

名簿管理プログラム

—独自コマンドを添えて—

氏名: 中嶋空偉 (NAKAJIMA, Sorai)
学生番号: 09B23549

出題日: 2024年6月12日
提出日: 2024年7月14日
締切日: 2024年7月24日

1 概要

構造体と配列を用いて、大量のデータを扱うC言語プログラムの基本を学ぶ。その過程で、標準入力から「ID, 氏名, 誕生日, 住所, 備考」からなる情報を管理する名簿管理プログラムを作成する。このプログラムでは、上の形式で入力するとデータが保存される。また、%から始まるコマンドを入力すると、内容を表示、プログラムの停止、保存数の参照、探索、ファイルへの書き込み、読み込み、データの整列などができる。

本レポートでは、探索、ファイルへの書き込み、読み込み、データの整列のコマンドを実装していく。

演習中に取り組んだ課題として、以下の課題4から課題7についての内容を報告する。

- 課題1 名簿管理プログラムの要件や仕様を踏まえたプログラム作成方針の検討
- 課題2 コマンド実装と考察(a)%Fコマンド
- 課題3 コマンド実装と考察(b)%Rコマンドと%Wコマンド
- 課題4 コマンド実装と考察(c)%Sコマンド
- 課題5 コマンド実装と考察(d)%独自コマンド
- 課題6 プログラム全体に対する考察
- 課題7 発展課題の実装

また、考察課題として、以下の観点での考察をおこない、??章にまとめた。

- 考察1 メモリ上のデータ配置と構造体のサイズに関する考察 (??節)
- 考察2 エラー処理に関する考察 (??節)

2 プログラムの作成方針

本演習で作成したプログラムが満たすべき要件と仕様として、「(1) 基本要件」と「(2) 基本仕様」を示す。

(1) 基本要件

1. プログラムは、その実行中、少なくとも10000件の名簿データをメモリ中に保持できるようにすること。
2. 名簿データは、「ID, 氏名, 誕生日, 住所, 備考」を一つのデータとして扱えるようにすること。
3. プログラムとしての動作や名簿データの管理のために、以下の機能を持つコマンドを実装すること。
 - (a) プログラムの正常な終了
 - (b) 登録された名簿データのデータ数表示
 - (c) 名簿データの全数表示, および, 部分表示
4. 標準入力からのユーザ入力を通して、データ登録やデータ管理の操作を可能とすること。
5. 標準出力にはコマンドの実行結果のみを出力すること。

(2) 基本仕様

1. 名簿データは、コンマ区切りの文字列（CSV入力と呼ぶ）で表されるものとし、Listing ??に示したようなテキストデータを処理できるようにする。
2. コマンドは、%で始まる文字列（コマンド入力と呼ぶ）とし、表1にあげたコマンドをすべて実装する
3. 1つの名簿データは、C言語の構造体 (`struct`) を用いて、構造を持ったデータとしてプログラム中に定義し、利用する。
4. 全名簿データは、”何らかのデータ構造”を用いて、メモリ中に保持できるようにする。
5. コマンドの実行結果以外の出力は、標準エラー出力に出力する。

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 (C)	
%P n	先頭からn件表示 (Print)	nが0 → 全件表示, nが負 → 後ろから-n件表示
%R file	file の読み込み (Read)	
%W file	file の書き出し (Write)	ファイルに上書きで書き出す.
%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 id	プロファイルの削除. (delete)	メモリの解放を行う

2.1 プログラムの全体像

2.2 名簿管理プログラムの実装方針

2.2.1 %Fコマンド

- 機能 引数として文字列を受け取る. 引数とどれかの要素が一致するものを探索し, 表示する.
- 実装方針 引数と一致するかどうかをそれぞれの要素でstrcmp関数を用いて判定する. 一致した場合, 表示する. この動きをforループによってすべてのプロファイルで行う.
- 動作確認 適当な数のプロファイルを与える. id, 誕生日, 名前, 住所, コメントで検索して正しいものが出力されることを確認する. また, 同じデータを持つものが複数あるときの動作も確認する. データにないものを入力して, 見つからなかった時の動きを確認する, 誕生日の入力形式についても確認する.

2.2.2 %Rコマンド

- 機能 引数としてファイル名を受け取る. 一行ずつ読み取ってプロファイルデータとして読み込み.
- 実装方針 fopen関数によって読み取りモードでファイルを開く. fgets関数をファイル形式に対応させて一行ずつ読み込む. 読み込んだデータはnew_profile関数で profile構造体形式にする. profile_data_storeに格納していく.
- 動作確認 適当な数プロファイルデータが入ったcsvファイルを用意する. %Rコマンドによって読み込んで%Pコマンドでfileと同じ内容か確認する, 同じ内容を二回読み込むことで上書きになっていないかを確認する.

2.2.3 %Wコマンド

- 機能 引数としてファイル名を受け取る。 `profile_data_store`に格納されているprofileを引数のfileに書き込んでいく。
- 実装方針 `fopen`関数によって書き込みモードでファイルを開く。 `profile_data_store`データを `fprintf`関数でyyyy-mm-dd形式に変換して fileに書き込む。 `for`ループによってすべてのプロファイルで行う。 最後にはファイルをクローズする。
- 動作確認 適当な数のプロファイルを与える。 fileに書き込んで同じ形式になっているか確認する。 特に正しく改行されているか、 データが入ったfileのとき上書きされるかも確認する。

2.2.4 %Sコマンド

- 機能 引数としてint型の整数を受け取る。 引数1,2,3,4,5それぞれでprofileのフィールドごとに並び替える。 ソートアルゴリズムはバブルソートを採用する。
- 実装方針 `switch`文を用いてそれぞれの操作を分ける。
比較用関数として `sort_id, birthday, name...`を用意する。 比較はすべて `sprintf`関数を用いて文字列に変換してから、 `strcmp`関数を用いて行う。 `swap_profile`関数で並び替える。
- 動作確認 適当な数のprofileデータを与える。 それぞれのフィールドで正しくソートされるかについて確認する。 また、 1-5以外の引数の時の動きを確認する。 異なるprofileデータが同じフィールドデータを持つときの動きも確認する。

2.2.5 %Dコマンド

- 機能 引数としてint型の整数を受け取る。 引数をidとして一致するidの profileを削除する。
- 実装方針 1からidが一致するものを探索する。 一致するものがあれば、 そのprofileに次のprofileデータをコピーしていき、 削除する。 探索は、 最後まで繰り返されるため、 同じidが存在した場合、 削除を繰り返す。 削除する前にcommentフィールドのメモリの解放をする。
- 動作確認 適当な数のプロファイルを与える。 Dコマンドを実行して引数と同じものが削除されることを確認する。 その際、 メモリが解放されたかのメッセージを確認する。 また、 同じidのprofileを複数用意して、 どのように動作するか確認する。 一致するprofileがない時の動作も確認する。

