LESSON 0
基礎語法與編程技巧

楊孟憲 Vincent

- 逢甲大學資訊工程學系大一
- 111年特選生
- C\C++\Python\Web
- 2022 年ICPC晉級亞洲區決賽
- CEP 專業級 pr96
- 入選 YTP 選訓營
- ITSA 績優 rk. 18/568
- 新化高中競程 語法班\算法班 教學

本單元會學習到的內容

- 解讀錯誤訊息
- 資料型態
- 判斷式
- 迴圈
- 常見的數學函式
- 字元字串
- 函式
- 考試中的coding技巧

錯誤訊息

AC	Accept
WA	Wrong Answer
RE	Runtime error
TLE	Time Limit Exceed
MLE	Memory Limit Exceed
CE	Compile Error
OLE	Output Limit Exceed

基本程式框架

```cpp
#include<iostream>
// 引入標頭檔
using namespace std;
// 標準命名空間 std


// 程式邏輯從 main 開始
int main(){
    // 主程式
    return 0; // 結束程式
}
```

資料型態

資料型態	程式表示	數值
字元	char	‘A’
字串	string	“Hello”
布林	bool	true, false / 1, 0
整數	int	-2^(31) ~ 2^(31) - 1
長長整數	long long int (請常使用)	-2^(63) ~ 2^(63) - 1
浮點數	float	小數點後 4 位
浮點數	double (請常使用)	較精準

存取資料的型態

算數運算子

算數運算子	功能	範例
+	加	a + b
-	減	a - b
*	乘	a * b
/	除	a / b
%	取餘數	a % b

注意：double 和 float 等浮點數不能取餘數

算數運算子

b006: 電電的梯形

輸入三個正整數T、B、H代表此梯形的上底、下底、高(H必為偶數)

輸出梯形面積

面積 =(上底+下底)*高 / 2

算數運算子

b006: 電電的梯形

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

int main(){
    int t, b, h;
    cin >> t >> b >> h;
    cout << (t + b) * h / 2 << "\n";
    return 0;
}
```

小數點處理

a016 糟糕，我發燒了！

輸入華氏溫度，輸出攝氏並精確到小數點後第三位

攝氏 = 5 / 9 * (華氏 - 32)

取到小數點後 n 位： cout << fixed << " " << setprecision(n) << number;
需要 include<iomanip>

小數點處理

a016 糟糕，我發燒了！

輸入華氏溫度，輸出攝氏並精確到小數點後第三位

將數字強制轉型

- float(5)
- 5.0

判斷式

判斷式

判斷輸入數值：

輸入一個整數，如果小於零，輸出Negative，
如果等於零輸出 Zero，否則輸出 Positive

關係運算子

關係運算子	功能	範例
<	小於	a < b
<=	小於等於	a <= b
>	大於	a > b
>=	大於等於	a >= b
==	相等	a == b
!=	不相等	a != b

判斷式

a053. Sagit's 計分程式

輸入一個整數 N (0<=N<=100)，代表學生在 ZeroJudge 系統上解出的題數。

1. 答對題數在 0~10 者，每題給6分。
2. 題數在 11~20 者，從第11題開始，每題給2分。(前10題還是每題給6分)
3. 題數在 21~40 者，從第21題開始，每題給1分。
4. 題數在 40 以上者，一律100分。

印出該同學得分。

邏輯運算子

邏輯運算子	功能	範例
!	非	!(條件a)
&&	且	(條件a) && (條件b)
||	或	(條件a) || (條件b)

三元運算子

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

int main(){

    cout << (條件 ? A : B) << "\n";
    // 當條件符合，輸出A，否則輸出B

    return 0;
}
```

- 當條件符合輸出A，否則輸出 B
- A 和 B 的型態必須一樣

判斷式

你成年了嗎？

輸入一個正整數代表你現在的年齡，如果已經成年，
輸出”Yes” 否則輸出 “No”

迴圈

迴圈

回音

輸入一個正整數n，輸出n次 “Hello world”。

迴圈

FOR LOOP

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

int main(){
    for(控制變數的設定; 迴圈結束條件測試; 控制變數的調整){

    }
}
```

三個迴圈都可以通用，
可以依照題目需求選擇較好實作的迴圈

WHILE LOOP

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

int main(){
    while(條件){

    }
}
```

