Function

函式

函式 Function

和數學上的函式(Function)類似

$x = 1$ ▶ $f(x) = x^2 + 2x + 1$ ▶ $f(x) = 4$

輸入數值x

得到唯一的數值$f(x)$

函式 Function

0個或多個輸入

參數
(parameter)

0個或1個輸出

回傳值(return value)

C++內建函式

C/C++中已有內建寫好的函式可供我們使用，在使用它們前，必須先引入其對應的函式庫

<cmath> sin, cos, tan

計算三角函數：

正弦(sine):

```
float/double/long double sin(float/double/long double arg);
```

餘弦(cosine):

```
float/double/long double sin(float/double/long double arg);
```

正切(tangent):

```
float/double/long double sin(float/double/long double arg);
```

* arg的單位是 弧度(rad)

<cmath> sin, cos, tan

計算三角函數：

M_PI是π的常數

```
sin(M_PI / 6)
```

回傳 0.5

```
cos(M_PI / 6)
```

回傳 0.866025

```
tan(M_PI / 6)
```

回傳 0.57735

<cmath> abs

計算絕對值：

```
int abs(int n);

long abs(long n);

long long abs(long long n);
```

<cmath> abs

計算絕對值：

```
abs(-3);
```

回傳 3

<cmath> pow

計算$base^{exp}$：

```
float pow(float base, float exp);

double pow(double base, double exp);

long double pow(long double base, long double exp);
```

<cmath> pow

計算$base^{exp}$：

```
pow(2, 3);
```

回傳 8

<cmath> sqrt

計算$\sqrt{arg}$：

```
float sqrt(float arg);

double sqrt(double arg);

long double sqrt(long double arg);
```

<cmath> sqrt

計算$\sqrt{arg}$：

```
sqrt(2);
```

回傳 1.41421

<algorithm> sort

- sort(first, last);
- 將first ~ last - 1之間的元素由小到大排列

```
int arr[5] = {4, 1, 5, 3, 2};
sort(arr, arr + 5);
```

即可將arr陣列從第0~4格排列

<algorithm> sort

- sort(first, last, cmp);
- 將first ~ last - 1之間的元素依照自訂比照函式排列

```
bool cmp(int a, int b){
    return a > b;
}

int arr[5] = {4, 1, 5, 3, 2};
sort(arr, arr + 5, cmp);
```

如果a>b回傳true，反之false

即可將arr陣列從第0~4格大到小排列

<algorithm> stable_sort

- `stable_sort(first, last);`
- 將first ~ last - 1之間的元素由小到大排列
- 保證相同數值的元素，在排序後關係不變

<algorithm> swap

```
swap(a, b);
```

將a和b交換

函式宣告

在C++中可以這樣宣告：

```
回傳值型態 函式名稱(參數類別1 參數名1, 參數類別2 參數名2, ...)
{
    程式區塊;
    return 回傳值;
}
```

函式一旦執行到return，就會回傳，略過之後的程式碼

函式宣告

範例：計算$f(x) = x^2 + 2x + 1$

```
int function_of_x(int x)
{
    return x*x+x*2+1;
}
```

函式宣告

範例：計算$f(x) = x^2 + 2x + 1$

回傳值型態

```
int function_of_x(int x)
{
    return x*x+x*2+1;
}
```

函式宣告

範例：計算$f(x) = x^2 + 2x + 1$

函式名稱

```
int function_of_x(int x)
{
    return x*x+x*2+1;
}
```

函式宣告

範例：計算$f(x) = x^2 + 2x + 1$

參數類別

```
int function_of_x(int x)
{
    return x*x+x*2+1;
}
```

函式宣告

範例：計算$f(x) = x^2 + 2x + 1$

參數名

```
int function_of_x(int x)
{
    return x*x+x*2+1;
}
```

函式宣告

範例：計算$f(x) = x^2 + 2x + 1$

```
int function_of_x(int x)
{
    return x*x+x*2+1;
}
```

回傳值

函式宣告

範例：輸出貓咪

```
void cat(){
    cout << "\
      |\\      _,,,---,,_\n\
ZZZzz /,`.-\'`\'    -.  ;-;;,_\n\
     |,4-  ) )-,_. ,\\ (  `\'-\'\n\
    \'---\'\'(_/--\'  `-\'\\_)  Felix Lee \n\
";
    return;
}
```

函式宣告

回傳值型態 沒有回傳值可以使用void

```
void cat(){
    cout << "\
      |\\      _,,,---,,_\n\
ZZZzz /,`.-\'`\'    -.  ;-;;,_\n\
     |,4-  ) )-,_. ,\\ (  `\'-\'\n\
    \'---\'\'(_/--\'  `-\'\\_)  Felix Lee \n\
";
    return;
}
```

Cambridge Dictionary

void (noun)

a large hole or empty space
空洞；空間；空白

She stood at the edge of the chasm and stared into the void.
她站在裂縫的邊緣凝視著下面的深淵。

Before Einstein, space was regarded as a formless void.
在愛因斯坦之前，太空被認為是無形的虛空之地。

函式宣告

函數名稱 貓咪

```
void cat(){
    cout << "\
      |\\      _,,,---,,_\n\
ZZZzz /,`.-\'`\'    -.  ;-;;,_\n\
     |,4-  ) )-,_. ,\\ (  `\'-\'\n\
    \'---\'\'(_/--\'  `-\'\\_)  Felix Lee \n\
";
    return;
}
```

