Ya-Chun Hsu @ Taipei Municipal Jianguo High School

試試看

- 設計一個函式swap，將傳入的兩個int變數數值交換

```
1  #include <iostream>
2  using namespace std;
3
4  void swap(                    ){
5
6
7
8  }
9  int main(){
10      int a=10,b=20;
11      cout<<"a="<<a<<" b="<<b<<endl;
12      swap(                );
13      cout<<"a="<<a<<" b="<<b<<endl;
14      return 0;
15 }
```

這樣寫？

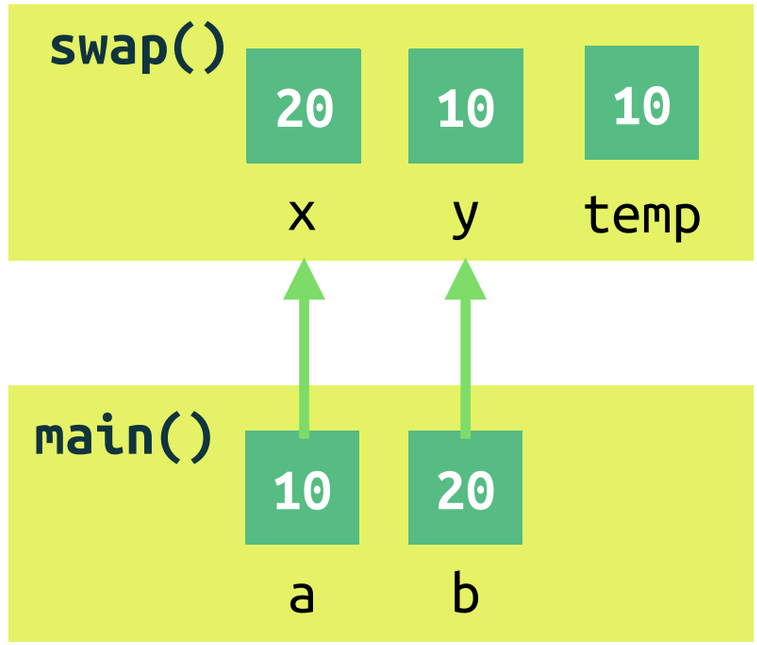
```
1  #include <iostream>
2  using namespace std;
3
4  void swap(int x, int y){
5      int temp=x;
6      x=y;
7      y=temp;
8  }
9  int main(){
10     int a=10,b=20;
11     cout<<"a="<<a<<" b="<<b<<endl;
12     swap(a,b);
13     cout<<"a="<<a<<" b="<<b<<endl;
14     return 0;
15 }
```

輸出結果

```
a=10, b=20
a=10, b=20
```

為什麼不會交換？

```
1  #include <iostream>
2  using namespace std;
3
4  void swap(int x, int y){
5      int temp=x;
6      x=y;
7      y=temp;
8  }
9  int main(){
10     int a=10,b=20;
11     cout<<"a="<<a<<" b="<<b<<endl;
12     swap(a,b);
13     cout<<"a="<<a<<" b="<<b<<endl;
14     return 0;
15 }
```



參數傳遞

- 截至目前為止，我們知道如何在呼叫函式時傳遞參數的值
  - 只有「傳值」(pass by value)，值被複製一份，傳遞者和接受者兩個變數各儲存一份，互不相干
- 我們需要能夠存取宣告於函式外的變數的方法

指標 pointer

- int儲存整數值，float/double儲存小數...
- 指標儲存**記憶體位置**

指標 pointer

- 宣告指標時要使用*符號

儲存整數變數記憶體位置的指標

```
int *p;
double *q;
```

儲存浮點數變數記憶體位置的指標

取址符號 &

- 取得變數的記憶體位置

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      int age=10;
6      float average=1.234;
7      cout<<&age<<endl<<&average<<endl;
8      return 0;
9  }
```

輸出結果