DO WHILE LOOP

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

int main(){
    do{

    }while(條件);
}
```

a038. 數字翻轉

輸入一個正整數n，將數字反轉後輸出(不包含前綴零)

迴圈

重複輸入

重複輸入兩個整數a, b。
當 a, b >= 0 輸出兩數。
當 a = b = 0 時，結束輸入。
當 a, b < 0 時，什麼都不做。

- break;
- continue;

迴圈

c039. 00100 - The 3n + 1 problem

1. 輸入 n
2. 印出 n
3. 如果 n = 1 結束
4. 如果 n 是奇數 那麼 n=3*n+1
5. 否則 n=n/2
6. GOTO 2

迴圈

d189. 11150 - Cola

輸數一個正整數 n ，代表你原來有幾瓶可樂，已知三瓶喝完的可樂可以換一瓶新的可樂，並且你可以先借一瓶可樂，但是最後要還。

輸出你最多能夠喝幾瓶可樂。

迴圈

d189. 11150 - Cola

- 回圈條件

  while(now >= 3)

- 判斷要不要借可樂

  如果最後剩下兩瓶，借一瓶後手上會有兩瓶，這樣就可以再換一瓶，
  並且把換的那瓶還回去。

  if(now == 2) now++;

進位轉換

- 十進位(decimal) 數值： 0 ~ 9
- 二進位(binary) 數值： 1、0
- 十六進位(hexadecimal) 數值： 0 ~ 9 , A ~ F

十進位制	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
二進位制	0	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111
八進位制	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17
十六進位制	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

迴圈

進位轉換

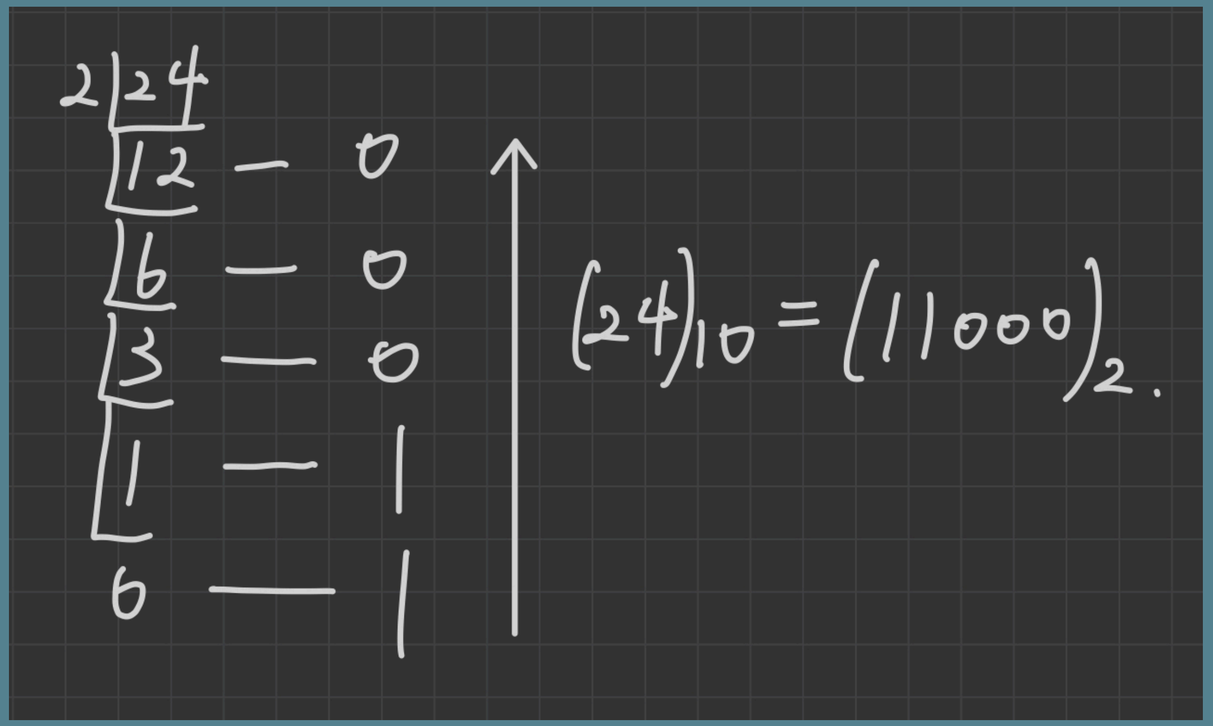
1. 15(decimal) → ?(binary)
2. 15(hexadecimal) → ?(binary)

exercise

1. 24(decimal) → ?(binary)
2. 24(hexadecimal) → ?(binary)

進位轉換

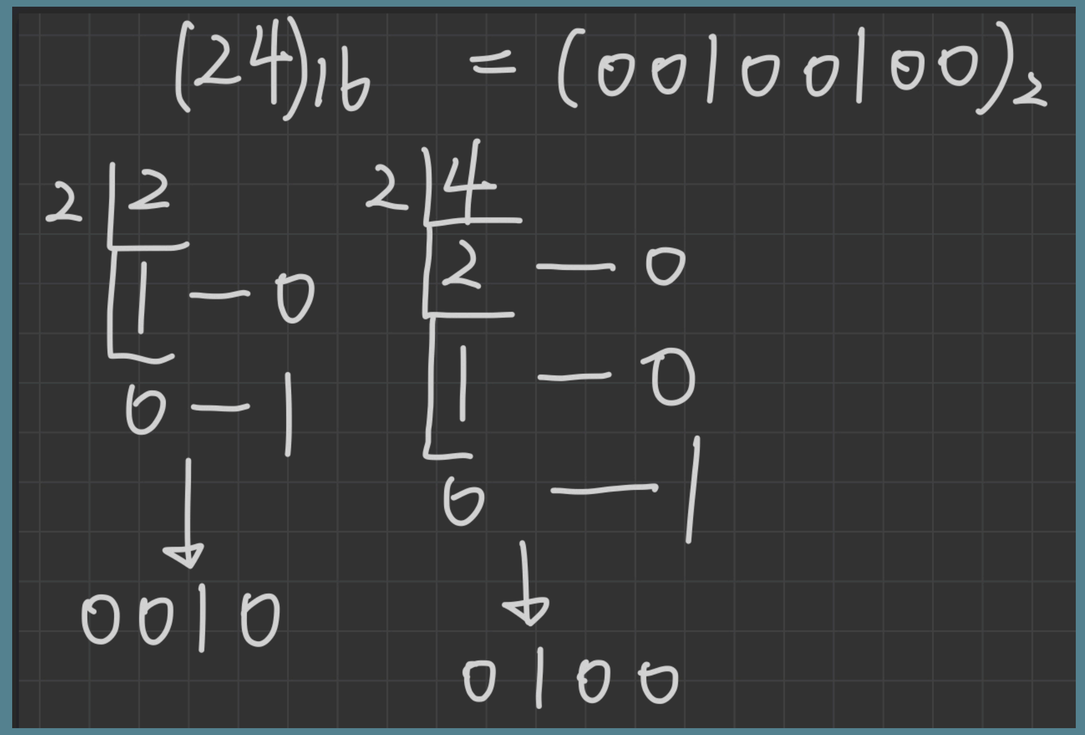
1. 24(decimal) → ?(binary)



迴圈

進位轉換

2. 24(hexadecimal) → ?(binary)



10019 - Funny Encryption Method

輸入一個正整數 n。

- b1 為當 n 為十進位轉為二進位後1出現的次數。
- b2 為當 n 為十六進位轉為二進位後1出現的次數。

輸出 b1 以及 b2

CMATH

CMATH

double pow(int, int)

輸入兩個整數 n, m(1<=n,m<=10)，輸出n的m次方

CMATH

double sqrt(double)

輸入一個整數 n 輸出 n 的平方根。

CMATH

int abs(int)

輸入一個整數 n 輸出 n 的絕對值。

d186. 11461 - Square Numbers

完全平方數就是平方根為整數的整數。例如 1, 4, 81 就是完全平方數。
給你兩個整數 a 和 b，請你求出 a 與 b 之間 (含) 有幾個完全平方數。
當 a = b = 0 時結束輸入。

字元字串

跳脫字元

常見的跳脫字元	代表意義
\n	換行
\t	tab(4個空白鍵)
\\	反斜線(\)
\"	雙引號(")
\'	單引號(')
\0	Null空字元(字串結尾)

- 試著輸出以下字串： The password is "123456".

  "" 在C++中用來當做字串開頭或結尾
  在你想跳脫的字元前加上反斜線 \ (在delete下面)

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

int main(){
    cout << "The password is \"123456\".";
}
```

字元字串

內容

串串是個有強迫症的人，他聊天時喜歡在每段話的開頭加入一個「\」符號，在每段話的結尾加入「"」符號，並且一定要換行。電電非常受不了，決定來解決這個問題。

輸入說明

（無）

輸出說明

請輸出以下兩行文字：

\I am Chuang Chuang"

\He is Diang Diang"

