プログラミングおよび実習II・第 10 回

2003 年 11 月 26 日

1　実力チェック

次のプログラムを実行した時に表示される内容を解答用紙に書きなさい。解答用紙のプログラム名は「strange.c」としなさい。

strange.c

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3
 4 struct Rec {
 5     char s[100];
 6     int y;
 7     int x;
 8 };
 9
10 int strange(struct Rec *p, int x)
11 {
12     printf("A\n");
13     if (x <= 1) {
14         p->y++;
15         printf("%s\n", p->s);
16         return x;
17     }
18     p->x++;
19     printf("B\n");
20     return strange(p, x-1) + x;
21 }
22
23 int main()
24 {
25     struct Rec r;
26
27     strcpy(r.s, "r.s");
28     r.x = r.y = 0;
29     printf("C\n");
30     printf("%d\n", strange(&r, strange(&r, 3)));
31     printf("D\n");
32     printf("%d\n%d\n", r.x, r.y);
33     return EXIT_SUCCESS;
34 }
```

2　いろいろな初期値で整列時間を計測する

実用に供されるアプリケーションプログラムの中で、選択ソート、バブルソート、クイックソートなどのソーティングアルゴリズムが使用される場合の状況はいろいろと考えられます。まったくばらばらな状態の配列を整列したいこともあるでしょうし、ほとんどの部分はすでに整列されている状態の配列を完全に整列したい場合もあることでしょう。与えられた状況が変れば、それに適したソーティングアルゴリズムも変ってくると予想できます。前回までの演習問題で作成した時間測定のプログラム群は、たとえば

```
for (i = 0; i < num; i++)
    data[i].score = rand();
```

のように、関数 rand の返す乱数を用いて Record 型の配列 data 中の各要素の score の部分 (メンバ) を初期化していましたが[1]、これを

```
for (i = 0; i < num; i++)
    data[i].score = i;
```

や

```
for (i = 0; i < num; i++)
    data[i].score = num - i;
```

のようにすると、始めから昇順になっている配列や、逆に降順になっている配列を、昇順に整列するための時間を計測することができるはずです。

いろいろな初期状態の配列を整列するために費される時間を計測する実験を行う際には、配列の初期化を行う仕事を独立させて、次のような関数として定義しておくと便利です。

```
 1 typedef int Mode;
 2 #define RANDOM  (1)
 3 #define INC     (2)
 4 #define DEC     (3)
 5
 6 void initdata(Record d[], int num, Mode mode)
 7 {
 8     int i;
 9
10     switch (mode) {
11     case RANDOM:
12         for (i = 0; i < num; i++)
13             d[i].score = rand();
14         break;
15     case INC:
16         for (i = 0; i < num; i++)
17             d[i].score = i;
18         break;
19     case DEC:
20         for (i = 0; i < num; i++)
21             d[i].score = num - i;
22         break;
23     }
24 }
```

このプログラムの 1 行目では Mode を int の別名として定義しています。関数 initdata は、Record 型の配列 d と要素の個数 num (int 型)、初期化の方法 mode (Mode 型) の 3 つの引数を受け取ると、配列 d の先頭から num 個の要素を、mode で指定された方法で初期化します。引数 mode の値が 1 (RANDOM) の場合は乱数で、 2 (INC) の場合は昇順となるように、 3 (DEC) の場合は降順となるように初期化されます。

switch 文　関数 initdata で使われている swicth 文は C 言語に用意されている構文の 1 つで、一般的には

```
switch (式) 文
```

[1]第 9 回 11 ページのプログラムの 25–26 行目や 38–39 行目

の形で 1 つの文を構成します。swicth 文が実行されるときには、まず「式」の部分が評価 (計算) されます。「式」は整数値を表すもの[2]でなければなりません。この式の計算結果が整数値 k であった場合、「文」およびそれを構成している文の中から、

```
case k:
```

という形式の case **ラベル**と呼ばれる名札を持つ文が探されて、その文にジャンプします。そのような case ラベルが見つからない場合は、