```
0022FF4C
0022FF48
```

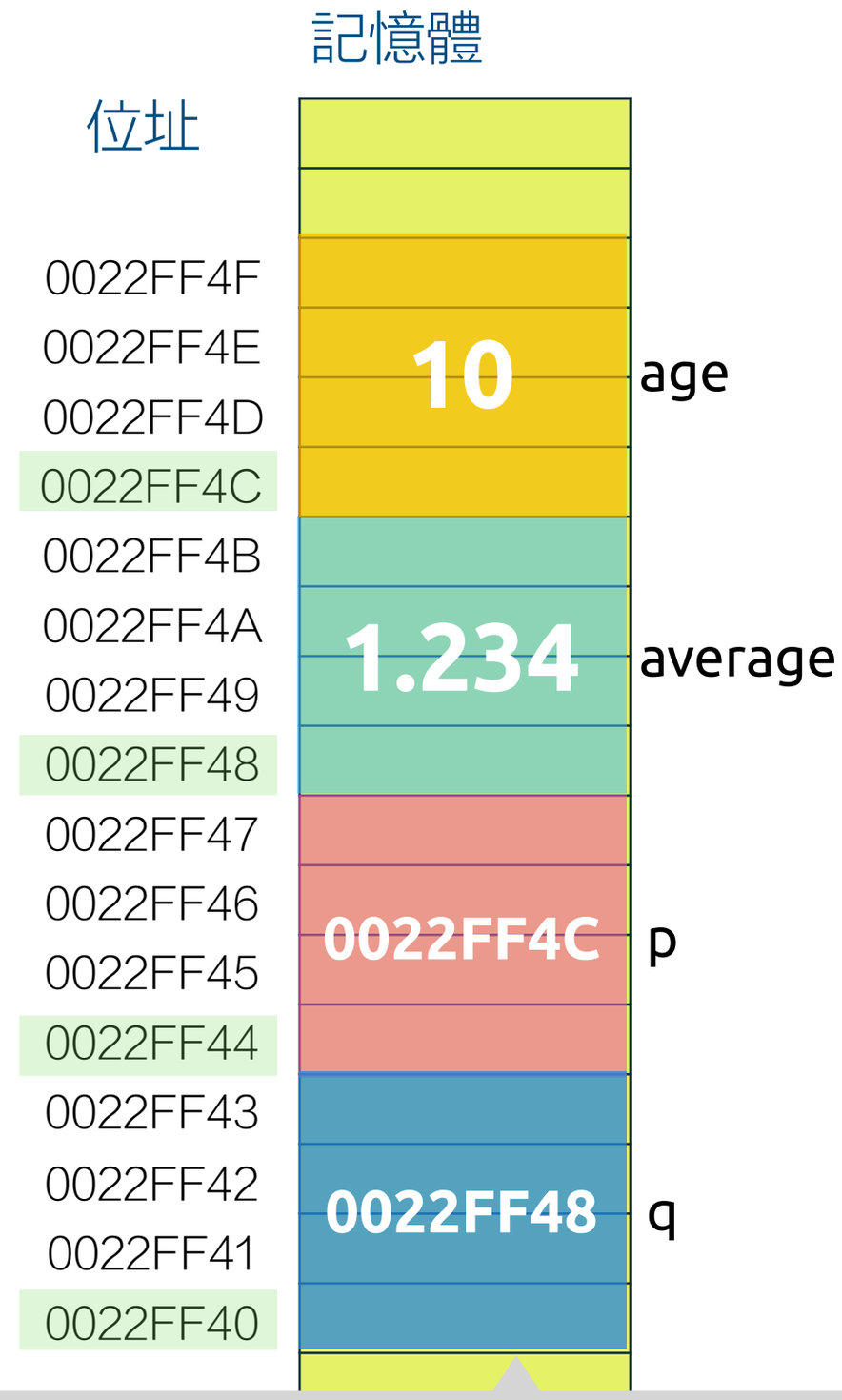
取址符號 &

```
 5      int age;
 6      float average;
 7      int *p;
 8      float *q;
 9
10      age=10;
11      average=1.234;
12      p=&age;
13      q=&average;
14
15      cout<<&age<<endl;
16      cout<<&average<<endl;
17      cout<<&p<<endl;
18      cout<<p<<endl;
19      cout<<&q<<endl;
```

輸出結果