測資資訊：

記憶體限制： 64 MB
公開 測資點#0 (100%): 1.0s , <1K

字串常使用的函式

函式	用法	輸出
size()	str.size();	回傳字串長度
substr(起始位置, 數量)	str.substr();	回傳子字串
reverse(起始迭代器, 終止迭代器)	reverse(str.begin(), str.end());	回傳將字串反轉
stoi(字串)	stoi(“123”)	回傳 字串轉為整數
to_string(數字)	to_string(123)	回傳 數字轉為字串

字元字串

字首字尾

輸入一個字串，輸出第一個和最後一個字元。

字元字串

字首字尾

- 字串可以看成是字元陣列

假設我有一個字串 str = “hello”

陣列表示	str[0]	str[1]	str[2]	str[3]	str[4]
值	‘h’	‘e’	‘l’	‘l’	‘o’

- 陣列從零開始

字元字串

a022. 迴文

輸入一個字串判斷是否為回文

字元字串

a022. 迴文

輸入一個字串判斷是否為回文

- 字串就是字元陣列
- For 迴圈解法
- While 迴圈解法

迴圈做法

一個索引值從前面往後搜尋，一個從後面往前，兩兩比對。

注意：只要搜尋一半就可以了

陣列索引	str[0]	str[1]	str[2]	str[3]	str[4]
數值	a	b	c	b	a

迴圈

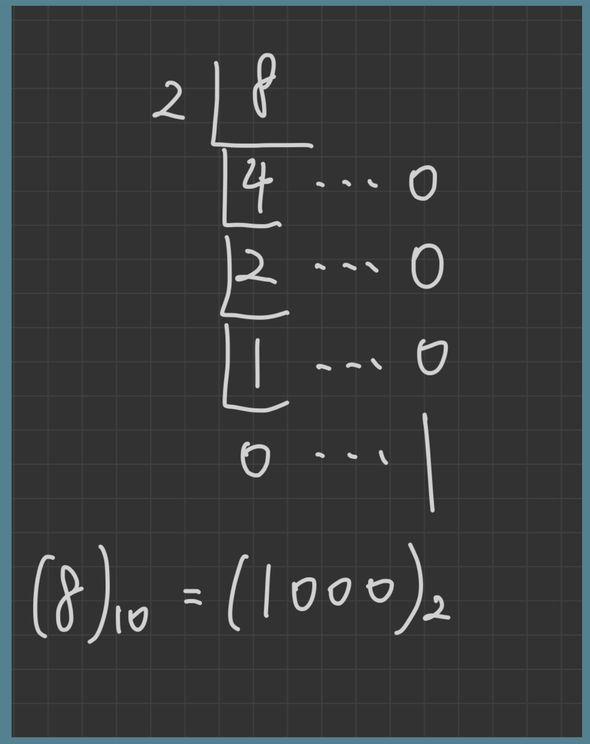
a132. 10931 - Parity

重複輸入一個整數 I 如果 I = 0 結束輸入，
求該數字的 2 進位以及2進位後1的個數 。

- 使用字串實作

a132. 10931 - Parity

十進位轉二進位



ASCII Code

- ASCCII Code

在電腦裡，每個字元都有對應的一個十進位數字。
例如 'A' 的 ascii code 為 65。

- 如何取得字元的 ascii code
  cout << (int)'a' << "\n";

- 如何從 ascii code 取得字元
  cout << (int)'a' << "\n";

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	*[NULL]*	32	20	*[SPACE]*	64	40	@	96	60	`
1	1	*[START OF HEADING]*	33	21	!	65	41	A	97	61	a
2	2	*[START OF TEXT]*	34	22	"	66	42	B	98	62	b
3	3	*[END OF TEXT]*	35	23	#	67	43	C	99	63	c
4	4	*[END OF TRANSMISSION]*	36	24	$	68	44	D	100	64	d
5	5	*[ENQUIRY]*	37	25	%	69	45	E	101	65	e
6	6	*[ACKNOWLEDGE]*	38	26	&	70	46	F	102	66	f
7	7	*[BELL]*	39	27	'	71	47	G	103	67	g
8	8	*[BACKSPACE]*	40	28	(	72	48	H	104	68	h
9	9	*[HORIZONTAL TAB]*	41	29	)	73	49	I	105	69	i
10	A	*[LINE FEED]*	42	2A	*	74	4A	J	106	6A	j
11	B	*[VERTICAL TAB]*	43	2B	+	75	4B	K	107	6B	k
12	C	*[FORM FEED]*	44	2C	,	76	4C	L	108	6C	l
13	D	*[CARRIAGE RETURN]*	45	2D	-	77	4D	M	109	6D	m
14	E	*[SHIFT OUT]*	46	2E	.	78	4E	N	110	6E	n
15	F	*[SHIFT IN]*	47	2F	/	79	4F	O	111	6F	o
16	10	*[DATA LINK ESCAPE]*	48	30	0	80	50	P	112	70	p
17	11	*[DEVICE CONTROL 1]*	49	31	1	81	51	Q	113	71	q
18	12	*[DEVICE CONTROL 2]*	50	32	2	82	52	R	114	72	r
19	13	*[DEVICE CONTROL 3]*	51	33	3	83	53	S	115	73	s
20	14	*[DEVICE CONTROL 4]*	52	34	4	84	54	T	116	74	t
21	15	*[NEGATIVE ACKNOWLEDGE]*	53	35	5	85	55	U	117	75	u
22	16	*[SYNCHRONOUS IDLE]*	54	36	6	86	56	V	118	76	v
23	17	*[END OF TRANS. BLOCK]*	55	37	7	87	57	W	119	77	w
24	18	*[CANCEL]*	56	38	8	88	58	X	120	78	x
25	19	*[END OF MEDIUM]*	57	39	9	89	59	Y	121	79	y
26	1A	*[SUBSTITUTE]*	58	3A	:	90	5A	Z	122	7A	z
27	1B	*[ESCAPE]*	59	3B	;	91	5B	[	123	7B	{
28	1C	*[FILE SEPARATOR]*	60	3C	<	92	5C	\	124	7C	|
29	1D	*[GROUP SEPARATOR]*	61	3D	=	93	5D	]	125	7D	}
30	1E	*[RECORD SEPARATOR]*	62	3E	>	94	5E	^	126	7E	~
31	1F	*[UNIT SEPARATOR]*	63	3F	?	95	5F	_	127	7F	*[DEL]*

cin 與 getline(cin, str)

- cin：讀到空格或結尾就會停止了

```
input:
Hello World

output:
Hello
```

- getline：一次讀取一整行，包含空格

注意：使用getline(cin, str) 時，會讀到前一個輸入的換行，使用 cin.ignore() 忽略換行。

```
input:
Hello World

output:
Hello World
```

字元字串

你好，新同學！

今天逢甲大學資工系轉來了一位新同學， 你不知道他的名字。
第一行會輸入一個字串 name，為新同學的名字，請你向他

Hello, name!

