Stack

ZeroJudge a.565
Yu-Hsuan Chen

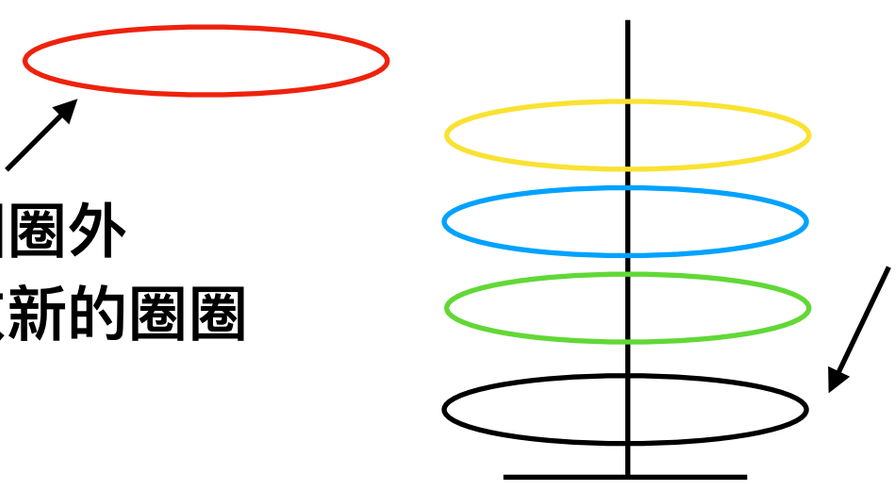
請下載本投影片
並以Adobe Reader開啟
用PgUP/PgDown切換上下頁
這樣才看得到投影片中動畫的呈現

＊貼心提醒＊

在開始講解這一題之前
需要先向你介紹資料結構中的Stack

什麼是Stack

- Stack(堆疊)，是一種資料結構。
- 滿足「後進先出」(LIFO, Last in First out)的特性
  也就是<u>最早進入Stack的，最晚被取出</u>
  <u>最晚進入Stack的，最先被取出</u>
- 現實的比喻：套圈圈



除了拿圈圈外
只能在上面放新的圈圈

想拿到下面的黑色圈圈
一定要先把上面的黃、藍、綠
三個圈圈<u>依序</u>拿走

Stack in program

Stack in program

- 用陣列實作：int array[12];

0	1	2	3	4	5	6	7	8	9	10	11

Stack in program

- 用陣列實作：int array[12];

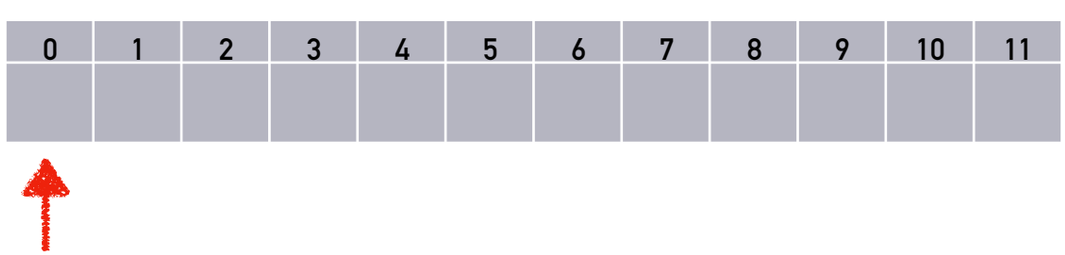
C++ STL有提供stack
但不是我們要討論的重點

0	1	2	3	4	5	6	7	8	9	10	11

Stack in program

C++ STL有提供stack
但不是我們要討論的重點

- 用陣列實作：int array[12];
- 輔助的變數：
  int top = 0;
  初始值 0（指在下一個空格上）
  同時top值也代表Stack目前已有多少資料

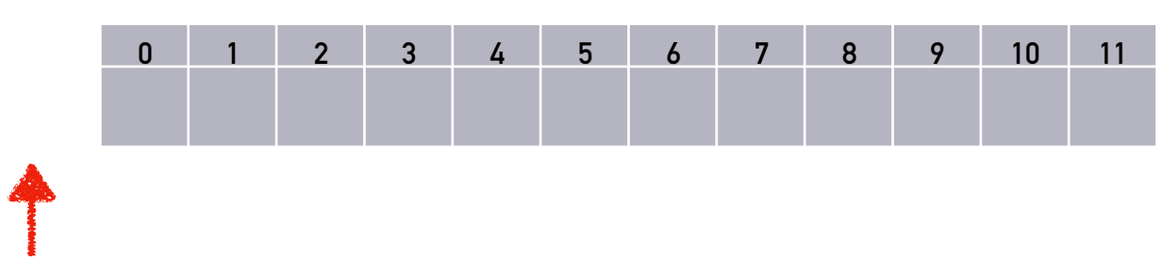


Stack in program

C++ STL有提供stack
但不是我們要討論的重點

- 用陣列實作：int array[12];

- 輔助的變數：
  int top = 0;
  初始值 0（指在下一個空格上）
  同時top值也代表Stack目前已有多少資料



另一種作法則是top = -1
讓top永遠指在最上面的data

Stack的操作

- Push : 把東西放入Stack
- Pop : 把最上面的東西從Stack取出
- isEmpty : 檢查Stack是否為空
- isFull : 檢查Stack是否已滿

Stack Is FULL

檢查堆疊是否滿了

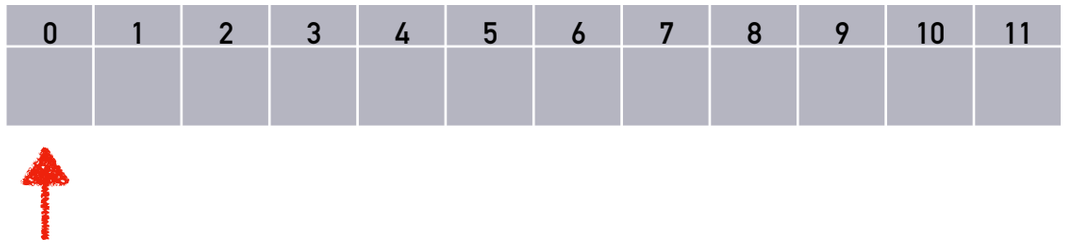
```
bool isFull(int top)

{

    if(top >= MAX_STACK_SIZE + 1)

        return true;

    else return false;

}
```

0	1	2	3	4	5	6	7	8	9	10	11
7	5	8	7	3	4	6	9	0	1	2	4

Stack Is EMPTY

檢查堆疊是否為空

```
bool isEmpty(int top)

{

    if(top <= 0)

        return true;

    else return false;

}
```



Push into Stack

加入新東西到Stack的最上面

```
void Push_Stack(int value, int *top)
{
    if(isFull(*top) == false)
    {
        stack[*top] = value;
        (*top)++;
    }
}
```

TOP = 2

0	1	2	3	4	5	6	7	8	9	10	11
3	4										

Push into Stack

加入新東西到Stack的最上面

```
void Push_Stack(int value, int *top)
{
    if(isFull(*top) == false)
    {
        stack[*top] = value;
        (*top)++;
    }
}
```