3 プログラムの実装と考察

3.1 コマンドの実装と考察(a)%Fコマンド

3.1.1 プログラムの説明

この関数が呼び出されると、与えられた引数を使って`profile_data_store`配列内のプロフィールを検索する。

- 引数 引数として文字列ポインタ`word`を受け取る。この`word`が探索する際のキーワードとなる。
- 動き
 1. `profile_data_store`配列を一つずつ見ていって、各プロフィールをチェックしていく。
 2. 各プロフィールに対して以下のチェックを行う。：
 - (a) プロファイルIDを`sprintf`を使って文字列に変換し、`strcat`を使って`word`と比較する。IDが一致した場合、そのプロフィールの情報を出力する。
 - (b) `word`とプロフィールのコメント、名前、住所を上と同じように比較し、いずれかが一致した場合、そのプロフィールの情報を出力する。
 - (c) プロファイルの誕生日を文字列に変換し、`word`と同じように比較する。誕生日が一致した場合、そのプロフィールの情報を出力する。

3.1.2 プログラムの動作確認

次のようなデータを読み込ませる。

```
5100046,The Bridge,1845-11-2,14 Seafield Road Longman Inverness,  
SEN Unit 2.0 Open  
5100127,Bower Primary School,1908-1-19,Bowermadden Bower Caithness,  
01955 641225 Primary 25 2.6 Open  
5100224,Canisbay Primary School,1928-7-5,  
Canisbay Wick,01955 611337 Primary 56 3.5 Open  
5100321,Castletown Primary School,1913-11-4,Castletown Thurso,  
01847 821256 01847 821256 Primary 137 8.5 Open  
5100429,Crossroads Primary School,1893-2-24,Dunnet Thurso,  
01847 851629 01847 851629 Primary 29 2.4 Open  
5100526,Dunbeath Primary School,1805-1-8,Dunbeath,  
01593 731286 Primary 13 1.1 Open
```

以下のようなコマンドを実行する。

```
%F 5100046  
%F Bower Primary School  
%F 1928-07-05
```

%F Castletown Thurso

%F 01847 851629 01847 851629 Primary 29 2.4 Open

出力 :

5100046,The Bridge,1845-11-2,14 Seafield Road Longman Inverness,

SEN Unit 2.0 Open

5100127,Bower Primary School,1908-1-19,Bowermadden Bower Caithness,

01955 641225 Primary 25 2.6 Open

5100224,Canisbay Primary School,1928-7-5,Canisbay Wick,

01955 611337 Primary 56 3.5 Open

5100321,Castletown Primary School,1913-11-4,Castletown Thurso,

01847 821256 01847 821256 Primary 137 8.5 Open

5100429,Crossroads Primary School,1893-2-24,Dunnet Thurso,

01847 851629 01847 851629 Primary 29 2.4 Open

5100526,Dunbeath Primary School,1805-1-8,Dunbeath,

01593 731286 Primary 13 1.1 Open

%F 5100046

idが一致しました

Id : 5100046

Name : The Bridge

Birth : 1845-11-02

Addr. : 14 Seafield Road Longman Inverness

Comm. : SEN Unit 2.0 Open

%F Bower Primary School

名前,住所,コメントのいずれかがヒットしました

Id : 5100127

Name : Bower Primary School

Birth : 1908-01-19

Addr. : Bowermadden Bower Caithness

Comm. : 01955 641225 Primary 25 2.6 Open

%F 1928-7-5

%F 1928-07-05

誕生日が一致しました

Id : 5100224

Name : Canisbay Primary School

Birth : 1928-07-05

Addr. : Canisbay Wick

Comm. : 01955 611337 Primary 56 3.5 Open

%F Castletown Thurso

名前,住所,コメントのいずれかがヒットしました

Id : 5100321

Name : Castletown Primary School

Birth : 1913-11-04

Addr. : Castletown Thurso

Comm. : 01847 821256 01847 821256 Primary 137 8.5 Open

%F 01847 851629 01847 851629 Primary 29 2.4 Open

名前,住所,コメントのいずれかがヒットしました

Id : 5100429

Name : Crossroads Primary School

Birth : 1893-02-24

Addr. : Dunnet Thurso

Comm. : 01847 851629 01847 851629 Primary 29 2.4 Open

3.1.3 実装にあたっての考察

この関数の実装にあたってatoiについてエラーが多く起こってしまったのでなぜうまくいかなかったのかについて考察する。

- 変換方法: atoiは文字列が数字でない場合、いくつかの動作を行う。例えば、12345abcが入力されたとき12345と数字以外のものがくるまで変換して返す。abcが入力されたときは、変換できずに0が返される。数字の0が返されたのか、文字列を変換した結果返されたのかが分からないため、エラーの原因になりえる。
- 比較方法: 文字列と整数の比較を行う場合、変換がうまくいかなかった場合、その後に影響する場合もある。特に、文字列である場合、エラーが起こる。

いずれかの理由でエラーがおこってしまう。そのため今回はsprintfを使用して整数値を文字列に合わせるほうが確実だと考え、そちらを採用した。

3.2 コマンドの実装と考察(b)%Rコマンドと%Wコマンド

3.2.1 %Rコマンドの説明

この関数が呼び出されると、与えられたファイルからデータを読み取り、プロファイルを作成し、`profile_data_store`配列に格納する。

- ファイルのオープン：ファイルの名前が引数として渡されると、`fopen`関数により引数と一致するファイルが読み取りモード"`r`"で開かれる。開くのに失敗したときエラーとして出力される。
- ファイルからのデータ読み込み： `fget`関数を使って`file`から`line`に一行ずつ格納する。その際、`subst`関数で改行文字を`null`文字に変える。不要な改行をなくす。
- プロファイルの作成と格納： `profile_data_item`が`MAX_PROFILES`以上であればエラーを返す。それ未満であれば、`profile`型の構造体を定義し、`profile`に格納していく関数`new_profile`に渡す。その際、`profile_data_item`を1増やす。作成できなかった場合はエラーを返す。
- ファイルのクローズ：データの読み込み後に`fclose`関数によってファイルを閉じる。

3.2.2 %Wコマンドの説明

この関数が呼び出されると、引数によって与えられたファイルに現在保持しているデータを書き込む、

- ファイルのオープン：ファイルの名前が引数として渡されると、`fopen`関数により引数と一致するファイルが書き込みモード"`w`"で開かれる。開くのに失敗したときエラーとして出力される。
- ファイルへのデータ読み込み： `profile_data_item`に基づいて`/textttfor`ループを実行する。それぞれのデータを構造体のポインタに格納する。その後、`csv`形式に合ったデータを`fprintf`関数によってファイルに保存していく。
- ファイルのクローズ：データの書き込み後に`fclose`関数によってファイルを閉じる。

3.2.3 %Rプログラムの動作確認

%Rコマンドによって読み込んで%Pコマンドでファイルと同じ内容か確認する。同じファイルを二回読み込むことで、データが上書きされていないことを確認する。例えば、次のようなデータを含む `result.R.csv` ファイルを用意する。

```
5100046,The Bridge,1845-11-2,14 Seafield Road Longman Inverness,  
SEN Unit 2.0 Open  
5100127,Bower Primary School,1908-1-19,Bowermadden Bower Caithness,  
01955 641225 Primary 25 2.6 Open  
5100224,Canisbay Primary School,1928-7-5,Canisbay Wick,  
01955 611337 Primary 56 3.5 Open  
5100321,Castletown Primary School,1913-11-4,Castletown Thurso,
```


