＜演習問題＞　クイックソート

【課題問題1】

　クイックソート関数（quick）によるソーティングの処理の流れを明らかにしたい．リスト1（次ページ）に示すクイックソート関数を用いて，図1に示すように，パスごとのソーティング開始前の，(1)左端の値とそれが格納されている配列の添え字，(2)右端の値とそれが格納されている配列の添え字，(3)ピボットの値とそれが格納されている配列の添え字，左端から右端の配列の値，さらに，ソーティング結果得られる左端から右端の配列の値を表示するプログラムを作成せよ．そして，作成した関数を用いて，10個の整数からなる数列をクイックソートしたとき，上記の(1)～(3)の値を印字出力してクイックソートが正しく行われていることを考察せよ．入力データは以下のようにすること．

int a[10] = {7, 2, 5, 3, 1, 10, 6, 8, 9, 4}

注意：パスとは関数quick()が呼び出された順番を表す数字とする．

＜印字出力の形式＞

1回目の呼び出し：

　　　　　＜開始時＞

　　　　　　左端の添え字：ｘ

　　　　　　右端の添え字：ｘ

　　　　　　ピボットの値：ｘ

　　　　　　左端～右端の値：ｘ　ｘ　ｘ　ｘ　ｘ　ｘ　ｘ　ｘ　ｘ　ｘ

　　　　　＜ソーティング結果＞

　　　　　　左端～右端の値：ｘ　ｘ　ｘ　ｘ　ｘ　ｘ　ｘ　ｘ　ｘ　ｘ

2回目の呼び出し：

　　　　　＜開始時＞

　　　　　　左端の添え字：ｘ

　　　　　　右端の添え字：ｘ

　　　　　　ピボットの値：ｘ

　　　　　　左端～右端の値：ｘ　ｘ　ｘ　ｘ　ｘ

　　　　　＜ソーティング結果＞

　　　　　　左端～右端の値：ｘ　ｘ　ｘ　ｘ　ｘ

(以下，続く)

ソート終了：ｘ　ｘ　ｘ　ｘ　ｘ　ｘ　ｘ　ｘ　ｘ　ｘ

　　　　　　　　　図１　印字形式

レポートには以下のものを添付すること．

1. main関数とクイックソート関数のプログラムリスト
2. 実行結果（図１の形式）
3. 印字出力の結果を参照して，左端の添え字，右端の添え字，新しい左端の添え字，新しい右端の添え字の値を用いて，パスごとに処理が正しく行われていることを説明せよ．

// クイックソート関数

void quick( int a[], int left, int right )

{

int pl=left; // left:　左端の添え字

int pr=right; // right:　右端の添え字

int pivot; // pivot:　pivotの値

pivot = a[(pl+pr)/2]; // ピボットは配列の中央に格納された値

while(1){

while( a[pl] < pivot ){

pl++;

}

while( a[pr] > pivot ){

pr--;

}

if( pr < pl )

break;

swap( &a[pl], &a[pr] )

pl++;

pr--;

}

if( left < pr ){

quick( a, left, pr );

}

if( pl < right ){

quick( a, pl, right );

}

}

void swap( int \*x, int \*y )

{

int tmp = \*x;

\*x = \*y;

\*y = tmp;

}

図2　クイックソート関数

ヒント：

パスとは関数quick()が呼び出される順番を表すので，まず0に初期化しておいて，quickが呼び出されるごとにインクリメントして関数quickの中で算出して表示すればよい．

【課題問題2】

クイックソート法の特性を調べたい．同一の数列に付いて，クイックソートプログラムの実行中に行われる比較と交換の回数を計数し，次に，バブルソートプログラムの実行中に行われる比較と交換の回数を計数して比較せよ．また，バブルソート法よりクイックソート法の方が比較と交換の回数が少ない理由を考察せよ．ただし，数列はrand関数を用いて生成せよ．使用方法は図4に示すとおりである．データの個数が，20個，40個，60個，80個，100個の5つの場合について，比較と交換の回数を計数する．

計数結果の出力形式

＜クイックソート法＞

　　データ数：20個

　　比較回数：ｘｘｘ回

　　交換回数：ｘｘｘ回

　　データ数：40個

　　比較回数：ｘｘｘ回

　　交換回数：ｘｘｘ回

　　(以下，続く)

＜バブルソート法＞

　　データ数：20個

　　比較回数：ｘｘｘ回

　　交換回数：ｘｘｘ回

　　データ数：40個

　　比較回数：ｘｘｘ回

　　交換回数：ｘｘｘ回

　　(以下，続く)

図3　出力形式

レポートには以下のものを添付すること

1. 回数を計数するプログラムを加えたバブルソートプログラムと，それを実行するmain関数のプログラリスト
2. 回数を計数するプログラムを加えたクイックソートプログラムと，それを実行するmain関数のプログラムリスト
3. 図3の出力形式に基づいた両プログラムの計数結果
4. バブルソート法よりクイックソート法の比較，交換の回数が少ない理由

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int main(void) {

int i, n, ran, sum;

srandom(time(NULL)); // 乱数発生初期化

printf("Number of trials: "); // 試行回数指定

scanf("%d",&n);

sum=0;

for ( i=0;i<n;i++) {

ran = random(); // 乱数発生

sum += ran; // 乱数の和を計算

}

printf("合計：%d \n", sum);

return 0;

}

図4　rand関数の使用例（n個の乱数を発生させて，その合計を計算するプログラム）

【課題問題3】

　クイックソートプログラムとバブルソートプログラムについて，同一のデータについて実行時間を測定して比較し考察せよ．データ数が，5千個，1万個，2万個，4万個，8万個の5つの場合について，それぞれ実行時間を計測する．比較，交換の回数を計数するプログラムは削除すること．

ヒント：実行時間の測定はclock関数を使用すればよい．使用方法を図5に示す．ただし＜time.h＞をインクルードすること．

レポートには以下のものを添付すること．

1. クイックソートプログラムと実行時間を測定するプログラムを追加したmain関数のプログラムリスト
2. バブルソートプログラムと実行時間を測定するプログラムを追加したmain関数のプログラムリスト
3. 両プログラムの実行結果（実行時間）
4. 実行結果に関する考察

#include <stdio.h>

#include <time.h>

#define NUM 10

int main(void) {

int a[] = {10,9,8,7,6,5,4,3,2,1};

time\_t start, end;

start = clock(); // 開始時間取得

quick( a, 0, NUM-1 );

end = clock(); // 終了時間取得

printf(”計算時間は%.3f秒です\n\n”, (float)(end-start)/CLOCKS\_PER\_SEC);

return 0;

}

図5　clock関数の使用例