```
0022FF4C
0022FF48
0022FF44
0022FF4C
0022FF40
```

每次執行結果，記憶體位址可能都不相同

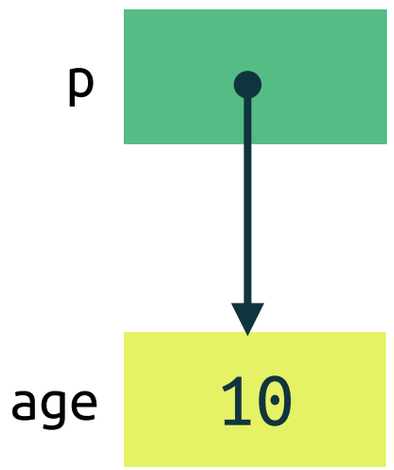


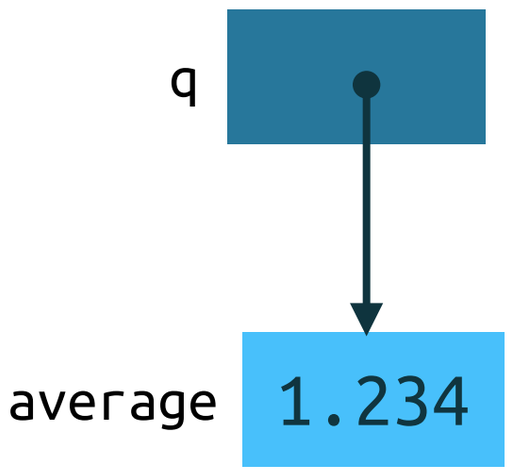
甚至無法保證變數儲存在相鄰的位址

取址符號 &

- 事實上，我們很少直接使用實際的記憶體位址來表示變數與指標間的關係
- 可以使用箭頭來表示它們之間的關係

```
 5      int age;
 6      float average;
 7      int *p;
 8      float *q;
 9
10      age=10;
11      average=1.234;
12      p=&age;
13      q=&average;
```

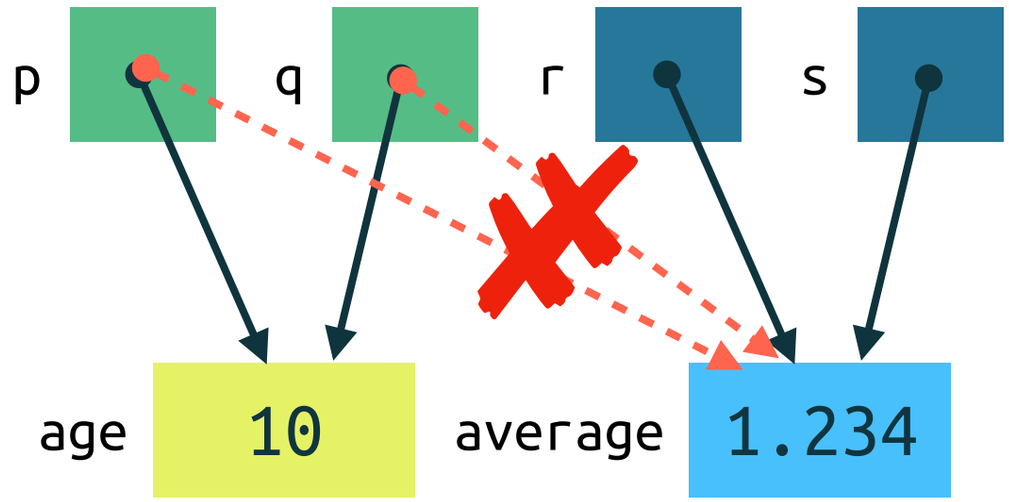




指派符號 =

- 型態相符才可以指派

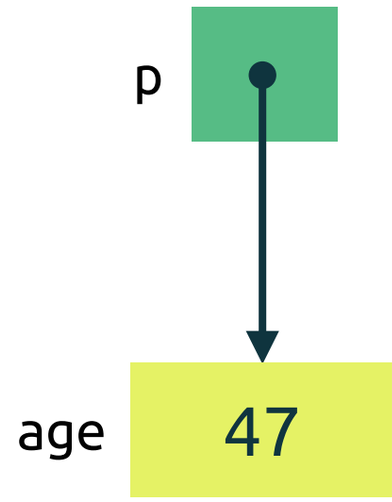
```
5       int age=10;
6       float average=1.234;
7       int *p,*q;
8       float *r,*s;
9
10      p=&age;
11      q=p;
12      r=&average;
13      s=r;
14
15      p=&average; //error
16      q=r;    //error
```



提領運算子 *

- 透過提領運算子 *存取指標指向的變數