01847 821256 01847 821256 Primary 137 8.5 Open
5100429,Crossroads Primary School,1893-2-24,Dunnet Thurso,
01847 851629 01847 851629 Primary 29 2.4 Open

以下のようなコマンドを実行する。

```
%R result_R.csv
Data read from result_R.csv successfully.
%P
```

出力:

Id : 5100046
Name : The Bridge
Birth : 1845-11-02
Addr. : 14 Seafield Road Longman Inverness
Comm. : SEN Unit 2.0 Open

Id : 5100127
Name : Bower Primary School
Birth : 1908-01-19
Addr. : Bowermadden Bower Caithness
Comm. : 01955 641225 Primary 25 2.6 Open

Id : 5100224
Name : Canisbay Primary School
Birth : 1928-07-05
Addr. : Canisbay Wick
Comm. : 01955 611337 Primary 56 3.5 Open

Id : 5100321
Name : Castletown Primary School
Birth : 1913-11-04
Addr. : Castletown Thurso
Comm. : 01847 821256 01847 821256 Primary 137 8.5 Open

Id : 5100429
Name : Crossroads Primary School
Birth : 1893-02-24
Addr. : Dunnet Thurso
Comm. : 01847 851629 01847 851629 Primary 29 2.4 Open

続けて同じファイルを再度読み込む。

```
%R result_R.csv
Data read from result_R.csv successfully.
%P
```

出力:

```
Id      : 5100046
Name    : The Bridge
Birth   : 1845-11-02
Addr.   : 14 Seafield Road Longman Inverness
Comm.   : SEN Unit 2.0 Open
```

```
Id      : 5100127
Name    : Bower Primary School
Birth   : 1908-01-19
Addr.   : Bowermadden Bower Caithness
Comm.   : 01955 641225 Primary 25 2.6 Open
```

```
Id      : 5100224
Name    : Canisbay Primary School
Birth   : 1928-07-05
Addr.   : Canisbay Wick
Comm.   : 01955 611337 Primary 56 3.5 Open
```

```
Id      : 5100321
Name    : Castletown Primary School
Birth   : 1913-11-04
Addr.   : Castletown Thurso
Comm.   : 01847 821256 01847 821256 Primary 137 8.5 Open
```

```
Id      : 5100429
Name    : Crossroads Primary School
Birth   : 1893-02-24
Addr.   : Dunnet Thurso
Comm.   : 01847 851629 01847 851629 Primary 29 2.4 Open
```

```
Id      : 5100046
Name    : The Bridge
```

Birth : 1845-11-02
Addr. : 14 Seafield Road Longman Inverness
Comm. : SEN Unit 2.0 Open

Id : 5100127
Name : Bower Primary School
Birth : 1908-01-19
Addr. : Bowermadden Bower Caithness
Comm. : 01955 641225 Primary 25 2.6 Open

Id : 5100224
Name : Canisbay Primary School
Birth : 1928-07-05
Addr. : Canisbay Wick
Comm. : 01955 611337 Primary 56 3.5 Open

Id : 5100321
Name : Castletown Primary School
Birth : 1913-11-04
Addr. : Castletown Thurso
Comm. : 01847 821256 01847 821256 Primary 137 8.5 Open

Id : 5100429
Name : Crossroads Primary School
Birth : 1893-02-24
Addr. : Dunnet Thurso
Comm. : 01847 851629 01847 851629 Primary 29 2.4 Open

3.2.4 %Wプログラムの動作確認

まず、次のようなデータを入力する。

5100046,The Bridge,1845-11-2,14 Seafield Road Longman Inverness,SEN Unit 2.0 Open
5100127,Bower Primary School,1908-1-19,Bowermadden Bower Caithness,01955 641225
Primary 25 2.6 Open
5100224,Canisbay Primary School,1928-7-5,Canisbay Wick,01955 611337 Primary 56
3.5 Open
5100321,Castletown Primary School,1913-11-4,Castletown Thurso,
01847 821256 01847 821256 Primary 137 8.5 Open
5100429,Crossroads Primary School,1893-2-24,
Dunnet Thurso,01847 851629 01847 851629 Primary 29 2.4 Open

Wコマンドを入力する.

```
%W result_W.csv
```

result_W.csvを確認する.

```
5100046,The Bridge,1845-11-2,14 Seafield Road Longman Inverness,  
SEN Unit 2.0 Open  
5100127,Bower Primary School,1908-1-19,Bowermadden Bower Caithness,  
01955 641225 Primary 25 2.6 Open  
5100224,Canisbay Primary School,1928-7-5,Canisbay Wick,  
01955 611337 Primary 56 3.5 Open  
5100321,Castletown Primary School,1913-11-4,Castletown Thurso,  
01847 821256 01847 821256 Primary 137 8.5 Open  
5100429,Crossroads Primary School,1893-2-24,Dunnet Thurso,  
01847 851629 01847 851629 Primary 29 2.4 Open
```

さらに、次のようなデータがすでに入ったresult_W1ファイルを用意する.

```
5100224,Canisbay Primary School,1928-07-05,Canisbay Wick,01955 611337  
Primary 56 3.5 Open  
5100321,Castletown Primary School,1913-11-04,Castletown Thurso,  
01847 821256 01847 821256 Primary 137 8.5 Open
```

次のようなデータを読み込ませる.

```
5100046,The Bridge,1845-11-02,14 Seafield Road Longman Inverness,SEN Unit 2.0 Open  
5100127,Bower Primary School,1908-01-19,Bowermadden Bower Caithness,  
01955 641225 Primary 25 2.6 Open
```

Wコマンドを入力する.

```
%W result_W1.csv
```

result_W1.csvを確認すると

```
5100046,The Bridge,1845-11-02,14 Seafield Road Longman Inverness,SEN Unit 2.0 Open  
5100127,Bower Primary School,1908-01-19,Bowermadden Bower Caithness,01955 641225  
Primary 25 2.6 Open
```

となって上書きされていることが分かる。

3.2.5 実装にあたっての考察

この関数の実装にあたって`get_line`関数を変更するべきか、それとも新しくすべきか考えたので考察していく。`get_line`関数を標準入力とファイルからの読み込みに対してどちらも対応した改良版のコードはこの後記載している。

- まとめることのメリット：`get_line`関数をまとめることによって同じような関数を2回定義しなくてよいので簡潔なコードになる。また、エラーや修正が必要な改善が求められたときに修正箇所が少なく済む。
- まとめることのデメリット：標準入力のときのみ追加したい事項やファイルのときのみ追加したいエラーなどがあった場合に実装しづらい。また、`%R`コマンドの時にのみしか必要ないにもかかわらず、多く使用されるであろう標準入力のときにも`if(file == NULL)`という条件分岐が発生するため、処理速度に影響があると推測できる。

