プログラミング演習I第13回レポート

学籍番号：2364902

名前：キム　ギュソク

(1) 課題番号と課題内容

A. 基本課題２

i. ファイルからデータを読み込みながら線形リストを生成し、登録番号を入力するとその選手をリストから削除し、削除後のリストの内容を表示する。該当者がいなければその旨を出力する。登録番号に0が入力されるまで、繰り返し削除ができるようにする。

(2) フローチャートまたは疑似言語によるアルゴリズムの記述

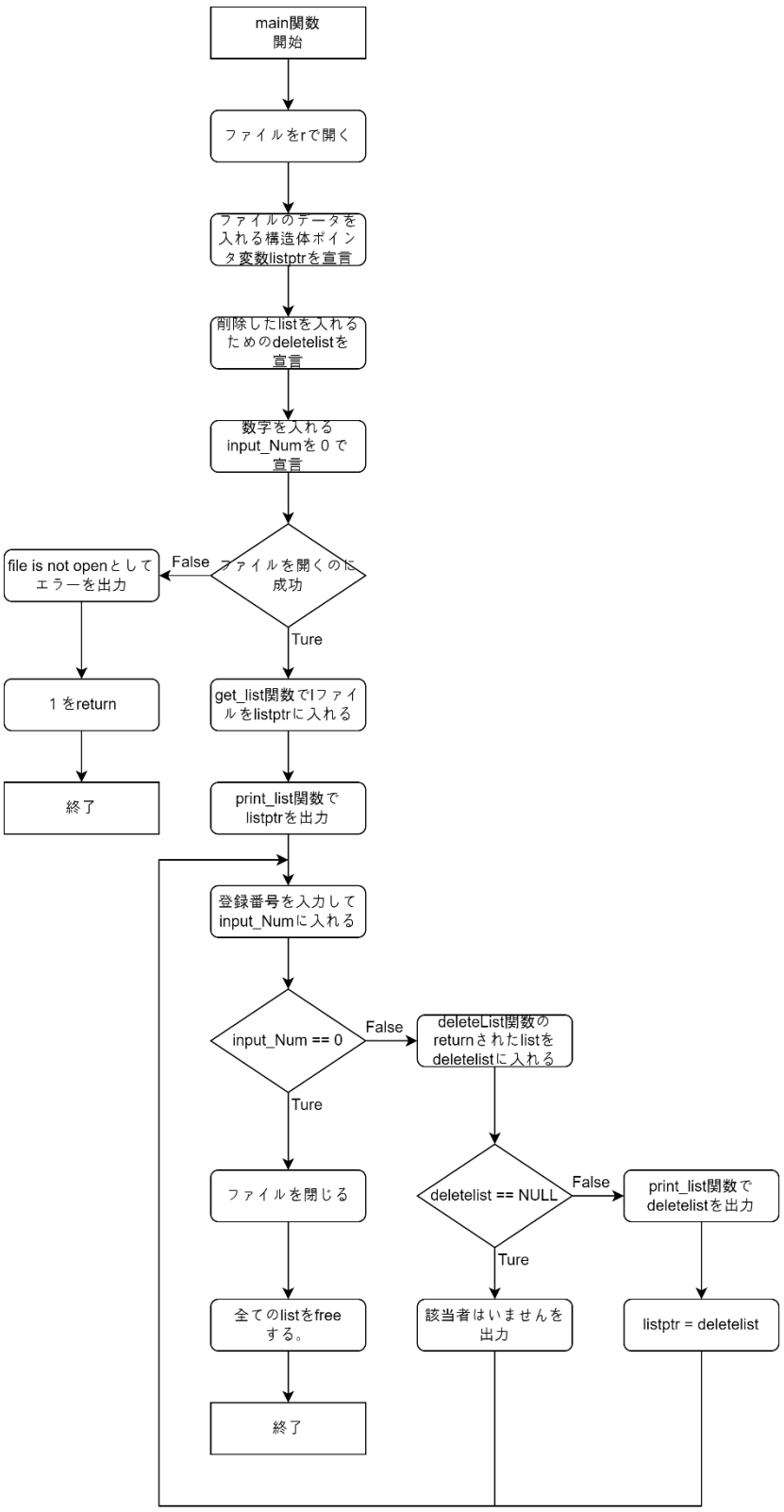