TOP = 2

0	1	2	3	4	5	6	7	8	9	10	11
3	4										

Push into Stack

加入新東西到Stack的最上面

```
void Push_Stack(int value, int *top)
{
    if(isFull(*top) == false)
    {
        stack[*top] = value;
        (*top)++;
    }
}
```

Push之前檢查Stack有沒有空間

TOP = 2

0	1	2	3	4	5	6	7	8	9	10	11
3	4										

Push into Stack

加入新東西到Stack的最上面

```
void Push_Stack(int value, int *top)
{
    if(isFull(*top) == false)
    {
        stack[*top] = value;
        (*top)++;
    }
}
```

Push之前檢查Stack有沒有空間

TOP = 2

0	1	2	3	4	5	6	7	8	9	10	11
3	4										

Push into Stack

加入新東西到Stack的最上面

```
void Push_Stack(int value, int *top)
{
    if(isFull(*top) == false)
    {
        stack[*top] = value;
        (*top)++;
    }
}
```

Push之前檢查Stack有沒有空間

TOP = 2

0	1	2	3	4	5	6	7	8	9	10	11
3	4	7									

Push into Stack

加入新東西到Stack的最上面

```
void Push_Stack(int value, int *top)
{
    if(isFull(*top) == false)
    {
        stack[*top] = value;
        (*top)++;
    }
}
```

Push之前檢查Stack有沒有空間

TOP = 2

0	1	2	3	4	5	6	7	8	9	10	11
3	4	7									

Push into Stack

加入新東西到Stack的最上面

```
void Push_Stack(int value, int *top)
{
    if(isFull(*top) == false)
    {
        stack[*top] = value;
        (*top)++;
    }
}
```

Push之前檢查Stack有沒有空間

TOP = 3

0	1	2	3	4	5	6	7	8	9	10	11
3	4	7									

Pop from Stack

把Stack最上面的東西取走

```
int Push_Stack(int *top)
{
    if(isEmpty(*top) == false)
    {
        int temp = stack[*top-1];
        (*top)--;
    }
    return temp;
}
```

TOP = 3

0	1	2	3	4	5	6	7	8	9	10	11
3	4	7									

Pop from Stack

把Stack最上面的東西取走

```
int Push_Stack(int *top)
{
    if(isEmpty(*top) == false)
    {
        int temp = stack[*top-1];
        (*top)--;
    }
    return temp;
}
```

TOP = 3

0	1	2	3	4	5	6	7	8	9	10	11
3	4	7									

Pop from Stack

把Stack最上面的東西取走

```
int Push_Stack(int *top)
{
    if(isEmpty(*top) == false)
    {
        int temp = stack[*top-1];
        (*top)--;
    }
    return temp;
}
```

Pop之前檢查Stack有沒有東西

TOP = 3

0	1	2	3	4	5	6	7	8	9	10	11
3	4	7									

Pop from Stack

把Stack最上面的東西取走

```
int Push_Stack(int *top)
{
    if(isEmpty(*top) == false)
    {
        int temp = stack[*top-1];
        (*top)--;
    }
    return temp;
}
```

Pop之前檢查Stack有沒有東西

TOP = 3

0	1	2	3	4	5	6	7	8	9	10	11
3	4	7									

Pop from Stack

把Stack最上面的東西取走

```
int Push_Stack(int *top)
{
    if(isEmpty(*top) == false)
    {
        int temp = stack[*top-1];
        (*top)--;
    }
    return temp;
}
```

Pop之前檢查Stack有沒有東西

temp = 7

TOP = 3

0	1	2	3	4	5	6	7	8	9	10	11
3	4	7									

Pop from Stack

把Stack最上面的東西取走

```
int Push_Stack(int *top)
{
    if(isEmpty(*top) == false)
    {
        int temp = stack[*top-1];
        (*top)--;
    }
    return temp;
}
```

Pop之前檢查Stack有沒有東西

temp = 7

TOP = 2

0	1	2	3	4	5	6	7	8	9	10	11
3	4	7									

Pop from Stack

把Stack最上面的東西取走

```
int Push_Stack(int *top)
{
    if(isEmpty(*top) == false)
    {
        int temp = stack[*top-1];
        (*top)--;
    }
    return temp;
}
```

Pop之前檢查Stack有沒有東西

temp = 7

TOP = 2

0	1	2	3	4	5	6	7	8	9	10	11
3	4	7									

Pop from Stack

把Stack最上面的東西取走

```
int Push_Stack(int *top)
{
    if(isEmpty(*top) == false)
    {
        int temp = stack[*top-1];
        (*top)--;
    }
    return temp;
}
```

剛剛Pop的7我們不會刻意從Array清掉
因為有Top值幫我們把關Stack的資料
下次有東西Push進來 7就被覆蓋了

temp = 7

TOP = 2

0	1	2	3	4	5	6	7	8	9	10	11
3	4	7									

如果看到這裡已經對題目有想法，那就別急著往下翻

去解題啦。

題目怎麼說

- 只要p跟q的位置是「面對面」（亦即「pq」），就代表彼此相看兩不厭，可被視為一對，其他的狀態，如背對背（亦即「qp」、「pp」或「qq」），則不能配對成功。成功配對後，該pq對就被移出等候配對名單，讓其他的p與q可以有機會繼續配對。

你會怎麼看？

..p..p.p...q.q.

- 找可能的組合：pq, p.q, p.....q
- 找到之後把該組p q去掉，重複動作

你會怎麼看？

..p..p.p...q.q.

已經配對的p q： 0

- 找可能的組合：pq, p.q, p.....q
- 找到之後把該組p q去掉，重複動作

你會怎麼看？

..p..p. ... .q.

已經配對的p q： 1

- 找可能的組合：pq, p.q, p.....q
- 找到之後把該組p q去掉，重複動作

你會怎麼看？

..p..p. ... .q.

已經配對的p q： 1

- 找可能的組合：pq, p.q, p.....q
- 找到之後把該組p q去掉，重複動作

你會怎麼看？

..p.. . ... . .

已經配對的p q： 2

- 找可能的組合：pq, p.q, p.....q
- 找到之後把該組p q去掉，重複動作

你會怎麼看？

..p.. . ... . .

已經配對的p q： 2

沒有其他的q可以與p配對了

- 找可能的組合：pq, p.q, p.....q
- 找到之後把該組p q去掉，重複動作

你會怎麼看？(cont.)

- 算p,q分別出現幾次？

你會怎麼看？(cont.)

- 算p,q分別出現幾次？

```
..p..p.p...q.q.
```

你會怎麼看？(cont.)

- 算p,q分別出現幾次？

..p..p.p...q.q.

P: 3
Q: 2
Matched: 2

你會怎麼看？(cont.)

- 算p,q分別出現幾次？

..p..p.p...q.q.

P: 3
Q: 2
Matched: 2

你會怎麼看？(cont.)

- 算p,q分別出現幾次？

..p..p.p...q.q.

P: 3
Q: 2
Matched: 2

.p...qq..p.pq.p..q.qpp..qpq

你會怎麼看？(cont.)