凱賽加密(補)

給定一個字串 str = “abc”，利用凱薩加密法，將每一個字元之 ascii 碼加上3。

會得到加密過後的字符串 str = “def”。

加密做法

```
string ss = "abc";
for(int i=0;i<ss.size(); i++){
    cout << (char)((int)ss[i] + 3);
}
```

a009. 解碼器

輸入一個字串，請依照範例測資提示解密。
輸出解密後的結果。

函式

區域變數、全域變數

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

int a = 100; // 全域變數

int main(){
    int a = 10; // 區域變數
    cout << a << "\n";
    return 0;
}
```

優先權： 區域 > 全域

區域變數、全域變數

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

int a = 100; // 全域變數

void func(){
    cout << a << "\n";
    return;
}

int main(){
    int a = 10; // 區域變數
    cout << a << "\n";
    a = 0;
    func();
    return 0;
}
```

main 裡的 a = ?

func 裡的 a = ?

函式

這是一個簡單的函示範例，輸入一個整數 a ，回傳 a 的立方。

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

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

int main(){
    int a;
    cin >> a;
    cout << func(a) << "\n";
    return 0;
}
```

- 被呼叫的函式要寫在上面

以這個例子來說， main 呼叫 func，所以func要寫在main上面。

函式

輸入三角形兩短邊，找出第三邊。

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

int main(){
    double a, b;
    cin >> a >> b;
    cout << side(a, b) << '\n';
    return 0;
}

double side(double a, double b){
    return sqrt(pow(a, 2) +  pow(b, 2));
}
```

- 這是錯誤的
- 如何修改？

函式

11332 - Summing Digits

輸入一個正整數 n。定義一個 function f(n) 為 n 的每一位數字總和。重複執行 f(n) 直到 n 為一位數。

競程技巧

講師：楊孟憲

萬用標頭檔

```cpp
#include<iostream>
#include<cmath>
#include<algorithm>
#include<vector>
.
.
.
.
.
.
.
.
using namespace std;


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

```cpp
#include<bits/stdc++.h>
using namespace std;

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

不想記一堆標頭檔

請使用別人寫好的萬用標頭檔

#DEFINE

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

long long addition(long long n, long long m){
    return n + m;
}

int main(){
    long long n, m;
    cin >> n >> m;
    cout << addition(n, m) << "\n";
    return 0;
}
```

(Define 前)

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

#define int long long

int addition(int n, int m){
    return n + m;
}

signed main(){
    int n, m;
    cin >> n >> m;
    cout << addition(n, m) << "\n";
    return 0;
}
```

(Define 後)

- 使用#define int long long讓程式碼看起來比較簡潔
- 要小心的是，因為 long long main 沒有被定義，會執行錯誤。將 int main 改成 signed main 即可。

IO 優化

```cpp
int main(){
    ios_base::sync_with_stdio(false); cin.tie(0); cout.tie(0);
    // 當資料量非常大時沒有加上 io 優化很容易超時。
    return 0;
}
```

當資輸入資料量很大的時候，使用 IO 預防 TLE

完全體

```cpp
#include<bits/stdc++.h>
using namespace std;

#define int long long

void solve(){

}

signed main(){
    ios_base::sync_with_stdio(false); cin.tie(0); cout.tie(0);
    return 0;
}
```

一維陣列

陣列

為何要使用陣列?

變數可以幫我們儲存一筆資料，非常方便；但是要儲存多筆資料的時候就顯得麻煩。

試想要記錄五個人的數學成績，只用變數宣告會寫成。

```
 6 void sovle(){
 7     int g1, g2, g3, g4, g5;
 8     cin >> g1 >> g2 >> g3 >> g4 >> g5;
 9     return;
10 }
```

那如果一個班有 100 個人呢？

陣列

語法 - 陣列宣告

```
資料型態 名稱[長度];

int arr[10]; // 宣告 10 個元素的整數陣列

double score[10]; // 宣告 10 個元素的浮點數陣列

char ascii[10]; // 宣告 10 個元素的字元陣列
```

陣列

語法 - 陣列初始化

```
int number[10] = {0};

double score[10] = {0.0};

char ascii[10] = {'\0'};

bool flag[10] = {false};
```

```
int number[5] = {0, 1, 2, 3, 4};

double score[5] = {87.0, 78.0, 99.5, 69.5, 82.5};

char ascii[5] = {'A', 'B', 'C', 'D', 'E'};

bool flag[5] = {false, true, false, true, false};
```

陣列

語法 - 陣列初始化 (請使用)

```
int arr[5];
memset(arr, 1, sizeof(arr)); // 0 -1 1

bool arr_2[5];
memset(arr_2, true, sizeof(arr_2)); // true/false, 1/0

char arr_3[5];
memset(arr_3, '\0', sizeof(arr_3)); // 空字元
```

陣列

語法 - 陣列使用（指派、輸入、輸出）

```
int arr[5] = {10, 20, 30, 40, 50};
```

長度為5 的陣列，索引值從 0 ~ 4

陣列索引	arr[0]	arr[1]	arr[2]	arr[3]	arr[4]
數值	10	20	30	40	50

陣列

語法 - 陣列使用（指派、輸入、輸出）

使用 for 迴圈輸入輸出數值

```
int arr[5];
for(int i=0;i<5;i++){
    cin >> arr[i];
}

for(int i=0;i<5;i++){
    cout << arr[i] << " ";
}
```

陣列

陣列大小應該開多少

輸入說明

第一行輸入兩個正整數 n, D，接下來有 n 個正整數，代表每個時間點股票的價格。

數字範圍

- $1 \leq n \leq 100, 1 \leq D \leq 100$
- $1 \leq a[i] \leq 100$

子題配分

- (50%): $n = 3$
- (50%): 無額外限制

題目會給定每個提到的數值範圍，
在這個題目輸入說明中，n 很明顯是陣列大小。

所以陣列大小至少要開 110。
為了預防溢位(overflow)，請多開幾個。

陣列

1. 程式交易

給定股票價格 a[1], a[2], a[3], a[4], a[5] ... a[n]，依照以下買賣規則輸出利潤總和。

1.同一時間手上只會持有一張股票，並會在時間點 1 花 a[1] 買進。

2.若當下持有股票且該股票買進價格為 x ，當遇到價格 y 大於等於 x + D 即時賣出，並賺進利潤 y - x。

3.若當下手上沒有股票且上一次賣出的價格為 x ，當遇到價格 y 小於等於 x - D 時則會買進股票。

陣列技巧：建表

質數表

多筆輸入，每行輸入兩個整數 a, b(2 <= a, b <= 300)，輸出閉區間[a, b]之間出現的所有質數。

陣列

直接硬做效率太低，可能會吃 TLE

```cpp
#include<bits/stdc++.h>
using namespace std;

#define int long long

signed main(){
    int a, b;

    while(cin >> a >> b){
        for(int i=a;i<=b;i++){
            bool check = 1;
            for(int k=2;k<i;k++){
                if(!(i%k)){
                    check = false;
                    break;
                }
            }
            if(check){
                cout << i << " ";
            }
        }
        cout << "\n";
    }
    return 0;
}
```