図 1。main関数

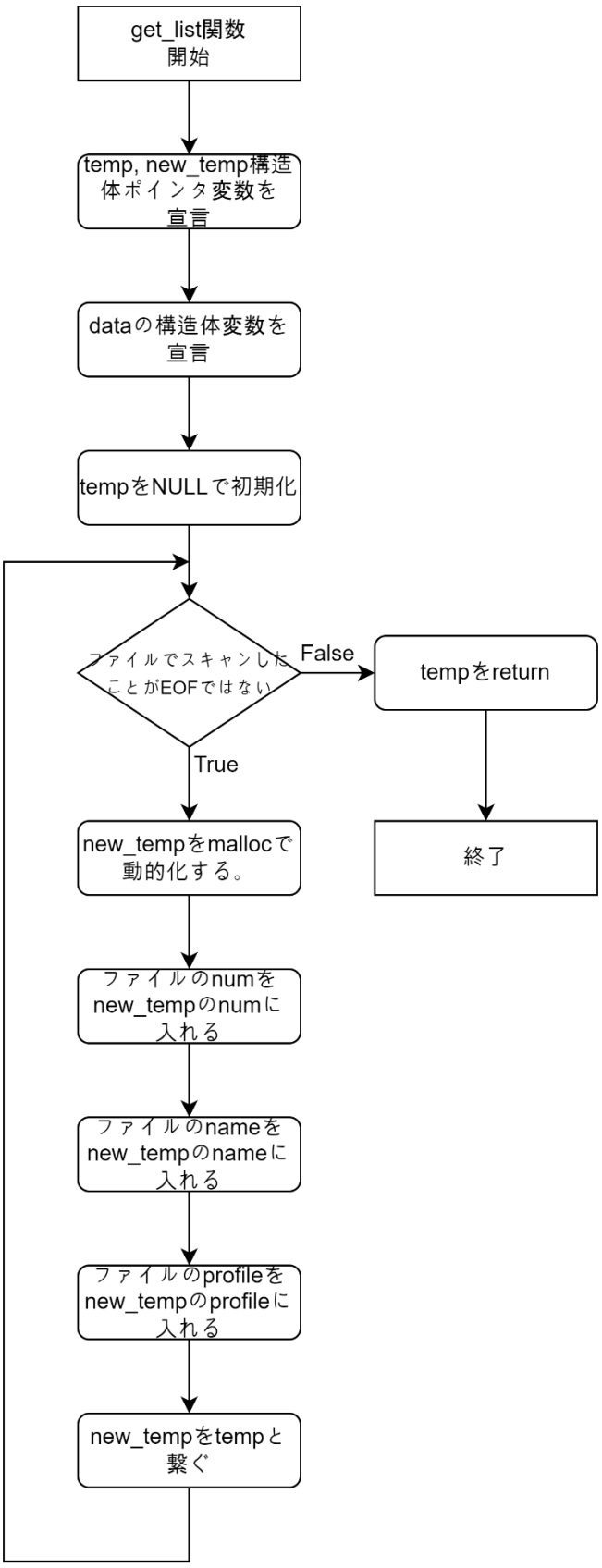


図 2。get_list関数

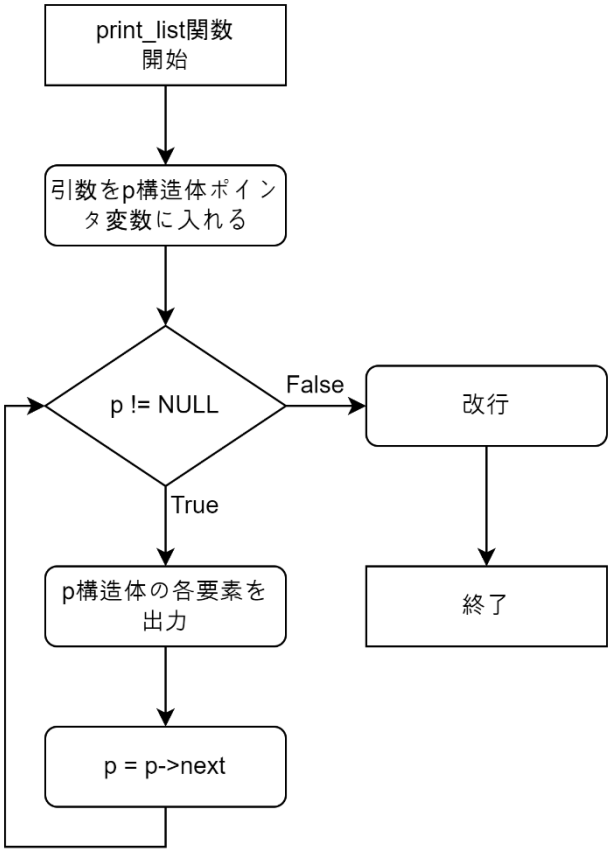


図 3。print_list関数