- 算p,q分別出現幾次？

`..p..p.p...q.q.`

P: 3
Q: 2
Matched: 2

`.p...qq..p.pq.p..q.qpp..qpq`

P: 7
Q: 7
Matched: 6

你會怎麼看？(cont.)

- 算p,q分別出現幾次？

`..p..p.p...q.q.`

P: 3
Q: 2
Matched: 2

`.p...qq..p.pq.p..q.qpp..qpq`

P: 7
Q: 7
Matched: 6

你會怎麼看？(cont.)

- 算p,q分別出現幾次？

..p..p.p...q.q.

.p...qq..p.pq.p..q.qpp..qpq

P: 3
Q: 2
Matched: 2

P: 7
Q: 7
Matched: 6

只數p q的出現次數
卻忽略了q在p前面的情況

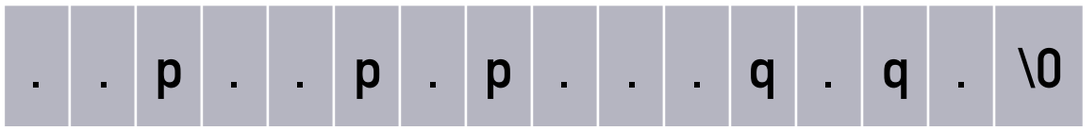
用程式的角度思考

用程式的角度思考

- 如何保存讀入的字串？

用程式的角度思考

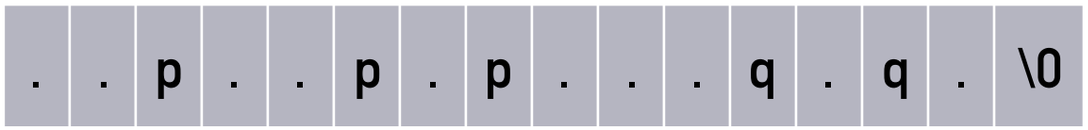
- 如何保存讀入的字串？



字串陣列
char input[17]

用程式的角度思考

- 如何保存讀入的字串？



字串陣列
char input[17]

- 用一個for迴圈，逐個對p,q做檢查

```
for (i = 0; i < strlen(input) ; i++)
{

}
```

用程式的角度思考

- 如何保存讀入的字串？

.	.	p	.	.	p	.	p	.	.	.	q	.	q	.	\0

字串陣列
char input[17]

- 用一個for迴圈，逐個對p,q做檢查

```
for (i = 0; i < strlen(input) ; i++)
{
        ?
}
```

分析P,Q的行為

女生加入等待的隊列
等待白馬王子出現

- 看到P → 加入排隊的清單，等待Q出現
- 看到 . → 什麼都不做
- 看到 Q →找尋是否已經有讀入的P

男生若看到隊列有女生
把隊伍中最後的女生牽走
否則只能離開不再回來

分析P,Q的行為

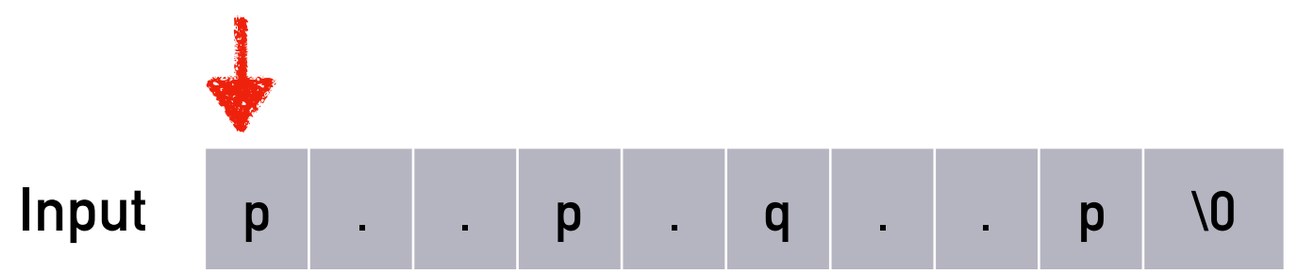
- 看到P → 將P Push到Stack
- 看到 . → 什麼都不做
- 看到 Q →找尋是否已經有讀入的P

男生若看到隊列有女生
把隊伍中最後的女生牽走
否則只能離開不再回來

分析P,Q的行為

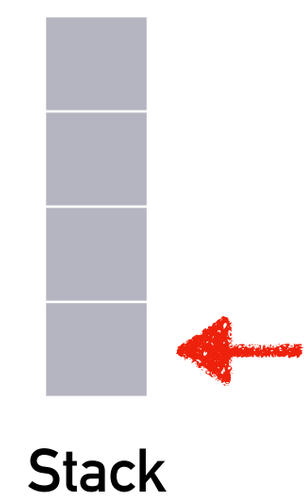
- 看到P → 將P Push到Stack
- 看到 . → 什麼都不做
- 看到 Q → 嘗試從Stack中Pop一個P
  如果Stack沒有P，這個Q永遠不會有P與它配對

Code Demo

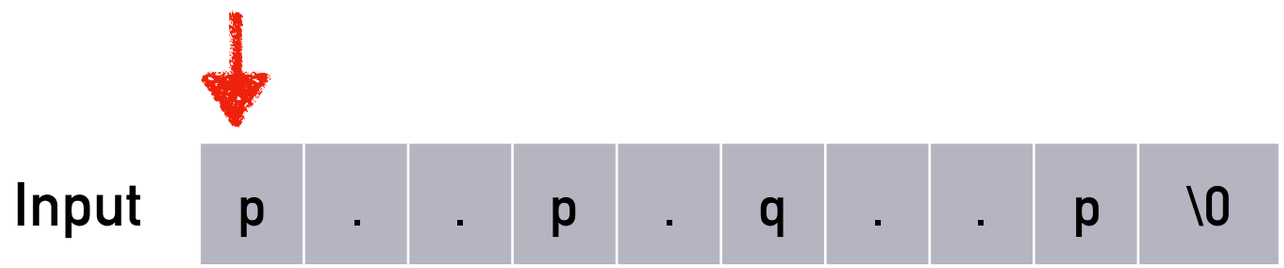



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
        Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
            Paired++;
}
```

Paired =	0
i =	0

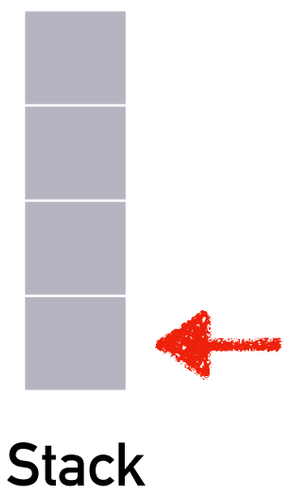


Code Demo

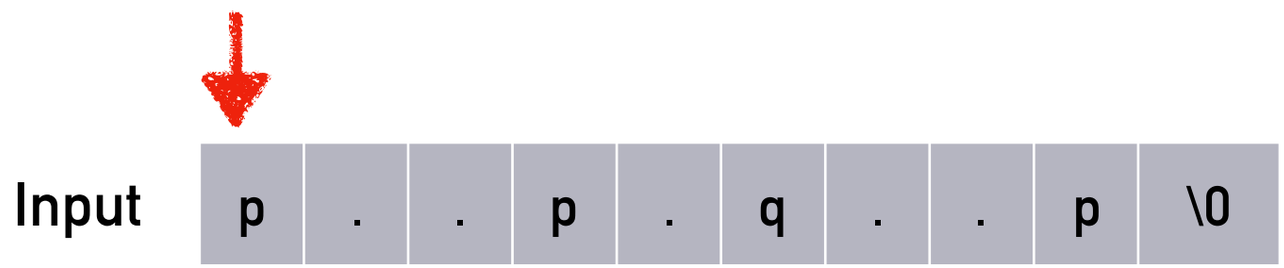



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == 'p')
         Push(P);
    else if(input == 'q')
        if(Pop() == true)
           Paired++;
}
```

