰を行う必要があるためプログラムが複雑になる．

(2) エラーのあった行を指摘して，終了する

この方法は，入力中に 1 つのエラーを発見することしかできない．しかし，エラーのあった入力データを無視してしまうと以降のデータ入力の正当性チェックにも影響がでるような場合には，この方法を採らざるを得ないこともある．

エラーのあった行を指摘せず，終了または無視するという方法も考えられるが，正常終了との区別が付かないため実用的でない．

今回は，エラーのある点を指摘して，無視する方法がよいと考えた．実際の処理について以下に示す．まず，取得した CSV データの分割数が 5 でなければエラーメッセージを出力するようにした．また，年月日の分割数が 3 でない場合も同様にエラーメッセージを出力するようにした．最後に，整数型で入力すべき場所に文字列が入力されていたときにエラーメッセージを出力するようにした．具体例な処理としては文字列の 1 文字目が整数であるか，文字列を整数型に変換したときの桁数と元の文字列の文字数が同じであるかどうかを判断基準とした．

6.4.2 `cmd_print` 関数（%P オプション）におけるエラー処理

CSV データの登録件数が 0 件である場合や，登録件数より多い件数のデータの表示が要求された場合の出力について考察する．

(1) CSV データの登録件数が 0 件である場合

入力された表示件数 n については，0 以外の値の場合は (2) に含まれるため考慮しないものとする．このときの出力に関しては，エラーメッセージを表示するというものとコマンドの解釈通り 0 件表示する（何も表示しない）という 2 通りが考えられる．今回は何も表示されないことで次の入力位置が分かりにくくなる，使用者がデータの確認のために%C コマンドを使用する必要がある等の理由から前者のエラーメッセージを表示する方法を採用した．

(2) 登録件数より多い件数のデータの表示が要求された場合

このときの出力としては，エラーメッセージのみを出力する方法，エラーメッセージを出力した上で全件表示する方法，エラーメッセージを出力せずに全件表示する方法が考えられる．エラーメッセージを出力することで使用者が異常な値を入力したことに気付くという利点があるため，今回はエラーメッセージを出力した上で全件表示するという方法を採用することにした．

6.5 `cmd_write` 関数や `cmd_bin_write` 関数におけるエラー処理

引数としてファイル名が与えられなかった場合や，データ件数が 0 件であったときの処理について考察する．

(1) ファイル名が与えられなかった場合

このときの処理としては，エラーを返すということが考えられるが，この場合保存したと勘違いしそのまま終了してしまう危険がある．したがって今回は，それぞれ `output.csv`, `bin_output.dat` というファイル名で保存する方法を採用した．

(2) データ件数が 0 件であった場合

データ件数が 0 件であるときにデータを上書き保存しようとすると，空のファイルで既存のファイルを上書きしてしまう可能性がある．このときの処理として，上書きするかどうかの確認を行う，上書き保存してしまう，エラーメッセージを表示し上書き保存しないといったものがある．今回は誤って既存のファイに上書きしてしまう可能性を減らすために，エラーメッセージを表示し上書き保存を行わない方法を採用することとした．

6.6 新規コマンドの実装

表 1 に示したように，新たに 4 つのコマンドを実装した．以下に各コマンドの詳細を述べる．

(1) %CP 関数

%C 関数の実装の際に，指定範囲のデータの表示を行う機能が必要であると感じ，%CP(Complex Print) 関数の実装を行った．実装の大まかな手順は%P と同じである．また，1 番目の数値が 2 番目の数値よりも大きい場合，表示順を逆にすることで視覚的に分かりやすくした．

(2) %D 関数

不足機能についての考察で行ったように，削除機能が必要と考えたため%D 関数の実装を行った．これは%D n のように入力した際に n 件目のデータを削除するものである．実装方法は単純に指定された位置以降の構造体を 1 つずつ前へずらすだけである．この処理はリスト構造を用いることで高速化できるため，他の関数の仕組みに合わせて構造体配列をリスト型に変更することも考えたい．

(3) %BW，%BR 関数

テキスト形式でない形式でのファイルの入出力を行う関数として，%BW と%BR の実装を行った．これらを実装することの意義や問題点は後述するため割愛する．これらの使用方法は基本的に%W，%R と同じである．これらの関数の実装は fwrite 関数や fread 関数を用いることで簡単に行うことができる．

6.7 テキスト形式とバイナリ形式

%BW コマンドと%BR コマンドによって行われるバイナリ形式のデータの扱いについて考察する．

バイナリ形式でデータを扱うことの長所は，数字等の扱いの際に容量の効率が良くその形式の都合上テキスト形式と比べ処理がやや速いという点である．また，この形式でデータを扱うことにはエンディアンなどの問題点が存在する．これは環境によって変わるものであり，異なる環境での動作を想定するのであれば考慮しなければならないものである．また，この形式のデータを扱う際にはデータの形式を知っている必要があり，使用用途も限られる．本プログラムにおいては，可変長である備考データや各文字列に含まれる終端文字以降の余分な容量の部分で余計な処理を行わない為に各データのサイズを各データの直前に 4 ビットで格納している．ただし，備考データが必ずしも 4 ビットの整数で表せるとは限らないため Windows での .mid ファイルのように各データの大きさも可変長で格納することが望ましいと思われる．

7 感想

本講義では，構造体配列の扱い方や文字列を操作する関数，ファイルポインタの扱い方やプログラムの動作確認の行い方について学ぶことができた．特に，不正な入力に対してエラーメッセー

ジを出力することや想定外の入力を考慮することが昨年度に比べてかなり増え，使用者に配慮したプログラムを制作していることが実感できた．また，仕様や要件が曖昧に設定されているため個人で必要だと思ったコマンドを実装したり想定外の動作に対する処理を行う関数を実装するなどの拡張性もあり，やりがいを感じることができた．ソートの実装については，1 学期に学んだクイックソートを活用することができるなど，学修した実感が得られることも多々あった．いくつか行った考察課題の発展の内容についても，それぞれの内容について理解を深められたと感じた．今学期の講義で得られたこの経験を，今後のプログラム製作にも活かしたいと思う．