第四週

王子銜, 陳毅軒, 吳尚龍

電機通訊程式設計

March 11, 2024

Outline

介紹

與數學的函式類似

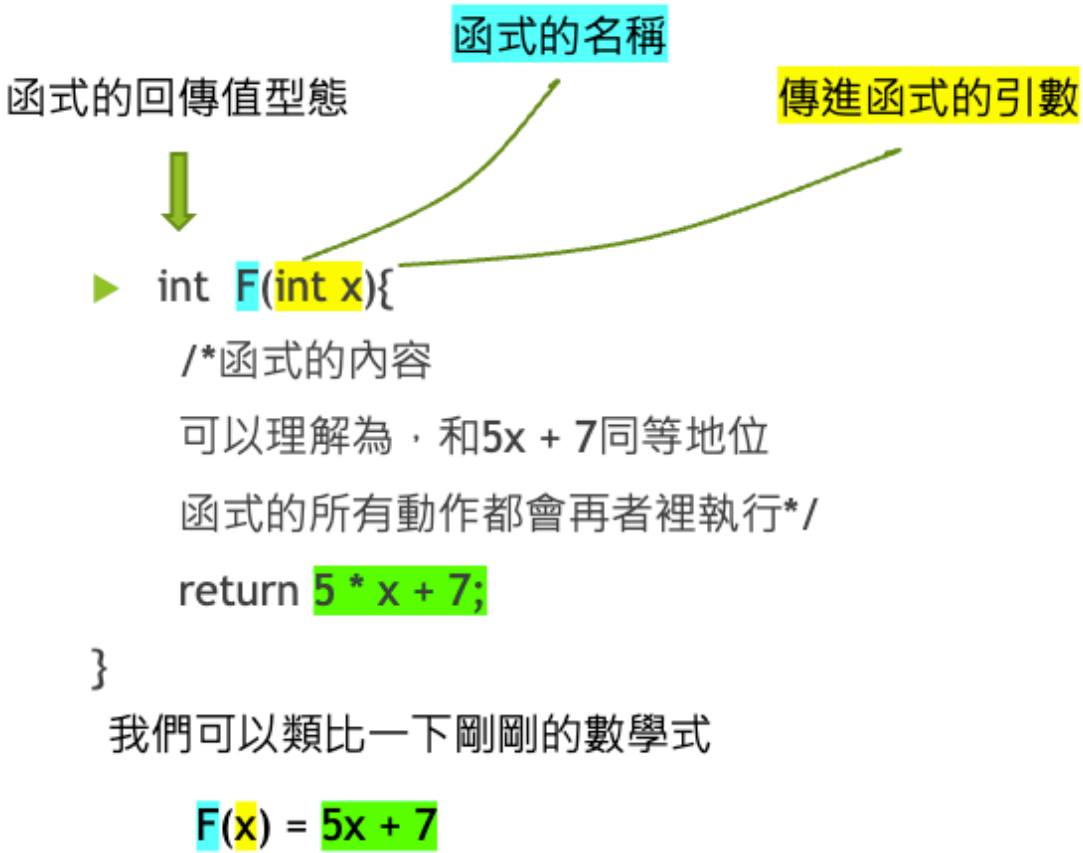
數學函示

以最簡單的一次函式來舉例:

- $F(x) = 5x + 7$

我們可以隨意丟入一個數字進這個函式裡，都會進行乘以 5 並且加上 7 這個動作。

函式的結構


函式的名稱
函式的回傳值型態
傳進函式的引數
int F(int x){
/*函式的內容
可以理解為，和5x + 7同等地位
函式的所有動作都會再者裡執行*/
return 5 * x + 7;
}
我們可以類比一下剛剛的數學式
F(x) = 5x + 7

參數及回傳值

```
1  int add(int a, int b)
```

在小括號裡，a 前面的 int 代表這個參數的型態，a 代表名稱
若有多個參數的話，可以用逗號隔開不同的變數

```
1  int add(int a, int b)
2  long long add(int a, int b)
3  double add(int a, int b)
```

回傳值跟一般的變數形態一樣，可以使用 int, char, double 等型態，但也可以不回傳，型態使用 void 就可以了

函式宣告擺放位置

第一種：一口氣寫完

```c
#include<stdio.h>

int add(int a, int b){
  return a + b;
}

int main(){
  printf("%d\n", add(4, 7));
}
```