Paired =	0
i =	0

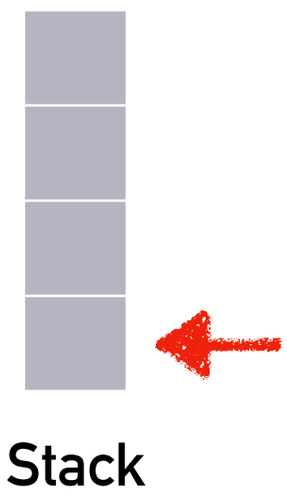


Code Demo

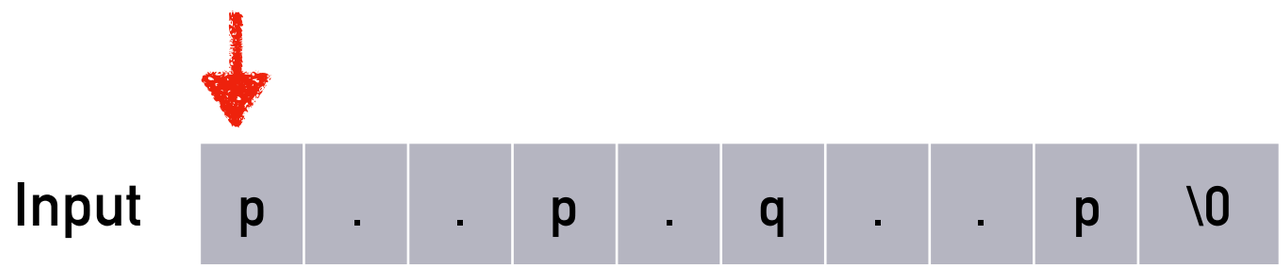



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
        Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
            Paired++;
}
```

Paired =	0
i =	0

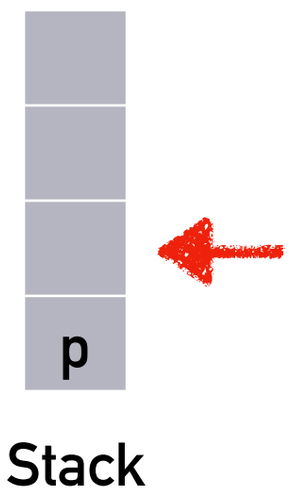


Code Demo

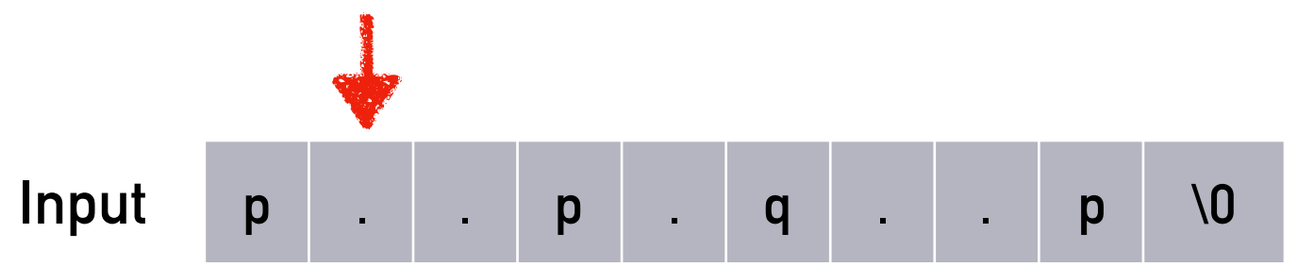



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
        Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
            Paired++;
}
```

Paired =	0
i =	0

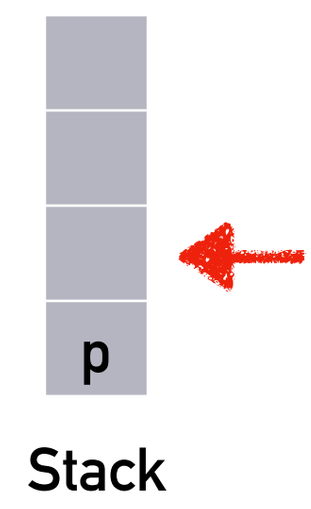


Code Demo

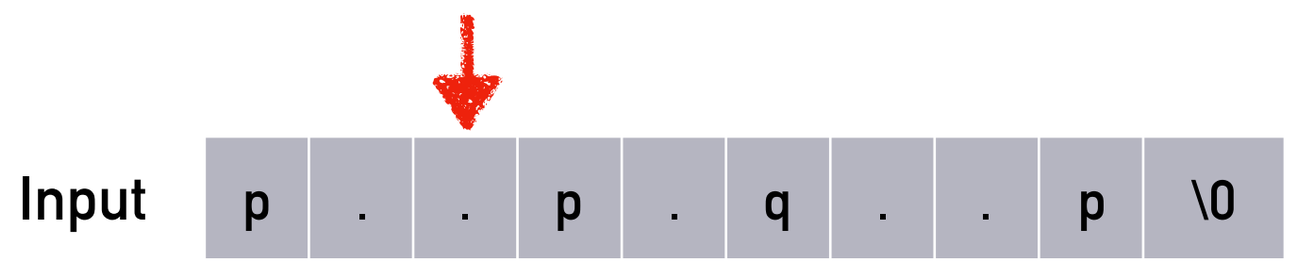



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
        Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
            Paired++;
}
```

Paired =	0
i =	1

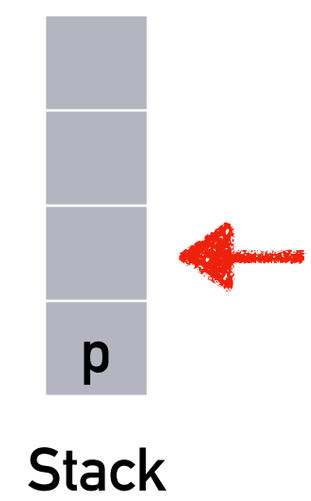


Code Demo

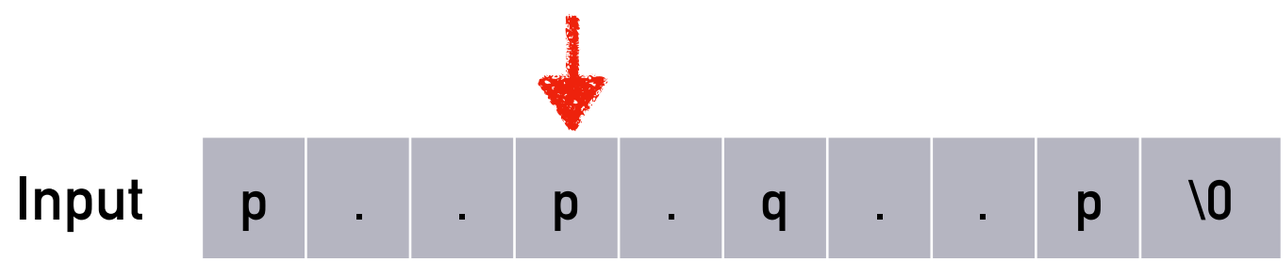



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
        Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
            Paired++;
}
```

