

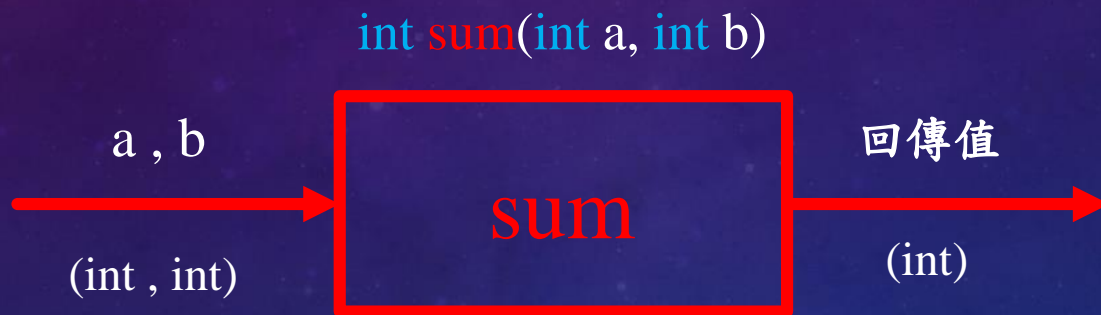
The background is a deep blue gradient with a subtle pattern of white dots. Overlaid on the left side is a large, semi-circular degree scale ranging from 150 to 260 degrees. Several concentric circles and arcs are scattered across the image, some with arrows indicating a clockwise direction. The overall aesthetic is technical and mathematical.

函式、遞迴

TA:顏瑋廷，蕭堯，楊祚暨，許晉偉

函式定義

回傳值的資料型態 函式名稱(參數的資料型態 參數名稱,...)



一個函式可以有0到多個參數,但只能有一個回傳值

函式定義

```
回傳值的資料型態 函式名稱(參數的資料型態 參數名稱,...){  
    /*程式碼*/  
    return 回傳值  
}
```

```
int sum(int a, int b){  
    return a+b;  
}
```

函式內不能再定義其他函式

函式使用

```
1 #include <stdio.h>
2
3
4 int sum(int ,int); ← 函式宣告
5
6
7 int main(void) {
8     int a=5,b=6;
9     printf("%d\n",sum(a,b)); ← 函式呼叫
10    return 0;
11 }
12
13
14 int sum(int a, int b){ ← 函式定義
15     return a+b;
16 }
17
18 |
```

函式呼叫前要先宣告或定義函式

函式參數若未宣告資料型態則預設為int

Ex:

```
int func(double x, y){
```

.....

```
}
```

此時 x 為double ，y 為 int

函式使用

```
1  #include <stdio.h>
2
3  int sum(int a,int b){
4      return a+b;
5  }
6
7  int main(void)
8  {
9      int a=5,b=6;
10     printf("%d\n",sum(a,b));
11     return 0;
12 }
13
```

函式宣告+定義

函式呼叫

注意事項

- 函式資料型別為void時，return會導致語法錯誤

```
1 #include<stdio.h>
2 void func(int a);
3
4 int main(){
5     //printf("hello\n");
6     printf("%d",func(3));
7 }
8
9 void func(int a){
10     return a;
11 }
```

```
test.c:12:3: error: void function 'func' should not return a value [-Wreturn-type]
   return a;
   ^
```

- 在函式內將此函式的某個參數定義成區域變數，將造成語法錯誤

```
int func(int a){
    int a;
    return a;
}
```

```
[matthew@yanweitingdeMBP C_class % gcc -o
test.c:10:7: error: redefinition of 'a'
   int a;
   ^
```

注意事項

- 請用有意義的變數名稱，naming rule很重要！
增加程式的可讀性。

Ex:

```
int leap_year ( int year ){  
    .....  
}
```

- 函式的大小盡量不要超過半頁，讀入的參數不宜過多

補充-變數宣告

- 變數名稱依宣告位置分為三種
 - 全域變數(Global variable)
 - 宣告在函式定義外
 - 因為容易造成名稱汙染，請避免使用全域變數
 - 區域變數(Local variable)
 - 宣告在函式定義內
 - 函式參數(Formal parameter)

