



程式設計

第3章 控制流程 Control Flow

蘇維宗(Wei-Tsung Su)
suwt@au.edu.tw
564D





目標

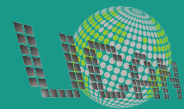
控制流程的目的是可以指定程式執行流程, 包含

1. 邏輯控制(`if`、`else`、`else if`)
2. 選擇器(`switch`)
3. 迴圈控制(`for`、`while`、`do-while`)
4. `Break`與`Continue`
5. `Goto`與`Labels`



邏輯控制

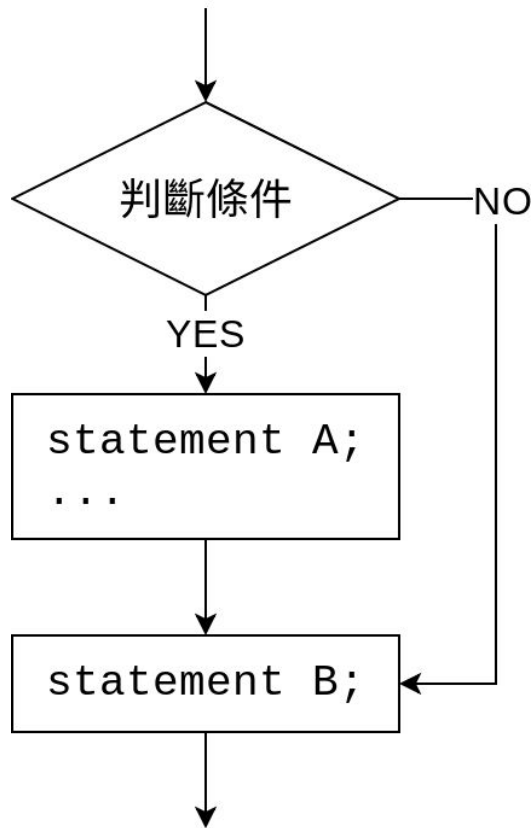
根據特定條件改變程式執行流程



if

如果**判斷條件**成立則執行程式區塊內的程式。

```
1.  if(判斷條件) { //成立時執行
2.      statement A1;
3.      ...
4.  }
5.  statement B;
```





if (續)

這段程式碼會印出？

```
1.  int num = 5;
2.  if(num > 10)
3.      printf("Hello\n");
4.      printf("Hello\n");
```

這段程式碼會印出？

```
1.  int num = 5;
2.  if(num > 10) {
3.      printf("Hello\n");
4.      printf("Hello\n");
5.  }
```