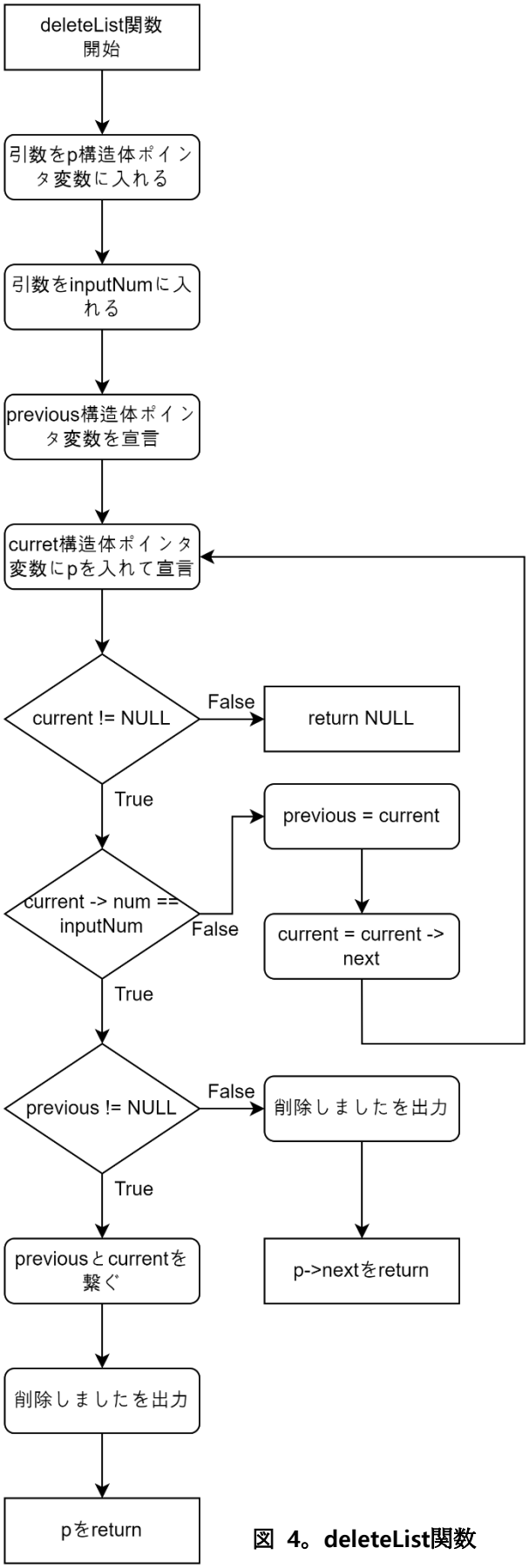


図 4。deleteList関数

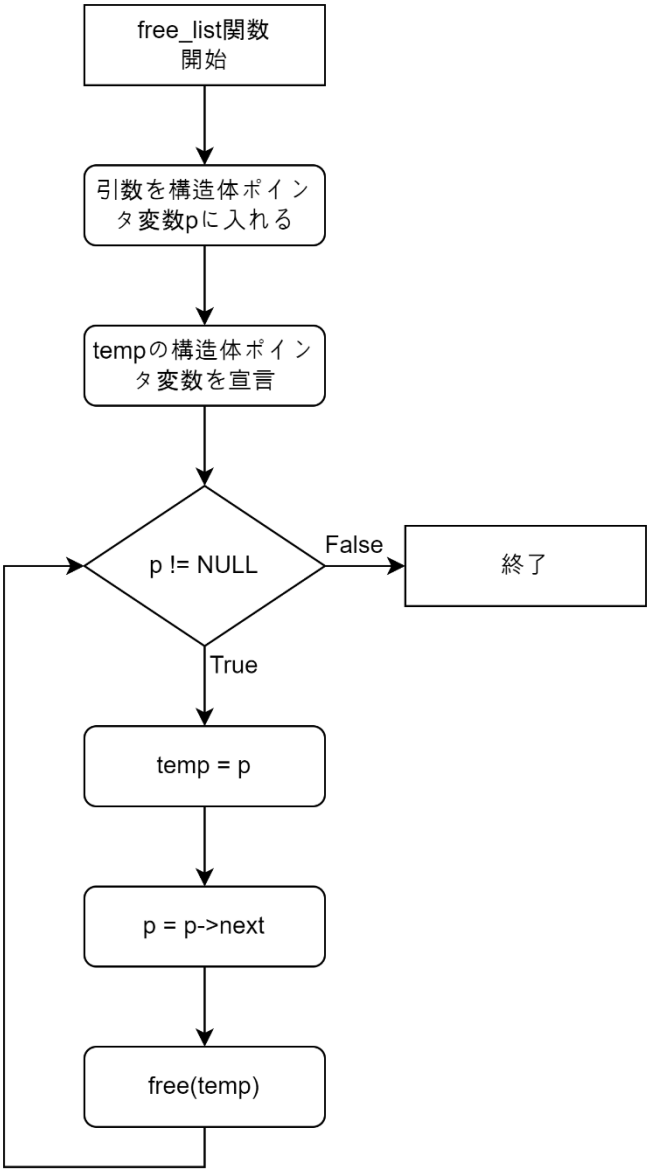
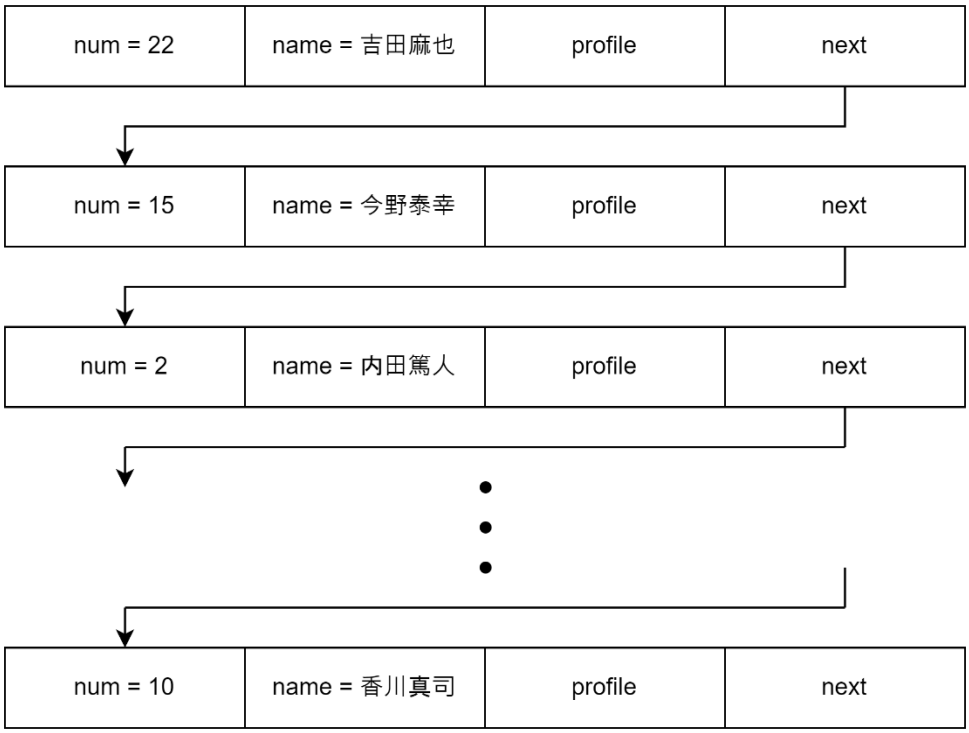