`get_line`関数をまとめることによって簡潔さなどが向上する反面、実際は時間のかかる処理ではないため、影響はないかもしれないが、1万という大きなデータを扱う上では大きな差になってしまうかもしれない。

```
1:      int get_line(char *line, int size, FILE *file) {
2:      if (file == NULL) {
3:          if (fgets(line, 1024, stdin) != NULL) {
4:              subst(line, '\n', '\0');
5:              return 1;
6:          }
7:      } else {
8:          if (fgets(line, 1024, file) != NULL) {
9:              subst(line, '\n', '\0');
10:             return 1;
11:          }
12:      }
13:      return 0;
14: }
```

3.3 コマンドの実装と考察(c)%Sコマンド

3.3.1 プログラムの説明

この関数は、`cmd.sort`が呼び出されると、`int`型の引数`column`によって `id`、名前、誕生日、住所、備考について並び替える。なお、並び替えた後の表示の機能はない。このソートはバブルソートのアルゴリズムに基づいている。

- 引数 引数として`int`型の引数`column`を受け取る。この`column`がソートする際のキーワードとなる。
- 動き
 1. `swap_profile`関数:メモリ効率のために構造体のポインタを通して`profile`を交換する。
 2. `switch`文により`column`の値に応じてそれぞれソートを行う。
 3. バブルソートを行っていく、要素同士を比較して入れ替える関数をそれぞれの要素に対して用意している。

idのとき, それぞれをsprintfによって文字列に変換する. strcmpにより比較する. この値が正の時, swap関数によって入れ替える.

birthdayのとき, それぞれをsprintfによって年, 月, 日付をつなげて文字列に変換する. strcmpにより比較する. この値が正の時, swap関数によって入れ替える.

name,address,commentのとき,strcmpにより比較する. この値が正の時, swap関数によって入れ替える.

3.3.2 プログラムの動作確認

以下のようなプロファイルデータを読み込ませる.

```
5100127,Bower Primary School,1908-1-19,Bowermadden Bower Caithness,  
01955 641225 Primary 25 2.6 Open  
5100046,The Bridge,1845-11-2,14 Seafield Road Longman Inverness,  
SEN Unit 2.0 Open  
5100224,Canisbay Primary School,1928-7-5,Canisbay Wick,  
01955 611337 Primary 56 3.5 Open  
5100321,Castletown Primary School,1913-11-4,Castletown Thurso,  
01847 821256 01847 821256 Primary 137 8.5 Open  
5100429,Crossroads Primary School,1893-2-24,Dunnet Thurso,  
01847 851629 01847 851629 Primary 29 2.4 Open
```

%Sコマンドによってそれぞれのデータでソートする.

```
%S 1  
%P 5  
Id      : 5100046  
Name    : The Bridge  
Birth   : 1845-11-02  
Addr.   : 14 Seafield Road Longman Inverness  
Comm.   : SEN Unit 2.0 Open  
  
Id      : 5100127  
Name    : Bower Primary School  
Birth   : 1908-01-19  
Addr.   : Bowermadden Bower Caithness  
Comm.   : 01955 641225 Primary 25 2.6 Open  
  
Id      : 5100224  
Name    : Canisbay Primary School  
Birth   : 1928-07-05  
Addr.   : Canisbay Wick
```

Comm. : 01955 611337 Primary 56 3.5 Open

Id : 5100321

Name : Castletown Primary School

Birth : 1913-11-04

Addr. : Castletown Thurso

Comm. : 01847 821256 01847 821256 Primary 137 8.5 Open

Id : 5100429

Name : Crossroads Primary School

Birth : 1893-02-24

Addr. : Dunnet Thurso

Comm. : 01847 851629 01847 851629 Primary 29 2.4 Open

%S 2

%P 5

Id : 5100127

Name : Bower Primary School

Birth : 1908-01-19

Addr. : Bowermadden Bower Caithness

Comm. : 01955 641225 Primary 25 2.6 Open

Id : 5100224

Name : Canisbay Primary School

Birth : 1928-07-05

Addr. : Canisbay Wick

Comm. : 01955 611337 Primary 56 3.5 Open

Id : 5100321

Name : Castletown Primary School

Birth : 1913-11-04

Addr. : Castletown Thurso

Comm. : 01847 821256 01847 821256 Primary 137 8.5 Open

Id : 5100429

Name : Crossroads Primary School

Birth : 1893-02-24

Addr. : Dunnet Thurso

Comm. : 01847 851629 01847 851629 Primary 29 2.4 Open

Id : 5100046

Name : The Bridge
Birth : 1845-11-02
Addr. : 14 Seafield Road Longman Inverness
Comm. : SEN Unit 2.0 Open

%S 3

%P 5

Id : 5100046
Name : The Bridge
Birth : 1845-11-02
Addr. : 14 Seafield Road Longman Inverness
Comm. : SEN Unit 2.0 Open

Id : 5100429
Name : Crossroads Primary School
Birth : 1893-02-24
Addr. : Dunnet Thurso
Comm. : 01847 851629 01847 851629 Primary 29 2.4 Open

Id : 5100127
Name : Bower Primary School
Birth : 1908-01-19
Addr. : Bowermadden Bower Caithness
Comm. : 01955 641225 Primary 25 2.6 Open

Id : 5100321
Name : Castletown Primary School
Birth : 1913-11-04
Addr. : Castletown Thurso
Comm. : 01847 821256 01847 821256 Primary 137 8.5 Open

Id : 5100224
Name : Canisbay Primary School
Birth : 1928-07-05
Addr. : Canisbay Wick
Comm. : 01955 611337 Primary 56 3.5 Open

%S 4

%P 5

Id : 5100046
Name : The Bridge

Birth : 1845-11-02
Addr. : 14 Seafield Road Longman Inverness
Comm. : SEN Unit 2.0 Open

Id : 5100127
Name : Bower Primary School
Birth : 1908-01-19
Addr. : Bowermadden Bower Caithness
Comm. : 01955 641225 Primary 25 2.6 Open

Id : 5100224
Name : Canisbay Primary School
Birth : 1928-07-05
Addr. : Canisbay Wick
Comm. : 01955 611337 Primary 56 3.5 Open

Id : 5100321
Name : Castletown Primary School
Birth : 1913-11-04
Addr. : Castletown Thurso
Comm. : 01847 821256 01847 821256 Primary 137 8.5 Open

Id : 5100429
Name : Crossroads Primary School
Birth : 1893-02-24
Addr. : Dunnet Thurso
Comm. : 01847 851629 01847 851629 Primary 29 2.4 Open

%S 5

%P 5

Id : 5100321
Name : Castletown Primary School
Birth : 1913-11-04
Addr. : Castletown Thurso
Comm. : 01847 821256 01847 821256 Primary 137 8.5 Open

Id : 5100429
Name : Crossroads Primary School
Birth : 1893-02-24
Addr. : Dunnet Thurso
Comm. : 01847 851629 01847 851629 Primary 29 2.4 Open

Id : 5100224
Name : Canisbay Primary School
Birth : 1928-07-05
Addr. : Canisbay Wick
Comm. : 01955 611337 Primary 56 3.5 Open

Id : 5100127
Name : Bower Primary School
Birth : 1908-01-19
Addr. : Bowermadden Bower Caithness
Comm. : 01955 641225 Primary 25 2.6 Open