練習：終極密碼戰v1

假設 $m=40$ ，讓使用者輸入一個整數 n 。
如果猜對(即 $n=m$)則輸出YES。

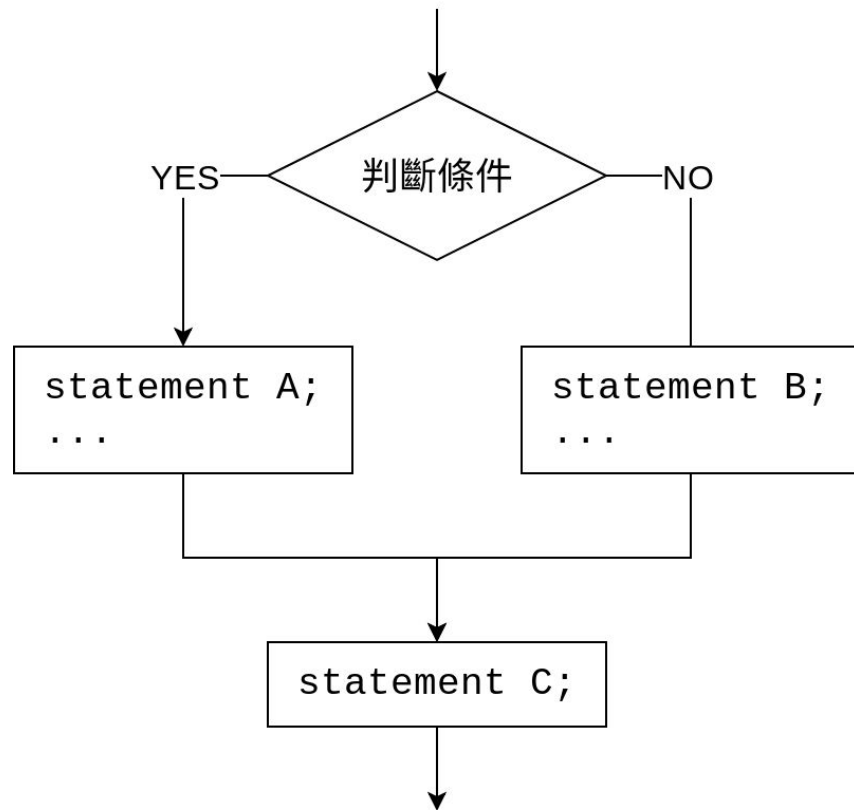
輸入	輸出
20	
50	
60	
40	YES

假設 $m=40$

if else

依據**判斷條件**將程式分為"成立"
與"不成立"**兩個程式區塊**

```
1.  if (判斷條件) {           // 成立時執行
2.      statement A;
3.      ...
4.  } else {                   // 不成立時執行
5.      statement B;
6.      ...
7.  }
8.  statement C;
```





練習：終極密碼戰v2

假設 $m=40$ ，讓使用者輸入一個整數 n 。
如果猜對(即 $n=m$)則輸出YES，否則輸出NO。

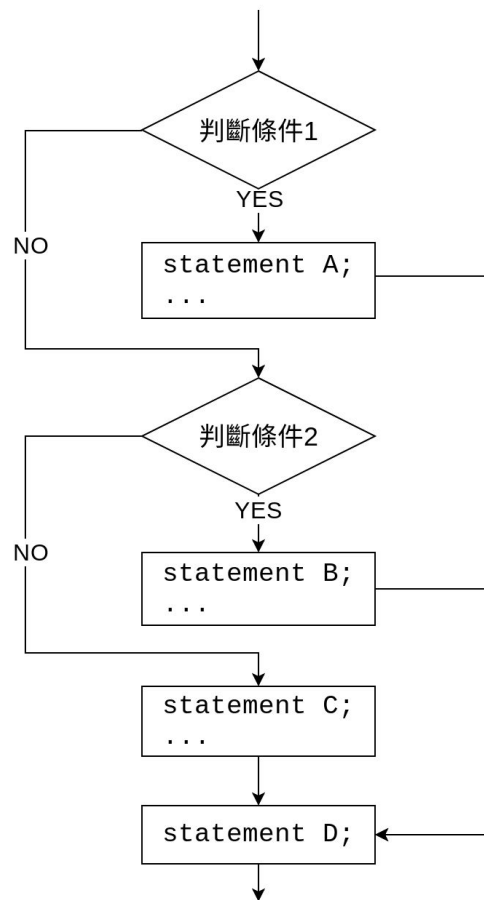
輸入	輸出
20	NO
50	NO
60	NO
40	YES

假設 $m=40$

else-if

依據多個特定條件將程式分為**多個程式區塊**。

```
1.  if(判斷條件1) {           //判斷條件1成立
2.      statement A;
3.      ...
4.  } else if(判斷條件2) {     //判斷條件2成立
5.      statement B;
6.      ....
7.  } else {                   //都不成立
8.      statement C;
9.      ...
10. }
11. statement D;
```





練習：終極密碼戰v3

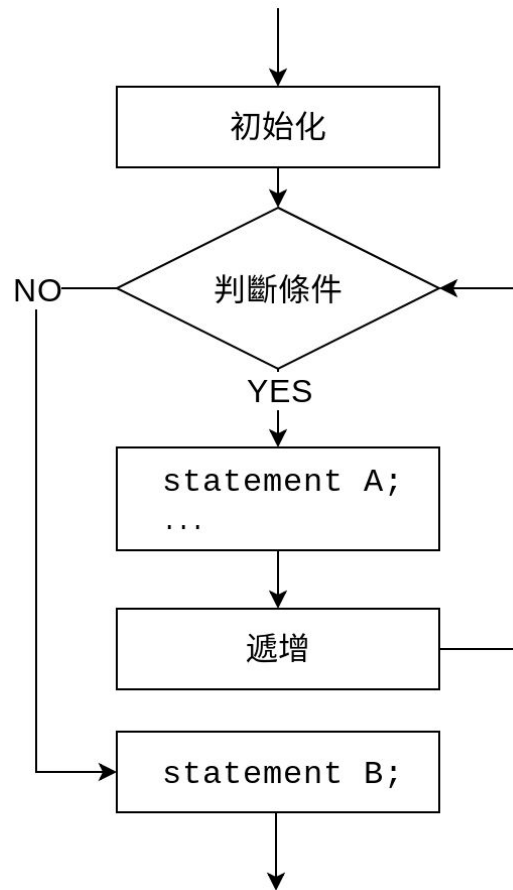
假設 $m=40$ ，讓使用者輸入一個整數 n 。
如果猜對(即 $n=m$)則輸出YES; 如果 $n>m$
輸出TOO LARGE; 如果 $n<m$ 輸出TOO
SMALL。