補充-變數宣告

```
1  #include<stdio.h>
2  int i;
3  int my_func1(int a){
4      i=a;
5  }
6
7  int my_func2(int a){
8      int i;
9      i = a;
10 }
```

全域變數

函式參數

區域變數

同一組區塊{}裡相同的變數名稱只能有一個

複習-變數宣告

```
1  #include<stdio.h>
2  int i;
3  int my_func1(int a){
4      i=a;
5  }
6
7  int my_func2(int a){
8      int i;
9      i = a;
10 }
```

```
12 int main(){
13     i=5;
14     my_func2(10);
15     printf("%d\n",i);
16     my_func1(20);
17     printf("%d\n",i);
18 {
19     int i ;
20     i = 40;
21 }
22     printf("%d\n",i);
23     int i;
24     i=80;
25     printf("%d\n",i);
26     return 0;
27 }
```

5

20

20

80


補充-變數宣告

- 要在程式碼中使用已宣告的變數需要滿足以下兩個條件
 - 在程式碼中位於該變數名稱宣告之後
 - 該變數是全域變數或被包含在該變數名稱宣告的區塊{}內

遞迴

一個函式在它的函式內使用它自身稱為遞迴使用。在遞迴使用中，執行遞迴函數將反復使用其自身，每使用一次就進入新的一層。

```
int A() {  
    ...  
    if (xxx){  
        ...  
    } else {  
        ...  
        A();  
    }  
}
```

A blue curved arrow originates from the 'A();' line within the function body and points back to the opening curly brace of the 'int A()' function definition, visually representing the function calling itself.

- 遞迴函式將無休止地使用其自身。所以為了防止遞迴使用無終止地進行，必須在函數內設定終止遞迴的條件。常用的辦法是加條件判斷，滿足某種條件後就不再作遞迴使用，然後逐層返回。

EX:階乘

```
#include<stdio.h>
```

```
#include<stdlib.h>
```

```
int Factorial(int n) {
```

```
    if (n==0) {
```

```
        return 1;
```

```
    } else {
```

```
        return n* Factorial(n-1);
```

```
    }
```

