プログラミング B レポート
課題 3

担当教員: 武政 淳二

大坪　祐清
学籍番号: 09B25014

2026 年 1 月 1 日

1　課題内容

基本課題および発展課題に取り組んだ.

2　アルゴリズムの説明

以下の手順で, 処理を進める.

1. csv ファイルの 1 行目からファイルの終わりまで以下の処理を繰り返す.
   （a）csv ファイルを読み込み,3 行目,7 行目,8 行目,9 行目を変数に読み込む.
   （b）読み込んだ変数を, 構造体配列に読み込む.
2. (検索 1) と (検索 2) のどちらの検索をしたいのか, あるいはプログラムを終了したいのか, ユーザー/入力ファイルからの入力を受け付ける.
3. (検索 1) あるいは (検索 2) をユーザー/入力ファイルが望んだ場合, 検索したい郵便番号ないしは住所の入力を受けつけ, 変数として受け取る.
4. 入力に応じて条件分岐を行う
5. (検索 1) の場合の処理
   （a）クイックソートを用いて, 郵便番号が昇順になるように構造体配列をソートする.
   （b）ユーザーからの郵便番号の入力と, ソートされた構造体配列をもとに, 二分探索法で目的の住所を探す.
   （c）条件にあったデータがあったら, 郵便番号と住所を出力する.
6. (検索 2) の場合の処理
   （a）n=1 から n=「1 で読み込んだ行数」まで, 以下の処理を繰り返す
      i. n 行目の都道府県名, 市区町村名, 町域名のいずれかに入力の文字列が含まれていたら, 配列に n を追加する.
      ii. 6（a）i の操作でいずれにも, 入力が含まれていなかったかつ,n 行目の住所に入力の文字列が含まれていたら, 配列に n を追加する.
   （b）配列をクイックソートで, 郵便番号が昇順になるようにソートする.
   （c）配列に入っている情報をもとに, 郵便番号と住所を出力する.
   （d）ユーザー/入力ファイルに, 絞り込み検索をしたいか尋ねる.
   （e）絞り込み検索をする場合, 絞り込みたい住所の入力を受けつけ, 変数として受け取る.
   （f）n=1 から n=「6（a）i の配列で追加した数」まで, 以下の処理を繰り返す.
      i. 6（a）i の配列の n 番目の要素の行の都道府県名, 市区町村名, 町域名のいずれかに入力の文字列が含まれていたら,6（a）i の配列とは別の配列に n を追加する.
      ii. 前述のいずれにも, 入力が含まれていなかったかつ,6（a）i の配列の n 番目の要素の行の住所に入力の文字列が含まれていたら,6（f）i の配列に n を追加する.

（g）ユーザー/入力ファイルに絞り込み検索をさらに, したいか尋ねる.

（h）絞り込み検索をしたい場合,6e ～ 6g の操作を同様に繰り返す.

7. いずれかの操作が終わったら, 再び,2 ～ 6 の操作を繰り返す.

3 プログラムの説明

本プログラムは,C 言語にて記述した.

3.1 入力形式

プログラムへの入力は, モードによって, 異なる.

プログラムに引数を与えずに, 実行する場合, インタラクティブモードとなる. この場合, プログラムは実行中に, 検索モードと検索クエリを標準入力から受け取り, 検索を行う.

プログラム実行時に, 検索モードと検索クエリの列を記入した入力ファイル名を引数として指定し, 実行する場合, ファイルモードとなる. この場合, プログラムは検索モードと検索クエリをファイルから読み取り, 検索する.

3.2 出力形式

プログラムの出力は, 検索の結果, 得られた住所である. 出力は,[郵便番号]:[住所] の形式で出力し, 結果が複数ある場合は, 郵便番号順が昇順になるように出力する.

3.3 データ構造

本プログラムのデータ構造を, 表 1 に示す.

