C 程式語言 4

do while迴圈

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
  int n,i=1,sum=0;   /* 設定迴圈初值 */
  do
  {
    printf("Please input the value of n (n>0):");
    scanf("%d",&n);
  }
  while (n<=0);                    /* 當n<=0時重複輸入n的值 */

  do
  {
    sum+=i;
    i++;
  }
  while (i <= n);       /* 當i<=n時執行累加的動作 */
  printf("1+2+...+%d=%d\n",n,sum);
  system("pause");
  return 0;
}
```

迴圈的跳離

雖然迴圈敘述可以在開頭或結尾測試結束條件，但是有時我們需要在迴圈之中測試結束條件，我們可以使用break關鍵字來跳出迴圈，如同switch條件敘述使用break關鍵字跳出程式區塊

迴圈的跳離

- break 敘述：
  - 略過迴圈主體的其餘部分，執行迴圈之後的敘述

break 敘述的語法

```
for(初值設定; 判斷條件; 設定增減量)
{
    敘述 1;
    敘述 2;
    …
    break;
    …
    敘述 n;
}
...
```

若執行break敘述，則此區塊內的敘述不會被執行

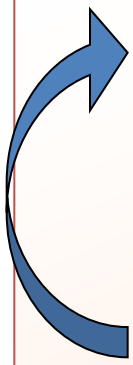
continue 敘述

在迴圈執行過程中，相對於上一個使用break關鍵字跳出迴圈，continue關鍵字可以馬上繼續下一次迴圈的執行，而不執行程式區塊位在continue關鍵字之後的程式碼

continue 敘述

- continue 敘述：
  - 略過迴圈主體的其餘部分，直接開始下一個迴圈循環

continue 敘述的語法



```
for(初值設定; 判斷條件; 設定增減量)
{
    敘述 1;
    敘述 2;
      …
    continue;
      …
    敘述 n;
}
  ...
```

若執行continue敘述，則此區塊內的敘述不會被執行

清除緩衝區的資料

- fflush可用來清除緩衝區的資料

fflush() 函數的用法

```
fflush(stdin);      /* 清除緩衝區內的資料 */
```

認識陣列 –使用多個變數

- 上述表格是小考成績，我們可以宣告5個int整數變數來儲存這5次成績，如下所示：

```
int test1 = 71;
int test2 = 83;
int test3 = 67;
int test4 = 49;
int test5 = 59;
```

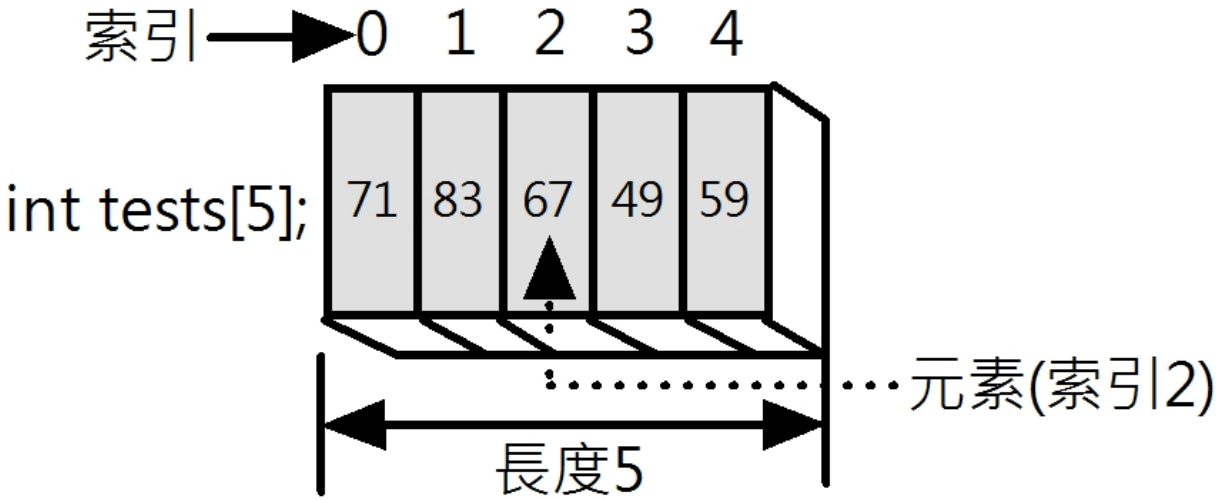
- 上述程式碼宣告5個變數和指定初值，5個數量還好，如果是一班50位學生的成績，我們需要50個變數；如果一個公司有500位員工時，在程式中就需要宣告大量變數，如此會造成程式碼變的十分複雜。

認識陣列 – 找出共同特性

- 讓我們再次觀察上述小考成績的5個變數，其擁有的共同特性，如下所示：
  - 變數的資料型態相同都是int。
  - 變數有循序性，擁有順序的編號1~5。

認識陣列 - 使用陣列

- 陣列（Array）是一種儲存大量循序資料的結構，我們可以將上述相同資料型態的5個int變數集合起來，使用一個名稱tests代表，如下圖所示：



一維陣列

- 陣列是相同型態之元素所組成的集合
- 在 C 語言中，陣列使用前必須先宣告：

一維陣列的宣告格式

資料型態 陣列名稱[個數];

- 下面是一維陣列宣告的範例：