Paired =	0
i =	2

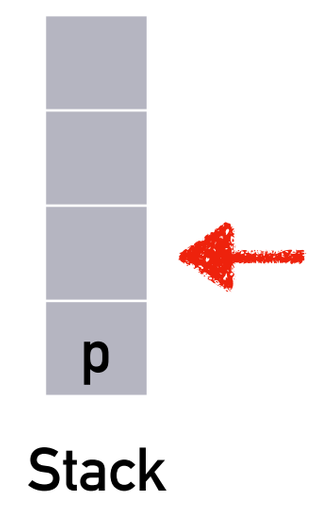


Code Demo

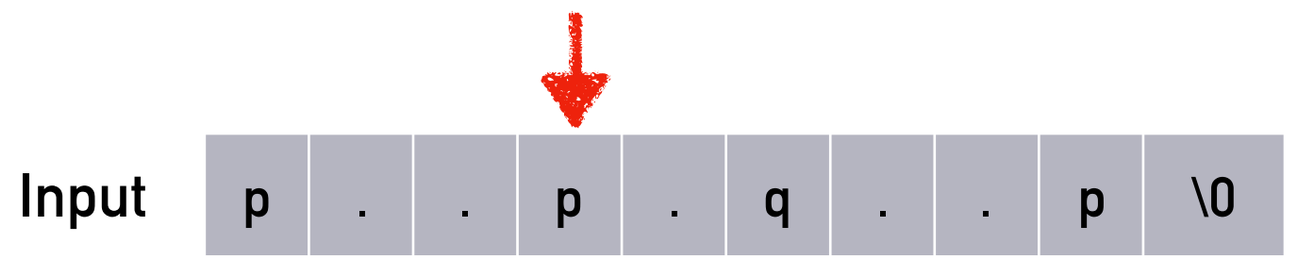



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
        Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
            Paired++;
}
```

Paired =	0
i =	3

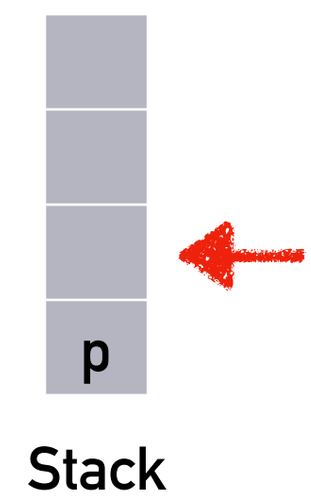


Code Demo

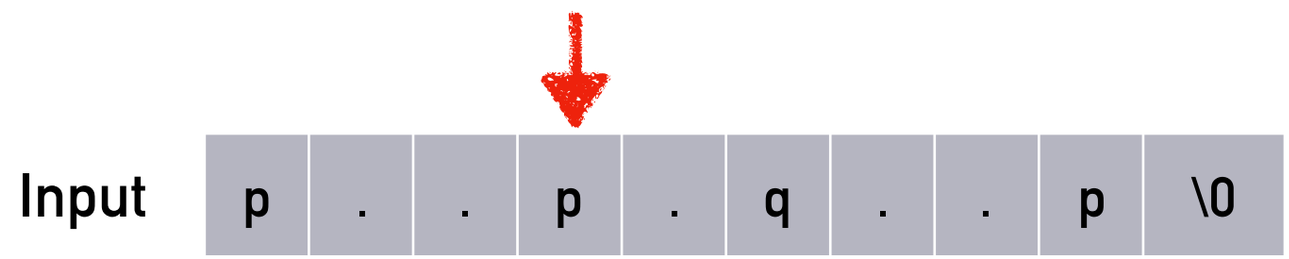



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
          Push(P);
    else if(input == ‘q’)
         if(Pop() == true)
            Paired++;
}
```