陣列

建表過後，只要查詢就好。

```cpp
#include<bits/stdc++.h>
using namespace std;

#define int long long

bool arr[100];

signed main(){
    int a, b;

    memset(arr, false, sizeof(arr));

    for(int i=1;i<=100;i++){
        bool check = true;
        for(int k=2;k<i;k++){
            if(!(i % k)){
                check = false;
                break;
            }
        }
        if(check) arr[i] = true;
    }


    while(cin >> a >> b){
        for(int i=a;i<=b;i++){
            if(arr[i]) cout << i << " ";
        }
        cout << "\n";
    }
    return 0;
}
```

陣列

11063 - B2-Sequence

給定一個數列 a 。
第一行輸入一個正整數 n ，代表數列長度，第二行輸入 n 個整數，從 a1 ~ an。
所謂「B2數列」係指一正整數數列 1<= b1 < b2 < b3 ...，其中所有的 bi + bj （i <= j）皆不相等。

試求 數列a 是不是 B2 數列。

陣列

Reverse Card Open!

有十張覆蓋的牌，輸入 n 代表掀開幾張覆蓋的牌，接下來 n 行有一個字串 s 及整數 k，
代表掀開地 k 張覆蓋的牌，名字為 s。

輸出十行代表 1 ~ 10 張牌，如果地 i 張牌覆蓋，輸出 “EMPTY”，否則輸出該張牌的牌名。

陣列

d562. 山寨版磁力蜈蚣

第一行輸入一個正整數 n ，第二行輸入 n 個整數， a1 ~ an。

第一行輸出最一開始的狀態。
第二行輸出「去除第一項之後，全部倒轉過後的結果」。
直到數字只到一個為止。

- while 迴圈搭配 for 迴圈實作。

陣列

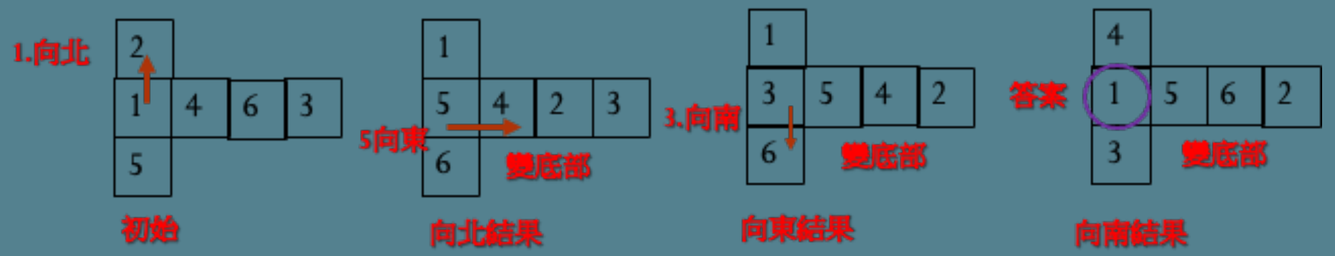
10340 - All in All

輸入字串 S、T，問 S 是否為 T 的子字串。

陣列

10409 - Die Game

將骰子六個面命名為 頂、底、東、西、南、北。
剛開始骰子的狀態為：
頂：1 底：6
西：3 東：4
北：2 南：5



多筆輸入 t ，代表有 t 比操作，接下來 t 行每行輸入一個字串，代表骰子往哪個方向滾，
輸出最後頂部的數字為多少。

當 t == 0 結束輸入。

排序

排序

std::sort

C++ STL之下內建的排序工具，使用前先導入標頭檔 algorithm

用法： sort(array_name + L, array_name + R); // sorting array's [L, R) elements

sort 裡面放的第一個參數與第二個參數都為 迭代器 的型態

可以把迭代器想像成一種指標，之後的單元會介紹到

排序

由小到大排序

sort(陣列名字, 陣列名字 + 陣列長度);

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

排序

由大到小排序

sort(起始指標, 結束指標, 排序規則);

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

排序

由大到小排序

sort(起始指標, 結束指標, 排序規則);

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

沒有指定排序歸則，就是默認 less<int>()

排序

11462 - Age Sort

給你某國家所有一歲以上 (含) 的人民的年齡。你知道該國沒有人活到 100 歲或更老。現在給你一個很簡單的工作，就是把所有的年齡由小到大排序。

排序

第 k 大

請在一串數字中找到第 k 大的數字。
給定一串數字，輸入到 -1 停止，接著輸入一個整數 k。

排序

成績指標

第一行輸入一個正整數 n ，代表有幾個學生，第二行有 n 個整數代表每個學生的成績。

有三行輸出。
第一行為 學生成績由小到大排序。
第二行印出最高不及格分數，如果全數及格，印出 best case。
第三行印出最低及格分數，如果全數不及格，印出 worst case。