函式宣告

範例：輸出貓咪

```
void cat(){
    cout << "\
      |\\       _,,,---,,_\n\
ZZZzz /,`.-\'`\'    -.  ;-;;,_\n\
     |,4-  ) )-,_. ,\\ (  `\'-\'\n\
    \'---\'\'(_/--\'  `-\'\\_)  Felix Lee \n\
";
    return;
}
```

當一個敘述過長時，可以使用\來斷行

編譯器將它們視為同一行敘述

函式宣告

範例：輸出貓咪

編譯器的眼中：

```
void cat(){
    cout << "       |\\       _,,,---,,_\nZZZzz /,`.-\'`\'    -.  ;-
;;,_\n     |,4-  ) )-,_. ,\\ (  `\'-\'\n    \'---\'\'(_/--\'  `-
\'\\_)  Felix Lee \n";
    return;
}
```

函式宣告

範例：輸出貓咪

```
void cat(){
    cout << "\
      |\\       _,,,---,,_\n\
ZZZzz /,`.-\'`\'    -.  ;-;;,_\n\
     |,4-  ) )-,_. ,\\ (  `\'-\'\n\
    \'---\'\'(_/--\'  `-\'\\_)  Felix Lee \n\
";
    return;
}
```

void沒有回傳值

return可加可不加

函式呼叫

函式的呼叫：

```
函式名稱(參數名1, 參數名2, ...);

變數 = 函式名稱(參數名1, 參數名2, ...);
```

函式呼叫

例如：

```
1   #include <iostream>
2   using namespace std;
3
4   int function_of_x(int x){
5       return x*x+x*2+1;
6   }
7
8   int main(){
9       cout << function_of_x(1);
10      return 0;
11  }
```

函式呼叫

例如：

```
1   #include <iostream>
2   using namespace std;
3
4   int function_of_x(int x){
5       return x*x+x*2+1;
6   }
7
8   int main(){
9       cout << function_of_x(1);
10      return 0;
11  }
```

宣告function_of_x

函式呼叫

例如：

```
1   #include <iostream>
2   using namespace std;
3
4   int function_of_x(int x){
5       return x*x+x*2+1;
6   }
7
8   int main(){
9       cout << function_of_x(1);
10      return 0;
11  }
```

宣告function_of_x

函式必須在函式main前宣告

否則無法在函式main中使用

函式呼叫

例如：

```
1   #include <iostream>
2   using namespace std;
3
4   int function_of_x(int x){
5       return x*x+x*2+1;
6   }
7
8   int main(){
9       cout << function_of_x(1);
10      return 0;
11  }
```

呼叫函式function_of_x

函式呼叫

例如：

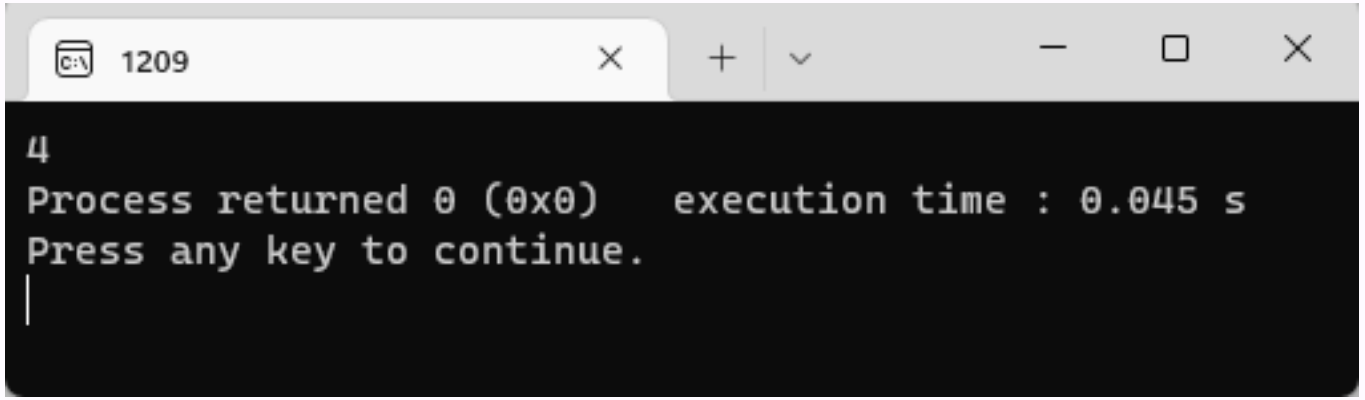
```
1   #include <iostream>
2   using namespace std;
3
4   int function_of_x(int x){
5       return x*x+x*2+1;
6   }
7
8   int main(){
9       cout << function_of_x(1);
10      return 0;
11  }
```

輸出回傳值

函式呼叫

例如：

```
1   #include <iostream>
2   using namespace std;
3
4   int function_of_x(int x){
5       return x*x+x*2+1;
6   }
7
8   int main(){
9       cout << function_of_x(1);
10      return 0;
11  }
```