Paired =	0
i =	3

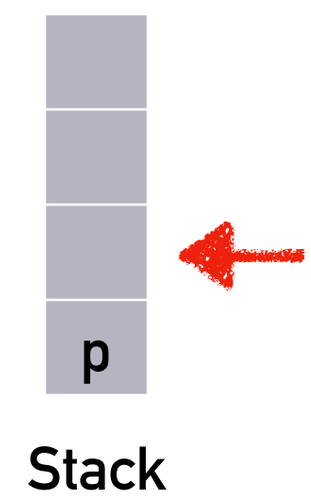


Code Demo

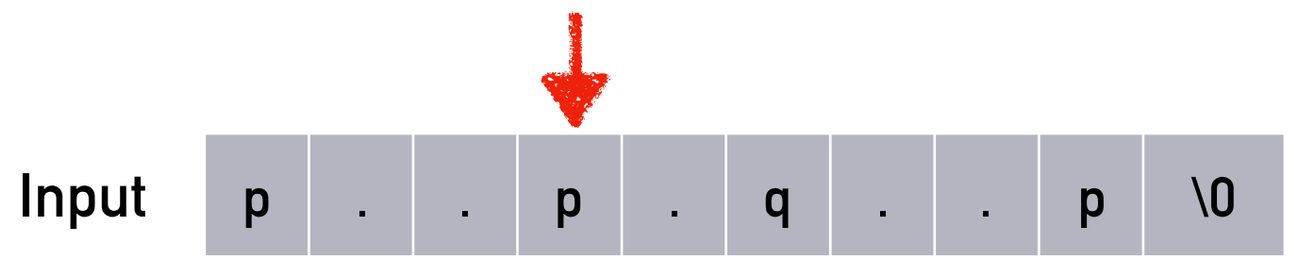



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
        Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
            Paired++;
}
```

Paired =	0
i =	3

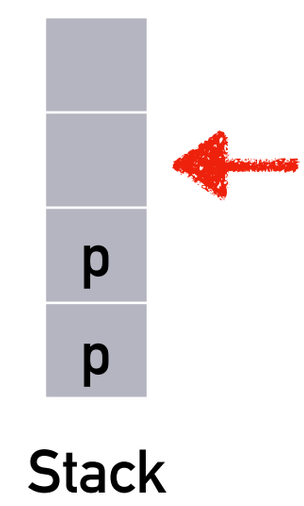


Code Demo

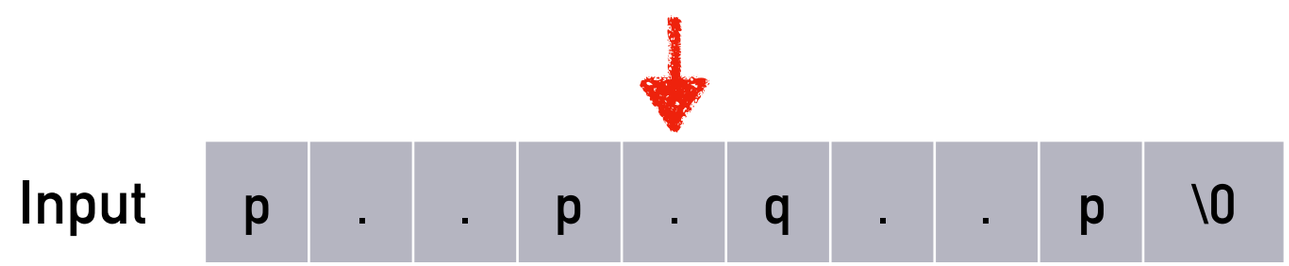



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
        Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
            Paired++;
}
```

Paired =	0
i =	3

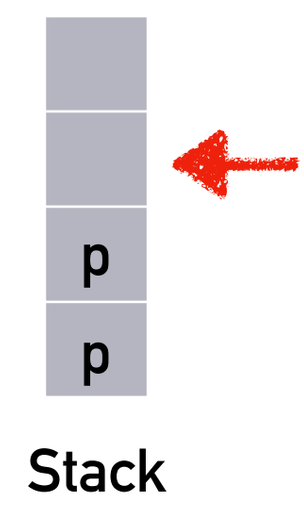


Code Demo

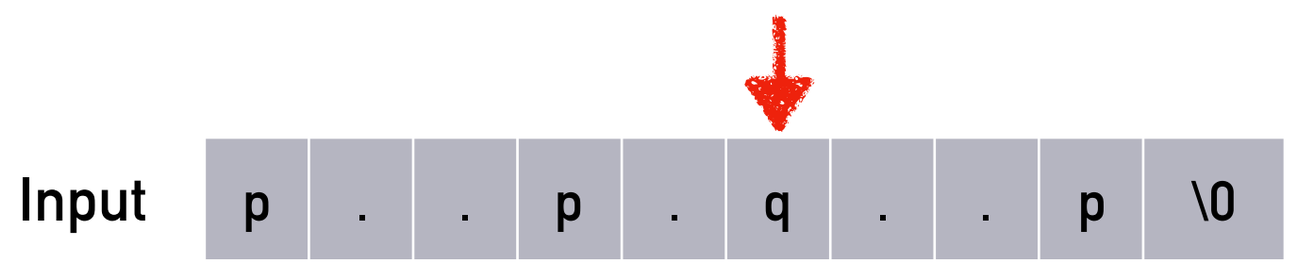



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
        Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
            Paired++;
}
```

Paired =	0
i =	4

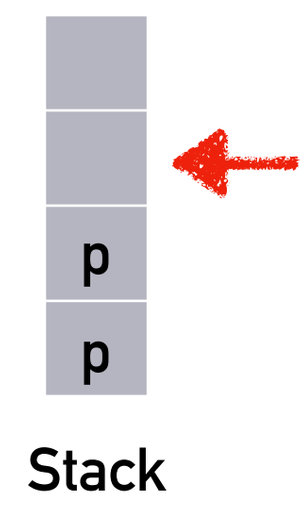


Code Demo

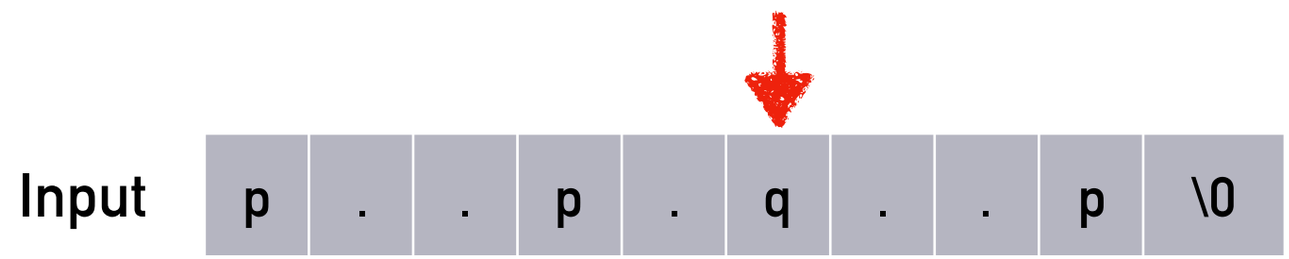



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
        Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
            Paired++;
}
```

