說明：

陣列是一種結構性的資料儲存空間，其同一陣列裡的資料性質呈一致性，元素與元素之間的記憶位置是相鄰的，通常我們利用一個變數來代表整體的資料。舉例而言，我們可以把陣列想成一群鳥窩，而陣列裡的變數個數代表鳥窩的數目，例如：麻雀[20]，我們可以想成麻雀的窩總共有二十間，每一間住著一間麻雀，假如我們想要知道第三間麻雀的名字，只要把麻雀[2]的值取出，便可知道住在第三間麻雀的姓名。C語言的陣列索引一定是從0的開始的。

格式：

根據陣列的結構而言，可以把陣列分為(1)一維陣列、(2)二維陣列、(3)多維陣列。

而其表示方法如下：

資料型態　陣列名稱[陣列大小];

資料型態　陣列名稱[陣列大小][陣列大小];

宣告陣列變數時,也可一併給與初始值:

```
int x[5] = {1,2,3,4,5};
int y[] = {1,2,3};
int z[3][4] = {{1,2,3,4},{5,6,7,8},{0,1,2,3}};
int a[];
```

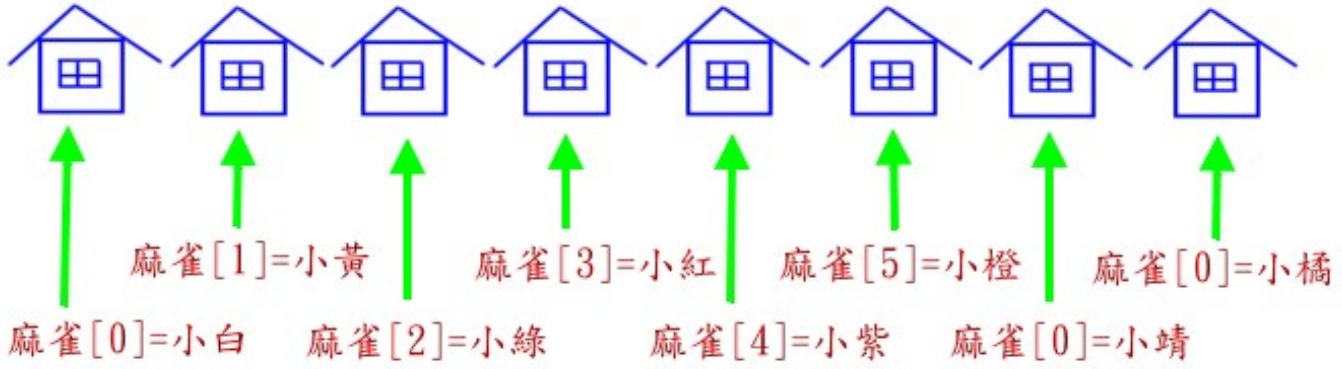
上面例子裡的y陣列大小,是由後面{}裡元素的個數決定。int a[]並沒有分配儲存陣列內容的空間,因此可視為指標宣告。

引用方式：

陣列名稱[索引值]

陣列名稱[索引值][索引值]

圖示：



範例：(輸入３個實數，並求其平均值)

```
#include<stdio.h>
void main() {
    float num[3], sum=0;
    int i;
    for (i = 0; i < 3; i++) {
        printf("Input a number to num[%d] : ", i);
        scanf("%f", &num[i]);
        sum = sum + num[i];
    }
    printf("The average is %f\n", sum / i);
}
```

陣列的空間分配方式

無論是幾維的陣列,C語言都以分配一塊連續的記憶體空間來處理。

```
int x[10];
```

分配10*sizeof(int)個bytes

```
int x[5][10];
```

分配5*10*sizeof(int)個bytes

```
int x[4][5][6];
```

分配4*5*6*sizeof(int)個bytes

```
void fun(int x[]) {
}
```

上面的x就沒有分配陣列的空間了,而是相當於int *x;這是因為C語言呼叫函數傳遞參數時,無法傳遞整個陣列(陣列可能大得不得了),而是傳遞陣列的開頭地址,也就是指標。因此在參數宣告時,指標和沒有宣告大小的陣列是可以混用的。