函式呼叫

例如：

```cpp
#include <iostream>
using namespace std;

void cat(){
    cout << "\
      |\\      _,,,---,,_\n\
ZZZzz /,`.-\'`\'    -.  ;-;;,_\n\
     |,4-  ) )-,_. ,\\ (  `\'-\'\n\
    \'---\'\'(_/--\'  `-\'\\_)  Felix Lee \n\
";
    return;
}

int main(){
    cat();
    return 0;
}
```

函式呼叫

例如：

```
1   #include <iostream>
2   using namespace std;
3
4   void cat(){
5       cout << "\
6         |\\      _,,,---,,_\n\
7   ZZZzz /,`.-\'`\'    -.  ;-;;,_\n\
8        |,4-  ) )-,_. ,\\ (  `\'-\'\n\
9       \'---\'\'(_/--\'  `-\'\\_)  Felix Lee \n\
10  ";
11      return;
12  }
13
14  int main(){
15      cat();
16      return 0;
17  }
```

宣告cat

函式呼叫

例如：

```
1   #include <iostream>
2   using namespace std;
3
4   void cat(){
5       cout << "\
6         |\\      _,,,---,,_\n\
7   ZZZzz /,`.-\'`\'    -.  ;-;;,_\n\
8        |,4-  ) )-,_. ,\\ (  `\'-\'\n\
9       \'---\'\'(_/--\'  `-\'\\_)  Felix Lee \n\
10  ";
11      return;
12  }
13
14  int main(){
15      cat();
16      return 0;
17  }
```

呼叫函式cat

函式呼叫

例如：

```
1   #include
2   using na
3
4   void cat
5       cout
6         |\
7   ZZZzz /,
8        |,4
9       \'--
10  ";
```

1209

```
      |\       _,,,---,,_
ZZZzz /,`.-'``'    -.  ;-;;,_
     |,4-  ) )-,_. ,\ (  ``'-'
    '---''(_/--'  `-'\_)  Felix Lee

Process returned 0 (0x0)   execution time : 0.056 s
Press any key to continue.
```

函式原型宣告
(function prototype declaration)

- 有的時候，我們自訂的函式會過長，使要在自訂函式後面的main函式難以尋找。
- 此時可以使用函式原型宣告(function prototype declaration)

```
1    #include <iostream>
2    using namespace std;
3
4    int my_function(int a, int b){
         ⋮
1000         return result;
1001     }
1002
1003     int main(){
1004         cout << my_function(3, 5);
1005         return 0;
1006     }
```

函式原型宣告
(function prototype declaration)

回傳值型態 函式名稱(**參數類別1** **參數名1**, **參數類別2** **參數名2**, ...);

函式原型宣告
(function prototype declaration)

回傳值型態 函式名稱(**參數類別1** **參數名1**, **參數類別2** **參數名2**, ...);

注意需要加;

函式原型宣告
(function prototype declaration)

```
1    #include <iostream>
2    using namespace std;
3
4    int my_function(int, int);
5
6    int main(){
7        cout << my_function(3, 5);
8        return 0;
9    }
10
11   int my_function(int a, int b){
             ⋮
1007     return result;
1008 }
```

函式原型宣告
(function prototype declaration)

```
1  #include <iostream>
2  using namespace std;
3
4  int my_function(int, int);
5
6  int main(){
7      cout << my_function(3, 5);
8      return 0;
```

函式原型

函式原型宣告
(function prototype declaration)

```
4   int my_function(int, int);
5
6   int main(){
7       cout << my_function(3, 5);
8       return 0;
9   }
10
11  int my_function(int a, int b){
```

Main函式

5

函式原型宣告
(function prototype declaration)

```
9   }
10
11  int my_function(int a, int b){
                ⋮
1007    return result;
1008 }
```

函式定義

函式原型宣告
(function prototype declaration)

```
1    #include <iostream>
2    using namespace std;
3
4    int my_function(int, int);
5
6    int main(){
7        cout << my_function(3, 5);
8        return 0;
9    }
10
11   int my_function(int a, int b){
             ⋮
1007     return result;
1008 }
```

如此一來，程式的可讀性
與美觀度都有了提升

函式執行流程

```
1   #include <iostream>
2   using namespace std;
3
4   void f(){
5       cout << "in function" << endl;
6   }
7
8   int main(){
9       cout << "function start" << endl;
10      f();
11      cout << "function end" << endl;
12      return 0;
13  }
```

函式執行流程

```
1   #include <iostream>
2   using namespace std;
3
4   void f(){
5       cout << "in function" << endl;
6   }
7
8   int main(){
9       cout << "function start" << endl;
10      f();
11      cout << "function end" << endl;
12      return 0;
13  }
```

程式的起點

output

[redacted]

函式執行流程

```
1   #include <iostream>
2   using namespace std;
3
4   void f(){
5       cout << "in function" << endl;
6   }
7
8   int main(){
9       cout << "function start" << endl;
10      f();
11      cout << "function end" << endl;
12      return 0;
13  }
```

output

```
function start
```

輸出”function start”

函式執行流程

```
1   #include <iostream>
2   using namespace std;
3
4   void f(){
5       cout << "in function" << endl;
6   }
7
8   int main(){
9       cout << "function start" << endl;
10      f();
11      cout << "function end" << endl;
12      return 0;
13  }
```