Paired =	0
i =	5

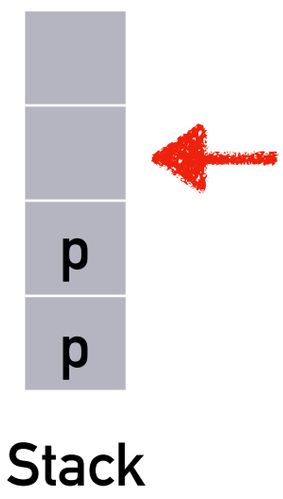


Code Demo

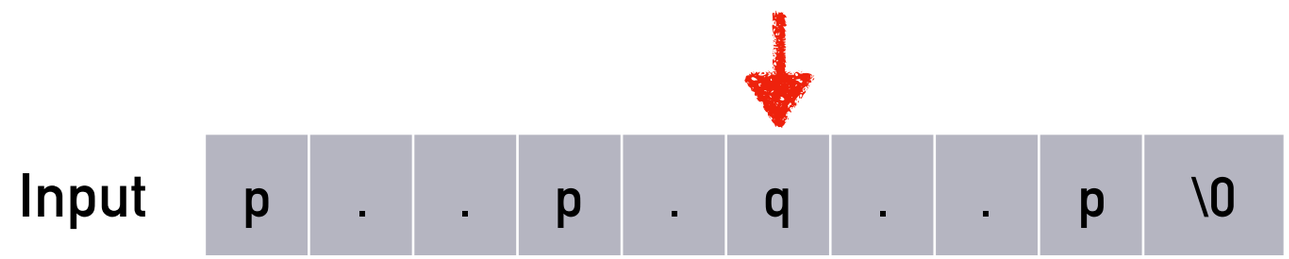



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
         Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
           Paired++;
}
```

Paired =	0
i =	5

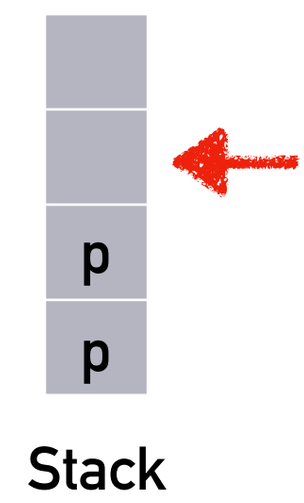


Code Demo

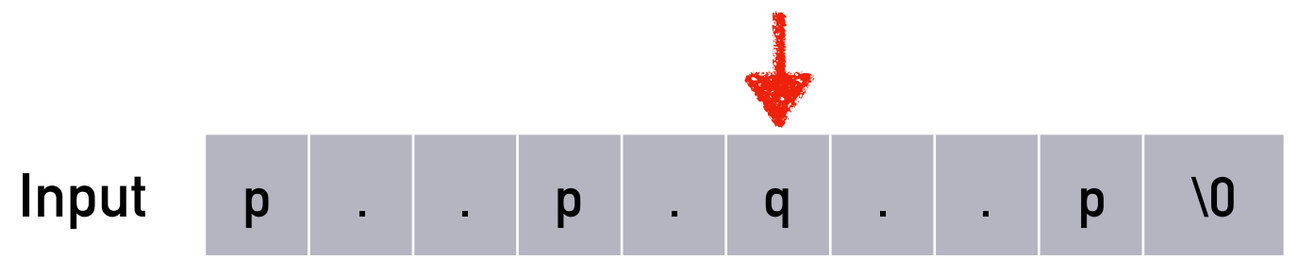



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
        Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
            Paired++;
}
```

Paired =	0
i =	5

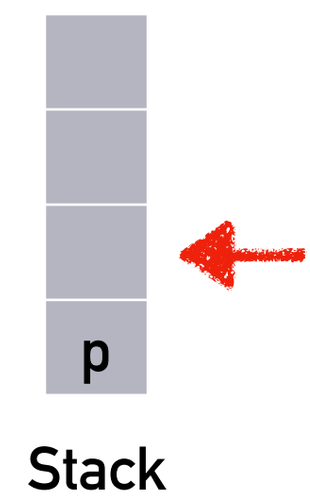


Code Demo

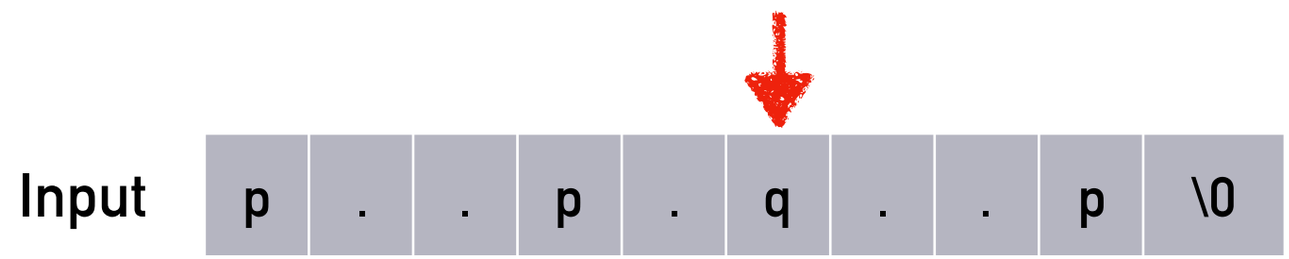



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
        Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
            Paired++;
}
```

Paired =	0
i =	5

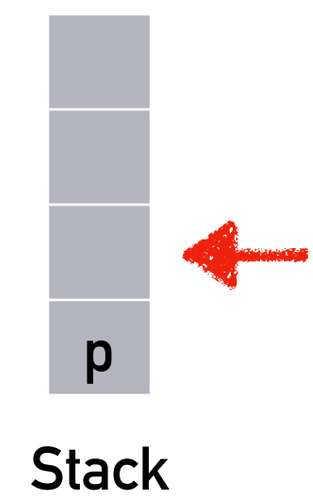


Code Demo

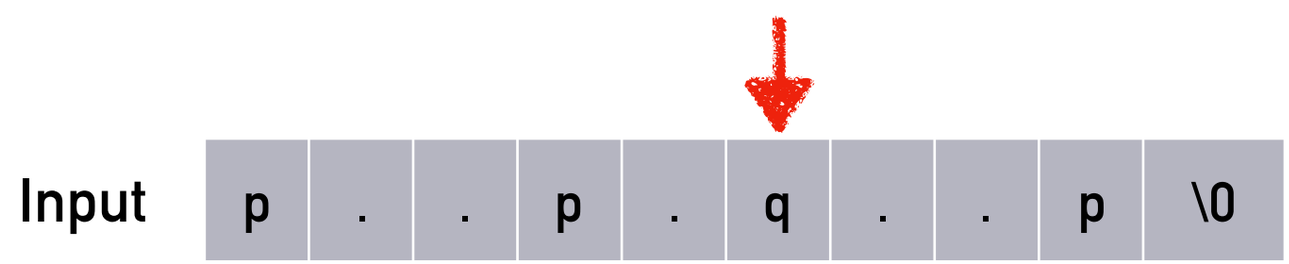



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
         Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
            Paired++;
}
```