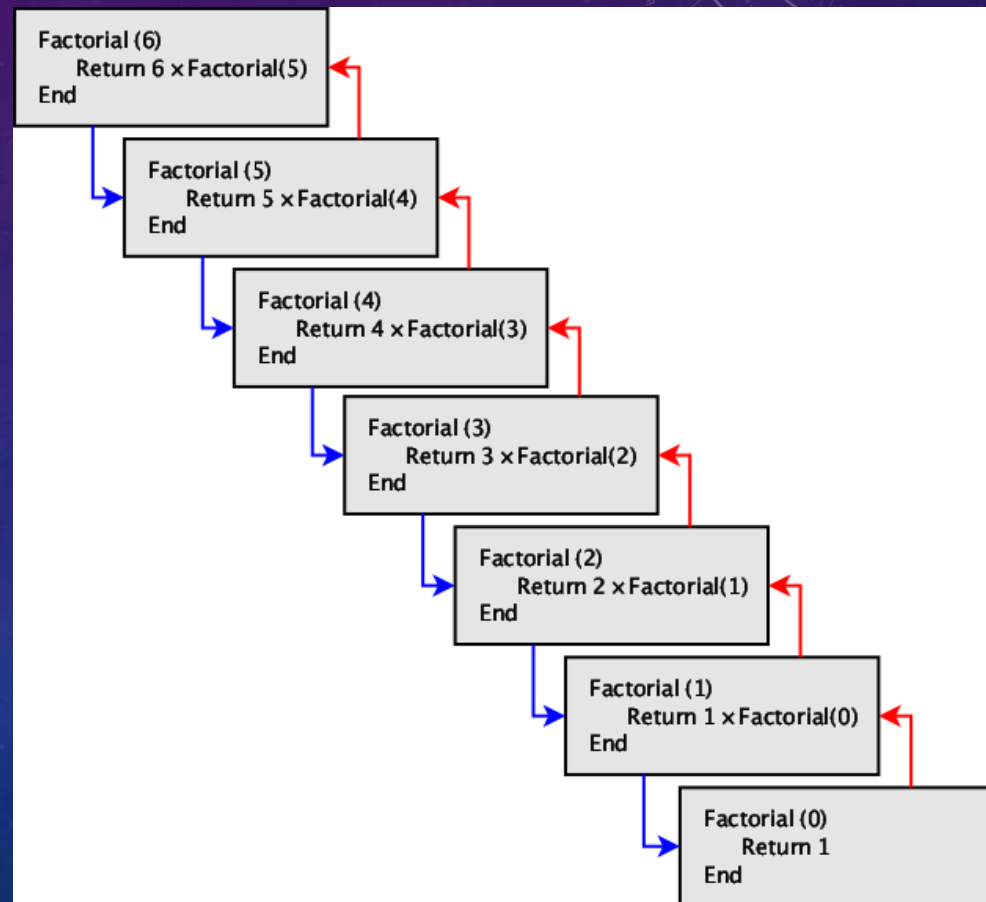
```
}
```

```
int main( ){
```

```
    printf("%d\n", Factorial(5) );
```

```
    return 0;
```

```
}
```



迭代法(Iterative)

- 迭代法是用迴圈去循環重複程式碼的某些部分來得到答案
- Ex : 求階乘 $n! = 1 * 2 * \dots * n$

```
1  #include<stdio.h>
2  int main(){
3      int n = 5;  //n!
4      int j;
5      int result = 1;
6      for(j=1; j≤n; j++){
7          result = result * j;
8      }
9      printf("%d\n", result);
10     return 0;
11 }
```



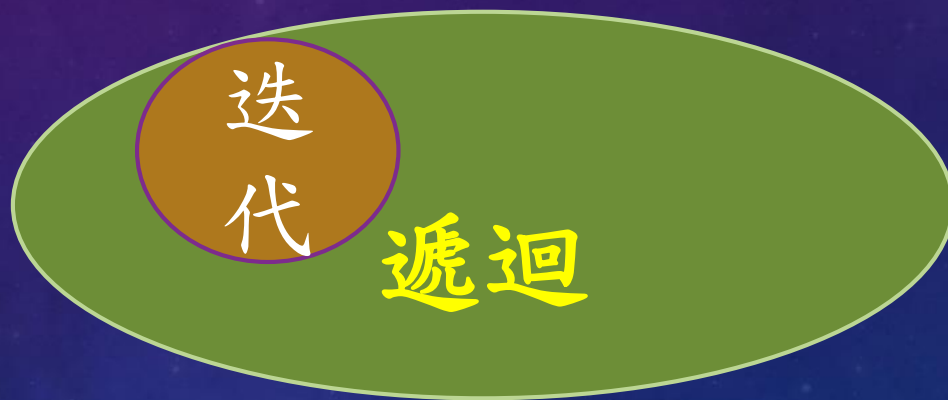
```
matthew@yanweitingdeMBP C_class
120
matthew@yanweitingdeMBP C_class
```

遞迴VS迭代

- **遞迴**：透過重複將問題分解為同類的子問題而解決問題的方法。在大部分的程式語言中都支援函數的自我呼叫，也就是說程式語言中可以**通過呼叫函數本身來進行遞迴**。所以使用遞迴實現一個計算邏輯往往只需要很短的程式碼就能解決，並且這樣的程式碼也比較容易理解。
- **迭代**：其目的通常是為了接近所需求的目標或結果。**每一次對過程的重複被稱為一次"迭代"**，每一次迭代得到的結果通常會被用來作為下一次迭代的初始值。其實遞迴都可以用迭代來代替。但相對於遞迴的簡單易懂，迭代就比較生硬難懂了。不過迭代的效率比遞迴要高，並且在空間消耗上也比較小。