既然無論是幾維的陣列,C語言都以分配一塊連續的記憶體空間來處理,那麼像是

```
int x[2][3];
x[0][2] = 0;
```

中的x[0][2]被翻譯到哪一塊記憶體去了? C語言是使用row major的方式來處理多維到一維的對應。簡單的說,就是右邊的索引先變化:

```
  000  000
```

這六個整數的順序為x[0][0],x[0][1],x[0][2],x[1][0],x[1][1],x[1][2]。

array和pointer的對應關係如下:

```
int x[p][q];
int y[p][q][r];
int z[p];
int *po;
po = x;
x[i][j] = 0;
*(po + i*q + j) = 0; // same with x[i][j];
po = y;
y[i][j][k] = 0;
*(po + i*q*r + j*r + k) = 0; // same with y[i][j][k]
po = z;
z[i] = 0;
*(po + i) = 0; // same with z[i]
```

陣列的參數傳遞

C語言只能傳遞指標,無法傳遞陣列的內容。假設我們要傳遞一個二維陣列,則C會幫我們將該陣列的起頭位置傳入,但參數宣告部分則有如下不同的方式:

```
void foo1(int x[][]) { // 編譯過,但Compiler不知如何翻譯, 還是用int *x自己計算地址比較好
    x[2][2] = 0; // 編譯錯誤!, Compiler不知如何翻譯
}
void foo2(int x[][3]) { // Compiler知道如何翻譯
    x[1][1] = 0; // Compiler知道要翻成*(x+1*3+1)
}
void foo3(int x[2][3]) { // Compiler知道如何翻譯
    x[1][1] = 0; // Compiler知道要翻成*(x+1*3+1)
}
void foo4(int x[3][]) { // 編譯錯誤
}
void foo5(int *x) { // 反正只能傳pointer
    *(x+1*3+1) = 0;
}
int main() {
    int m[2][3];
    int *p;
    int k[4][4];
    foo2(m);
    foo2(p); // Compiler warning, incompatible pointer type
```

```
    foo3(k); // Compiler warning, incompatible pointer type
    foo5(m); // Compiler warning, incompatible pointer type
    foo5(p);
    foo5((int *)m); // 強迫轉型, Compiler就不會抱怨了
}
```

動態空間分配

宣告陣列時,C compiler就已經分配好空間了。例如

```
int main() {
    int x[10][20];
}
```

compiler會在進入main時於堆疊上分配給x放置200個整數所需的空間,而在離開main時將空間回收。對指標來說,則只有分配紀錄指標的空間,但對於透過該指標所能存取的記憶體,卻沒有分配。有許多情況是設計者無法事先預知所需空間的大小,必須等到run time才能知道。C語言提供了一系列的函數可於執行期間分配或釋放記憶體空間。

```
void *malloc(size_t size);
void *calloc(size_t nelem, size_t elsize);
void free(void *ptr);
```

使用以上函數必須#include <stdlib.h>。malloc會自heap(堆積)取得size個byte, 並傳回該區塊的起始位置。calloc分配nelem個大小為elsize個byte的空間,並把該區塊所有的內容設為0。free則是釋放由malloc或calloc所分配的記憶體空間。

要特別提醒讀者的是, malloc和free要小心使用, 如果malloc所得的空間沒有用free釋放, 則應用程式就會一直佔住該記憶體, 此一現象稱為memory leakage。如果同一空間free了兩次, 或已經free了卻繼續使用, 也都可能出問題。

範例

Selection Sort

```
#include <stdio.h>
#define SIZE 6
int main() {
    int i, j, min, at, tmp;
    int data[] = {5,7,1,4,3, 2};
    for (i = 0; i < SIZE - 1; i++) {
        min = data[i];
        at = i;
        for (j = i+1; j < SIZE; j++) {
            if (data[j] < min) {
                min = data[j];
                at = j;
            }
        }
        tmp = data[i];
        data[i] = min;
```

```
        data[at] = tmp;
    }
    for (i = 0; i < SIZE; i++) printf("%d\n", data[i]);
}
```

