複合データのソート・表探索

郵便番号データのソート・検索

郵便番号データベース（元）

- 約12万5千件

0600000 北海道札幌市中央区以下に掲載がない場合
0640941 北海道札幌市中央区旭ケ丘
0600041 北海道札幌市中央区大通東
0600042 北海道札幌市中央区大通西（１～１９丁目）
0640820 北海道札幌市中央区大通西（２０～２８丁目）
0600031 北海道札幌市中央区北一条東
0600001 北海道札幌市中央区北一条西（１～１９丁目）
0640821 北海道札幌市中央区北一条西（２０～２８丁目）
0600032 北海道札幌市中央区北二条東

郵便番号データベース（ソート済み）

- 約12万5千件

0010000 北海道札幌市北区以下に掲載がない場合
0010010 北海道札幌市北区北十条西（１～４丁目）
0010011 北海道札幌市北区北十一条西（１～４丁目）
0010012 北海道札幌市北区北十二条西（１～４丁目）
0010013 北海道札幌市北区北十三条西（１～４丁目）
0010014 北海道札幌市北区北十四条西（１～４丁目）
0010015 北海道札幌市北区北十五条西（１～５丁目）
0010016 北海道札幌市北区北十六条西（１～６丁目）
0010017 北海道札幌市北区北十七条西（１～６丁目）

課題

1. 郵便番号または住所（の任意の一部）から検索（線形探索）。ただし、「郵便番号　住所」一括を検索結果とする
   - 構造体使用せず。文字列検索関数strstr()を利用
2. 郵便番号による住所検索（2分探索）
   - 構造体使用
3. 郵便番号でバブルソート・クイックソート、速度を確認
   - 構造体使用
4. 郵便番号または住所（の任意の一部）から検索（線形探索）。ただし、郵便番号か住所のいずれ（つまり答え）のみを検索結果とする…発展
   - 構造体使用せず。文字列検索関数strstr()を利用
   - ポインタを活用

課題1：郵便番号または住所（の任意の一部）から検索（線形探索）

- 「郵便番号 住所」を一括出力する
- 該当なしの場合は出力なし
- 実行例：

```
./a.out
input search key (end with 'q'): 2790031
2790031 千葉県浦安市舞浜
input search key (end with 'q'): 此花区桜島
5540031 大阪府大阪市此花区桜島
input postal code (end with 'q'): 5202194   ← 該当なし（龍谷大学瀬田学舎）
input search key (end with 'q'): q
```

13万件もデータがあるのに、なぜか大学は「該当なし」（他大学も）。みなさんの自宅の郵便番号はたぶんすべて検索可能！

課題1の処理手順

1. 検索したい郵便番号、または住所の一部を標準入力から文字列変数keyに読み込む。特定の記号（上記例では'q'）が入力されれば処理終了
2. 郵便番号・住所のデータを1行、文字列変数lnに読み込む
3. ln中にkeyが存在するかをチェックし、存在すればlnを標準出力
4. ファイルの終わりであれば、手順1.へ。でなければ、手順2.へ

（線形探索）

課題1のプログラミング

- 構造体は使わない。1次元の文字列変数を使う
- keyへの「検索」文字列読み込みはscanf関数を使えばよい
- lnへの「郵便番号・住所」文字列読み込みは後述のfgets関数を使う必要がある
  - 郵便番号・住所のデータも一行ずつ読み込んで処理するので、2次元配列は不要
  - ファイルから1行ずつ読み込むプログラムの書き方は後のプログラム例を参照すればよい
- ln中にkeyあるかのチェックはstrstr関数を用いればよいでしょう
- 手順1に戻る度に、郵便番号・住所のデータを先頭から検索していく必要があるので、ファイルポインタをファイルの先頭に戻す処理が必要。つまり、rewind関数を用いる必要がある（後のプログラム例を参照）