呼叫函式f

output

```
function start
```

函式執行流程

```
1   #include <iostream>
2   using namespace std;
3
4   void f(){
5       cout << "in function" << endl;
6   }
7
8   int main(){
9       cout << "function start" << endl;
10      f();
11      cout << "function end" << endl;
12      return 0;
13  }
```

函式f開始

output

```
function start
```

函式執行流程

```
1   #include <iostream>
2   using namespace std;
3
4   void f(){
5       cout << "in function" << endl;
6   }
7
8   int main(){
9       cout << "function start" << endl;
10      f();
11      cout << "function end" << endl;
12      return 0;
13  }
```

輸出”in function”

output

```
function start
in function
```

函式執行流程

```
1   #include <iostream>
2   using namespace std;
3
4   void f(){
5       cout << "in function" << endl;
6   }
7
8   int main(){
9       cout << "function start" << endl;
10      f();
11      cout << "function end" << endl;
12      return 0;
13  }
```

函式f結束　沒有回傳值

output

```
function start
in function
```

函式執行流程

```
1   #include <iostream>
2   using namespace std;
3
4   void f(){
5       cout << "in function" << endl;
6   }
7
8   int main(){
9       cout << "function start" << endl;
10      f();
11      cout << "function end" << endl;
12      return 0;
13  }
```

output

```
function start
in function
function end
```

輸出”function end”

函式執行流程

```
1   #include <iostream>
2   using namespace std;
3
4   void f(){
5       cout << "in function" << endl;
6   }
7
8   int main(){
9       cout << "function start" << endl;
10      f();
11      cout << "function end" << endl;
12      return 0;
13  }
```

函式main結束，程式結束

output

```
function start
in function
function end
```

函式執行流程

```
1   #include <iostream>
2   using namespace std;
3
4   void f(){
5       cout << "in function" << endl;
6   }
7
8   int main(){
9       cout << "function start" << endl;
10      f();
11      cout << "function end" << endl;
12      return 0;
13  }
```

1209

```
function start
in function
function end

Process returned 0 (0x0)   exe
 s
Press any key to continue.
```

參數的傳遞

```
1   #include <iostream>
2   using namespace std;
3
4   int add(int a, int b){
5       return a+b;
6   }
7
8   int multiply(int x, int y){
9       int Sum = 0;
10      for (int i = 0; i < y; i++){
11          Sum = add(Sum, x);
12      }
13      return Sum;
14  }
15
16  int main(){
17      cout << multiply(3, 5);
18      return 0;
19  }
```

參數的傳遞

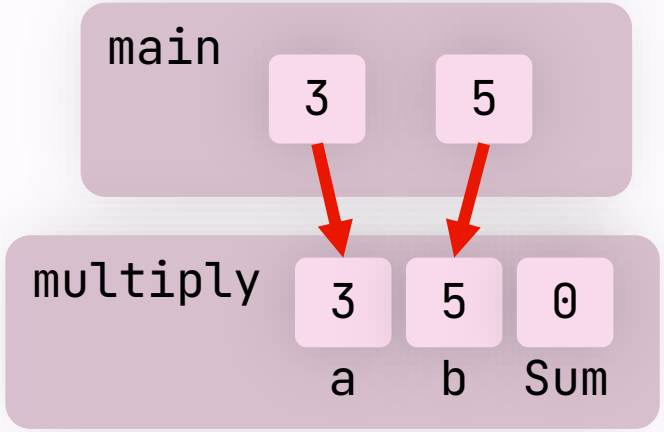
```
1   #include <iostream>
2   using namespace std;
3
4   int add(int a, int b){
5       return a+b;
6   }
7
8   int multiply(int x, int y){
9       int Sum = 0;
10      for (int i = 0; i < y; i++){
11          Sum = add(Sum, x);
12      }
13      return Sum;
14  }
15
16  int main(){
17      cout << multiply(3, 5);
18      return 0;
19  }
```

程式開始

參數的傳遞

```
1   #include <iostream>
2   using namespace std;
3
4   int add(int a, int b){
5       return a+b;
6   }
7
8   int multiply(int x, int y){
9       int Sum = 0;
10      for (int i = 0; i < y; i++){
11          Sum = add(Sum, x);
12      }
13      return Sum;
14  }
15
16  int main(){
17      cout << multiply(3, 5);
18      return 0;
19  }
```

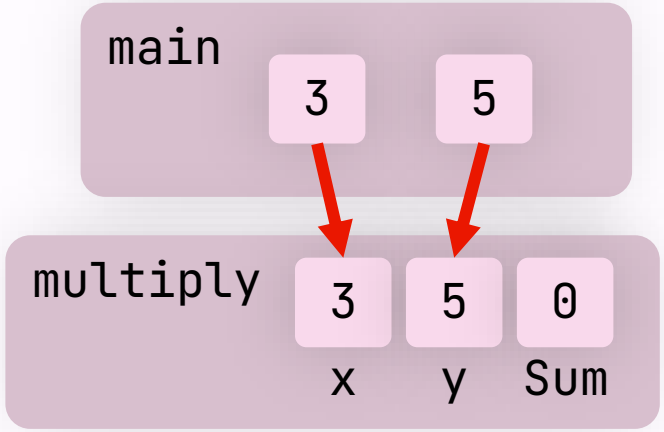
呼叫multiply(3, 5)