Insertion Sort

```
/**
 * Program Name: sort.c
 * Purpose: insertion sort
 * Author: Shiuh-Sheng Yu, Department of Information Management
 *         National ChiNan University
 * Since: 2006/10/16
 */
#include <stdio.h>
#define SIZE 6
int main() {
    int i, j, tmp;
    int data[SIZE] = {5,7,1,4,3,2};
    for (i = 1; i < SIZE; i++) {
        tmp = data[i];
        for (j = i-1; j >= 0 && data[j] > tmp; j--) {
            data[j+1] = data[j];
        }
        data[j+1] = tmp;
    }
}
```

Pascal Triangle

下圖為n=6的情況

```
            1
          1   1
        1   2   1
      1   3   3   1
    1   4   6   4   1
  1   5   10  10  5   1
1   6   15  20  15  6   1
```

其規則是最外層是1, 裡面每個數字都是上方兩個數字的和. Pascal Triangle是$(x + y)^n$每個項次的係數.

提示: 如果把上圖左邊的空白拿掉則會變成下面的圖形, 除了最左邊和最右邊的數字是1以外, 裡面的每一個數字都是其正上方和左上方數字的和. 你可以用陣列來計算和儲存這些數字, 然後再以上圖的格式印出來.

```
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
1   5   10  10  5   1
1   6   15  20  15  6   1
```

所以只要你能回答下面問題, 程式就寫完了:

- 如果要用*號印出這種形狀的三角形, 該怎麼寫? (迴圈範例裡已經練習過了)
- 最左邊和最右邊如何表達?
- 內部每一個數字都是正上方和左上方數字, 請問正上方和左上方這兩個位置的陣列索引如何表達?
- 每一個row前面要加上幾個空白才能讓圖形看起來是正三角形?

以下是程式的範例:

```
/**
 * Program Name: pas2.c
 * Purpose: print pascal triangle on screen
 * Author: Shiuh-Sheng Yu, Department of Information Management
 *         National ChiNan University
 * Since: 2004/12/10
 */
#include <stdio.h>
int main() {
    int n, i, j;
    unsigned tri[51][51];
    printf("Please input size of Pascal triangle(0 to 50): ");
    scanf("%d", &n);
    if (n < 0 || n > 50) {
        printf("I can only print Pascal triangle between 0 and 50.\n");
        return;
    }
    for (i = 0; i <= n; i++) {
        for (j = 0; j < n - i; j++) {
            printf("   ");
        }
        for (j = 0; j <= i; j++) {
            printf(" %5u",tri[i][j] = (j==0 || j==i) ? 1 : tri[i-1][j-1]+tri[i-1][j]);
        }
        printf("\n");
    }
}
```

列出整數陣列的所有排列

```
#include <stdio.h>
/* 列出由at左邊所有的排列 */
void permutation(int data[], int at, int arraySize) {
    int i, tmp;
    // 如果已經排到最後了,印出結果
    if (at == arraySize) {
        for (i = 0; i < arraySize; i++) {
            printf("%d ", data[i]);
        }
        printf("\n");
        return;
    }
```

```
    for (i = at; i < arraySize; i++) { // choose ith element
        // exchange at with i
        tmp = data[i];
        data[i] = data[at];
        data[at] = tmp;
        permutation(data, at + 1, arraySize);
        // exchange back, so that we undo previous exchange
        tmp = data[i];
        data[i] = data[at];
        data[at] = tmp;
    }
}
int main() {
    int data[] = {1,2,3};
    permutation(data, 0, 3);
}
```