文字列の入出力関数

関数名	使い方	戻り値	機能
fgets	#define N 64 char s[N]; fgets(s, sizeof(s), fp);	成功：sを返すが、ファイルの終わりまたは失敗：NULL	fpが示す位置から文字列をsに読み込む。文字は ・sizeof(s)-1個の文字まで.....sizeof(s)=N ・改行まで ・ファイルの終わりまで のいずれかの条件が満たされるまで読み込まれる
fputs	#define N 64 char s[N]; fputs(s, fp);	成功：非負の値 失敗：EOF	fpが示す位置に文字列sを書き出す

文字列のフォーマット化入出力関数

関数名	使い方	戻り値	機能
fscanf	fscanf(fp, “...”, ...);	成功：入力項目数 ファイルの終わりまたは失敗：EOF	scanfと同じ。ただし、標準入力からでなくファイルから読み込む
fprintf	fprintf(fp, “...”,...);	成功：書き出された文字の数 失敗：負の値	printfと同じ。ただし、標準出力でなくファイルに書き出す

scanf/fscanfとfgetsの違いについて

- 文字列“ab cd”を文字列変数sに読み込むとき、
  - s: ab ← scanf(“%s”, s)/fscanf(fp, “%s”, s)
  - s: ab cd ¥n ← fgets(s, sizeof(s), stdin)/ fgets(s, sizeof(s), fp)

- ちなみに、
  - s[0]: a, s[1]: b, s[2]: ¥0
  - s[0]: a, s[1]: b, s[2]: (半角空白), s[3]: c, s[4]: d, s[5]: ¥n, s[6]: ¥0
    （s[5]=‘¥0’;で改行記号を除去することができる）

文字列処理

関数名	使い方	戻り値の型	機能
strcmp	strcmp(s, t);	int	文字列sとtを比較。s==tなら0を、s<tなら負の値を、s>tなら正の値を返す。ここでの大小は辞書順である。また、大文字<小文字
strlen	strlen(s);	int	sの長さ（文字数）を返す
strcpy	strcpy(s, t);	char *	文字列tをsにコピーし、sを返す
strstr	strstr(s, t)	char *	s中にtが最初に現れる位置へのポインタ、ないときはNULLを返す。

- ヘッダファイル<string.h>が必要
- ライブラリ関数内では、変数s：char *, 変数t：const char *
  - ✓ main関数側であればchar s[N];　1次元配列の場合
  - ✓ 関数の引数であればchar *s, or, char s[]; 1次元配列の場合

ファイルの入出力について

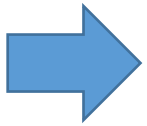
- 複数のデータファイルを読み書きする場合や、今回のような、住所データをファイルから読み込み、検索用の郵便番号を標準入力から入力する場合のプログラミングでは、ファイルの入出力機能が必要となる
- 例：

ipt1.dat

```
Tanaka  79
Nakata  73
```

ipt2.dat

```
Hashimoto  89
Nakahashi  67
Nakamoto  99
```



opt.dat

```
Tanaka    79
Nakata    73
Hashimoto  89
Nakahashi    67
Nakamoto    99
```

プログラム例

```
#include <stdio.h>
#define N 64

int main(void)
{
  char line[N];
  FILE *fp_ipt1, *fp_ipt2, *fp_opt;

  fp_ipt1=fopen(“ipt1.dat”, “r”);
  fp_ipt2=fopen(“ipt2.dat”, “r”);
  fp_opt=fopen(“opt.dat”, “w”);
```

```
  while(fgets(line, N, fp_ipt1) != NULL)
     fputs(line, fp_opt);
  while(fgets(line, N, fp_ipt2) != NULL)
     fputs(line, fp_opt);
   ⋮
   rewind(fp_ipt1); rewind(fp_ipt2);
   ⋮
  fclose(fp_ipt1);
  fclose(fp_ipt2);
  fclose(fp_opt);

  return 0;
}
```

ファイルのopen/close関数

- ファイルは読み書きを行う前にオープンし、使い終わったらクローズする

関数名	使い方	戻り値	機能
fopen	FILE *fp; fp=fopen(file, mode);	成功：ファイルポインタ 失敗：NULL	ファイルをmodeの状態でオープンする
fclose	FILE *fp; fclose(fp);	成功: 0 失敗：EOF	ファイルをクローズする
fflush	FILE *fp; fflush(fp);	成功：0 失敗：EOF	バッファをファイルに書き出す