Id : 5100046
Name : The Bridge
Birth : 1845-11-02
Addr. : 14 Seafield Road Longman Inverness
Comm. : SEN Unit 2.0 Open

同じデータを持つものが複数あるときの動作も確認する。以下のようなプロファイルデータを読み込ませる。

5100127,Bower Primary School,1908-1-19,Bowermadden Bower Caithness,
01955 641225 Primary 25 2.6 Open
5100046,The Bridge,1845-11-2,14 Seafield Road Longman Inverness,
SEN Unit 2.0 Open
5100046,Canisbay Primary School,1928-7-5,Canisbay Wick,
01955 611337 Primary 56 3.5 Open
5100321,Castletown Primary School,1913-11-4,Castletown Thurso,
01847 821256 01847 821256 Primary 137 8.5 Open
5100046,Crossroads Primary School,1893-2-24,Dunnet Thurso,
01847 851629 01847 851629 Primary 29 2.4 Open

同じidを持つ場合以下のような結果になる。

%S 1
%P
Id : 5100046
Name : The Bridge
Birth : 1845-11-02
Addr. : 14 Seafield Road Longman Inverness

Comm. : SEN Unit 2.0 Open

Id : 5100046

Name : Canisbay Primary School

Birth : 1928-07-05

Addr. : Canisbay Wick

Comm. : 01955 611337 Primary 56 3.5 Open

Id : 5100046

Name : Crossroads Primary School

Birth : 1893-02-24

Addr. : Dunnet Thurso

Comm. : 01847 851629 01847 851629 Primary 29 2.4 Open

Id : 5100127

Name : Bower Primary School

Birth : 1908-01-19

Addr. : Bowermadden Bower Caithness

Comm. : 01955 641225 Primary 25 2.6 Open

Id : 5100321

Name : Castletown Primary School

Birth : 1913-11-04

Addr. : Castletown Thurso

Comm. : 01847 821256 01847 821256 Primary 137 8.5 Open

同じidのとき、最初にデータが入っていた順番になる。

3.3.3 実装にあたっての考察

この関数の実装にあたって計算量がどのようになっているのか気になったので各関数の計算量、改善案について考察する。

- 計算量： `swap_profile`関数は構造体のコピーが3回ある。コピー操作は $O(1)$ なので全体は $O(1)$ 。 `sort_id`関数などの比較して並び替える関数は与えられたデータを文字列に変換して比較し、必要であれば`swap_profile`関数を呼び出す。それぞれの操作は $O(1)$ なので全体の操作は $O(1)$ 。 `cmd_sort`関数はそれぞれの分岐に対してforループにより N 回ループの中で N 回ループが実行される。その中で計算量 $O(1)$ の関数が実行される。よって全体の計算量は $O(N^2)$ となる
- 改善案1： クイックソートやマージソートなどのソート法に変える。これらのソート法は平均計算量が $N \log N$ なので計算量を減らすことができる。
- 改善案2： 比較する際の文字列の変換を減らす。現状、比較する際にすべてのデータを文字列

に変換しているが、idやbirthdayは数値のまま比較することで計算量は変換するために必要な分だけ計算量は減る。

3.4 コマンドの実装と考察(d)独自コマンド

3.4.1 %Dコマンドの説明

この関数が呼び出されると、idが一致するものを探し、そのprofileデータを削除する。

- 探索：idが引数として渡されると、引数と一致するidをもつプロファイルが1から順に探索される。
- メモリの解放：free関数を用いて、探索で見つかったプロファイルのcommentフィールドのメモリを解放する。メモリの解放がないとのちにメモリの圧迫により、エラーや動作が重くなる原因になる。
- profileの削除：見つかったプロファイルがあった場所にその次のプロファイルを格納していく。forループによって一つずつずらしてプロファイルを削除する。最後にprofile_data_itemsを1減らす。

3.4.2 プログラムの動作確認

```
5100046,The Bridge,1845-11-2,14 Seafield Road Longman Inverness,  
SEN Unit 2.0 Open  
5100127,Bower Primary School,1908-1-19,Bowermadden Bower Caithness,  
01955 641225 Primary 25 2.6 Open  
5100224,Canisbay Primary School,1928-7-5,Canisbay Wick,  
01955 611337 Primary 56 3.5 Open  
5100321,Castletown Primary School,1913-11-4,Castletown Thurso,  
01847 821256 01847 821256 Primary 137 8.5 Open  
5100429,Crossroads Primary School,1893-2-24,Dunnet Thurso,  
01847 851629 01847 851629 Primary 29 2.4 Open
```

```
%D 5100127
```

```
Memory was freed successfully
```

```
%P
```

```
Id      : 5100046
```

```
Name    : The Bridge
```

```
Birth   : 1845-11-02
```

```
Addr.   : 14 Seafield Road Longman Inverness
```

```
Comm.   : SEN Unit 2.0 Open
```

```
Id      : 5100224
```

```
Name    : Canisbay Primary School
```

Birth : 1928-07-05
Addr. : Canisbay Wick
Comm. : 01955 611337 Primary 56 3.5 Open

Id : 5100321
Name : Castletown Primary School
Birth : 1913-11-04
Addr. : Castletown Thurso
Comm. : 01847 821256 01847 821256 Primary 137 8.5 Open

Id : 5100429
Name : Crossroads Primary School
Birth : 1893-02-24
Addr. : Dunnet Thurso
Comm. : 01847 851629 01847 851629 Primary 29 2.4 Open

%D 5100429

Memory was freed successfully

%P

Id : 5100046
Name : The Bridge
Birth : 1845-11-02
Addr. : 14 Seafield Road Longman Inverness
Comm. : SEN Unit 2.0 Open

Id : 5100224
Name : Canisbay Primary School
Birth : 1928-07-05
Addr. : Canisbay Wick
Comm. : 01955 611337 Primary 56 3.5 Open

Id : 5100321
Name : Castletown Primary School
Birth : 1913-11-04
Addr. : Castletown Thurso
Comm. : 01847 821256 01847 821256 Primary 137 8.5 Open

%D 000000

%P

Id : 5100046
Name : The Bridge

Birth : 1845-11-02
Addr. : 14 Seafield Road Longman Inverness
Comm. : SEN Unit 2.0 Open

Id : 5100224
Name : Canisbay Primary School
Birth : 1928-07-05
Addr. : Canisbay Wick
Comm. : 01955 611337 Primary 56 3.5 Open

Id : 5100321
Name : Castletown Primary School
Birth : 1913-11-04
Addr. : Castletown Thurso
Comm. : 01847 821256 01847 821256 Primary 137 8.5 Open

5100046,The Bridge,1845-11-2,14 Seafield Road Longman Inverness,
SEN Unit 2.0 Open
5100046,The Bridge,1845-11-2,14 Seafield Road Longman Inverness,
SEN Unit 2.0 Open
5100321,Castletown Primary School,1913-11-4,Castletown Thurso,
01847 821256 01847 821256 Primary 137 8.5 Open

%D 5100046

Memory was freed successfully

Memory was freed successfully

%P

Id : 5100321
Name : Castletown Primary School
Birth : 1913-11-04
Addr. : Castletown Thurso
Comm. : 01847 821256 01847 821256 Primary 137 8.5 Open