表 1: プログラムのデータ構造

データ型	型名	概要
構造型	ADDRESS	CSV の 1 行分のデータを記憶する. 要素は表 2 に示す

3.4 大域変数

本プログラムの大域変数を, 表 3 に示す.

構造型配列 address_data[200000] に郵便番号と住所のデータが入る. データの取り出しは,address_data[16].code のように取り出す. これ以降,address_data に指定する, 配列の添字を index と呼ぶ.

表 2: ADDRESS 型の要素

要素の型	変数名	概要
int	code	郵便番号を記憶する
char 型配列	pref[20]	住所の都道府県名を記憶する.
char 型配列	city[32]	住所の市区町村名を記憶する.
char 型配列	town[116]	住所の町域名を記憶する.

表 3: プログラムの大域変数

型名	変数名	概要
ADDRESS	address_data[200000]	CSV の全行のデータを ADDRESS 型配列として記憶する.
int	mode	検索モードを記憶する.
int	refine_flag	絞り込み検索をするかどうか記憶する.
char	query[200]	検索クエリ (郵便番号または住所) を記憶する
long	total_count	何行データを読み込んだかを記憶する.
int	hit_index_list[200000]	住所検索で何行目がヒットしたかを記憶する.
int	hit_list_count	住所検索でヒットした回数を記憶する.

3.5 関数の設計

設計した各関数を表 4 に示す.

表 4: 設計した関数

関数名	引数	返り値	処理内容
main	int , int **	int	コマンドラインから, 引数を受け取り, 各関数に渡す
init	引数なし	返り値なし	構造体配列 address_data[2000] の初期化を行う
scan	引数なし	返り値なし	住所データを CSV から読み込み, 構造体配列に記憶させる.
run_from_file	const char *	返り値なし	コマンド、検索対象の郵便番号、 または住所の列をテキストファイル から受け取り、コード検索を行う。
respond	引数なし	返り値なし	クエリを読み込み, 応答する

input	引数なし	返り値なし	検索モードとクエリの入力を読み込む
code_search	引数なし	返り値なし	郵便番号によって住所を検索し, 検索結果を出力する
comp_adr	const void * , const void *	int	q_sort 関数の比較関数. 第 1 引数の index の郵便番号が, 第 2 引数の index の郵便番号より 大きかったら, 自然数を返す.
binary_search	int , int , int , ADDRESS *	返り値なし	第 1 引数に指定された値を, 第 4 引数に指定された ADDRESS の index=left から index=right まで二分探索法で探す
search_around	int,int *, int *, ADDRESS *	返り値なし	第 1 引数に指定された, index の周囲を探索し, index の郵便番号と同じ値を探す. 返り値として, 引数に指定されたアドレスに 同じ郵便番号の始点と終点を返す
address_search	引数なし	返り値なし	クエリの住所を探索し, 検索結果を出力する.
search	int, int *	返り値なし	(第 1 引数) 行目のデータと クエリを比較し, 条件に合致したら, 第 2 引数の値に 1 を足して返す.
comp_code	const void *, const void *	int	q_sort 関数の比較関数. 第 1 引数の index の郵便番号が, 第 2 引数の index の郵便番号より 大きかったら, 自然数を返す.
re_input	引数なし	返り値なし	絞り込み検索をするか, ユーザーにたずねる. 絞り込む場合, 絞り込み検索の内容を query に入れる.
refinement	引数なし	返り値なし	絞り込み検索を実施し, 検索結果を出力する

関数の呼び出し関係を図 1 に示す.

main 関数から処理が始まり, まず init 関数で構造体の初期化が行われる. 次に, 関数に引

数にあるかどうかで処理が分かれる．引数がある場合は run_from_file 関数が実施される．run_from_file 関数での呼び出し関係を図 2 に示す．引数がない場合,respond 関数が呼び出される.respond 関数での呼び出し関係を図 3 に示す.