參數的傳遞

```
1   #include <iostream>
2   using namespace std;
3
4   int add(int a, int b){
5       return a+b;
6   }
7
8   int multiply(int x, int y){
9       int Sum = 0;
10      for (int i = 0; i < y; i++){
11          Sum = add(Sum, x);
12      }
13      return Sum;
14  }
15
16  int main(){
17      cout << multiply(3, 5);
18      return 0;
19  }
```

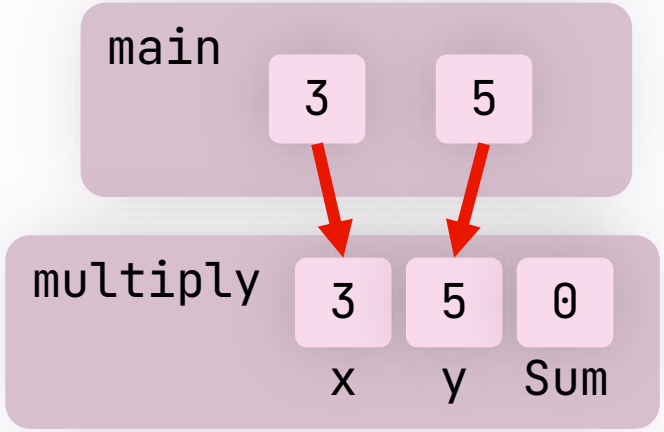
宣告一整數Sum



參數的傳遞

```
1   #include <iostream>
2   using namespace std;
3
4   int add(int a, int b){
5       return a+b;
6   }
7
8   int multiply(int x, int y){
9       int Sum = 0;
10      for (int i = 0; i < y; i++){
11          Sum = add(Sum, x);
12      }
13      return Sum;
14  }
15
16  int main(){
17      cout << multiply(3, 5);
18      return 0;
19  }
```

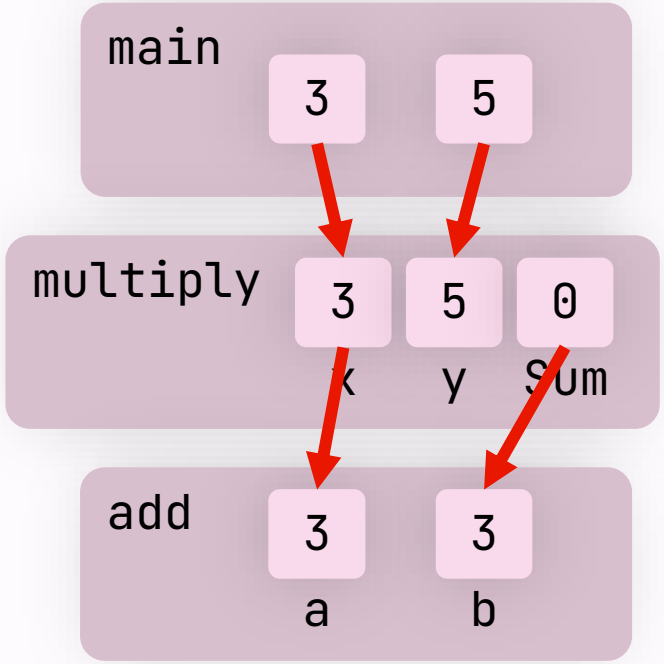
重複執行y次



參數的傳遞

```
1   #include <iostream>
2   using namespace std;
3
4   int add(int a, int b){
5       return a+b;
6   }
7
8   int multiply(int x, int y){
9       int Sum = 0;
10      for (int i = 0; i < y; i++){
11          Sum = add(Sum, x);
12      }
13      return Sum;
14  }
15
16  int main(){
17      cout << multiply(3, 5);
18      return 0;
19  }
```

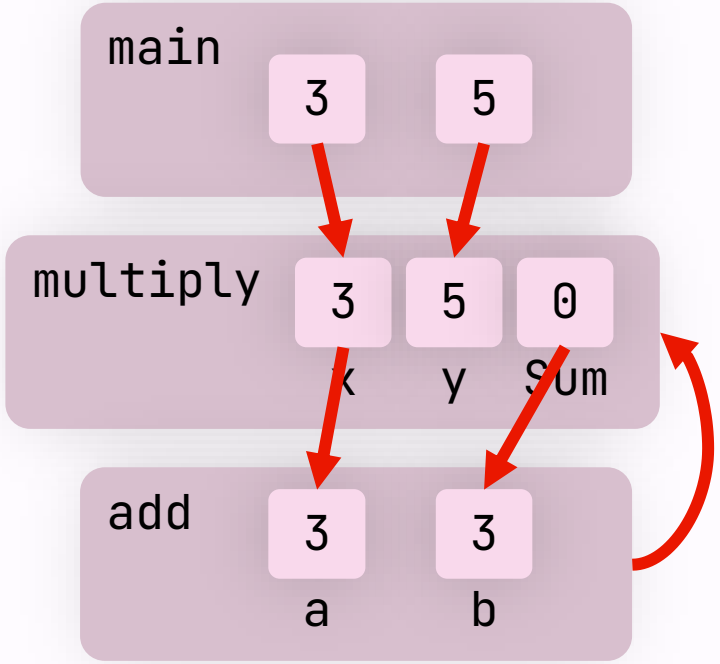
呼叫add(Sum, x)