図 5。free_list関数

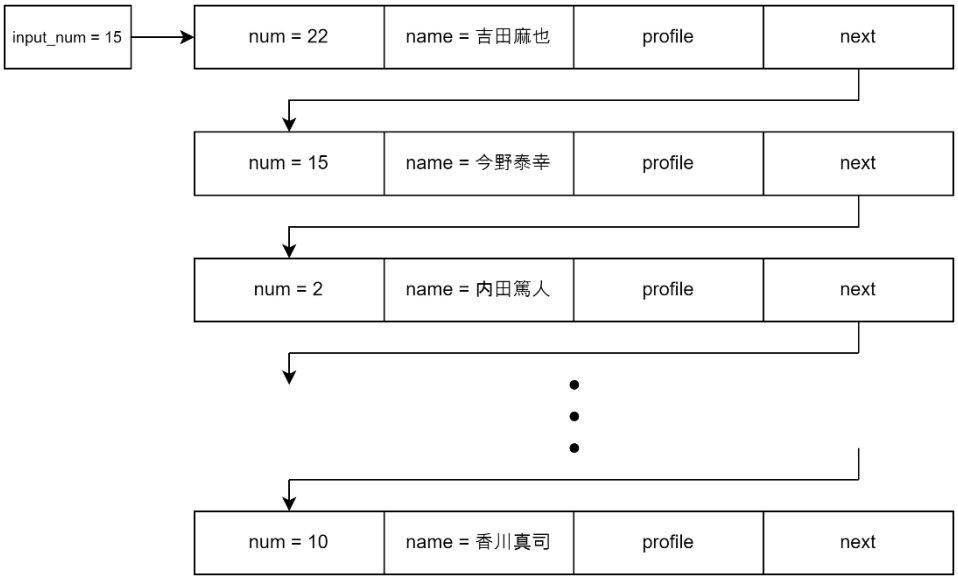
(3) アルゴリズムが正しいことの説明

A. 正当性

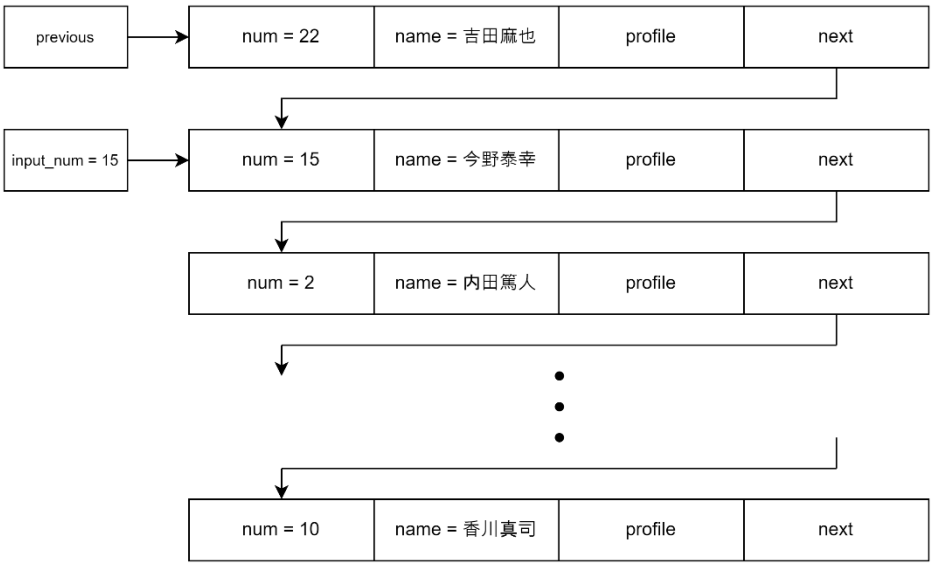
i. ファイルから読み取ったデータは次の図のようにリストで保存されている



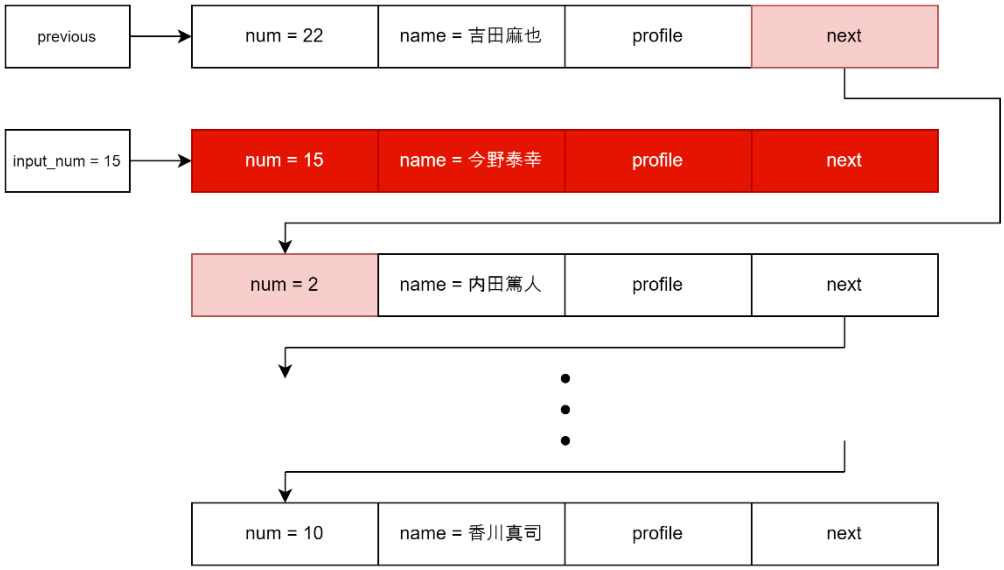