效果：

- 優點：節省行數，一目瞭然，可以直接使用函式
- 缺點：main() 會被壓縮到很下面，函式互相呼叫時會有順序問題

函式宣告擺放位置

第二種：先打格式，再打內容

```c
#include<stdio.h>

int add(int, int);

int main(){
  printf("%d\n", add(4, 7));
}

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

效果：

- 優點：不用考慮函式順序，main() 還在上面
- 缺點：行數變多，沒辦法一目瞭然

使用函式原因

為什麼要用函式？

使用函式原因

增加可讀性、減少重複的程式碼、簡潔
不使用函式：

```
1  #include<stdio.h
2  #define int long long
3
4  signed main(){
5    // 區間和
6    int s, e;
7    scanf("%lld %lld", &s, &e);
8    printf("%lld\n", (s + e) * (e - s + 1) / 2);
9    scanf("%lld %lld", &s, &e);
10   printf("%lld\n", (s + e) * (e - s + 1) / 2);
11   scanf("%lld %lld", &s, &e);
12   printf("%lld\n", (s + e) * (e - s + 1) / 2);
13 }
```

使用函式原因

使用函式：

```
#include<stdio.h>
#define int long long

// 區間和
int sum(int s, int e){
  return (s + e) * (e - s + 1) / 2;
}

signed main(){
  int s, e;
  scanf("%lld %lld", &s, &e);
  printf("%lld\n", sum(s, e));
  scanf("%lld %lld", &s, &e);
  printf("%lld\n", sum(s, e));
  scanf("%lld %lld", &s, &e);
  printf("%lld\n", sum(s, e));
}
```

使用函式原因

分工容易

```
1  void checkdates();
2  void submitdata();
3  void papers();
4  void randompicker();
5  int main()
```

爲什麼要用遞迴？

一個函式自己呼叫自己

使用遞迴

累加,$1 + 2 + 3 + ... + n$

```c
#include<stdio.h>

int adder(int n){
  if(n == 0)return 0;
  return n+adder(n-1);
}

int main(){
  int n,total;
  printf("Enter 1 integer numbers: ");
  scanf("%d", &n);
  total = adder(n);
  printf("total = %d\n", total);
}
```

使用遞迴

累加

```
1  //output
2    Enter 1 integer numbers: 3
3    total = 6
```

使用遞迴

累加流程

adder(3); n=3; 3 + adder(2)

↓

adder(2); n=2 3+2+adder(1)

↓

adder(1); n=1 3+2+1+adder(0)

↓

adder(0); n=0 3+2+1+0

↓

return n; total = 6

使用遞迴

複習歐幾里得演算法-迴圈版

```c
#include<stdio.h>

int main(){
  int a,b,r;
  printf("Enter 2 integer numbers: ");
  scanf("%d %d", &b, &r);
  do{
      a = b;
      b = r;
      r = a % b;
      printf("a = %2d, b = %2d, r = %2d\n", a, b, r);
  }while(r != 0);
  printf("GCD = %d\n", b);
}
```

使用遞迴

歐幾里得演算法-遞迴版

```c
#include<stdio.h>

int gcd(int a, int b){
  printf("a = %2d, b = %2d\n", a, b);
  if(b == 0)return a;
  return gcd(b, a%b);
}

int main(){
  int a,b,GCD;
  printf("Enter 2 integer numbers: ");
  scanf("%d %d", &a, &b);
  GCD = gcd(a, b);
  printf("GCD = %d\n", GCD);
}
```

使用遞迴

歐幾里得演算法-遞迴版

```
1  //output
2    Enter 2 integer numbers: 98 35
3    a = 98, b = 35
4    a = 35, b = 28
5    a = 28, b =  7
6    a =  7, b =  0
7    GCD = 7
```

使用遞迴

歐幾里得演算法-遞迴版流程

gcd(98, 35); a=98,b=35

↓

gcd(35, 28); a=35,b=28

↓

gcd(28, 7); a=28,b=7

↓

gcd(7, 0); a=7,b=0

↓

return a; a=7