```
default:
```

という名札を持つ文が探されて、そこへジャンプします。この defualt: も見つからない場合、「文」は全く実行されません。いずれかの名札にジャンプした場合は、そこから「文」の残りの部分が実行されて行きますが、その途中で break 文が実行されると、以降の部分は実行されずに switch 文全体の実行がそこで終了してしまいます。ただし、その break 文が while 文、for 文、do 文の中に出てきた場合は、その繰り返しが終了するだけで switch 文の実行に影響を与えることはありません。break 文は、それを囲んでいる最も内側の siwtch 文や while 文、for 文、do 文に対して効果を持ちます。

switch 文の「文」の部分はブロック[3]の形をしていることがほとんどで、典型的には

```
switch (式) {
case 定数式1:
    文の並び1
    break;
case 定数式2:
    文の並び2
    break;
case 定数式3:
    文の並び3
    break;
  ⋮
case 定数式n:
    文の並びn
    break;
default:
    文の並びn+1
    break;
}
```

のような形で用いられます。ただし、

```
printf("xは");
switch (x) {
case 2: case 3: case 5: case 7:
    printf("素数です\n");
    break;
case 1: case 4: case 6: case 8: case 9: case 10:
    printf("素数ではありません\n");
    break;
default:
    printf("素数かどうか分かりません\n");
}
```

[2] char や signed char、unsigned char、short、unsigned short、int、unsigned int、long、unsigned long のような整数型や enum (列挙) 型などの式となります

[3] { と }で囲まれた宣言や文の並びのことです

のように、1 つの文に複数の case ラベルを付けることも可能ですし、

```
printf("x以下の素数は");
switch (x) {
case 10: case 9: case 8: case 7:
    printf(" 7");
case 6: case 5:
    printf(" 5");
case 4: case 3:
    printf(" 3");
case 2:
    printf(" 2 です\n");
    break;
case 1:
    printf("ありません\n");
    break;
default:
    printf("いくつあるか分かりません\n");
}
```

のように、各 case ラベルに対応する文の並びの後に break 文を置かないで、そのまま switch 文の残りの部分の実行を継続することも可能です。また、これらの例では default: というラベルの付いた文の並びは最後に置かれていますが、どこに置いても構いませんし、省略してしまうこともできます。

3 CSV 形式で実験結果を出力する

いろいろな要素数といろいろな初期値を持つ配列について、いろいろなソーティングアルゴリズムによる整列時間を計測する実験を行い、そこで得られたデータを整理したり分析したりする際には、表計算ソフト[4]と総称されるアプリケーションプログラムが有用です。実験結果を簡単にグラフにすることができますし、平均値や標準偏差の計算、度数分布表 (ヒストグラム) の作成などの統計処理も容易にできます。

前回までの実験用プログラムでは、

```
1000個の要素の整列に要した時間（11.240 - 1.280）/ 47456 = 0.0002098786 秒
```

のように、人間にとって分かりやすいような形式で実験結果を出力していましたが、表計算ソフトでの読み込みを容易にするためには、要素数や測定された時間、繰り返しの回数などの各数値データを「,」(カンマ) で区切っただけの **CSV 形式**と呼ばれる形式にしておくのが便利です。 CSV 形式の場合、先の出力例は

```
1000, 11.240, 1.280, 47456, 0.0002098786
```

のようになります。また、前回に紹介したバッチファイルと (シェルの) リダイレクション機能を利用して、たとえば 5、 10、 20、 50、 100、 200、 500、 1000、 2000、 5000 の 10 通りの要素数で得られた整列時間のデータは、 CSV 形式だと次のような内容のファイルとして保存することになります。

[4] Microsoft Excel や OpenOffice Calc などのアプリケーションプログラムのことをこう呼びます