FILE *fp; ファイルポインタ変数の宣言

ファイルオープンのmode

mode	動作	ファイルが存在しない場合	ファイルが存在した場合
“r”	ファイルから読み込み	エラー	
“w”	ファイルへ書き出し	作成する	内容は失われる
“a”	ファイルへ追加書き出し	作成する	ファイルの最後から追加

ファイル位置関数

関数名	使い方	戻り値	機能
feof	feof(fp);	ファイルの終わりでない：0 ファイルの終わり：≠0	fpで示すファイルがファイルの終わり（エンドオブファイル）かどうかを調べる
fseek	fseek(fp, offset, origin);	実行終了：0 エラー：≠0	fpで示すファイルポインタをoriginからoffset文字（バイト）だけ離れたところにセット。originは SEEK_SET: ファイルの始め SEEK_CUR: 現在位置 SEEK_END: ファイルの終わり
rewind	rewind(fp);	実行終了：0 エラー：≠0	fpで示すファイルポインタをファイルの先頭に移動する

課題2の処理手順

1. 郵便番号・住所のデータ（ソート済み）を配列に読み込む。ただし、
   - 配列を郵便番号と住所をメンバーとする構造体の配列とする
   - いずれのメンバーも文字列変数とする（郵便番号も0で始まるものがありintは困難）
2. 検索したい郵便番号を標準入力。特定の記号（上記例では’q’）が入力されれば処理終了
3. 2分探索関数を呼び出し、郵便番号で探索し、配列中の位置情報（該当なしなら-1）を返す
4. その位置情報の中身が-1でなければ、該当する配列要素の住所メンバーの値を出力する。そうでなければ「該当なし」を出力する
5. 手順2.に戻る

（2分探索）

課題2：郵便番号による住所探索（2分探索）

- 実行例：

./a.out

input postal code (end with 'q'): 5540031

大阪府大阪市此花区桜島　　← USJ

input postal code (end with ‘q’): 5202194　← 龍谷大学瀬田学舎

該当なし

input postal code (end with 'q'): 2790031

千葉県浦安市舞浜　　← ディズニーランド

input postal code (end with 'q'): q　　← 終了

課題2のプログラミング

- 上記処理手順に沿ってプログラムの作成を行う
- プログラムの作成は後の「構造体の使用例」を参考にするとよい。ただし、
  1. 構造体は以下のように定義するとよいでしょう

     typedef struct rc{

       char pc[N], add[N]; // pc: postal code 郵便番号, add: address 住所

     } RC
  2. 2分探索関数をint BSearch(int n, int a[], int x)からint BSearch(int n, RC a[], char key[])に変更
  3. 構造体の配列が大きすぎるため、「構造体の使用例」のように宣言すると、実行時にsegmentation errorが出るかも。そのとき

     main関数側のみにおいて、RC a[M]の代わりにstatic RC a[M]で宣言するとよいでしょう
  4. a[i].pcとkeyが両方とも文字列変数なので、それらの比較には文字列の比較関数strcmpを用いなければならない

構造体とは

- 構造体は、char, int, doubleなど異なるデータ型を必要に応じてまとめ、自ら定義したデータ型である
- たとえば以下のような表を扱うとき、1レコード（1行）を1データとして考えて、それを1つの構造体で定義すると便利

学籍番号	氏名	所属学科	成績
000010	龍谷太朗	数理情報	100
000105	瀬田花子	電子情報	90
⋮	⋮	⋮	⋮
000100	大津三郎	知能情報	80

構造体の定義(例)

学籍番号	氏名	所属学科	成績
000010	龍谷太朗	数理情報	100
000105	瀬田花子	電子情報	90
⋮	⋮	⋮	⋮
000100	大津三郎	知能情報	80

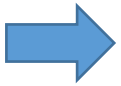
```
typedef  struct rc{
  char id[128], name[128], dep[128];
  int score;
} RC;
```