```
int score[4];          /* 宣告整數陣列score，可存放4個元素 */
float temp[7];         /* 宣告浮點數陣列temp，可存放7個元素 */
char name[12];         /* 宣告字元陣列name，可存放12個元素 */
```

陣列的索引值

- 陣列中的元素是以索引值來標示存放的位置
- 陣列索引值的編號必須由0開始

```
int score[4];
```

陣列的範例

- 一維陣列的基本操作：

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
  int i,score[4];                  /* 宣告整數變數i與整數陣列score */
  score[0]=78;                     /* 設定陣列的第一個元素為78 */
  score[1]=55;                     /* 設定陣列的第二個元素為55 */
  score[2]=92;                     /* 設定陣列的第三個元素為92 */
  score[3]=80;                     /* 設定陣列的最後一個元素為80 */

  for(i=0;i<=3;i++)
    printf("score[%d]=%d\n",i,score[i]);  /* 印出陣列的內容 */

  system("pause");
  return 0;
}
```

陣列初值的設定

- 一維陣列初值的設定格式：

一維陣列初值設定的格式

資料型態 陣列名稱[個數n]={初值1,初值2,…,初值n};

- 初值設定的範例：
  - `int score[4]={78,55,92,80};`
  - `int score[]={60,75,48,92};` `/* 省略元素的個數 */`

sizeof 關鍵字

- 查詢陣列所佔的記憶空間

查詢整個陣列所佔的位元組

```
sizeof(陣列名稱)    /* 查詢陣列所佔的位元組 */
```

陣列的應用－最大與最小值

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
  int A[5]={7,48,30,17,62};
  int i,min,max;
  min=max=A[0];                    /* 將max與min均設為陣列的第一個元素 */

  for(i=0;i<5;i++)
  {
    if(A[i]>max)          /* 判斷A[i]是否大於max */
      max=A[i];
    if(A[i]<min)          /* 判斷A[i]是否小於min */
      min=A[i];
  }
  printf("The maximum value of the array element :%d\n",max);
  printf("The minimum value of the array element :%d\n",min);

  system("pause");
  return 0;
}
```

陣列界線的檢查

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5                    /* 定義MAX為5 */
int main(void)
{
  int score[MAX];                /* 宣告score陣列，可存放MAX個整數 */
  int i=0,num;
  float sum=0.0f;
```

陣列界線的檢查

```
printf("Please input the score, input 0 to end:\n");
  do
  {
    if(i==MAX)                          /* 當i的值為MAX時，表示陣列已滿，即停止輸入 */
    {
      printf("Array space has been used up!!\n");
      i++;                              /* 此行先將i值加1，因為23行會把i的值減1掉 */
      break;
    }
    printf("Please input the score:");
    scanf("%d",&score[i]);
  }while(score[i++]>0);                 /* 輸入成績，輸入0時結束 */
  num=i-1;
  for(i=0;i<num;i++)
    sum+=score[i];                      /* 計算平均成績 */
  printf("average score  %.2f\n",sum/num);

  system("pause");
  return 0;
}
```

-The End-