輸入	輸出
20	TOO SMALL
50	TOO LARGE
60	TOO LARGE
40	YES

假設 $m=40$

for loop

```
1.  for ( 初始化 ; 判斷條件 ; 遞增 ) {  
2.      statement A;  
3.      ...  
4.  }  
5.  statement B;
```





練習：印出三角形

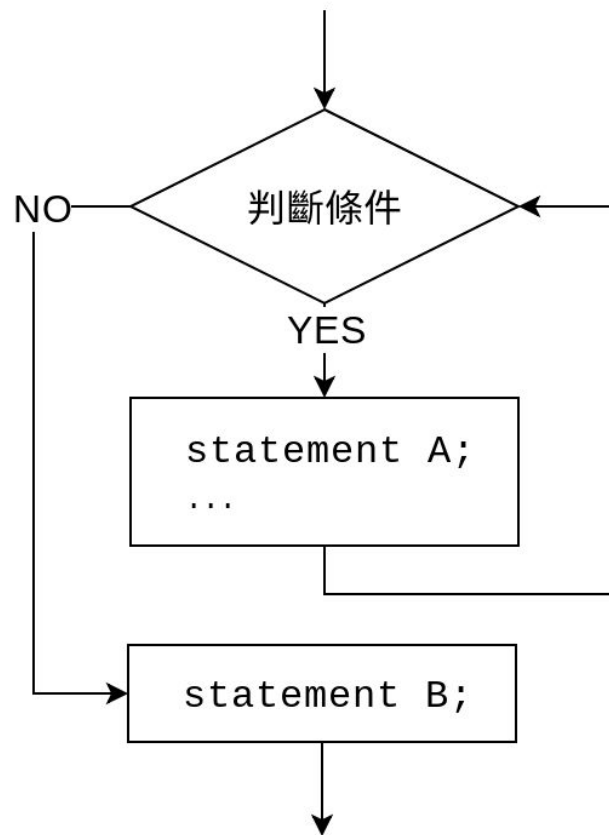
讓使用者輸入一個整數 n ，以 **for** 迴圈顯示用 "*" 印出的直角三角形。

輸入	輸出
2	* **
4	* ** *** ****



while loop

```
1. while(判斷條件) {  
2.     statement A;  
3.     ...  
4. }  
5. statement B;
```