參數的傳遞

```
1   #include <iostream>
2   using namespace std;
3
4   int add(int a, int b){
5       return a+b;
6   }
7
8   int multiply(int x, int y){
9       int Sum = 0;
10      for (int i = 0; i < y; i++){
11          Sum = add(Sum, x);
12      }
13      return Sum;
14  }
15
16  int main(){
17      cout << multiply(3, 5);
18      return 0;
19  }
```

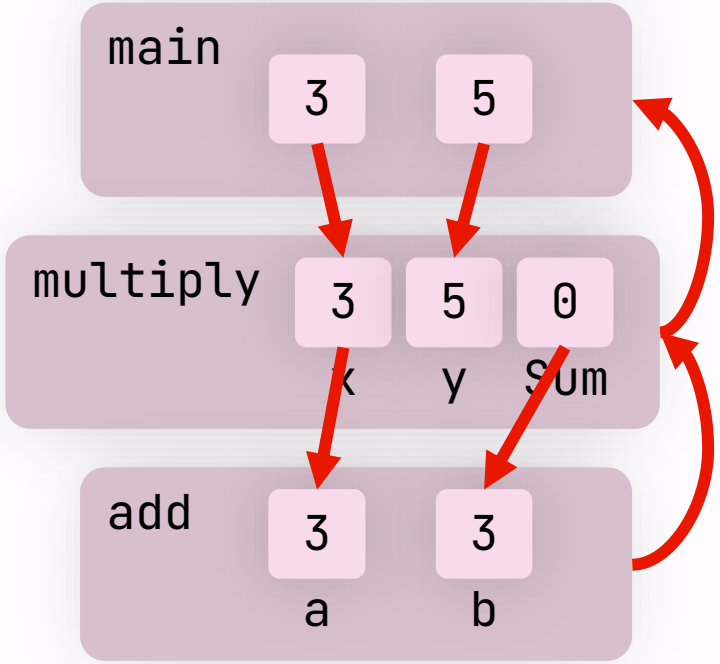
回傳a+b



參數的傳遞

```
1   #include <iostream>
2   using namespace std;
3
4   int add(int a, int b){
5       return a+b;
6   }
7
8   int multiply(int x, int y){
9       int Sum = 0;
10      for (int i = 0; i < y; i++){
11          Sum = add(Sum, x);
12      }
13      return Sum;
14  }
15
16  int main(){
17      cout << multiply(3, 5);
18      return 0;
19  }
```

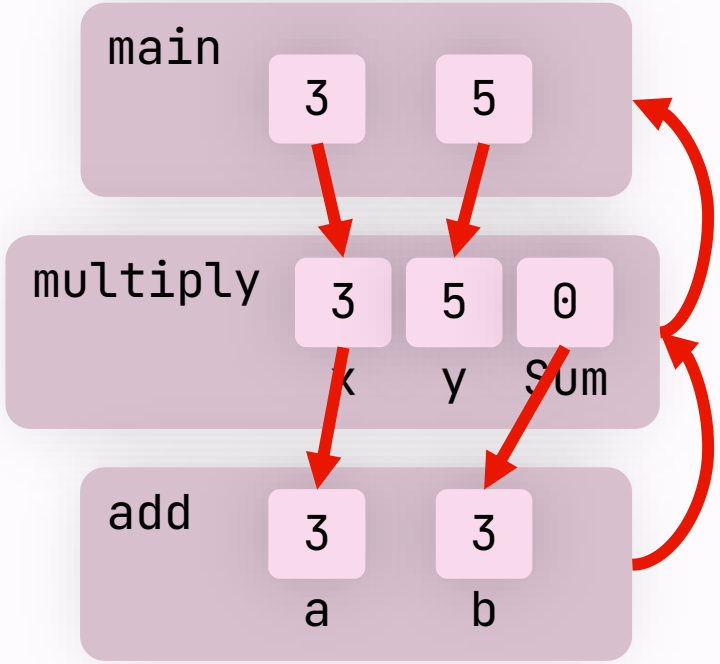
回傳Sum



參數的傳遞

```
1   #include <iostream>
2   using namespace std;
3
4   int add(int a, int b){
5       return a+b;
6   }
7
8   int multiply(int x, int y){
9       int Sum = 0;
10      for (int i = 0; i < y; i++){
11          Sum = add(Sum, x);
12      }
13      return Sum;
14  }
15
16  int main(){
17      cout << multiply(3, 5);
18      return 0;
19  }
```

