

1. 寫一個二維陣列 $A[ROW][COL]$ ，輸入 ROW 及 COL，並以一個二維指標存取此二維陣列 $A[ROW][COL]$ ，用指標顯示 $A[i][j]$ 值，並顯示 $A[i][j]$ 位址，參考如圖輸出。(10%)

```
Please enter number of ROW : 2
Please enter number of COL : 3
A[0][0]=0,      address of A[0][0]=0x876F28
A[0][1]=1,      address of A[0][1]=0x876F2C
A[0][2]=2,      address of A[0][2]=0x876F30
A[1][0]=3,      address of A[1][0]=0x874750
A[1][1]=4,      address of A[1][1]=0x874754
A[1][2]=5,      address of A[1][2]=0x874758
```

再以相同方法寫一個矩陣 B [ROW][COL]，透過迴圈寫出 $A + B$ (10%)、 $A - B$ (10%)。

最後，再以相同方法寫一個矩陣 C [COL][ROW]，透過迴圈寫出 $A * C$ (15%)、 C^T (transpose) (10%)，並釋放 A、B、C 陣列的記憶體(5%)。

```
C[0][0] = 0
C[0][1] = 1
C[1][0] = 2
C[1][1] = 3
C[2][0] = 4
C[2][1] = 5
```

- 請使用上課所教的動態陣列。
- Hint: $A[i][j] = COL * i + j$ 。
- A、B、C 陣列的值一律從 0 開始遞增，遞增量為 1。

.cpp



HW9_Q1 - Microsoft Visual Studio

檔案(F) 編輯(E) 檢視(V) 專案(P) 建置(B) 偵錯(D) 小組(M) 工具(T) 測

Debug x86 本機

工具箱

HW9_Q1.cpp

HW9_Q1

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main()
4 {
5     int R, C;
6     printf("Please enter your number of ROW:");
7     scanf_s("%d", &R);
8     printf("Please enter your number of COL:");
9     scanf_s("%d", &C);
10    int *A = (int *)malloc(R*C * sizeof(int));
11    int *B = (int *)malloc(R*C * sizeof(int));
12    int *arrC = (int *)malloc(R*C * sizeof(int));
13    for (int i = 0; i < R; i++) {
14        for (int j = 0; j < C; j++) {
15            A[i*C + j] = i * C + j;
16            B[i*C + j] = A[i*C + j];
17        }
18    }
19    for (int i = 0; i < R; i++) {
20        for (int j = 0; j < R; j++) {
21            arrC[i*R + j] = i * R + j;
22        }
23    }
24    for (int i = 0; i < R; i++) {
25        for (int j = 0; j < C; j++) {
26            printf("A[%d][%d] = %d", i, j, A[i*R + j]);
27            printf("\taddress of A[%d][%d] = 0x%x", i, j, &A[i*R + j]);
28            printf("\n");
29        }
30    }
31
32    //I'm separator line.
33
34    printf("\n\nA + B =\n");
35    for (int i = 0; i < R; i++) {
36        for (int j = 0; j < C; j++) {
37            printf("A[%d][%d] + B[%d][%d] = %d", i, j, i, j, A[i*R + j] + B[i*R + j]);
38            printf("\n");
39        }
40    }
```

22 %

HW9_Q1 - Microsoft Visual Studio

檔案(E)編輯(E)檢視(V)專案(P)建置(B)偵錯(D)小組(M)

Debug

x86

HW9_Q1.cpp

HW9_Q1