遞迴VS迭代

- 遞迴中一定有迭代，但是迭代中不一定有遞迴，大部分可以相互轉換。



- 能用迭代的不要用遞迴，遞迴呼叫函式不僅浪費空間，如果遞迴太深的話還容易造成堆疊的溢位。

課後練習

輸入年分，以**1950年1月1號**為星期日為基準，請印出輸入年份之萬年曆(整年)，輸入年份為1950年後，**格式請嚴格按照要求**。

請寫一個副函式 **leap_year**，讀入參數請命名為**year**，代表要印出的那年，功能為判斷平年或閏年，且僅有此功能。

```
int leap_year( int year){  
    .....  
}
```

其他功能不可寫在**leap_year**副函式內，可全部在主程式內完成，也可自行另外設立副函式使用。

```
Please enter a year:1950  
  
JAN  
Sun Mon Tue Wed Thu Fri Sat  
01 02 03 04 05 06 07  
08 09 10 11 12 13 14  
15 16 17 18 19 20 21  
22 23 24 25 26 27 28  
29 30 31  
  
FEB  
Sun Mon Tue Wed Thu Fri Sat  
01 02 03 04  
05 06 07 08 09 10 11  
12 13 14 15 16 17 18  
19 20 21 22 23 24 25  
26 27 28  
  
MAR  
Sun Mon Tue Wed Thu Fri Sat  
01 02 03 04  
05 06 07 08 09 10 11  
12 13 14 15 16 17 18  
19 20 21 22 23 24 25  
26 27 28 29 30 31  
  
APR  
Sun Mon Tue Wed Thu Fri Sat  
01 02 03 04 05 06 07  
08 09 10 11 12 13 14  
15 16 17 18 19 20 21  
22 23 24 25 26 27 28  
29 30  
  
MAY  
Sun Mon Tue Wed Thu Fri Sat  
01 02 03 04 05 06  
07 08 09 10 11 12 13  
14 15 16 17 18 19 20  
21 22 23 24 25 26 27  
28 29 30 31  
  
JUN  
Sun Mon Tue Wed Thu Fri Sat  
01 02 03  
04 05 06 07 08 09 10  
11 12 13 14 15 16 17
```


課後練習 Hint

Hint1: 假設今年1月1號為禮拜一，則

- 1.若今年為閏年，則明年1月1號為禮拜三
- 2.若今年為平年，則明年1月1號為禮拜二

Hint2: 平潤年判斷方式

1. 西元年份除以4不可整除，為平年。
2. 西元年份除以4可整除，且除以100不可整除，為閏年。
3. 西元年份除以100可整除，且除以400不可整除，為平年
4. 西元年份除以400可整除，為閏年。

* 禁止使用任何網路上的公式，請按照上述邏輯去計算星期而非直接套用公式

Please enter a year:1950

JAN

Sun	Mon	Tue	Wed	Thu	Fri	Sat
01	02	03	04	05	06	07
08	09	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

FEB

Sun	Mon	Tue	Wed	Thu	Fri	Sat
			01	02	03	04
05	06	07	08	09	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28				

MAR

Sun	Mon	Tue	Wed	Thu	Fri	Sat
			01	02	03	04
05	06	07	08	09	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

APR

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						01
02	03	04	05	06	07	08
09	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

MAY

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	01	02	03	04	05	06
07	08	09	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

JUN

Sun	Mon	Tue	Wed	Thu	Fri	Sat
				01	02	03
04	05	06	07	08	09	10
11	12	13	14	15	16	17