輸出結果



參數的傳遞

```
1   #include <iostream>
```

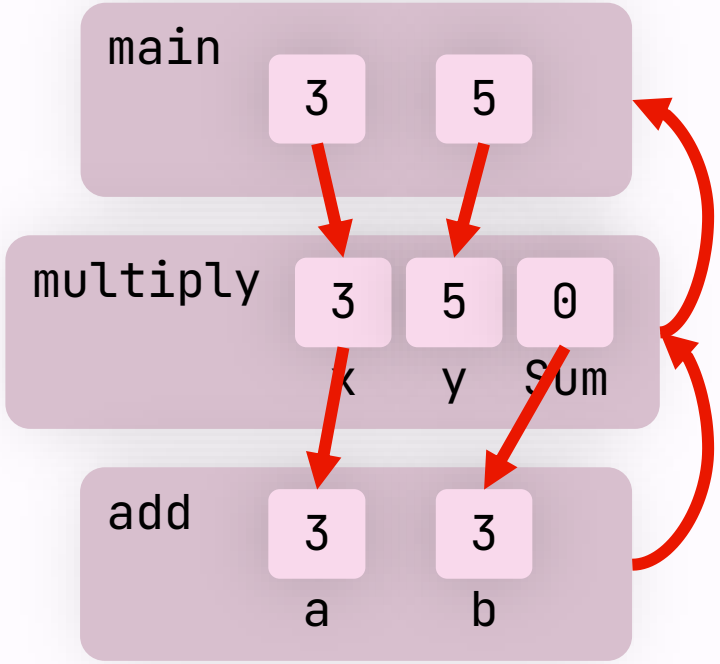
```
15
{
< multiply(3, 5);
0;
```

輸出結果

```
14  }
```

```
1209

15
Process returned 0 (0x0)   execution time : 0.05
1 s
Press any key to continue.
```



參數的傳遞

```
1   #include <iostream>
```

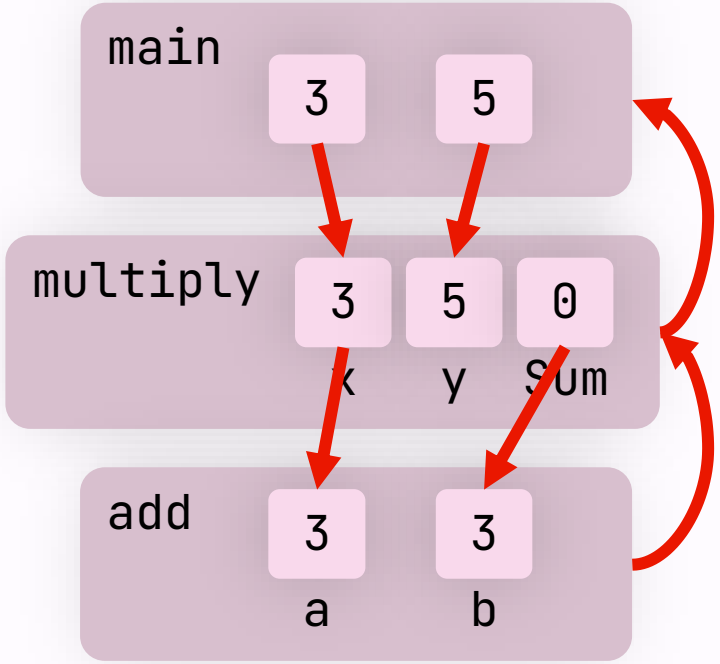
```
15
{
< multiply(3, 5);
0;
```

輸出結果

```
14  }
```

```
1209

15
Process returned 0 (0x0)   execution time : 0.05
1 s
Press any key to continue.
```



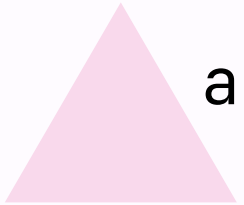
函式多載(overload)

- 有時，我們可能需要重複定義多個相同名稱但參數個數與類別不同的函式

函式多載(overload)

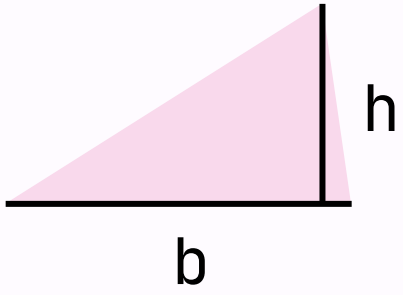
- 例如，計算三角形面積(A)

邊長為a的
正三角形



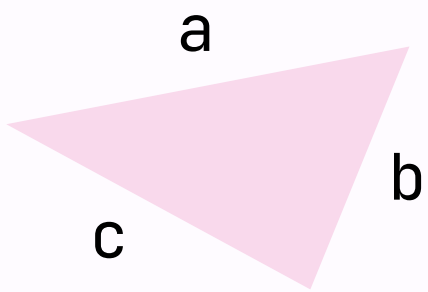
$$A = \frac{\sqrt{3}}{2}a^2$$

底為b，高為h的
三角形



$$A = \frac{1}{2}bh$$

三邊長分別為a,
b, c的三角形



$$s = \frac{a+b+c}{2}$$

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

函式多載(overload)

```
1   #include <iostream>
2   #include <cmath>
3   using namespace std;
4
5   double area(double a){
6       return sqrt(3) / 2 * a * a;
7   }
8
9   double area(double b, double h){
10      return 1 / 2.0 * b * h;
11  }
12
13  double area(double a, double b, double c){
14      double s = (a+b+c) / 2;
15      double A = sqrt(s * (s-a) * (s-b) * (s-c));
16      return A;
17  }
18
19  int main(){
20      cout << area(3) << endl;
21      cout << area(5, 10) << endl;
22      cout << area(3, 7, 5) << endl;
23      return 0;
24  }
```

