1071 NTOUCSE 程式設計 1C 期末考 參 考 答 案

108/01/08 (二)

1. 請根據下列要求撰寫一個 int isPowerOf2(long long int number) 函式，判斷傳入的 long long 型態正整數 number 是否是 2 的次方數 (1 算是 2^0)，是的話請傳回 1，不是的話請傳回 0

   a. [5] 請運用迴圈撰寫

   Sol:

```
int isPowerOf2(long long int number)
{
    while (number%2==0) number /= 2; // 把所有 2 的因數除掉, 2^k
    return number == 1;              // 沒有其它因數時 (不可以是 3,5,7,9,….)
}
```

   或是迴圈執行次數少一半，稍微快一點

```
int isPowerOf2(long long int number)
{
    while (number%4==0) number /= 4;
    return (number != 3);
}
```

   或是只用乘法

```
int isPowerOf2(long long int number)
{
    long long int prod = 1;
    while (prod<number) prod *= 2;
    return number == prod;
}
```

   b. [5] 請運用函式遞迴撰寫

   Sol:

```
int isPowerOf2(long long int number)
{
    if (number == 1) return 1;
    if (number%2==0)
        return isPowerOf2(number/2);
    else
        return 0;
}
```

   c. [5] 請使用運算式來完成這個函式，**不要**使用迴圈、遞迴、或是 math.h 裡面的函式 (這些函式都運用迴圈或是遞迴，需要使用的常數請以十六進位常數，例如: 0x12345abef，或是位元運算表示)

   Sol:

```
int isPowerOf2(long long int number) // number is a positive integer
{
    return 0x4000000000000000LL%number == 0; // 正整數 2^62 是 long long 所能表示最大的 2 的次方數
}
```

   或是

```
int isPowerOf2(long long int number) // number is a positive integer
{
    return (1LL<<62)%number == 0; // 正整數 2^62 是最大的 2 的次方數
}
```

   或是計算 n&(n-1)==0

```
int isPowerOf2(long long int number) // number is a positive integer
{
    return number&(number-1)==0; // or return !(number&(number-1));
}
```

2. 考慮有理數 p/q (p<q)，其中 p 與 q 為 long long 型態可以表示的正整數:

   a. [5] 如果 q<10000，請撰寫程式在 1 秒內計算 p/q 是否可以用十進位小數精確表示，並輸出

小數點右邊有幾位數? (例如: 123/512=0.240234375 可以精確地表示出來, 小數點右邊有 9 位數)

Sol.

下面模擬長除法來完成判斷, 如果可以精確表示的話, 長除法的餘數會得到 0, 如果連續除以 q 3 次才整除就表示小數點右邊有 3 位數字, 如果不能夠精確表示就會有循環小數, 長除法的餘數會重複出現而永遠不會是 0, 餘數一定小於 q, 所以用 q 作為下面迴圈的上限

```
#include <stdio.h>
int main()
{
    long long p, q;
    int i;
    scanf("%lld%lld", &p, &q);

    for (i=0; i<q&&p%q!=0; i++)
        p = (p*10)%q;

    if (i<q)
        printf("小數點右邊有 %d 位數字\n", i);
    else
        printf("不能精確表示\n");
    return 0;
}
```

b. [5] 如果 q 為 long long 型態且 $q>2^{50}$, 請撰寫程式在 1 秒內判斷 p/q 是否可以用十進位小數精確表示, 並輸出小數點右邊有幾位數?

Sol.