このアルゴリズムでは上の22番のリストから入力された数字とnumが一致するかを確認する。もし、入力された数字が15である場合は次のように作動する。



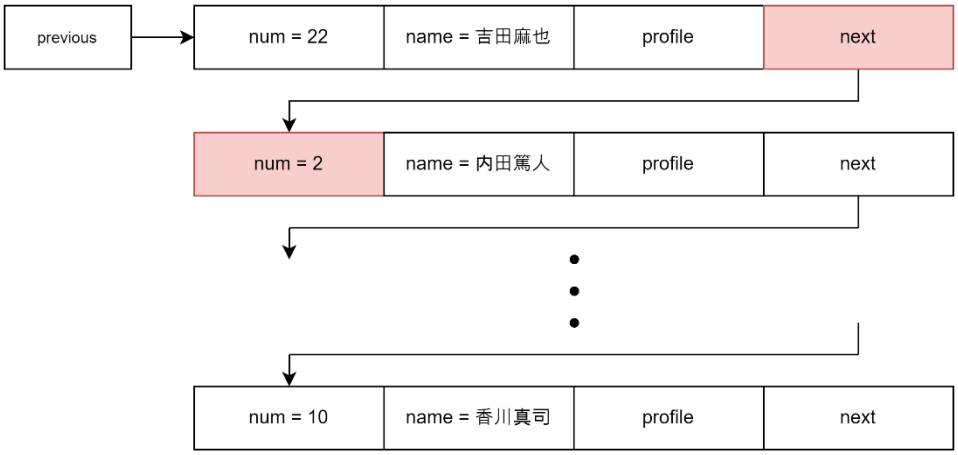
繰り返し文を利用して一番最初のリストから順番に数字が同じであるか確認する。そして、違う場合は次のリストに移動する。