函式多載(overload)

```
1   #include <iostream>
2   #include <cmath>
3   using namespace std;
4
5   double area(double a){
6       return sqrt(3) / 2 *
7   }
8
9   double area(double b, double h){
10      return 1 / 2.0 * b * h;
11  }
12
```

```
13  double area(double a, double b, double c){
14      double s = (a+b+c) / 2;
15      double A = sqrt(s * (s-a) * (s-b) * (s-c));
16      return A;
```

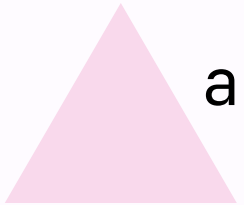
相同名稱的函數，
但參數個數不同

```
){
20      cout << area(3) << endl;
21      cout << area(5, 10) << endl;
22      cout << area(3, 7, 5) << endl;
23      return 0;
24  }
```

函式多載(overload)

```
1   #include <iostream>
2   #include <cmath>
3   using namespace std;
4
5   double area(double a){
6       return sqrt(3) / 2 * a * a;
7   }
8
9   double area(double b, double h){
10      return 1 / 2.0 * b * h;
11  }
```

邊長為a的
正三角形

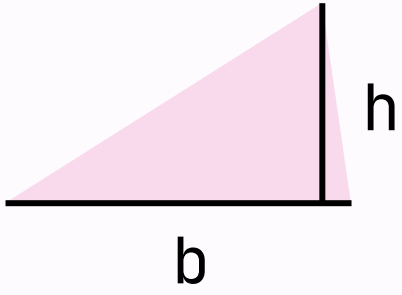


$$A = \frac{\sqrt{3}}{2} a^2$$

函式多載(overload)

```
5   double area(double a){
6       return sqrt(3) / 2 * a * a;
7   }
8
9   double area(double b, double h){
10      return 1 / 2.0 * b * h;
11  }
12
13  double area(double a, double b, double c){
14      double s = (a+b+c) / 2;
15      double A = sqrt(s * (s-a) * (s-b) * (s-c));
16      return A;
```

底為b，高為h的三角形

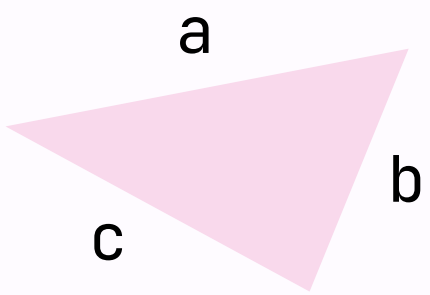


$$A = \frac{1}{2}bh$$

函式多載(overload)

```
10      return 1 / 2.0 * b * h;
11  }
12
13  double area(double a, double b, double c){
14      double s = (a+b+c) / 2;
15      double A = sqrt(s * (s-a) * (s-b) * (s-c));
16      return A;
17  }
18
19  int main(){
20      cout << area(3) << endl;
21      cout << area(5, 10) << endl;
```

三邊長分別為a, b, c的三角形



$$s = \frac{a+b+c}{2}$$

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

函式多載(overload)

```
double area(double a, double b, double c){
    return sqrt(s * (s-b) * (s-c));
}

int main(){
    cout << area(3) << endl;
    cout << area(5, 10) << endl;
    cout << area(3, 7, 5) << endl;
    return 0;
}
```

呼叫

```
double area(double a){
    return sqrt(3) / 2 * a * a;
}
```

函式多載(overload)

```
15  double area(double a, double b, double c){
                                    * (s-b) * (s-c));
16      return A;
17  }
18
19  int main(){
20      cout << area(3) << endl;
21      cout << area(5, 10) << endl;
22      cout << area(3, 7, 5) << endl;
23      return 0;
24  }
```

呼叫

```
 9  double area(double b, double h){
10      return 1 / 2.0 * b * h;
11  }
```

函式多載(overload)

```
14      double s = (a+b+c) / 2;
                                  * (s-b) * (s-c));
17  }
18
19  int main(){
20      cout << area(3) << endl;
21      cout << area(5, 10) << endl;
22      cout << area(3, 7, 5) << endl;
23      return 0;
24  }
```

呼叫

```
13  double area(double a, double b, double c){
14      double s = (a+b+c) / 2;
15      double A = sqrt(s * (s-a) * (s-b) * (s-c));
16      return A;
17  }
```

函式多載(overload)

```
1   #include <iostream>
2   #include
3   using nam
4
5   double ar
6       retur
7   }
8
9   double ar
10      retur
11  }
12
13  double area(double a, double b, double c){
                                    (s-b) * (s-c));
                                    endl;
                                    ) << endl;
                                    5) << endl;
```

triangle_area

```
7.79423
25
6.49519

Process returned 0 (0x0)   execution time : 0.038
s
Press any key to continue.
```

函式多載(overload)

⚠ 注意 ⚠

重載方法之間，若回傳型別不同，
將使程式碼 難以維護、理解，若有新的意圖應取新的方法名稱。

軟體開發大師 Kent Beck