當 q 值很大且 p/q 不能夠精確地表示出來的時候, 題 a 裡的方法迴圈需要執行 q 次才會停下來, 如此需要很久的時間才能夠判別, 一種方法是判斷是否開始循環, 但是如下題所述判斷循環需要以記憶體紀錄哪些出現過, 實際上不需要這樣子暴力以迴圈模擬長除法來判別, 如果能夠用 10 進位 k 位小數精確表示的話, p/q 需要寫成 x/10^k 的形式, 其中 gcd(x,2)=1 且 gcd(x,5)=1, 所以只要計算 p 和 q 裡面各有幾個 2 的因數和 5 的因數就可以知道可以用幾位數字表示出來了 例如 123/512 => 小數點右邊 9 位數, $123/(2^9*5^3)$ => 9 位數, $(123*2^3*5)/(2^7*5^6)$ => 5 位數

```
#include <stdio.h>
int main()
{
    long long p, q, q1;
    int count2=0, count5=0;
    scanf("%lld%lld", &p, &q);

    for (q1=q,count2=0; q1%2==0; q1/=2) count2++;
    while (p%2==0) p/=2, count2--; // q/p 可以被 2^count2 整除

    for (count5=0; q1%5==0; q1/=5) count5++;
    while (p%5==0) p/=5, count5--; // q/p 可以被 5^count5 整除

    if (q1==1) // q 只有 2 和 5 的因數
        printf("小數點右邊有 %d 位數字\n", count2>count5 ? count2 : count5);
    else
        printf("不能精確表示\n");
    return 0;
}
```

c. [5] 當有理數 p/q 沒辦法用十進位小數精確表示時, 最右邊有幾位數字循環(repeating decimal)? 例如 1/3=0.**3**333… 最後 1 位數字循環, 452/555=0.8**144**144144… 最後 3 位數字循環, 本題考慮 0<p<q≤5000

Sol.

以迴圈模擬長除法時, 如果 p/q 沒有辦法用十進位小數精確表示時, 會發現餘數重複出現, 以陣列紀錄餘數是否出現過, 如果 q 很大就會需要很久的時間

```
#include <stdio.h>
int main()
{
    int p, q, p1, period;
    char r[5001]={0};
    scanf("%d%d", &p, &q);

    r[0] = r[p] = 1;
    for (p1=p*10,period=1; period<q; r[p1]=1,p1*= 10,period++)
        if (r[p1=p1%q]) break;                  // 第 period 位數開始出現循環的餘數 p1
    while (p%q!=p1) period--, p=(p*10)%q;   // 第一次出現餘數 p1 的位置

    printf("last %d digits repeating\n", period);
    return 0;
}
或是每一個陣列元素多用一個位元組的空間來存放第一次出現的位置
#include <stdio.h>
int main()
{
    int p, q, p1, period;
    short r[5001]={0};
    scanf("%d%d", &p, &q);

    r[0] = r[p] = 1;
    for (p1=p*10,period=1; period<q; r[p1]=++period,p1*= 10)
        if (r[p1=p1%q]) break;                  // 第 period 位數開始出現循環的餘數 p1

    printf("last %d digits repeating\n", p1==0?0:period-r[p1]+1);
    return 0;
}
```

3. 請**依照各子題要求**撰寫程式，請不要使用全域變數，不要使用 C99 可變長度陣列(Variable length array)語法，以 Leibniz 公式計算 n×n 矩陣 A 的行列式值 (n≤10), Leibniz 公式如下:

$$\det(A)=\sum_{\sigma\in S_n}\operatorname{sgn}(\sigma)\prod_{i=0}^{n-1}a_{i,\sigma_i},\ n\text{-by-}n \text{ matrix } A=[a_{i,j}],\ \text{permutation } \sigma=\sigma_0\sigma_1...\sigma_{n-1}$$

S_n is the set of all permutations of {0,1,...,n-1} (e.g. {0123,0132,0213,...,3210}

$$\operatorname{sgn}(\sigma)=\begin{cases}1 & \text{if number of out of order pairs is even (e.g. } \sigma\text{=0123 or 0231)}\\ -1 & \text{if number of out of order pairs is odd (e.g. } \sigma\text{=3012 or 2031)}\end{cases}$$

a. [5] 如果整數陣列 int perm[10]; 紀錄某一個 0 到 n-1 的排列, (n 的數值不大於 10), 請撰寫一個函數 int sgn(int perm[], int n) 計算 sgn(σ) 的數值

Sol:

```
Number of out of order pairs 就是「逆序數」
int sgn(int perm[], int n)
{
    int i, j, sign;
    for (sign=i=0; i<n; i++)
        for (j=i+1; j<n; j++)
            if (perm[i]>perm[j]) sign++;
    return 1-2*(sign%2);
}
```