排序

字串的比較

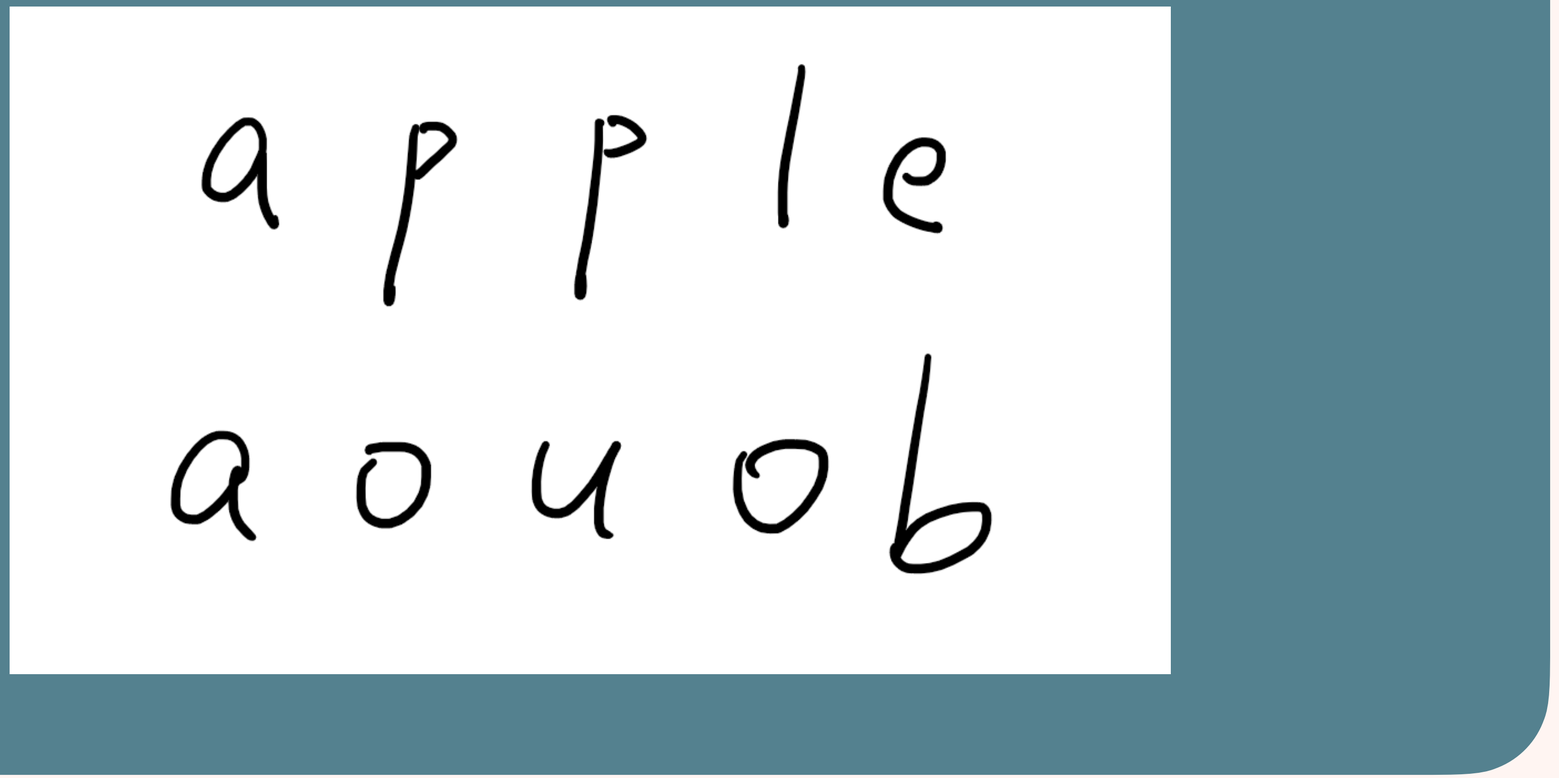
字串的比較上預設是以 字典序 來作為比較基準

字典序會以 ASCII 的先後順序而定

兩個字串在比較時回從左到右依序比較

排序

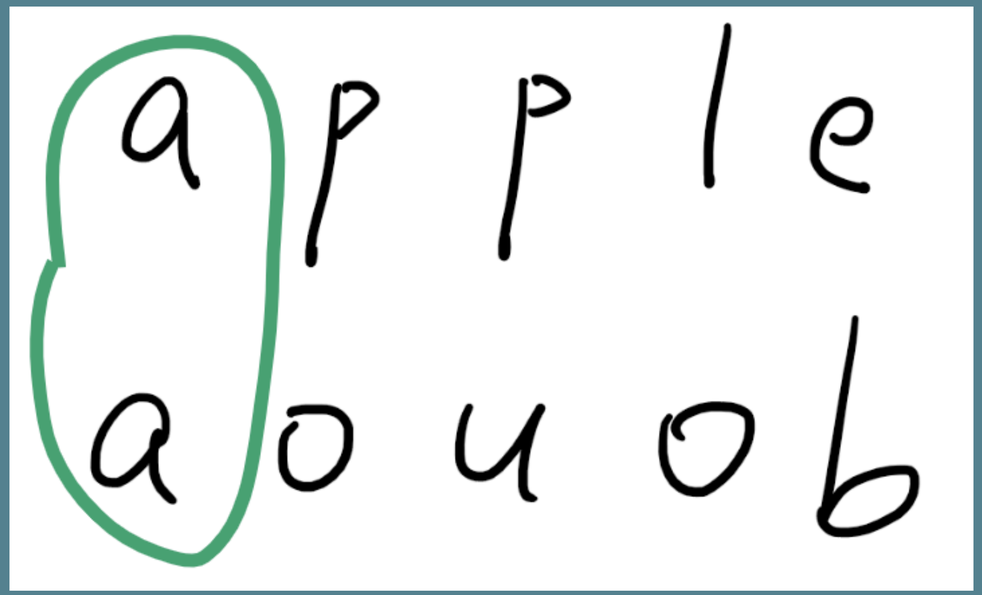
字串比較的例子



排序

由左往右比較

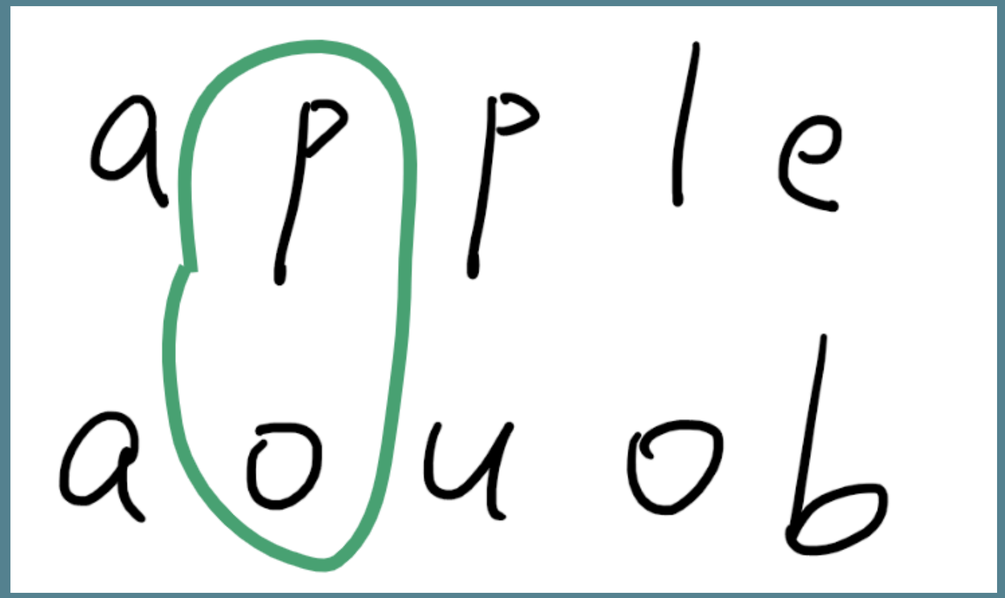
ascii 一樣，繼續往右比較



排序

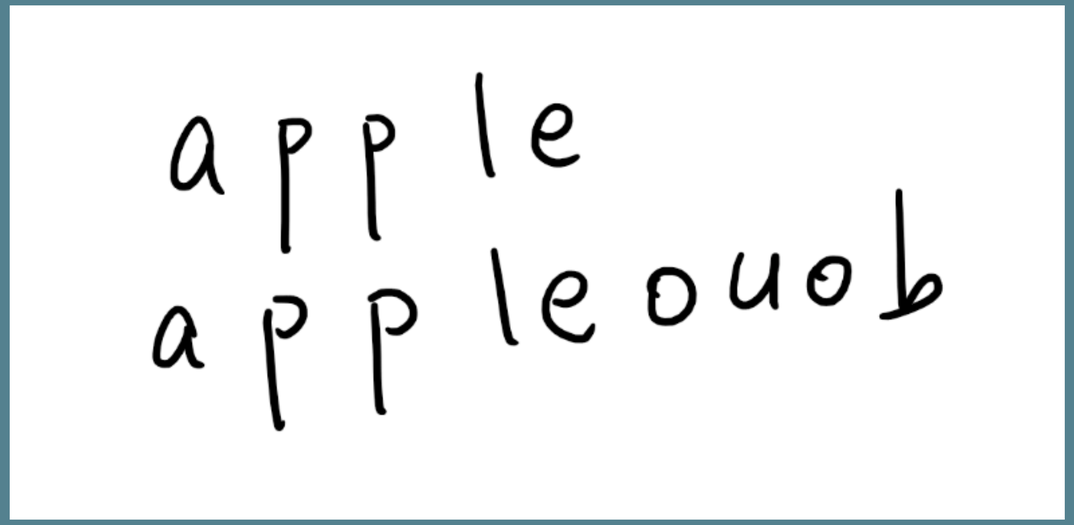
繼續比較，一旦分出大小則停止比較

'p' 比 'o' 的 ascii 還大
因此 apple 比 aouob 還大



排序

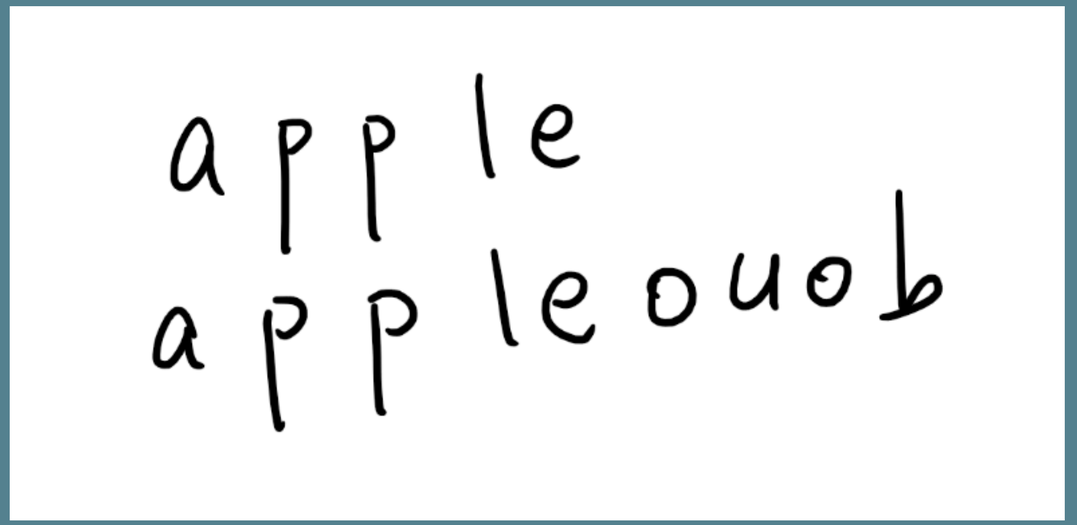
那如果兩個字串一長一短而且前半部分都一樣呢？



排序

那如果兩個字串一長一短而且前半部分都一樣呢？

比較長的字串比較大



二維陣列

陣列

語法 - 陣列宣告

```
資料型態 名稱[第一維度長度][第二維度長度]

int arr[5][10]; // 宣告 5*10 個元素的整數陣列

char ascii[5][10]; // 宣告 5*10 個元素的字元陣列

string str[5][10]; // 宣告 5*10 個元素的字串陣列
```

陣列

語法 - 陣列初始化

一樣使用 memset(name, val, sizeof(name));

