プログラミングおよび実習II・第 12 回

2003 年 12 月 10 日

1 実力チェック

DateOfBirth という名前のテキストファイルに、300 名余の名前と生年月日が次のような形式で格納されている。

```
Mika 1983 10 23
Hiroyuki 1982 3 11
Yuka 1983 5 13
Ken 1982 8 19
Daisuke 1983 12 10
  ⋮
Jiro 1981 3 11
```

次の birthday.c というプログラムは、このファイルを読み込んだ後、月と日を標準入力 (キーボード) から入力すると、その日が誕生日である人の情報をすべて出力するものである。実行例を参考にして、このプログラムの関数 main を完成しなさい。ただし、解答用紙のプログラム名は「birthday.c」とし、関数 main の定義のみを書きなさい。

birthday.c

```
#include <stdio.h>
#include <stdlib.h>

#define MAX_DATA        (1000)
#define NAME_LEN        (100)

typedef struct {
    char name[NAME_LEN];
    int year;
    int month;
    int day;
} Birthday;

void list(Birthday b[], int num, int m, int d)
{
    int i;

    for (i = 0; i < num; i++) {
        if (b[i].month == m && b[i].day == d) {
            printf("%12s %d %d %d\n", b[i].name,
                   b[i].year, b[i].month, b[i].day);
        }
    }
}

int main()
{
    ⋮
    return EXIT_SUCCESS;
}
```

birthday.c の実行例

```
s1542h017% ./birthday
誕生日を入力してください : 5 13
        Yuka 1983 5 13
```

```
誕生日を入力してください : 3 11
    Hiroyuki 1982 3 11
        Miki 1983 3 11
        Jiro 1981 3 11
誕生日を入力してください : 10 10
誕生日を入力してください : 10 23
        Mika 1983 10 23
誕生日を入力してください : .
s1542h017%
```

2 二分探索法を用いて郵便番号から住所を検索する

前回、郵便番号の一覧が格納されたファイル /home/sample/mprog2/zipcode を読み込んで、線形探索法で検索を行うプログラムを作成しましたが、今回は、二分探索法[1]を用いて同じことを行うプログラム prog12-1.c を作ってみましょう。このプログラムの実行例も前回の prog11-1.c と同様になるはずです。ただし、入力された郵便番号を持つ行が複数ある場合に、どの行が出力されるかは予想が付きません。

実現の方針 (1) 前回の prog11-1.c と同様に、ファイル /home/sample/mprog2/zipinfo から読み取ったデータは、ZipInfo 型の配列 data に格納しておく。また、読み取ったデータの件数を int 型の変数 num に格納する。

(2) ファイルの内容を読み込む配列 data とは別に、ZipInfo 型へのポインタ型の配列 dp を用意しておき、各要素を data 中の対応する要素のアドレスで初期化する。

(3) 配列 dp の各要素が指す構造体 (ZipInfo 型) の code の部分が昇順になるように、dp をクイックソート法で整列する。

(4) 探索すべき郵便番号 (整数) を標準入力 (キーボード) から scanf を使って読み取り、配列 dp に対して二分探索を行って、見つかった情報を標準出力 (端末ウィンドウ) に出力する。関数 main は scanf が整数 (郵便番号) の読み込みに失敗するまで、これを繰り返す。

(5) クイックソート法による整列は、次のように宣言することのできる関数として定義して、この関数を main から呼び出すようにする。

```
void quicksort(ZipInfo *d[], int h, int t);
```

(6) 二分探索法による探索は、次のように宣言することのできる関数として定義しておき、これを main から呼び出すようにする。

```
int binarysearch(ZipInfo *d[], int num, int code);
```

関数 binarysearch は、整列済みの配列 d から、引数 code を郵便番号として持つ要素[2]を二分探索法を用いて探し、見つかった要素の添字を返す。見つからなかった場合は -1 を返す。

[1] 教科書「C 言語によるアルゴリズムとデータ構造入門」の 138 ページからを適宜参照してください。

[2] その要素の指す先の ZipInfo 型のデータの code メンバが、関数 binarysearch の引数 code と等しいものを探します。

関数 binarysearch で実現する二分探索のアルゴリズムは次のようなものとなります。

アルゴリズム (1) 探索範囲となる配列 d、その要素数 num、見つけたい郵便番号 $code$ を引数として受け取る。

(2) 配列 d の先頭と末尾の要素の添字を、それぞれ l と h とする。

(3) $l \leqq h$ が成り立っている間、次の 4 つ手順を繰り返す。

(3-1) l と h の平均 (小数点以下切り捨て) を m とする。

(3-2) 配列 d の添字 m の要素が指す構造体の code メンバが、見つけたい郵便番号 $code$ より大きい場合は、h を $m-1$ に変更する。

(3-3) 小さい場合は l を $m+1$ に変更する。

(3-4) 等しい場合は二分探索を終了し、m を返り値として呼び出し元へ戻る。

(4) -1 を返り値として呼び出し元へ戻る。

3 演習問題

次の演習問題に取り組み、prog12-1.c については自分が作成したプログラムを mprog2 コマンドで提出してください。教員か TA によるチェックも必要です。