在上課及實習時我們以下列遞迴函式列印出 n 個數字的所有排列

```
void permutation(int perm[], int start, int end) {
    if (start == end) { printPerm(perm, end+1); return; }
    for (int i=start; i<=end; i++) {
        int tmp = perm[start], perm[start] = perm[i], perm[i] = tmp;
        permutation(perm, start+1, end);
        tmp = perm[start], perm[start] = perm[i], perm[i] = tmp;
    }
}
```

其中 perm 整數陣列一開始的資料為 0,1,…,n-1,

b. [5] 如果輸入資料如右，代表一個 nxn 的實數矩陣(n≤10)，第一列是 n，接下來是 n 列資料，每一列有 n 個實數資料，請撰寫一個函數 int readMatrix(double ***A) 讀取並且回傳矩陣的維度、動態配置大小為 n 的 double 指標陣列以及每列恰好 n 個元素的二維 double 陣列、並且讀取矩陣 n^2 個元素到這個陣列中

```
4
1.2 2 3 4
3 1 3.5 1
2 1.25 0 -1
2 4 5.3 1.9
```

Sol:

```
#include <stdlib.h>
int readMatrix(double ***A)
{
    int i, j, n;

    scanf("%d", &n);
    *A = (double **) malloc(n*sizeof(double *));
    for (i=0; i<n; i++)
        (*A)[i] = (double *)malloc(n*sizeof(double));
    for (i=0; i<n; i++)
        for (j=0; j<n; j++)
            scanf("%lf", &(*A)[i][j]);

    return n;
}
```

c. [5] 請撰寫一個函式 double product(int perm[], double **A, int n)，傳入的參數 perm 陣列紀錄一個 0,...,n-1 的排列，矩陣大小 n，以及題 b.動態配置的矩陣 A，計算並且回傳 $\prod_{i=0}^{n-1} a_{i,\sigma_i}$ 的 double 數值

Sol:

```
double product(int perm[], double **A, int n)
{
    int i;
    double result = 1.0;
    for (i=0; i<n; i++)
        result *= A[i][perm[i]];
    return result;
}
```

d. [5] 請寫一個函式 void freeMatrix(int **A) 釋放動態配置的記憶體

Sol:

```
void freeMatrix(double **A, int n)
{
    for (int i=0; i<n; i++)
        free(A[i]);
    free(A);
}
```

e. [10] 請撰寫 main() 函式，定義整數陣列 perm，運用 readMatrix() 讀入矩陣資料，初始化 perm 陣列，修改 permutation()函式，運用上面的 sgn()和 product() 函式計算矩陣的行列式值，列印矩陣的行列式值，最後釋放動態配置的記憶體

Sol:

```
void calcDet(double *det, int perm[], int start, int end, double **A)
{
    int tmp;
    if (start == end)
    {
        *det += sgn(perm, end+1) * product(perm, A, end+1);
        return;
    }
    for (int i=start; i<=end; i++)
    {
```

```
            tmp = perm[start], perm[start] = perm[i], perm[i] = tmp;
            calcDet(det, perm, start+1, end, A);
            tmp = perm[start], perm[start] = perm[i], perm[i] = tmp;
        }
    }

    int main()
    {
        int perm[NSIZE], n, i;
        double **A, det = 0.;

        n = readMatrix(&A);
        for (i=0; i<n; i++) perm[i] = i;

        calcDet(&det, perm, 0, n-1, A);

        printf("Determinant of %d-by-%d matrix is %f\n", n, n, det);
        freeMatrix(A, n);
        return 0;
    }
```

4. 請**依照下列要求**撰寫一個**遞迴**函數，以 Laplace 公式計算 n×n 矩陣 A 的行列式值 (n≤10), Laplace 公式如下:

$$\det(A)=\sum_{j=0}^{n-1}(-1)^{i+j}a_{i,j}\det(A_{i,j}),i\in\{0,...,n-1\},\ \text{n-by-n matrix } A=[a_{i,j}]_{i,j\in\{0,...,n-1\}},$$