メンバー

構造体の使用例：表データを読み込み任意のフィールドでソートし書き出すプログラム

ipt.dat

000001	龍谷太郎	知能	99
000101	瀬田花子	数理	100
000003	大津次郎	電子	95
000210	滋賀三郎	数理	90
000540	京都四郎	電子	98



opt.dat

000101	瀬田花子	数理	100
000210	滋賀三郎	数理	90
000001	龍谷太郎	知能	99
000003	大津次郎	電子	95
000540	京都四郎	電子	98

構造体の使用例：表データを読み込み任意のフィールドでソートし書き出すプログラム

```
#include <stdio.h>
#define N 128
#define M 1000 // レコードの数

typedef  struct rc{
  char id[N], name[N], dep[N];
  int score;
} RC;

void QSort(int n, RC a[]){...}

int main(void)
{
  int i, n=0;
  RC a[M];
  FILE *fp_ipt, *fp_opt;
  fp_ipt=fopen("ipt.dat", "r");
  fp_opt=fopen("opt.dat", "w");

  while(fscanf(fp_ipt, "%s %s %s %d",
                  a[n].id, a[n].name, a[n].dep, &a[n].score) != EOF)
     n++;

  QSort(n, a); // 指定フィールドでソート

  for(i=0; i<n; i++)
    fprintf(fp_opt, "%s %s %s %d", a[i].id, a[i].name, a[i].dep, a[i].score);

  fclose(fp_ipt);
  fclose(fp_opt);
  return 0;
}
```

課題3：郵便番号データを郵便番号でソートする

- 課題2と同様の構造体を使用
- 構造体の配列は大きすぎるため、RC a[M]の代わりにstatic RC a[M]で宣言するとよいでしょう
- バブルソート関数をvoid BSort(int n, int a[])からvoid BSort(int n, RC a[])に、クイックソート関数をvoid QSort(int f, int t, int a[])からvoid QSort(int f, int t, RC a[])に変更
- a[i].pcとkeyが両方とも文字列変数なので、それらの比較には文字列の比較関数strcmpを用いなければならない（文字列の比較という意味で、前回の講義資料ALGO-4Q-2-表探索-文字列データ.pdfが参考になるかも）

課題3：郵便番号データを郵便番号でソートする

- 約12万5千件（元データ）

最初部分のデータは以下の通り：

0600000 北海道札幌市中央区以下に掲載がない場合
0640941 北海道札幌市中央区旭ケ丘
0600041 北海道札幌市中央区大通東
0600042 北海道札幌市中央区大通西（１～１９丁目）
0640820 北海道札幌市中央区大通西（２０～２８丁目）
0600031 北海道札幌市中央区北一条東
0600001 北海道札幌市中央区北一条西（１～１９丁目）
0640821 北海道札幌市中央区北一条西（２０～２８丁目）
0600032 北海道札幌市中央区北二条東

約12万5千件（ソート後のデータ）

最初部分のデータは以下の通り：

0010000 北海道札幌市北区以下に掲載がない場合
0010010 北海道札幌市北区北十条西（１～４丁目）
0010011 北海道札幌市北区北十一条西（１～４丁目）
0010012 北海道札幌市北区北十二条西（１～４丁目）
0010013 北海道札幌市北区北十三条西（１～４丁目）
0010014 北海道札幌市北区北十四条西（１～４丁目）
0010015 北海道札幌市北区北十五条西（１～５丁目）
0010016 北海道札幌市北区北十六条西（１～６丁目）
0010017 北海道札幌市北区北十七条西（１～６丁目）

課題4：郵便番号または住所（の任意の一部）から検索（発展）

- 答え（郵便番号か住所の）答えのみを出力する
- 該当なしの場合は出力なし
- 実行例：

```
./a.out
input search key (end with 'q'): 2790031
千葉県浦安市舞浜
input search key (end with 'q'): 大阪市此花区桜島
5540031
input search key (end with 'q'): 5202194   ← 該当なし（龍谷大学瀬田学舎）
input search key (end with 'q'): q
```