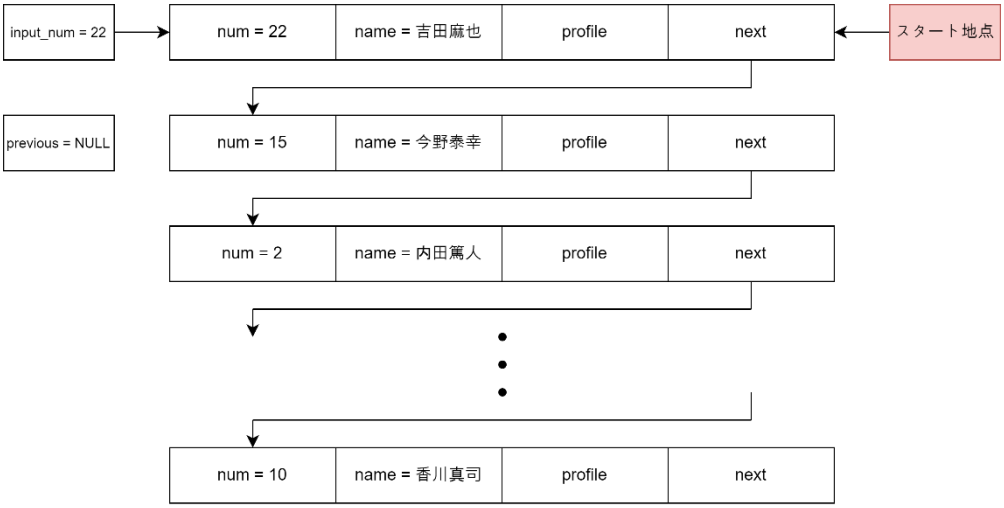
移動して入力された数字と同じ数字が出たときにはそのリストを削除するために前のリストのnextと次のリストを繋ぐ必要がある。そのためにpreviousが示しているリストのnextに現在のリストのnextを入れる必要がある。



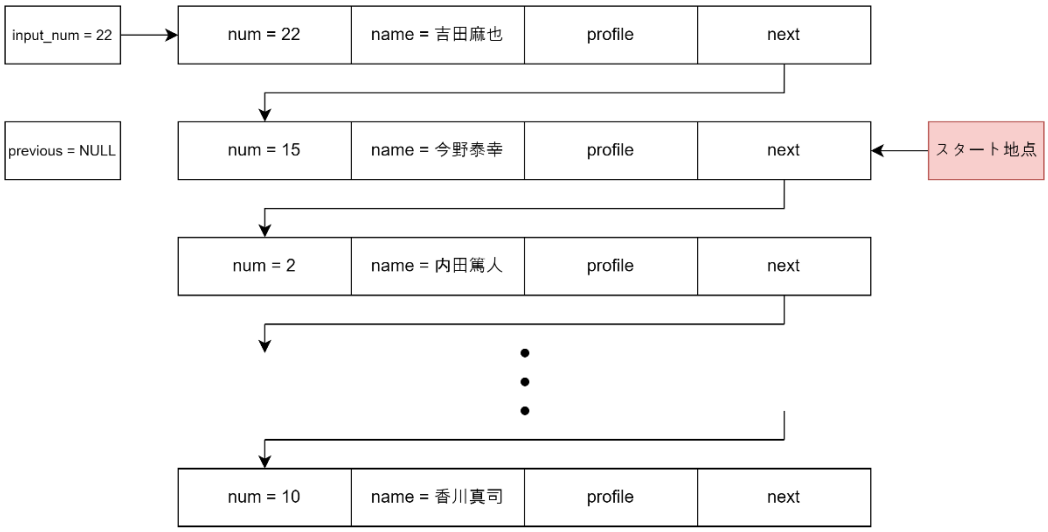
その後、15の繋がりが切れている状態のリストをアップデートすることで入力された数字の15のデータを削除することができる。



しかし、この方法では最初のデータを削除したいときに問題が生じるので最初のリストを削除するための条件をつけることが必要である。よって、最初のリストの場合previousがNULLであるためもし数字が見つかった時にpreviousの値がNULLの場合リストのスタート地点であるreturn値を二つ目のリストにすることで一番目のリストを削除することができる。



スタート地点がnum22のリストであるが



num15のリストに移ることで最初のリストを削除することができる。よって、このアルゴリズム通りでは入力された数字のリストを削除することが出来ると考えられる。

B. 停止性

i. このアルゴリズムで使われるループはmain関数で使われる 0 を入力するまで繰り返すループと、ファイルが終わるまで繰り返して読み取るループ、リストの全ての要素を出力するためのprint_list関数のループ、入力された数字と同じ数字を探すまで繰り返すdeleteList関数の中のループ、そして最後に全てのリストをfreeするためのfree_list関数の中のループが存在する。全てのループは決められた条件があるためその条件を満たすときに停止するように設計されている。例えば、Userが 0 を入力したときに停止するように条件を付けるなど、ファイルの最後であるEOFを読み取った場合繰り返しを停止するなど、正しい条件を立てることで正しく停止する。