```
5       int age=30;
6       int *p;
7
8       p=&age;
9       cout<<"age="<<*p<<endl;
10
11      *p=45;
12      cout<<"age="<<*p<<endl;
13
14      age=*p+2;
15      cout<<"age="<<*p<<endl;
```

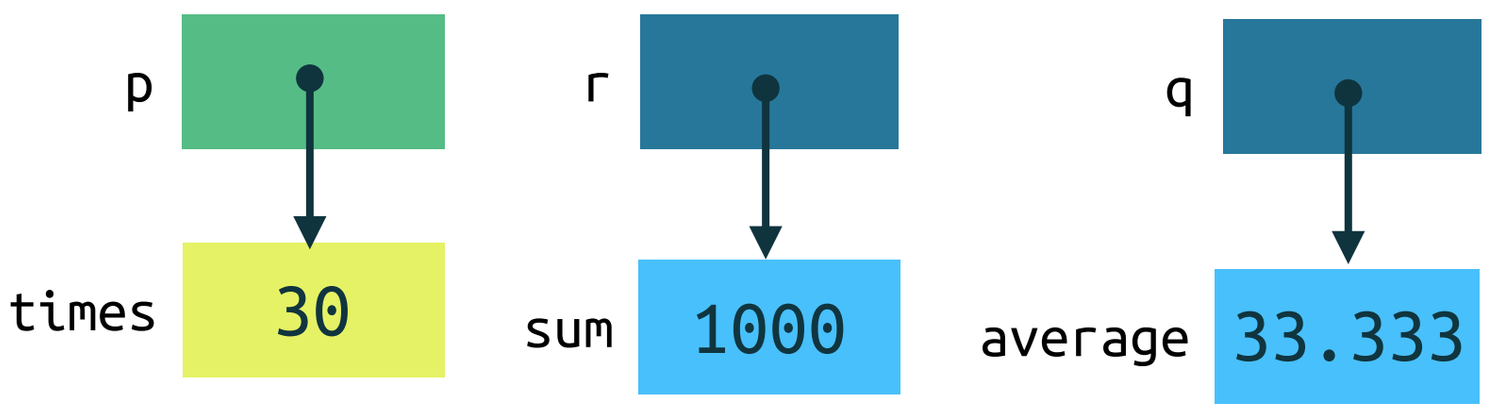


輸出結果
age=30
age=45
age=47

提領運算子 *