課題4の処理手順

1. 検索したい郵便番号、または住所の一部を標準入力から文字列変数keyに読み込む。特定の記号（上記例では'q'）が入力されれば処理終了
2. 郵便番号・住所のデータを1行、文字列変数lnに読み込む
3. ln中にkeyが存在するかをチェックし、ln中のkeyの出現位置（ポインタp）を得る（存在しなければp=NULL）
   3.1 pがlnの先頭を指すかどうかをチェックし、yesなら郵便番号での検索がわかるため、住所部分を標準出力
   3.2 noでかつNULLでなければ、住所で検索がわかるので、郵便番号部分を出力
4. ファイルの終わりであれば、手順1.へ。でなければ、手順2.へ

（線形探索）

補足

- 課題2のように、（fscanfを用いて）一行を郵便番号と住所に分けて読み込み、keyとこれらの比較で答えのみを出力する方法も考えられる
- しかし、この方法だと、汎用性がない。つまり、郵便番号と住所を空白でなくカンマで区切る場合や、住所中に空白があると、うまく行かなくなる
- 以下の例が、郵便番号の元データ（CSVデータ）。本授業ではすでにプログラミングしやすいように整形されている

"0640941","北海道","札幌市　中央区","旭ケ丘","HOKKAIDO","SAPPORO SHI CHUO KU","ASAHIGAOKA"

"0600041","北海道","札幌市　中央区","大通東","HOKKAIDO","SAPPORO SHI CHUO KU","ODORIHIGASHI"

第3回演習課題

1. 講義資料の「課題1：郵便番号または住所からの検索（郵便番号と住所を一括出力する）」プログラム（線形探索）を作成しなさい（ex03-pcodeLS.c）。郵便番号・住所データ（pcode.dat）は講義資料のWebサイトからダウンロードして使ってください。

   実行例（よかったら自宅などを試してみてください）：

   ./a.out

   input search key (end with 'q'): 此花区桜島

   5540031 大阪府大阪市此花区桜島　　← USJ

   input search key (end with 'q'): 2790031

   2790031 千葉県浦安市舞浜　　← ディズニーランド

   input search key (end with 'q'): あいうえお　　← 該当なしの場合は出力なし

   input search key (end with 'q'): q　　← 終了

第3回演習課題

2. 郵便番号から住所を検索する、講義資料の「課題2：郵便番号による住所検索」のプログラム（2分探索）を作成しなさい（ex03-pcodeBS.c）。郵便番号・住所のソート済みデータ（pcode-sort.dat）は講義資料のWebサイトからダウンロードして使ってください。

実行例：

./a.out

input postal code (end with 'q'): 5540031

大阪府大阪市此花区桜島

input postal code (end with 'q'): 5202194

該当なし

input postal code (end with 'q'): 2790031

千葉県浦安市舞浜

input postal code (end with ‘q’): q

第3回演習課題

3. 郵便番号・住所データ（pcode.dat）を郵便番号で（バブル・クイック）ソートするプログラム（ex03-pcode-sort.c）を作成しなさい。ただし、ソート結果をpcode-sort-new.datに書き出す。また、2つのソート法を2つの関数として定義し、それぞれ呼び出すことでプログラムを実行する（main関数で実行時に片方をコメントアウトする方法でOK）。そして、

① 両者を実行してみてその処理時間の差を観察しよう。

② Ubuntu端末上で

```
diff  pcode-sort.dat pcode-sort-new.dat
```

を両ソート手法について実行してみよう。diffコマンドの意味を調べるとともに、結果について考察してみよう。

第3回演習課題

4. （発展問題）講義資料の「課題4：郵便番号または住所からの検索（答えのみを出力する」プログラム（線形探索）を作成しなさい（ex03-pcodeLS-adv.c）。郵便番号・住所のデータはpcode.datを用いる。

実行例：

./a.out

input search key (end with 'q'): 此花区桜島

5540031

input search key (end with 'q'): 2790031

千葉県浦安市舞浜

input search key (end with ‘q’): あいうえお　←　該当なしの場合は出力なし

input search key (end with 'q'): q