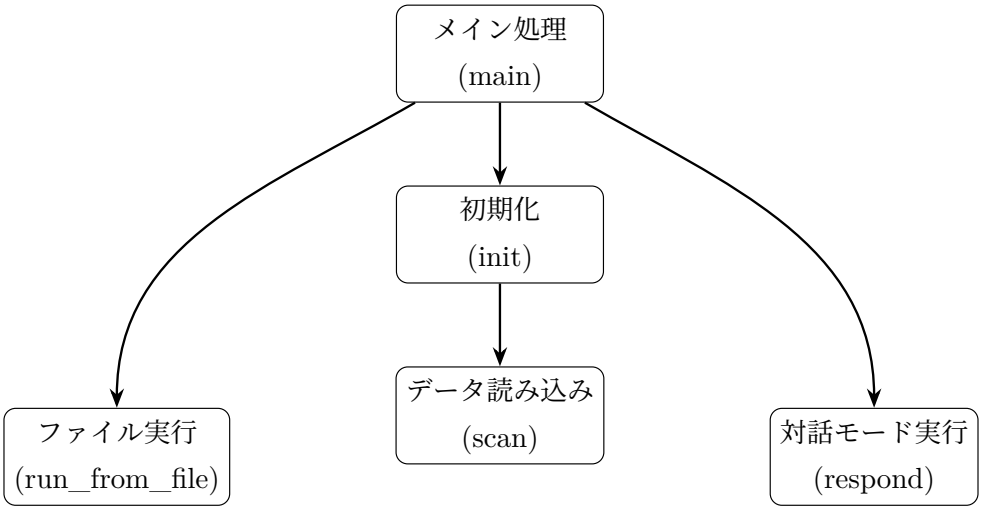

図 1: 全体の関数呼び出し関係図

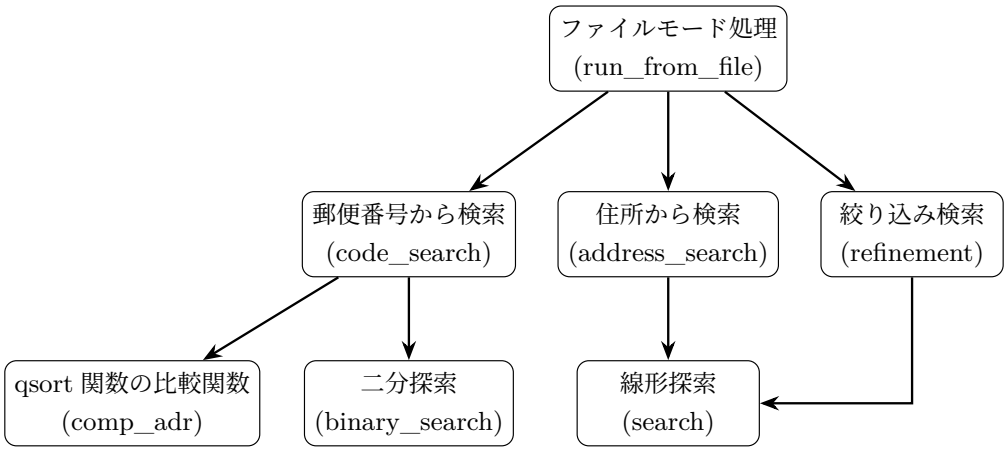

図 2: ファイル実行モードにおける呼び出し

3.6 処理の流れ・意図

プログラムの処理は, 以下の流れで実行される．() 内には,「2 アルゴリズムの説明」で対応する, 順序番号を示す.