```
5, 21.590, 11.960000, 9996024, 0.0000009634
10, 15.230, 4.820000, 3177012, 0.0000032767
20, 11.720, 1.690000, 839366, 0.0000119495
50, 10.530, 0.500000, 144687, 0.0000693221
100, 10.240, 0.240000, 36162, 0.0002765334
200, 10.110, 0.110000, 8916, 0.0011215792
500, 10.040, 0.040000, 1394, 0.0071736011
1000, 10.020, 0.020000, 330, 0.0303030303
2000, 10.070, 0.010000, 82, 0.1226829268
5000, 10.240, 0.010000, 13, 0.7869230769
```

4 演習問題

以下の 3 つの演習問題に取り組み、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。今回の演習問題については教員や TA のチェックは必要ありません。

4.1 prog10-1.c

まず、前回作成した選択ソート法の時間を測定するプログラム prog9-1.c を prog10-1.c にコピーしなさい。つぎに、このプログラムを改良し、コマンドラインの第 2 引数として random、inc、dec のいずれかの文字列を与えて起動すると、それぞれ次のように初期化されたデータを整列する時間を測定して (実行例のように) CSV 形式で結果を出力するようなプログラムにしなさい。

random	配列中の各要素 (Record 型) の score の部分を関数 rand の返り値で初期化する。
inc	配列中の各要素 (Record 型) の score の部分が昇順になるように初期化する。
dec	配列中の各要素 (Record 型) の score の部分が降順になるように初期化する。

このとき、次のような実現の方針を取りなさい。

実現の方針 (1) 「2. いろいろな初期値で整列時間を計測する」の節で紹介した関数 initdata を定義しておく。

(2) 関数 checktime に Mode 型の引数を追加し、checktime の引数を関数 initdata に渡して配列の (各要素の score の部分の) 初期化を行うようにする。

(3) 関数 main では、2 番目のコマンドライン引数がどのような文字列であるかで場合分けして、関数 checktime を呼び出す際に渡す最後の (追加された) 引数の値を変えるようにする。

prog10-1.c の実行例

```
s1542h017% ./prog10-1
要素数と初期化の方法が指定されていません。
s1542h017% ./prog10-1 1000 ran
初期化方法の指定が正しくありません。
s1542h017% ./prog10-1 random 1000
要素数の指定が正しくありません。
s1542h017% ./prog10-1 1000 random
1000, 10.070, 0.050, 915, 0.0109508197
s1542h017% ./prog10-1 1000 inc
```