(4) ソース・プログラムの説明

A. ソースコードの説明

i. 基本ライブラリであるstdio.hと文字列をコピーするためのstring.hライブラリ、そして動的メモリー活用であるmallocを使うためのstdlib.hライブラリをincludeする。そして、読み取るデータを入れるための構造体struct listを番号、名前、プロフィール、次のリストの住所をで宣言する。各関数のプロトタイプを宣言してmain関数でフ

ファイルポイントfpを宣言して決められたデータファイルをrタイプで開く。その後、出力を行うためのlistptr構造体ポインタ変数とdeletelist構造体ポインタ変数を宣言する。もし、ファイルを開くことに失敗したときにエラーメッセージを出力して終了する。もし、成功すればget_list関数にfpを入れて呼び出して返り値をlistptrに入れる。そして、print_list関数を利用してlistptrを出力する。その後、反復処理を利用して 0 を入力するまで繰り返して数字を入力してもらう。入力された数字は予め宣言したinput_Numに入れてその番号のリストを削除するためにdeliteListに数字とlistptrを入れて返り値をdeletelistに入れる。もし、返り値がNULLであるときには該当する番号がないのでその旨を出力する。もし、ある時は返り値のスタート地点のリスト住所を利用してprint_listで出力する。その後、listptrにdeletelistを入れることでlistptrをアップデートする。Userが 0 を入力するとbreakを利用してループを脱出して、ファイルを閉じて、全てのリストをfreeする。get_list関数では引数としてもらったファイルのポインタをfpに入れてtempとnew_tempを利用して読み取ったデータをnew_tempに入れてそれをtempに順番的に繋げることでデータをリストに入れることができる。そして、print_list関数はpがNULLになるまで次のポインタに移りながら要素を出力することで全てのリストの要素を出力することができる。deleteList関数の場合前のリストの位置を確認するためのprevious構造体ポインタ変数を宣言してfor文を利用して入力された数字と同じ番号を持っているリストを探す。見つかった場合前のリストと次のリストをつなぐことで現在のリストを削除する。もし、previousがNULLの場合は2番目のリストの住所をreturnする。その後、previousにcurrentを入れて繰り返す。全てのリストを確認して出なかった場合はNULLをreturnして終了。free_list関数の場合引数としてもらったリストを全てのfreeする関数である。

(5) 考察

A. ここで動的リストを利用することで得られる利点はたくさんあると思われる。今回のようにファイルからデータを読み取るときにはファイルのデータの数がどれだけあるかわからないので既存の方法では構造体配列を利用して最大のデータ数で配列を宣言してデータを読み取った。しかし、この方法ではデータの数が少ないときに無駄に使われているメモリー空間が多くなるためメモリー活用面で効率的ではない。よって、ファイルのデータの数によって変わることができる動的リストは現在のファイルから読み取ったデータを前回読み取ったデータに繋げることでファイルのデータ数にぴったりなリストを作ることができる。よって、無駄に使われるメモリー空間が無くなり効率的にメモリーを活用することができる。また、特定のリストを削除するときにも配列を利用するときにはその空間に 0 やNULLなどを入れて存在しない空間のように使われている。しかし、この方法ではメモリー空間は存在したままなので削除するたびに無駄なメモリー空間が生じる問題がある。それに比べて、動的リストの場合削除するときに前のリストと次のリストを繋げて現在のリストを解除することで削除されたメモリー空間はまた別の作業で使われるようになるので無駄なメモリー空間が生じない。よって、今回のような決められていないデータを利用する場合は動的リストを利用することがメモリー活用の面で効率的であると考える。

(6) 感想

A. 動的メモリー活用の利点について考えるようになる良い経験でした。また、リストの利点が動的メモリー活用だけではなく削除するときにも無駄なメモリーを生み出さない利点もあることに気づいた。