```
37     printf("A[%d][%d] + B[%d][%d] = %d", i, j, i, j, A[i*R + j] + B[i*R + j]);
38     printf("\n");
39 }
40 }
41 printf("\n\nA - B =\n");
42 for (int i = 0; i < R; i++) {
43     for (int j = 0; j < C; j++) {
44         printf("A[%d][%d] - B[%d][%d] = %d", i, j, i, j, A[i*R + j] - B[i*R + j]);
45         printf("\n");
46     }
47 }
48 //I'm separator line.
49 printf("A * C = \n");
50 int sum;
51 for (int i = 0; i < R; i++) {
52     for (int j = 0; j < R; j++) {
53
54         sum = 0;
55         for (int k = 0; k < C; k++) {
56             sum += A[i*C + k] * arrC[k*R + j];
57         }
58         printf("%d\t", sum);
59     }
60     printf("\n");
61 }
62 //I'm separator line.
63 printf("C transpose = \n");
64 for (int i = 0; i < C; i++) {
65     for (int j = 0; j < R; j++) {
66         printf("%d\t", arrC[R*C - (i*R + j) - 1]);
67     }
68     printf("\n");
69 }
70 free(A);
71 free(B);
72 free(arrC);
73 }
74 }
75 }
```

22 %

執行結果

Microsoft Visual Studio 偵錯主控台

Please enter your number of ROW:2

Please enter your number of COL:3

A[0][0] = 0 adress of A[0][0] = 0xebe420

A[0][1] = 1 adress of A[0][1] = 0xebe424

A[0][2] = 2 adress of A[0][2] = 0xebe428

A[1][0] = 2 adress of A[1][0] = 0xebe428

A[1][1] = 3 adress of A[1][1] = 0xebe42c

A[1][2] = 4 adress of A[1][2] = 0xebe430

A + B =

A[0][0] + B[0][0] = 0

A[0][1] + B[0][1] = 2

A[0][2] + B[0][2] = 4

A[1][0] + B[1][0] = 4

A[1][1] + B[1][1] = 6

A[1][2] + B[1][2] = 8

A - B =

A[0][0] - B[0][0] = 0

A[0][1] - B[0][1] = 0

A[0][2] - B[0][2] = 0

A[1][0] - B[1][0] = 0

A[1][1] - B[1][1] = 0

A[1][2] - B[1][2] = 0

A * C =

10 13

28 40

C transpose =

): 已載入 'C:\Windows\SysWOW64\msvcrt.dll'。找不到或無法開啟 PDB

```

A - B =
A[0][0] - B[0][0] = 0
A[0][1] - B[0][1] = 0
A[0][2] - B[0][2] = 0
A[1][0] - B[1][0] = 0
A[1][1] - B[1][1] = 0
A[1][2] - B[1][2] = 0
A * C =
10      13
28      40
C transpose =
5        4
3        2
1        0

```

2. 試寫一程式，將十進制數字轉成十六進制，程式需滿足以下要求：

- 請將十六進制數值的各個位數以陣列方式儲存。(15%)
- 宣告陣列時，請使用動態記憶體的方式配備剛好的記憶體容量給此陣列。(5%)
- 請印出你需要多少記憶體給轉換陣列。(10%)

輸出結果如下：

```

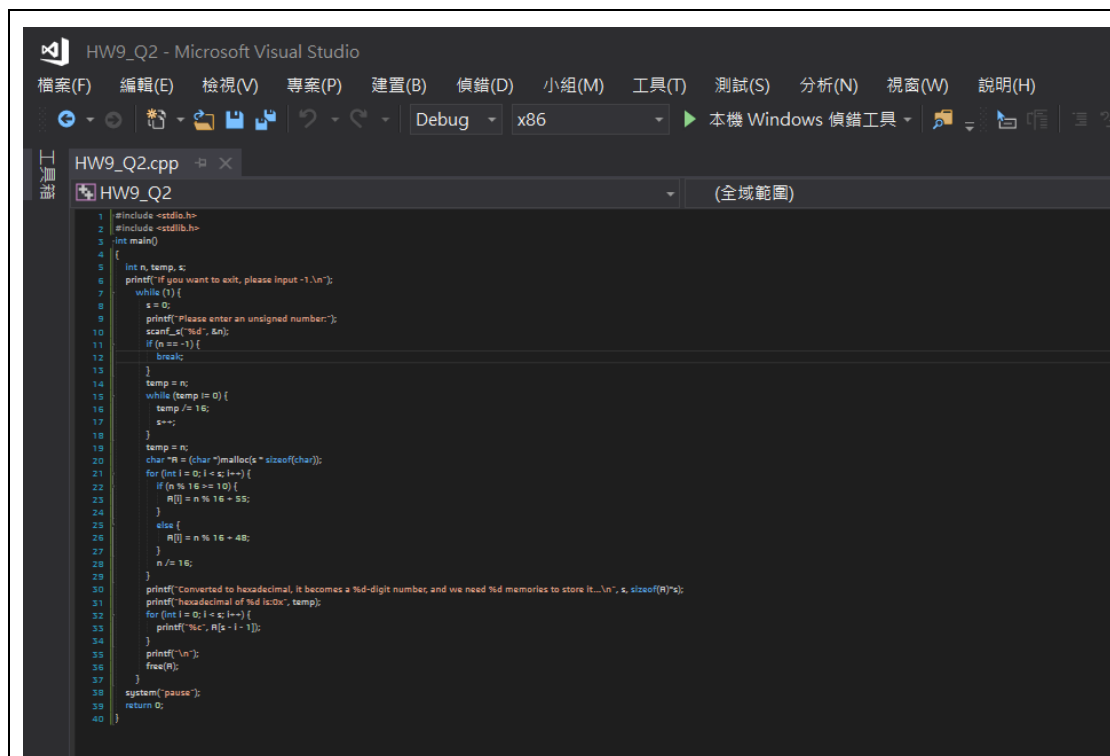
輸入一個十進制的正整數: 12345
十六進制後為4位數，配置16 bytes來儲存...
12345的十六進制為：0x3039