3.1 prog12-1.c

「2. 二分探索法を用いて郵便番号から住所を検索する」の節で解説した実現の方針に従って、プログラム prog12-1.c を完成しなさい。プログラムが完成したら mprog2 コマンドで提出してください。教員か TA によるチェックも必要となります。関数 quicksort や関数 binarysearch をこのような名前で定義しないと mprog2 コマンドが受理しませんので注意して下さい。前回の prog11-1.c をコピーして改造すると簡単です。関数 quicksort の定義は prog10-3.c などのプログラム中のもの流用して修正しましょう。

3.2 prog12-2.c と prog12-3.c

余裕のある受講者は、第 9 回に解説したのと同様の方法で、線形探索法 (prog11-1.c) と二分探索法 (prog12-1.c) とで、1 回の探索を行うときの所要時間を調べて表示するようなプログラム prog12-2.c と prog12-3.c を、それぞれ作成してみましょう。この問題のプログラムは mprog2 コマンドで提出する必要はありません。

/home/sample/mprog2/zipinfo を配列に読み込んだら、その先頭からコマンドライン引数で指定された個数だけの要素を探索範囲とします。探索範囲に含まれる郵便番号の最小値と最大値を調べて、その範囲の乱数を rand 関数を使って生成し[3]、この郵便番号を探索範囲から探索するときに要する時間を測定しましょう。2 分探索法の場合は、読み込んだデータの整列が完了した状態で、1 回の探索に必要な時間を計測します。

[3]最小値が min、最大値が max であるとき、rand() % (max - min + 1) + min で生成しましょう。

次は prog12-2.c (線形探索) と prog12-3.c (二分探索) の実行例です。

prog12-2.c と prog12-3.c の実行例

```
s1542h017% ./prog12-2 10
10件からの探索に要した時間 (78.950 - 58.850) / 134477173 = 0.0000001495 秒
s1542h017% ./prog12-2 100
100件からの探索に要した時間 (13.770 - 3.720) / 8018346 = 0.0000012534 秒
s1542h017% ./prog12-2 1000
1000件からの探索に要した時間 (10.310 - 0.280) / 719060 = 0.0000139488 秒
s1542h017% ./prog12-2 10000
10000件からの探索に要した時間 (10.000 - 0.000) / 12974 = 0.0007707723 秒
s1542h017% ./prog12-2 100000
100000件からの探索に要した時間 (10.000 - 0.000) / 1135 = 0.0088105727 秒
s1542h017% ./prog12-3 10
10個の要素からの探索に要した時間 (34.490 - 24.870) / 57179413 = 0.0000001682 秒
s1542h017% ./prog12-3 100
100件からの探索に要した時間 (24.800 - 16.190) / 35429320 = 0.0000002430 秒
s1542h017% ./prog12-3 1000
1000件からの探索に要した時間 (20.000 - 10.010) / 22858177 = 0.0000004370 秒
s1542h017% ./prog12-3 10000
10000件からの探索に要した時間 (15.960 - 6.170) / 14154099 = 0.0000006917 秒
s1542h017% ./prog12-3 100000
100000件からの探索に要した時間 (12.660 - 2.900) / 6380104 = 0.0000015298 秒
```

プログラミングおよび実習Ⅱ・第 12 回・終り

付録. 前回の演習問題の解答例

prog11-1.c

```
#include <stdio.h>
#include <stdlib.h>

#define FILE_NAME       "/home/sample/mprog2/zipcode"

#define MAX_DATA        (200000)
#define PREF_NAME_LEN   (20)
#define CITY_NAME_LEN   (40)
#define AREA_NAME_LEN   (100)

typedef struct {
    int code;                   /* 郵便番号 */
    char pref[PREF_NAME_LEN];   /* 都道府県名 */
    char city[CITY_NAME_LEN];   /* 市町村名 */
    char area[AREA_NAME_LEN];   /* 町域名 */
} ZipInfo;

int linearsearch(ZipInfo data[], int num, int code)
{
    int i;

    for (i = 0; i < num; i ++) {
        if (data[i].code == code)
            return i;
    }
    return -1;
}

int main()
{
    static ZipInfo data[MAX_DATA];
    FILE *fp;
    int code;
    int num, pos;

    fp = fopen(FILE_NAME, "r");
    if (fp == NULL) {
        printf("%s というファイルが読めません\n", FILE_NAME);
        exit(EXIT_FAILURE);
    }

    num = 0;
    while (fscanf(fp, "%d %s %s %s", &data[num].code,
                  data[num].pref, data[num].city, data[num].area) == 4) {
        num ++;
    }

    fclose(fp);
    printf("%s から %d 件のデータを読み込みました\n", FILE_NAME, num);

    while (1) {
        printf ("郵便番号を入力してください : ");
        if (scanf("%d", &code) != 1)
            break;
        pos = linearsearch(data, num, code);
        if (pos < 0)
```

```
            printf("対応する地域がありません。\n");
        else
            printf("%07d %s %s %s\n", data[pos].code,
                   data[pos].pref, data[pos].city, data[pos].area);
    }
    printf("検索を終了します。\n");
    return EXIT_SUCCESS;
}
```