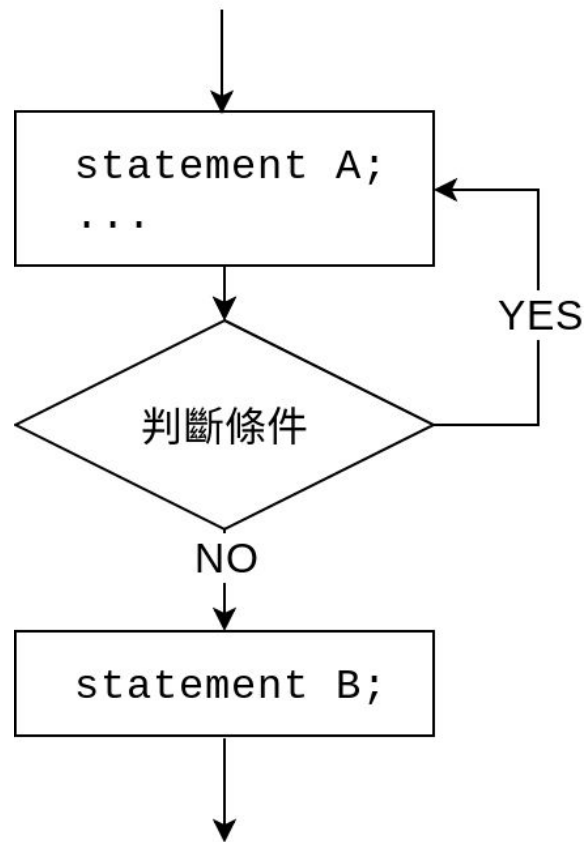
練習：印出三角形

讓使用者輸入一個整數 n ，以 **while** 迴圈顯示用 "*" 印出的直角三角形。

輸入	輸出
2	* * *
4	* * * * * * * * * *

do-while loop

```
1.  do {  
2.    statement A;  
3.    ...  
4.  } while(判斷條件);  
5.  statement B;
```





練習：印出三角形

讓使用者輸入一個整數 n ，以 **do-while** 迴圈顯示用 "*" 印出的直角三角形。

輸入	輸出
2	* **
4	* ** *** ****



練習：終極密碼戰v4

隨機產生一個0~100的數字 m ，讓使用者輸入一個整數 n 。如果猜對(即 $n=m$)則輸出YES與猜測次數；如果 $n>m$ 輸出TOO LARGE；如果 $n<m$ 輸出TOO SMALL。

註：以迴圈讓使用者可以持續猜數字，直到猜對為止。

輸入	輸出
20	TOO SMALL
50	TOO LARGE
60	TOO LARGE
40	YES 4



continue

在迴圈中執行continue會馬上離開本次迴圈執行下一次回圈

```
1.  for (初始化;判斷條件;遞增) {  
2.      statement A;  
3.      continue;  
4.      statement B;  
5.  }  
6.  statement C;
```





continue (續)

請問下面程式的輸出結果？

```
1.  for(int i=1;i<10;i++) {  
2.      if(i%3 == 0)  
3.          continue;  
4.      printf("%d ", i);  
5.  }  
6.  printf("\n");
```

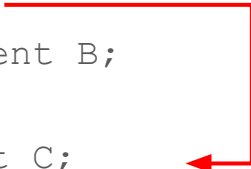




break

在迴圈(或switch)中執行break會馬上離開程式區塊

```
1.  for (初始化;判斷條件;遞增) {  
2.      statement A;  
3.      break;  
4.      statement B;  
5.  }  
6.  statement C;
```





break (續)

請問下面程式的輸出結果？

```
1.  for(int i=1;i<10;i++) {  
2.      if(i%3 == 0)  
3.          break;  
4.      printf("%d ", i);  
5.  }  
6.  printf("\n");
```





練習：判斷質數

輸一個正整數 n ，如果 n 是質數輸出YES，否則輸出NO。

想辦法利用`continue`與`break`提高執行效率

輸入	輸出
24	NO
11	YES
1009	YES



switch

```
1.  switch(expr) {  
2.  case opt1: //如果expr = opt1從這裡開始執行  
3.      statement A;  
4.      //break;  
5.  case opt2: //如果expr = opt2從這裡開始執行  
6.      statement B;  
7.      break;  
8.  default:  //都不符合從這裡開始執行  
9.      statement C;  
10.     break;  
11. }  
12. statement D;
```





練習：簡易計算機


試撰寫一程式，讓使用者選擇運算子 (+、-、*、/)。如果使用者輸入的不是這些運算子輸出 ERROR，否則讓使用者輸入兩個數字後印出計算後的結果。

註：以迴圈讓使用者可以持續計算，直到輸入Q為止。

輸入	輸出
+ 3 5	8
E	ERROR
* 5 4	20
Q	BYE

Goto與Labels

使用goto語法可以讓程式直接跳到以Labels標注的特定位置，例如：

```
1.  ...  
2.  if(error) {  
3.      goto ERROR;   
4.  }  
5.  ...  
6.  ERROR:  // Label  
7.  ...
```



Goto與Labels (續)

注意:過度使用goto語法會讓程式流程難以維護。

常見的goto時機是跳離多層的巢狀迴圈。

```
1.  for(...)
2.      for(...)
3.          for(...) {
4.              if(error)
5.                  goto END;
6.          {
7.  END:
8.      printf("Bye\n");
```



Q & A