列出整數陣列的所有組合

```
#include <stdio.h>
/**
 * n: data array的長度
 * m: 希望選m個數字
 * after: 只能選after.. n-1位置的數字
 * got: 目前已選的數字個數
 */
void com(int data[], int n, int m, int after, int got) {
    int i;
    if (m == got) { // 已選滿m個數字
        for (i = 0; i < m; i++) {
            printf("%d ", data[i]);
        }
        printf("\n");
        return;
    }
    for (i = after; i < n; i++) {
        // 選第i個
        int tmp = data[got];
        data[got] = data[i];
        data[i] = tmp;
        com(data, n, m, i + 1, got+1);
        tmp = data[got];
        data[got] = data[i];
        data[i] = tmp;
    }
}
void combination(int data[], int n, int m) {
    com(data, n, m, 0, 0);
}
int main() {
    int data[10] = {1,2,3,4,5,6,7,8,9,10};
    combination(data, 10, 3);
}
```

找出八皇后的所有解

西洋棋的棋盤大小為8 X 8, 其中皇后這個棋子可以向八個方向走任意距離, 如果要把8個皇后放到棋盤上, 而且彼此吃不到, 請問該如何放置, 合法的方法有多少種?

```
/**
 * Program Name: eightqueen.c
 * Purpose: 找出八皇后問題總共有幾解
 * Author: Shiuh-Sheng Yu
 * Dept. of Information Management, National ChiNan University
 * Since: 2004/12/20
 */
/* 使用加上邊框的10*10盤面 */
char initBoard[100] = {
    -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
    -1, 0, 0, 0, 0, 0, 0, 0, 0,-1,
    -1, 0, 0, 0, 0, 0, 0, 0, 0,-1,
    -1, 0, 0, 0, 0, 0, 0, 0, 0,-1,
    -1, 0, 0, 0, 0, 0, 0, 0, 0,-1,
    -1, 0, 0, 0, 0, 0, 0, 0, 0,-1,
    -1, 0, 0, 0, 0, 0, 0, 0, 0,-1,
    -1, 0, 0, 0, 0, 0, 0, 0, 0,-1,
    -1, 0, 0, 0, 0, 0, 0, 0, 0,-1,
    -1, 0, 0, 0, 0, 0, 0, 0, 0,-1,
    -1,-1,-1,-1,-1,-1,-1,-1,-1,-1
};
int solutions = 0;
void showBoard(char board[100]) {
    int i, j;
    for (i = 1; i <= 8; i++) {
        for (j = 1; j <= 8; j++) {
            (board[10 * i + j] == 'Q') ? printf("Q ") : printf(". ");
        }
        printf("\n");
    }
    printf("\n");
}

/**
 *  此函數試圖在board上面放第n個皇后,
 *  若找到可以放的位置, 就遞迴呼叫自己以便放上第n個皇后,
 */
void putOneQueen(char board[100], int n) {
    if (n > 8) {
        showBoard(board);
        solutions++;
        return;
    }
    int dir[3] = {9,10,11}; // 下方的三個方向
    int col, pos, i, j;
    // 檢查nth row上的每一個column(有8個要檢查)
    for (col = 1; col <= 8; col++) {
        // 如果要放上去的地方不在任何一個皇后的勢力範圍內
        if (board[pos = 10 * n + col] == 0) {
```

```
            // 放上新皇后
            board[pos] = 'Q';
            // 建立新皇后下方的勢力範圍
            for (i = 0; i < 3; i++)
                for (j = pos + dir[i]; board[j] >= 0; j += dir[i])
                    board[j]++;
            putOneQueen(board, n + 1);
            // 移除此皇后
            board[pos] = 0;
            // 移除此皇后的勢力範圍
            for (i = 0; i < 3; i++)
                for (j = pos + dir[i]; board[j] > 0; j += dir[i])
                    board[j]--;
        }
    }
}
int main() {
    putOneQueen(initBoard, 1);
    printf("八皇后問題共有%d組解\n", solutions);
}
```

如果在執行期間由使用者輸入棋盤的大小n, 你如何寫一程式找出n皇后的所有解?