```
5       int times=30;
6       float sum=1000, average=0;
7       int *p;
8       float *q,*r;
9
10      p=&times;
11      q=&average;
12      r=&sum;
13
14      *q=(*r)/(*p);
15      cout<<average<<endl;
```

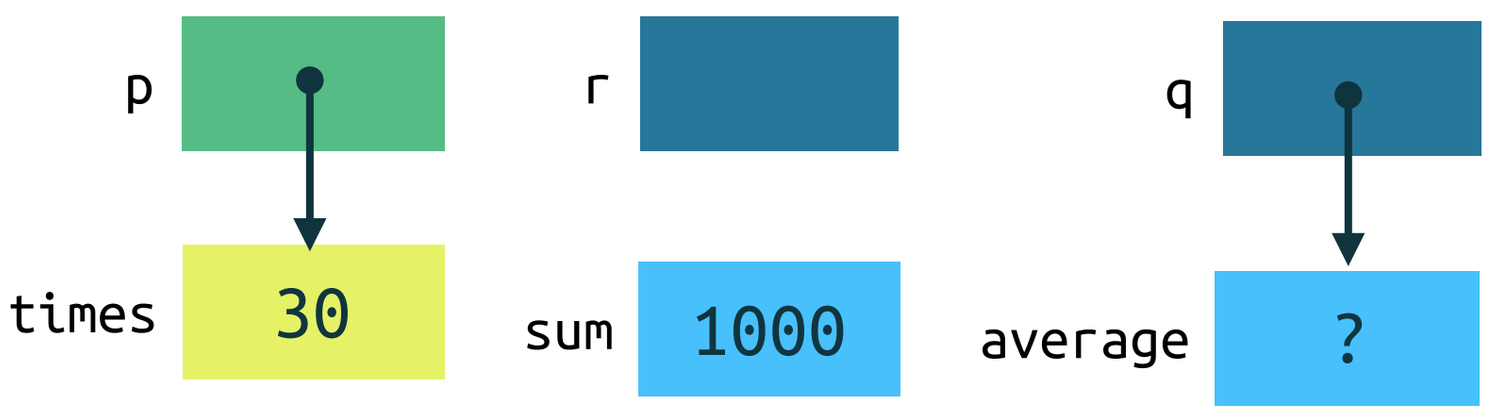
輸出結果
33.33333



提領運算子 *

- 常見的錯誤！

```
5       int times=30;
6       float sum=1000, average=0;
7       int *p;
8       float *q,*r;
9
10      p=&times;
11      q=&average;
12
13      *q=(*r)/(*p);
14      cout<<average<<endl;
```



Exercise

```
5       int a=10,b=20;
6       int *p=&a,*q=&b,*r;
7       *p=*q+b;
8       cout<<a<<" "<<b<<endl;
9
10      r=p;
11      a=30;
12      cout<<*q<<" "<<*r<<endl;
13
14      *r+=3;
15      *q=*r;
16      cout<<a<<" "<<b<<endl;
17
18      *p=*q+*r;
19      cout<<a<<" "<<b<<endl;
20      cout<<*q<<" "<<*r<<endl;
21
22      p=q;
23      q=r;
24      r=p;
25      cout<<*q<<" "<<*r<<endl;
```

輸出結果

```
40 20
20 30
33 33
66 33
33 66
66 33
```

*有三種...

- 乘法運算子
- 定義指標
- 提領運算子

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      int x,y;
6      int *p;          定義指標
7
8      p=&x;
9
10     y=x*20;          乘法運算子
11     *p=y+100;        提領運算子
12     return 0;
13 }
```

哪幾行有問題？

```
4      int x;
5      int *p = &x;
6      int *q = 100;
7
8      p = &x;
9      p = 100;
10     *p = 100;
11     *p = &x;
```