```
int arr[20][20];
memset(arr, 0, sizeof(arr));
```

陣列

語法 - 陣列使用（指派、輸入、輸出）

```
int arr[3][3] = {{1, 1, 1}, {2, 2, 2}, {3, 3, 3}};
```

行 \ 列			
	arr[0][0]	arr[0][1]	arr[0][2]
	arr[1][0]	arr[1][1]	arr[1][2]
	arr[2][0]	arr[2][1]	arr[2][2]

行 \ 列			
	1	1	1
	2	2	2
	3	3	3

陣列

語法 - 陣列使用（指派、輸入、輸出）

使用 for 迴圈輸入輸出數值

```
int arr[n][m];
for(int i=0;i<n;i++){
    for(int k=0;k<m;k++){
        cin >> arr[i][k];
    }
}
```

陣列

我閉著眼

一共輸入5*6 = 30個數字
每一橫排6個，共5排
數字只有1~6，同數字就是同種類

若有任何3個同種類連成一線(直的或橫的)，就輸出"Yes" (不含雙引號)
否則輸出"No"

陣列

00272 - TeX Quotes

輸入是若干列的文字，其中有偶數個雙引號（ " ），以 end-of-file 做結束。
輸出的文字必須和輸入的一模一樣，除了：
- 每一組雙引號的第一個 " 必須用兩個 ` 字元（就是 ``）來代替
- 每一組雙引號的第二個 " 必須用兩個 ' 字元（ 就是 ''）來代替。