3.4.3 実装にあたっての考察

この関数を実装するうえで計算時間が多くなっているのではないかと考えたので考察する。

- 計算量：

- 改善案1：二分探索法の実装

4 プログラム全体の考察

4.1 二分探索法の導入についての考察

二分探索法は、ソートされたデータに対して効率的に検索を行うアルゴリズムである。これを導入することで得られるメリットとデメリットを挙げる。

メリット

- 検索速度: 線形探索の時間計算量が $O(n)$ であるのに対し、二分探索法は $O(\log n)$ である。このため、データのサイズが大きくなるにつれて、二分探索法の検索速度の速さが目立ってくる。本プログラムでは1万近くのデータを管理するため、これは大きなメリットである。
- シンプルな実装: 二分探索法はほかの検索に比べてシンプルなため実装しやすい。さらに、バグが起きたときに問題を突き止めやすい。

デメリット

- ソートが必要: 二分探索法のためには、データがあらかじめソートされている必要がある。本プログラムのデータのソートには $O(N^2)$ の時間がかかるため、データがソートされていないときとされているときの探索時間の差が非常に大きい。
- 動的データへの不適応: 二分探索法は、上にも述べたようにデータのソートが必要である。データが頻繁に追加・削除される本プログラムのような動的なデータにはあまり適していない。データが変更されるたびに再ソートが必要となり、これが非効率的である。

以上のように、二分探索法は特定の条件下で非常に有用であるが、使用する際には条件を十分に考慮する必要がある。

4.2

5 発展課題：二分探索法(binary search)の導入

発展課題として全体考察で述べた二分探索法の導入を行う。 %S コマンドに組み込むことで、ソートされている場合は二分探索法、されていない場合は従来の方法も考えた。しかしながら、 %S コマンドが毎回ソートを確認しなければならなくなってしまうため、新しい %B コマンドとして実装していく。本課題では id のソート用としていく。

5.1 実装方針

- 機能 引数として int 型の整数を受け取る。引数と id が一致する profile を探索し、表示する。
- 実装方針 引数を受け取ると、二分探索法のため、ソートされているか確認する。確認すると、二分探索のアルゴリズムを開始する。見つかると print_profile 関数によって出力される。
- 動作確認 適当な数のプロファイルを与える。ソートされているときと、されていない時の動作を確認する。されている場合、一致するものを出力するか確認する。

- 比較 %Fコマンドと比較する．比較用にidのみの%Fコマンド，ソートされているか確認をしない%Bコマンド，それぞれ%f, %bとして比較用コマンドを作っていく．理論上， $n : \log(n)$ の割合の実行速度になるはずである．

5.2 コマンドの実装と考察(c)%Bコマンド

5.2.1 プログラムの説明

この関数は、cmd_sortが呼び出されると、int型の引数columnによって id, 名前, 誕生日, 住所, 備考について並び替える．なお、並び替えた後の表示の機能はない．このソートはバブルソートのアルゴリズムに基づいている．

- 引数 引数としてint型の引数idを受け取る．このidが探索する際のキーワードとなる．
- 動き
 1. ソート確認：まず，profile_data_storeのprofileがidでソートされているか比較文を用いて確認する．ソートされていないなら，エラー文を出力する．
 2. 二分探索：left,right,mid変数を用意する．その時にmidを計算しておく．midが目的のidと一致していればmidを返して終了．大きければmid+1をleftとして，小さければmid-1をrightとして繰り返す．最後まで見つからなかったとき-1を返す．その後見つからなかったことを出力する．-1以外の時はmidのprofileをprint_profile関数で出力して終了する．

5.2.2 プログラムの動作確認

まず，ソートされたデータを与える．

```
1,The Bridge,1845-11-2,14 Seafield Road Longman Inverness,
SEN Unit 2.0 Open
2,Bower Primary School,1908-1-19,Bowermadden Bower Caithness,
01955 641225 Primary 25 2.6 Open
3,Canisbay Primary School,1928-7-5,Canisbay Wick,
01955 611337 Primary 56 3.5 Open
```

1と4を探索する．

```
%B 1
binarysearchを行います
二分探索法によってこのidを持つものが見つかりました
ID: 1
Name: The Bridge
Birthday: 1845-11-2
Address: 14 Seafield Road Longman Inverness
Comment: SEN Unit 2.0 Open
```


%B 4

binarysearchを行います

このidを持つprofileは存在しません

上のように二分探索によって探索されたことが確認できる。存在しない場合は出力する。次にソートされていないデータを与える。

1,The Bridge,1845-11-2,14 Seafield Road Longman Inverness,

SEN Unit 2.0 Open

3,Bower Primary School,1908-1-19,Bowermadden Bower Caithness,

01955 641225 Primary 25 2.6 Open

2,Canisbay Primary School,1928-7-5,Canisbay Wick,

01955 611337 Primary 56 3.5 Open

%B 1

binarysearchを行います

このデータはソートされていません

上のようにソートされていないと出力する。

5.2.3 線形探索との比較

比較ではfindコマンドをidのみように変えたものとソートされているか確認しない二分探索法のコマンドを用意した。それぞれ探索開始時間、終了時間から時間を計測できるようなプログラムになっている。実験用関数は最後の章に掲載している。ソートされたものという限定条件があるものの合

インデックス	ID	線形探索の実行時間 (%f)	二分探索の実行時間 (%b)
1番目	5100046	0.000002 秒	0.000036 秒
500番目	5250927	0.000077 秒	0.000003 秒
1000番目	5506220	0.000014 秒	0.000003 秒
2000番目	8242143	0.000022 秒	0.000023 秒
2886番目	8681139	0.000375 秒	0.000003 秒
合計		0.000490 秒	0.000068 秒

表2 線形探索と二分探索の実行時間比較

計時間にはかなりの違いがみられた。それぞれ見てみると、線形探索は初めのほうは実行時間がかなり速いが、最後には約200倍になっている。それ比べて二分探索は安定して早いという結果になった。