答案
Line 6, 9, 11

初始化

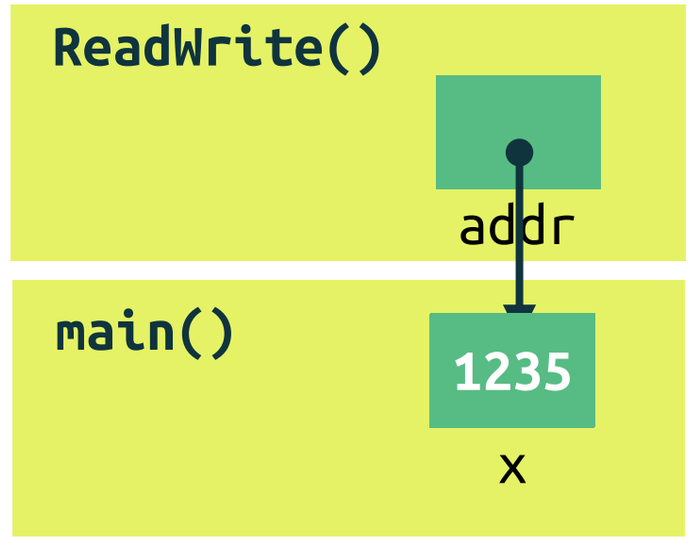
- 設定空指標可以用：
  - 0 是唯一可以可以放進指標的數字
  - NULL 是C語言常用的
  - nullptr 是C++ 11用來表示空指標的常數

初始化

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      int x;
6      int *p=nullptr; //set p as a null pointer
7      //do something...
8      if(p!=nullptr){ //safety check
9          *p*=2;
10     }
11     return 0;
12 }
```

傳位址到函式裡

```
 4 void ReadOnly(int val){
 5     cout<<"val="<<val<<endl;
 6     val=1235;
 7 }
 8 void ReadWrite(int *addr){
 9     cout<<"*addr="<<*addr<<endl;
10     *addr=1235;
11 }
12 int main(){
13     int x=1234;
14     ReadOnly(x);
15     cout<<"After ReadOnly(), x="<<x<<endl;
16     ReadWrite(&x);
17     cout<<"After ReadWrite(), x="<<x<<endl;
18     return 0;
19 }
```



輸出結果

```
val=1234
After ReadOnly(), x=1234
*addr=1234
After ReadWrite(), x=1235
```

傳位址到函式裡

- 到底應該傳值還是傳址？
- 傳值 (Pass by value)：呼叫的函式只要讀取變數值
- 傳址 (Pass by address)：呼叫的函式需要修改變數值

Exercise

```
4  void reset(int *p){
5      *p=0;
6  }
7  void calc(int x,int y,int *z){
8      x++;
9      y++;
10     *z=x+y;
11 }
12 bool divide(int x,int y,int *q,int *r){
13     if(y==0)
14         return false;
15     *q=x/y;
16     *r=x%y;
17     return true;
18 }
19 int main(){
20     int a=10,b=20;
21     calc(11,12,&a);
22     cout<<a<<" "<<b<<endl;
23     if(divide(a,b,&a,&b))
24         cout<<a<<" "<<b<<endl;
25     reset(&b);
26     if(divide(a,b,&a,&b))
27         cout<<a<<" "<<b<<endl;
28     return 0;
29 }
```

輸出結果

```
25 20
1 5
```

傳位址到函式裡

- 到底應該回傳數值還是回傳地址？
- 如果是一個值，可以選擇回傳值，也可以回傳地址

```
 4  int sum1(int a,int b){
 5      return a+b;
 6  }
 7  void sum2(int a,int b,int *ans){
 8      *ans=a+b;
 9  }
10  int main(){
11      int x=0;
12      x=sum1(10,20);
13      sum2(33,44,&x);
14      return 0;
15  }
```

傳位址到函式裡

- 基本上，回傳值比較方便

```c
int year = 0;
// input ...

if (isLeapYear(year)){
    // do something special
}
```

```c
int year = 0, check = 0;
// input ...

isLeapYear(year, &check);
if(check){
    // do something special
}
```

- 然而有超過一個時，就得要傳地址！