挑戰題

10908 - Largest Squares

b266. 矩陣翻轉

Mutant Flatworld Expolrers

10908 - LARGEST SQUARES

將程式碼分成幾個函式以便除錯。
get_max_val 函式會回傳最大邊長給 main 函式。

這邊因為要回傳最大值，所以 for 迴圈從 101 開始判斷。

需要判段起點有沒有超過邊長，所以如果起始點超過
在呼叫 check_ret() 判斷是否為矩形。

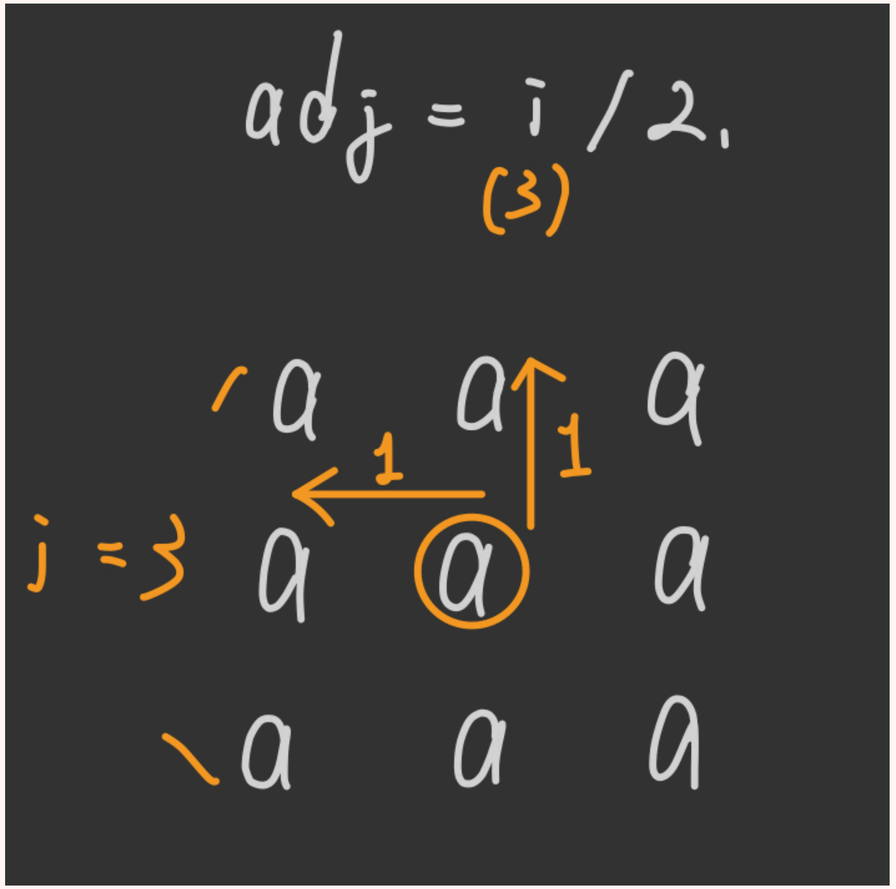
```
int get_max_val(int r, int c){

    for(int i=101; i >= 1; i-=2){
        int adj = i/2;
        if(r - adj < 0 || c - adj < 0){
            continue;
        }

        if(check_ret(r-adj, c-adj, i) == true){
            return i;
        }
    }

    return 1;
}
```

10908 - LARGEST SQUARES

for 迴圈枚舉 邊長的可行性。

保證矩形邊長一定為奇數，所以我們從 101 開始枚舉。

以左邊的 3 * 3 矩形，可以發現平移到矩形最左上角的步數剛好為 i / 2。



10908 - LARGEST SQUARES

check_ret() 函式回傳 true/false 為矩形邊長是否成立。

先判斷邊長是否超出陣列範圍，如果超出範圍 return false

再來判斷矩形內字元是否都一樣，我們可以跟 起始點 (r, c) 字元做比較。 如果不同 return false

不可能的情況判斷完後，代表矩形成立。 return true

```
bool check_ret(int r, int c, int l){
    char start = mp[r][c];

    for(int i=r; i<=r+l-1; i++){
        if(i >= m) return 0;

        for(int k=c;k<=c+l-1; k++){
            if(k >= n) return 0;

            if(mp[i][k] != start){
                return 0;
            }
        }
    }

    return 1;
}
```

矩陣翻轉

輸入處理：

```
cin >> n >> m >> q;

for(int i=0;i<n;i++){
    for(int k=0;k<m;k++){
        cin >> mp[i][k];
    }
}

for(int i=0;i<q; i++){
    cin >> oper[i];
}
```

矩陣翻轉

因為是要反轉回原來的樣子，所以要先把操作存起來再從後面往前一步一步還原。

判斷如果 oper[i] 為 1 時，呼叫 rotate 旋轉陣列。
否則呼叫 rev() 逆時針翻轉陣列

```
for(int i=q-1;i>=0;i--){
    if(oper[i] == 1){
        rotate();
    }

    else{
        rev();
    }
}
```

結構

結構

struct：自己定義一個資料型態

定義一個資料型態，裏頭可以存放學生名字，數學分數，英文分數。

```
 6 struct student{
 7     string name;
 8     int Math_score;
 9     int English_score;
10 };
```

ps.可以想像成包裝的概念

結構

宣告 struct 型別的資料並輸入

定義資料型態為 student。
如果想指定 student 裡面的某個資料，直接 .name 就可以指定了。

```
student St[100];
for(int i=0;i<100;i++){
    cin >> St[i].name >> St[i].Math_score >> St[i].English_score;
}
```

結構

宣告 struct 型別的資料並輸入

資料型態為 student。
如果想指定 student 裡面的某個資料，直接 .name 就可以指定了。

```
student St[100];
for(int i=0;i<100;i++){
    cin >> St[i].name >> St[i].Math_score >> St[i].English_score;
}
```

結構

我閉著眼

一共輸入5*6 = 30個數字
每一橫排6個，共5排
數字只有1~6，同數字就是同種類

若有任何3個同種類連成一線(直的或橫的)，就輸出"Yes" (不含雙引號)
否則輸出"No"

自定義排序

排序

這是之前教的 sort

sort(起始指標, 結束指標, 排序規則);

sort 第三個參數為排序規則。
自定義排序就是自己寫排序規則

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

排序

如果你想要自定義規則的畫就要自己寫一個 compare 函式

compare function 的型態為 布林(bool)，而解包含兩個參數

參數的型態依照排序的序列型態而定。(假設排序的陣列型態是 int，兩個參數就要放 int)。

回傳的值只有 true 和 false

排序

排序原理：

先交換、再詢問。

什麼時候回傳 true 什麼時候回傳 false

compare 回傳的規則定義為：

第一個參數是否有比第二個參數小

因此如果當兩個參數相同大小時，切記！！！一定要回傳 false

不然你將會得到 Runtime Error

排序

一切就緒！！將 sort 函式加入第三個參數即可

第三個參數要放什麼取決於你的 compare 的函式名稱

名稱叫 apple 就放 apple，banana 就放 banana

```
sort(a,a+5,cmp);
```

排序

compare 範例 (由大到小)

```cpp
bool cmp(int a, int b){
    if(a != b){
        return a < b;
    }

    return false;
}

void solve(){

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

    for(int i=0;i<5;i++){
        cout << arr[i] << " ";
    }

    cout << "\n";

    return;
}

signed main(){
    fastIO
    solve();
    return 0;
}
```

排序

字母排序 (Letters)

第一行輸入排序規則，第二行輸入 需排序的字串。
輸出排序後的字串。

挑戰題

00555 - Bridge Hands