6 作成したプログラムのソースコード

作成したプログラムを以下に添付する。なお、1章に示した課題については、

```
1: #include <stdio.h>
2: #include <stdlib.h>
3: #include <string.h>
4:
5: #define MAX_STR_LEN 69
6: #define MAX 1024
7: #define MAX_PROFILES 10000
8:
9: int subst(char *str, char c1, char c2) {
10:     int count = 0;
11:     while (1) {
12:         if (*str == c1) {
13:             *str = c2;
14:             count++;
15:         }
16:         str++;
17:         if (*str == '\0') {
18:             return count;
19:         }
20:     }
21: }
22:
23: int get_line(char *line) {
24:     if (fgets(line, 1024, stdin) != NULL) {
25:         subst(line, '\n', '\0');
26:         return 1;
27:     } else {
28:         return 0;
29:     }
30: }
31:
32: int split(char *str, char *ret[], char sep, int max) {
33:     int i = 0;
34:     ret[i] = str;
35:     while (*str) {
36:         if (*str == sep) {
37:             *str = '\0';
38:             ret[++i] = str + 1;
39:         } else if (*str == '\0') {
40:             return i + 1;
41:         }
42:         str++;
43:     }
44:     return i + 1;
45: }
46: }
47:
48: struct date {
49:     int y;
50:     int m;
51:     int d;
52: };
53:
54: struct profile {
55:     int id;
56:     char name[MAX_STR_LEN + 1];
```

```

57:     struct date birthday;
58:     char address[MAX_STR_LEN + 1];
59:     char *comment;
60: };
61:
62: struct date *new_date(struct date *d, char *str) {
63:     char *ptr[3];
64:
65:     if (split(str, ptr, '-', 3) != 3)
66:         return NULL;
67:
68:     d->y = atoi(ptr[0]);
69:     d->m = atoi(ptr[1]);
70:     d->d = atoi(ptr[2]);
71:
72:     return d;
73: }
74:
75: struct profile *new_profile(struct profile *p, char *csv) {
76:     char *ptr[5];
77:
78:     if (split(csv, ptr, ',', 5) != 5 )
79:         return NULL;
80:
81:     p->id = atoi(ptr[0]);
82:     strncpy(p->name, ptr[1], MAX_STR_LEN);
83:     p->name[MAX_STR_LEN] = '\0';
84:
85:     if (new_date(&p->birthday, ptr[2]) == NULL)
86:         return NULL;
87:
88:     strncpy(p->address, ptr[3], MAX_STR_LEN);
89:     p->address[MAX_STR_LEN] = '\0';
90:
91:     p->comment = (char *)malloc(sizeof(char) * (strlen(ptr[4]) + 1));
92:     if (p->comment == NULL) {
93:         fprintf(stderr, "Memory allocation failed for comment.\n");
94:         return NULL;
95:     }
96:     strcpy(p->comment, ptr[4]);
97:
98:     return p;
99: }
100:
101: struct profile profile_data_store[MAX_PROFILES];
102: int profile_data_nitems = 0;
103: void swap_profile(struct profile *p1, struct profile *p2)
104: {
105:     struct profile tmp;
106:
107:     tmp = *p2;
108:     *p2 = *p1;
109:     *p1 = tmp;
110: }
111: void print_profile(struct profile p) {
112:     printf("ID: %d\n", p.id);
113:     printf("Name: %s\n", p.name);
114:     printf("Birthday: %d-%d-%d\n", p.birthday.y, p.birthday.m, p.birthday.d);
115:     printf("Address: %s\n", p.address);
116:     printf("Comment: %s\n", p.comment);
117: }
118: void sort_id(struct profile *p, struct profile *q){
119:     char str_p[MAX] = "";

```

```

120:     char str_q[MAX]="";
121:     sprintf(str_p,"%d",p->id);
122:     sprintf(str_q,"%d",q->id);
123:     if (strcmp(str_p,str_q)>0){
124:         swap_profile(p,q);
125:     }
126: }
127: void sort_birthday(struct profile *p,struct profile *q){
128:     char str_p[MAX] ="";
129:     char str_q[MAX]="";
130:     sprintf(str_p,"%04d,%02d,%02d",p->birthday.y,p->birthday.m,p->birthday.d);
131:     sprintf(str_q,"%04d,%02d,%02d",q->birthday.y,q->birthday.m,q->birthday.d);
132:
133:     if (strcmp(str_p,str_q)>0){
134:         swap_profile(p,q);
135:     }
136: }
137: void sort_name(struct profile *p,struct profile *q){
138:     char str_p[MAX] ="";
139:     char str_q[MAX]="";
140:     sprintf(str_p,"%s",p->name);
141:     sprintf(str_q,"%s",q->name);
142:     if (strcmp(str_p,str_q)>0){
143:         swap_profile(p,q);
144:     }
145: }
146: void sort_add(struct profile *p,struct profile *q){
147:     char str_p[MAX] ="";
148:     char str_q[MAX]="";
149:     sprintf(str_p,"%s",p->address);
150:     sprintf(str_q,"%s",q->address);
151:     if (strcmp(str_p,str_q)>0){
152:         swap_profile(p,q);
153:     }
154: }
155: void sort_comm(struct profile *p,struct profile *q){
156:     char str_p[MAX] ="";
157:     char str_q[MAX]="";
158:     sprintf(str_p,"%s",p->comment);
159:     sprintf(str_q,"%s",q->comment);
160:     if (strcmp(str_p,str_q)>0){
161:         swap_profile(p,q);
162:     }
163: }
164:
165:
166: void cmd_quit() {
167:     exit(0);
168: }
169:
170: void cmd_check() {
171:     printf("%d profile(s)\n", profile_data_nitems);
172:
173: }
174:
175: void cmd_print(int count) {
176:     int j = 0;
177:
178:     if (count == 0) {
179:         count = profile_data_nitems;
180:     }
181:     if (count < 0) {
182:         count = -count;

```

```

183:         if (count > profile_data_nitems) {
184:             count = profile_data_nitems;
185:         }
186:         for (j = profile_data_nitems - count; j < profile_data_nitems; ++j) {
187:             printf("Id      : %d\n", profile_data_store[j].id);
188:             printf("Name    : %s\n", profile_data_store[j].name);
189:             printf("Birth   : %04d-%02d-%02d\n", profile_data_store[j].birthday.y, profile_data_store[j].birthday.m, profile_data_store[j].birthday.d);
190:             printf("Addr.   : %s\n", profile_data_store[j].address);
191:             printf("Comm.   : %s\n\n", profile_data_store[j].comment);
192:         }
193:     } else {
194:         if (count > profile_data_nitems) {
195:             count = profile_data_nitems;
196:         }
197:         for (j = 0; j < count; ++j) {
198:             printf("Id      : %d\n", profile_data_store[j].id);
199:             printf("Name    : %s\n", profile_data_store[j].name);
200:             printf("Birth   : %04d-%02d-%02d\n", profile_data_store[j].birthday.y, profile_data_store[j].birthday.m, profile_data_store[j].birthday.d);
201:             printf("Addr.   : %s\n", profile_data_store[j].address);
202:             printf("Comm.   : %s\n\n", profile_data_store[j].comment);
203:         }
204:     }
205: }
206:
207:
208:
209: void cmd_read(char *filename) {
210:     // fprintf(stderr, "Read (%s) command is invoked with arg: '%s'\n", filename);
211:     FILE *file = fopen(filename, "r");
212:     if (file == NULL) {
213:         perror("Failed to open file for reading");
214:         return;
215:     }
216:
217:     char line[MAX + 1];
218:     while (fgets(line, sizeof(line), file)) {
219:         subst(line, '\n', '\0');
220:         if (profile_data_nitems < MAX_PROFILES) {
221:             struct profile new_profile_data;
222:             if (new_profile(&new_profile_data, line) != NULL) {
223:                 profile_data_store[profile_data_nitems] = new_profile_data;
224:                 profile_data_nitems++;
225:             } else {
226:                 fprintf(stderr, "Failed to create profile from line: %s\n", line);
227:             }
228:         } else {
229:             fprintf(stderr, "Maximum number of profiles reached.\n");
230:             break;
231:         }
232:     }
233:
234:     fclose(file);
235:     printf("Data read from %s successfully.\n", filename);
236: }
237:
238: void cmd_write(char *filename) {
239:     // fprintf(stderr, "Write (%s) command is invoked with arg: '%s'\n", filename);
240:     FILE *file = fopen(filename, "w");
241:     if (file == NULL) {
242:         perror("Failed to open file for writing");
243:         return;
244:     }
245:

```

```

246:     for (int i = 0; i < profile_data_nitems; i++) {
247:         struct profile *p = &profile_data_store[i];
248:         fprintf(file, "%d,%s,%04d-%02d-%02d,%s,%s\n",
249:             p->id,
250:             p->name,
251:             p->birthday.y,
252:             p->birthday.m,
253:             p->birthday.d,
254:             p->address,
255:             p->comment);
256:     }
257:
258:     fclose(file);
259:     printf("Data written to %s successfully.\n", filename);
260: }
261:
262: void cmd_find(char *word) {
263:     // fprintf(stderr, "Find (%F) command is invoked with arg: '%s'\n", word);
264:     int i;
265:     struct profile *p ;
266:
267:
268:     for(i=0; i<sizeof(profile_data_store)/sizeof(profile_data_store[0]); i++) {
269:
270:         p = &profile_data_store[i];
271:         char s[20];
272:         sprintf(s, "%d", p->id);
273:         if (strcmp(s, word) == 0){
274:
275:             printf("idが一致しました\n");
276:             printf("Id      : %d\n", profile_data_store[i].id);
277:             printf("Name    : %s\n", profile_data_store[i].name);
278:             printf("Birth   : %04d-%02d-%02d\n", profile_data_store[i].birthday.y, profile_data_st
279:             printf("Addr.   : %s\n", profile_data_store[i].address);
280:             printf("Comm.   : %s\n\n", profile_data_store[i].comment);
281:             continue;
282:         }
283:
284:
285:         if (p->comment && (!strcmp(word, p->comment) || !strcmp(word, p->name) || !strcmp(word,
286:             printf("名前,住所,コメントのいずれかがヒットしました\n");
287:             printf("Id      : %d\n", profile_data_store[i].id);
288:             printf("Name    : %s\n", profile_data_store[i].name);
289:             printf("Birth   : %04d-%02d-%02d\n", profile_data_store[i].birthday.y, profile_data_st
290:             printf("Addr.   : %s\n", profile_data_store[i].address);
291:             printf("Comm.   : %s\n\n", profile_data_store[i].comment);
292:             continue;
293:         }
294:
295:
296:         char birth[20];
297:         sprintf(birth, "%04d-%02d-%02d", p->birthday.y, p->birthday.m, p->birthday.d);
298:         if (!strcmp(word, birth)) {
299:             printf("誕生日が一致しました\n");
300:             printf("Id      : %d\n", p->id);
301:             printf("Name    : %s\n", p->name);
302:             printf("Birth   : %04d-%02d-%02d\n", p->birthday.y, p->birthday.m, p->birthday.d);
303:             printf("Addr.   : %s\n", p->address);
304:             printf("Comm.   : %s\n\n", p->comment);
305:         }
306:     }
307: }
308:

```

```

309: void cmd_sort(int column) {
310:     // fprintf(stderr, "Sort (%%S) command is invoked with arg: '%d'\n", column);
311:     int i = 0, j = 0;
312:
313:     switch(column){
314:
315:     case 1:
316:         for (i = 0; i < profile_data_nitems; i++) {
317:             for (j = 0; j < profile_data_nitems - 1; j++) {
318:                 sort_id(&profile_data_store[j], &profile_data_store[j+1]);
319:             }
320:         }
321:         break;
322:     case 2:
323:         int i = 0, j = 0;
324:         for (i = 0; i < profile_data_nitems; i++) {
325:             for (j = 0; j < profile_data_nitems - 1; j++) {
326:                 sort_name(&profile_data_store[j], &profile_data_store[j+1]);
327:             }
328:         }
329:         break;
330:     case 3:
331:
332:         for (i = 0; i < profile_data_nitems; i++) {
333:             for (j = 0; j < profile_data_nitems - 1; j++) {
334:                 sort_birthday(&profile_data_store[j], &profile_data_store[j+1]);
335:             }
336:         }
337:         break;
338:     case 4:
339:         for (i = 0; i < profile_data_nitems; i++) {
340:             for (j = 0; j < profile_data_nitems - 1; j++) {
341:                 sort_add(&profile_data_store[j], &profile_data_store[j+1]);
342:             }
343:         }
344:         break;
345:     case 5:
346:         for (i = 0; i < profile_data_nitems; i++) {
347:             for (j = 0; j < profile_data_nitems - 1; j++) {
348:                 sort_comm(&profile_data_store[j], &profile_data_store[j+1]);
349:             }
350:         }
351:         break;
352:     }
353: }
354: void cmd_delete(int id){
355:     int i=0, j=0;
356:     for (i = 0; i < profile_data_nitems; i++){
357:         if (profile_data_store[i].id == id){
358:             free(profile_data_store[i].comment);
359:             printf("Memory was freed successfully\n");
360:             for(j = i; j < profile_data_nitems; j++){
361:                 profile_data_store[j] = profile_data_store[j+1];
362:             }
363:             profile_data_nitems--;
364:             i--;
365:         }
366:     }
367: }
368:
369:
370: }
371:

```

```

372: void exec_command(char cmd, char *param) {
373:     switch (cmd) {
374:         case 'Q': cmd_quit(); break;
375:         case 'C': cmd_check(); break;
376:         case 'P': cmd_print(atoi(param)); break;
377:         case 'R': cmd_read(param); break;
378:         case 'W': cmd_write(param); break;
379:         case 'F': cmd_find(param); break;
380:         case 'S': cmd_sort(atoi(param)); break;
381:         case 'D': cmd_delete(atoi(param)); break;
382:
383:         default:
384:             fprintf(stderr, "Invalid command %c: ignored.\n", cmd);
385:             break;
386:     }
387:
388: }
389:
390: void parse_line(char *line) {
391:     char cmd;
392:     char *param;
393:
394:     if (line[0] == '%') {
395:         cmd = line[1];
396:         param = line + 3;
397:         exec_command(cmd, param);
398:     } else {
399:         if (profile_data_nitems < MAX_PROFILES) {
400:             struct profile_data new_profile_data;
401:             if (new_profile(&new_profile_data, line) != NULL) {
402:                 profile_data_store[profile_data_nitems] = new_profile_data;
403:                 profile_data_nitems++;
404:             } else {
405:                 fprintf(stderr, "Failed to create profile.\n");
406:             }
407:         } else {
408:             fprintf(stderr, "Maximum number of profiles reached.\n");
409:         }
410:     }
411: }
412: void free_profiles() { /*メモリ解放用関数*/
413:     int i = 0;
414:     for (i = 0; i < profile_data_nitems; i++) {
415:         free(profile_data_store[i].comment);
416:     }
417: }
418: int main() {
419:     char line[MAX + 1];
420:     while (get_line(line)) {
421:         parse_line(line);
422:     }
423:     free_profiles();
424:     return 0;
425: }
426:
427:

```