輸入一個十進制的正整數: 1234
十六進制後為3位數，配置12 bytes來儲存...
1234的十六進制為：0x4D2

輸入一個十進制的正整數: 123
十六進制後為2位數，配置8 bytes來儲存...
123的十六進制為：0x7B

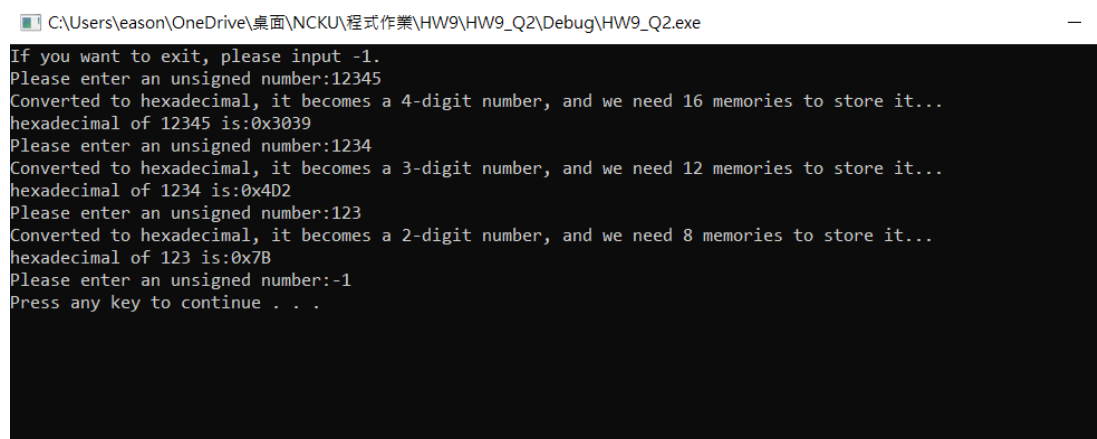
```

.cpp



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main()
4 {
5     int n, temp, c;
6     printf("If you want to exit, please input -1.\n");
7     while (1) {
8         s = 0;
9         printf("Please enter an unsigned number:");
10        scanf("%d", &n);
11        if (n == -1) {
12            break;
13        }
14        temp = n;
15        while (temp != 0) {
16            temp /= 16;
17            s++;
18        }
19        temp = n;
20        char *R = (char *)malloc(s * sizeof(char));
21        for (int i = 0; i < s; i++) {
22            if (n % 16 >= 10) {
23                R[i] = n % 16 + 55;
24            }
25            else {
26                R[i] = n % 16 + 48;
27            }
28            n /= 16;
29        }
30        printf("Converted to hexadecimal, it becomes a %d-digit number, and we need %d memories to store it...\n", s, sizeof(R));
31        printf("hexadecimal of %d is: ", temp);
32        for (int i = 0; i < s; i++) {
33            printf("%c", R[s - i - 1]);
34        }
35        printf("\n");
36        free(R);
37    }
38    system("pause");
39    return 0;
40 }
```

執行結果



```
C:\Users\yason\OneDrive\桌面\NCKU\程式作業\HW9\HW9_Q2\Debug\HW9_Q2.exe
If you want to exit, please input -1.
Please enter an unsigned number:12345
Converted to hexadecimal, it becomes a 4-digit number, and we need 16 memories to store it...
hexadecimal of 12345 is:0x3039
Please enter an unsigned number:1234
Converted to hexadecimal, it becomes a 3-digit number, and we need 12 memories to store it...
hexadecimal of 1234 is:0x4D2
Please enter an unsigned number:123
Converted to hexadecimal, it becomes a 2-digit number, and we need 8 memories to store it...
hexadecimal of 123 is:0x7B
Please enter an unsigned number:-1
Press any key to continue . . .
```

3. 說明一般變數和指標變數的差異。(10%)

一般變數最主要的功能就是儲存資料型態像是 int, float, double, char 的值，不同資料型態的變數也都會有對應不同的記憶體大小(1byte, 2bytes, 4bytes)，而每一個變數都像一個鐵櫃，裡面放置的是我們的值像是數字或者字元，這也是一般變數再儲存的東西，然後每個鐵

櫃都會有一個自己的編號，而指標變數的功能就是儲存這些變數的編號，然後因為這些變數的功能都是儲存位址，所以不管儲存的是什麼資料型態的位址，它本身的記憶體大小都是 4bytes。