まず,init 関数が実行される．そして,init 関数から scan 関数が呼び出される．scan 関数では, マクロ命令で定義された DATEFILE から CSV ファイルを読み込み, 構造体配列 address_data に

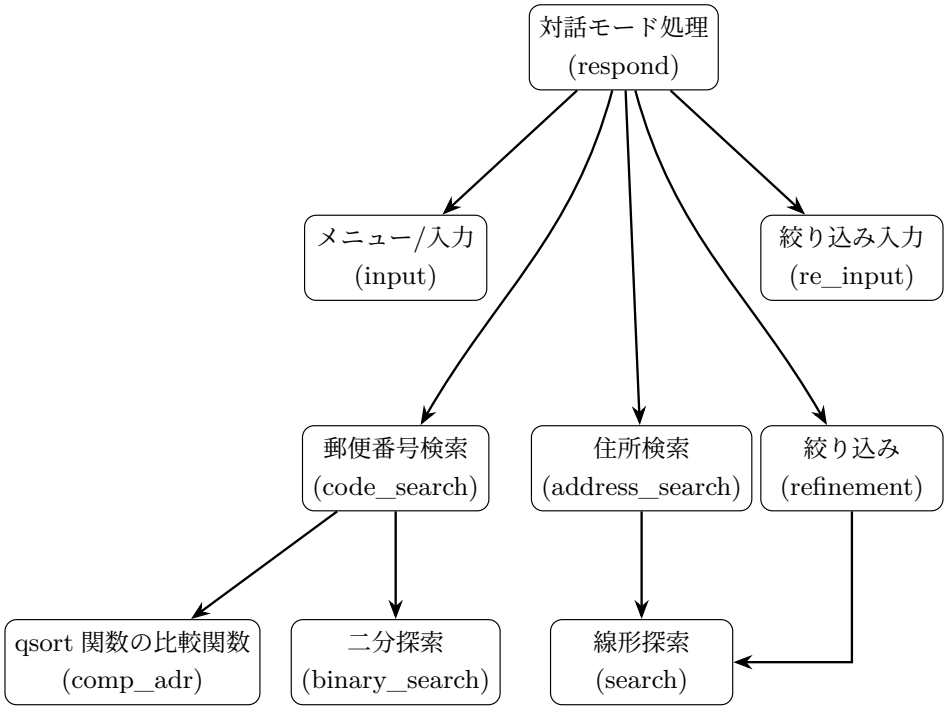

図 3: 対話モード（respond）における呼び出し

読み込む.(1)

その後,main 関数に処理が戻り, プログラム実行時に渡された引数の数で処理が分岐する．引数が 1 つ以上渡されていた場合には,run_from_file 関数が呼び出され, そうでない場合には,respond 関数が呼び出される．

run_from_file 関数では, ファイルの検索モードに応じて,code_search 関数,address_search 関数が呼ばれる．(2, 3,4)

respond 関数では, まず,input 関数が呼び出される．input 関数では, ユーザーの望みの検索モードと検索するクエリが, 変数 mode,query のそれぞれに入る．その後、mode に応じて,code_search 関数,address_search 関数が呼ばれる．(2, 3,4)

code_search 関数では, まず,address_data の要素のポインタを static ポインタ配列の address_search に読み込む．新しく, ポインタ配列を作ることで, もとの address_data の情報を残しておくと同時に, この後行うクイックソートの処理を軽くしている．クイックソートを実行した後,static 変数 isSorted に 1 を入れている．これにより,1 回目の code_search 関数の実行が終わった後に, 再度,code_search 関数が呼ばれた際に,1 回目のソートされた static ポインタ配列を再利用できるようにしている．(5a)

その後, 二分探索を binary_search 関数で実行し, クエリに一致するデータがあったら,search_around 関数を実行する．search_around 関数は,1 つの郵便番号に対して複数の住所が割り当てられている場合のための関数である．search_around 関数の実行後, 結果を出力する．(5b, 5c)

address_search 関数では, 読み込んだファイルの行数だけ,search 関数を実行する. search 関数では, 先に都道府県, 市区町村名, 町域名にクエリが含まれているかどうかを確認してる. これは, クエリが「大阪」のように短い場合, その後の snprintf 関数による, 文字列結合という重い処理を回避するためである. (6 (a) i, 6 (a) ii)

その後, クエリに一致したデータをクイックソートで郵便番号が昇順になるように並び替え, 出力する. (6b,6c)

4　工夫した点

5　考察

6　感想

7　作業工程

付録 A　プログラムリスト