$(n-1)$-by-$(n-1)$ matrix $A_{i,j}$ is the sub-matrix of A without the i-th row and the j-th column, i.e., $A_{i,j}=[a_{i',j'}]_{i'\in\{0,...,n-1\}\setminus\{i\},\ j'\in\{0,...,n-1\}\setminus\{j\}}$

a. [10] 請撰寫 double detLaplace(double A[][10], int n) 函式，參數是 double 型態的二維 10x10 陣列的矩陣資料以及 int 型態的矩陣維度 n，回傳值是 double 型態的行列式值。請在函式內另外定義二維 10x10 的 double 陣列來存放子矩陣 $A_{i,j}$，計算時請將上式 i 值指定為 0，請撰寫迴圈，藉由遞迴呼叫 detLaplace() 函式完成 Laplace 公式

Sol:

```
#define NSIZE 10
...
double detLaplace(double A[][NSIZE], int n)
{
    int i, j, k, k1;
    double sum=0, dA[NSIZE][NSIZE];

    if (n==1) return matrix[0][0];

    for (j=0; j<n; j++)
    {
        for (i=1; i<n; i++)
            for (k=k1=0; k<n; k++)
            {
                if (k==j) continue;
                dA[i-1][k1++] = A[i][k];
            }
        sum += (1-2*(j%2)) * A[0][j] * detLaplace(dA, n-1);
    }
    return sum;
}
```

b. [5] 如果在 main() 函式中讀入一個 8x8 的矩陣，請估計 detLaplace()函式在執行時總共會被呼叫幾次？某一個時間同時執行的 detLaplace() 函式最多有幾個?

Sol:

會呼叫 1+8*7*6*5*4*3*2= 40321 次，同一時間最多有 8 個 detLaplace()函數執行

c. [5] 接題 b, 因為區域變數有一個 10x10 的 double 陣列, 系統的堆疊至少需要有多少位元組程式才能順利正確完成這個遞迴函數的呼叫 (每一個函式呼叫, 至少需要在系統堆疊上保留區域變數以及函式參數的記憶體空間, 其它需要的記憶體空間請暫時不要考量, 另外也不要考慮 main() 函式所佔據的堆疊記憶體)?

Sol:

需要 8*(sizeof(double (*)[NSIZE])+sizeof(int)+
4*sizeof(int)+ NSIZE * NSIZE *sizeof(double)+sizeof(double)) = 8*(4+4+16+800+8)=6656 位元組

d. [5] 請用 struct 語法以及 typedef 語法定義一個結構型態 Matrix, 其中包含 10x10 的二維 double 型態陣列存放矩陣內容, 以及矩陣的維度 n

Sol:

```
struct Matrix_t
{
    double data[NSIZE][NSIZE];
    int n;
};
typedef struct Matrix_t Matrix;
```

e. [5] 請撰寫函數 Matrix readMatrix() 讀取右圖矩陣的維度 n 以及 n^2 個元素, 回傳整個結構

Sol:

```
Matrix readMatrix()
{
    int i, j;
    Matrix x;

    scanf("%d", &x.n);
    for (i=0; i<x.n; i++)
        for (j=0; j<x.n; j++)
            scanf("%lf", &(x.data[i][j]));

    return x;
}
```

```
4
1.2 2 3 4
3 1 3.5 1
2 1.25 0 -1
2 4 5.3 1.9
```

f. [10] 接題 d,e 請重新撰寫題 a 的 double detLaplace(const Matrix *A) 函式, 由於在 detLaplace() 函式內不會修改原來矩陣的內容, 所以 detLaplace()函式的參數使用常數結構型態 Matrix 變數的指標, 以避免過度的資料拷貝, 函式內部子矩陣 $A_{i,j}$ 請使用動態的記憶體配置及釋放所需要的結構變數?

Sol:

```
double detLaplace(const Matrix *A)
{
    int i, j, k, k1;
    double sum=0;
    Matrix *dA = (Matrix *) malloc(sizeof(Matrix));

    if (A->n==1)
        return A->data[0][0];

    for (j=0; j<A->n; j++)
    {
        for (i=1; i<A->n; i++)
            for (k=k1=0; k<A->n; k++)
            {
                if (k==j) continue;
                dA->data[i-1][k1++] = A->data[i][k];
            }
        dA->n = A->n - 1;
        sum += (1-2*(j%2)) * A->data[0][j] * detLaplace(dA);
```

```
    }
    free(dA);
    return sum;
}
```