```
1000, 10.040, 0.040, 938, 0.0106609808
s1542h017% ./prog10-1 1000 dec
1000, 10.000, 0.020, 694, 0.0143804035
s1542h017%
```

4.2 prog10-2.c

前回作成したバブルソート法の時間を測定するプログラム prog9-2.c に対して、prog10-1.c と同様のことができるプログラム prog10-2.c を作成しなさい。

4.3 prog10-3.c

さらに、前々回のクイックソート法の時間を測定するプログラム prog8-2.c に対して、prog10-1.c と同様のことができるプログラム prog10-3.c を作成しなさい。

5 レポート課題

今回の演習問題で作成した prog10-1.c、prog10-2.c、prog10-3.c の 3 つを含むいくつかの実験用プログラムを用意して、整列に費される時間をそれぞれ調べる実験を行い、分かったことを以下の要領でレポートとして提出しなさい。Record 型へのポインタ型とは違ったデータ型 (たとえば int 型や double 型、文字列型など) の要素の整列、配布資料では扱わなかった整列アルゴリズム、異る初期値データ (乱数、昇順、降順以外) についての実験を行ってみることを奨励します。ただし、整列時間の比較を行う際には、同じ機種の計算機を用いた実験を行うように注意すること。

レポートの形式 Microsoft Word 形式 (あるいは OpenOffice Writer 形式) のファイルで 5 〜 10 ページ程度 (A4 判)

提出期限 2003 年 12 月 17 日 13:30

提出先 実習室の Windows 環境の「R:a89023\プログラミングおよび実習II」に用意されている受講者ごとの提出用フォルダ

評価方法 50 点満点で採点し、平常点 (実力チェックや演習問題などの成績) の一部とします

レポートの構成

1. 表紙 (表題、学籍番号、氏名)
2. 実験の目的
   - この実験で何を明らかにしようとしているかについて自分なりの考えを書きます。
3. 実験の方法
   - 実験に用いた計算機とプログラムはどのようなものであったか、どのように実験を進めたかなど、実験の方法について説明します。この科目を受講していない人に説明するつもりで書いてください。
4. 実験結果

- ソーティングアルゴリズム (関数)、初期値として与えられるデータの並び方 (乱数、昇順、降順等)、整列する要素数ごとに、何度か整列に費される時間を測定し、その平均値、標準偏差を求めて表にします。
- また、同じ実験結果について、ソーティングアルゴリズム (関数) の違いや初期データの並び方 (乱数、昇順、降順) の違いで、必要な時間がどのように変ってくるか、その比較がしやすいように工夫して (いくつかの) グラフにまとめます。グラフは、以下の「実験結果に対する考察」中の適当な場所に置いても構いません。

5. 実験結果に対する考察

- 実験方法や実験結果の標準偏差から見て、実験結果はおおよそどの程度の精度を持っていると考えられるか評価してください。
- 要素数を N とする場合、選択ソートやバブルソートの整列時間はおおよそ N^2 に、クイックソートの整列時間はおおよそ $N \log N$ に比例すると言われている。この仮説を採用したとして、その比例定数が要素数によってどのように変化するかを、初期データの並び方 (乱数、昇順、降順) の違いごとにグラフにするなどして、分かったことをまとめてください。
- クイックソート法では、初期データの並び方によっては、整列に必要な時間が N^2 に比例してしまうことが知られています。余裕のある受講者は、データをどのように初期化しておくとこのようなことが起きるのか調べてみましょう。
- あるソフトウエアの一部でデータのソーティングが必要となった時、プログラマはどのようなことに気をつけてソーティングアルゴリズムを選択すればよいと考えられるでしょうか?
- 実験結果のどのような点についてはプログラムを実行する計算機によらず、どのような点については計算機によって変ってくると考えられるでしょうか?
- その他、実験の結果から分かったこと、まだ分からないこと等、思い当たった事項について説明してください。

注意　実験に用いたソースプログラムやバッチファイル、 CSV 形式の実験結果のファイルは、すべて、 **Linux 環境の** `Prog2` ディレクトリに `Report` というサブディレクトリを作成して、そこに置いておくようにしてください。

プログラミングおよび実習Ⅱ・第 10 回・終り

付録 A. Linux 環境から Windows 環境へのファイル転送

計算機実習室の Linux 環境で作成したファイルは、`smbclient` というコマンドを使って、 Windows 環境へ転送することができます。次の実行例では、 Linux 環境でのカレントディレクトリに置いてある `bubble.csv` と `quick.csv` という 2 つのテキストファイルを、 Windows 環境の「マイドキュメント」

(Q: ドライブ) へ転送しています。smbclient のコマンドライン引数中の t000000 は、各自のユーザー ID に読み替えてください。「Password:」の行では、 Windows 環境のログオンパスワードを入力して Enter キーを押します。入力したパスワードは表示されませんので注意して入力して下さい。

Linux から Windows へのファイル転送

```
s1542h017% smbclient '\\rs1-file.sun.smile.ryukoku.ac.jp\t000000$' -W SMILE
added interface ip=133.83.23.17 bcast=133.83.23.255 nmask=255.255.255.0
session request to RS1-FILE.SUN.SM failed (Called name not present)
Password:
Domain=[SUN] OS=[Windows 5.0] Server=[Windows 2000 LAN Manager]
smb: \> put bubble.csv
putting file bubble.csv as \bubble.csv (20.9262 kb/s) (average 20.9263 kb/s)
smb: \> put quick.csv
putting file quick.csv as \quick.csv (20.9262 kb/s) (average 20.9263 kb/s)
smb: \> quit
s1542h017%
```

逆に、 Windows 環境から Linux 環境にファイルを転送するためには、put コマンドの代りに get コマンドを使います。

Linux 環境と Windows 環境では、オブジェクトファイルの形式や (C のソースファイルを含む) テキストファイルの日本語文字コードが異りますので、これらのファイルを 2 つの環境間で転送する際には別に注意が必要となります。また、 Windows 環境ではファイル名中の大文字小文字は区別されませんので、この点についても注意してください。

付録 B. 前回の実力チェックの解答例

関数 selectsort の定義

```
void selectsort(Record *d[], int num)
{
    Record *tmp;
    int start, pos, i;
    int min;

    for (start = 0; start < num-1; start++) {
        min = d[start]->score;
        pos = start;
        for (i = start+1; i < num; i++) {
            if (d[i]->score < min) {
                min = d[i]->score;
                pos = i;
            }
        }
        if (pos != start) {
            tmp = d[start];
            d[start] = d[pos];
            d[pos] = tmp;
        }
    }
}
```