Paired =	1
i =	5

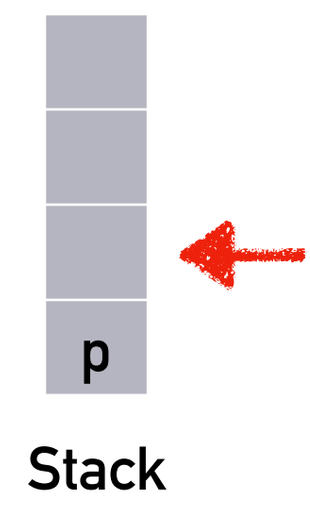


Code Demo

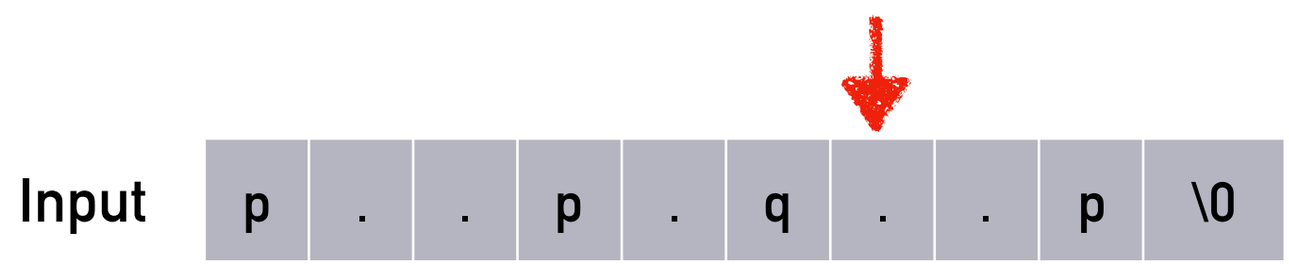



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
        Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
            Paired++;
}
```

Paired =	1
i =	5

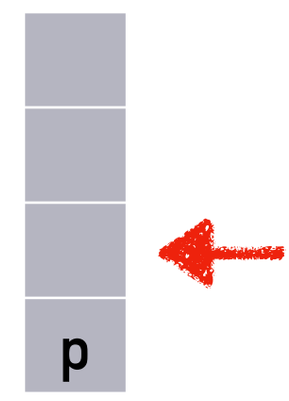


Code Demo



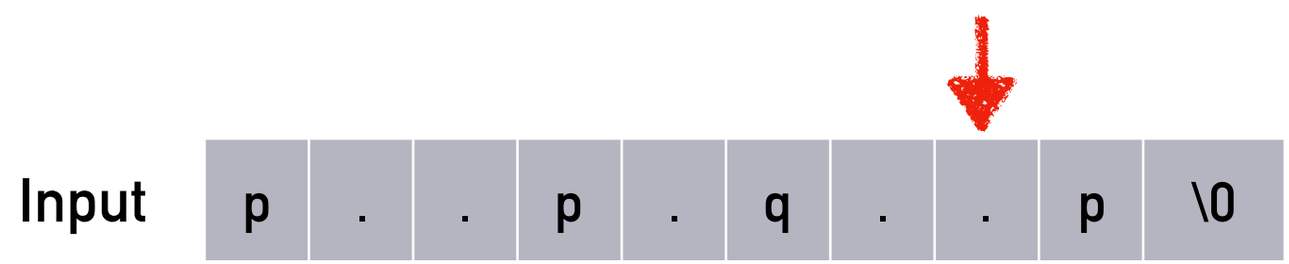

```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
        Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
            Paired++;
}
```

Paired =	1
i =	6



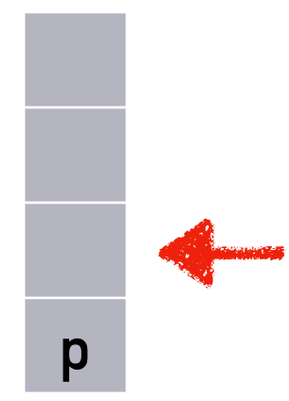
Stack

Code Demo



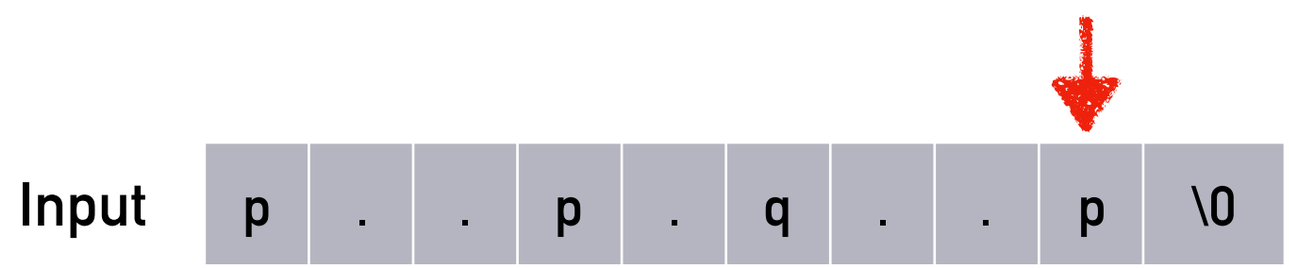

```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
        Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
            Paired++;
}
```

Paired =	1
i =	7



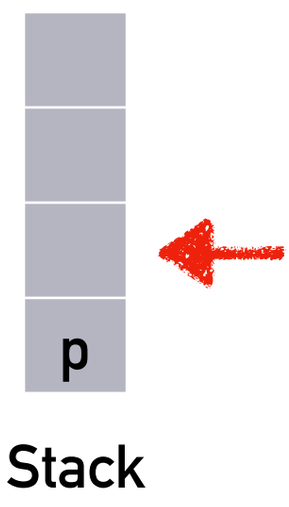
Stack

Code Demo

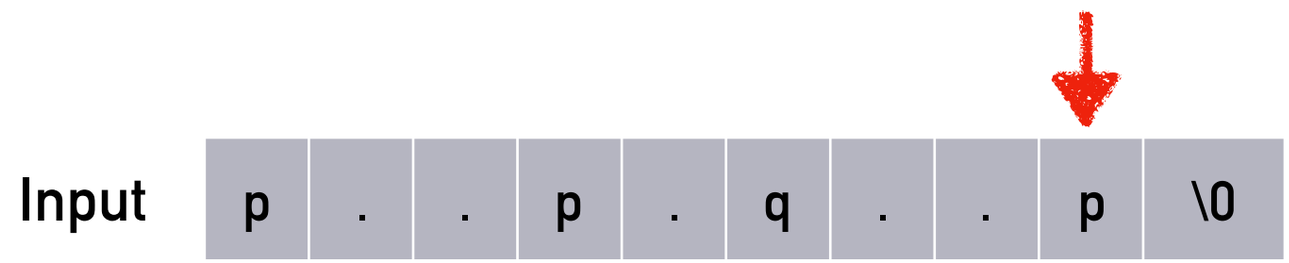



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
        Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
            Paired++;
}
```

Paired =	1
i =	8

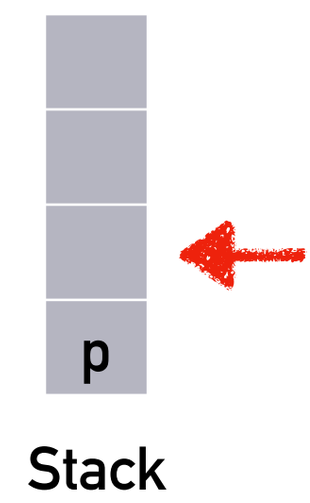


Code Demo

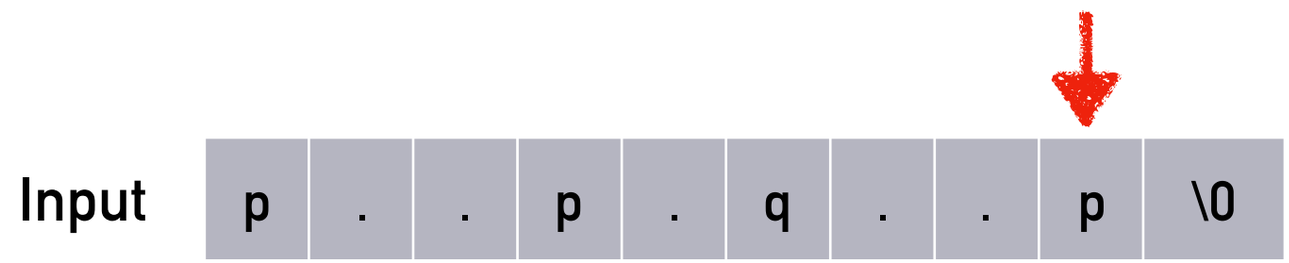



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == 'p')
         Push(P);
    else if(input == 'q')
        if(Pop() == true)
           Paired++;
}
```

Paired =	1
i =	8

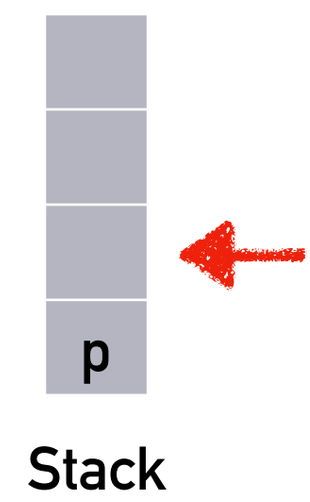


Code Demo

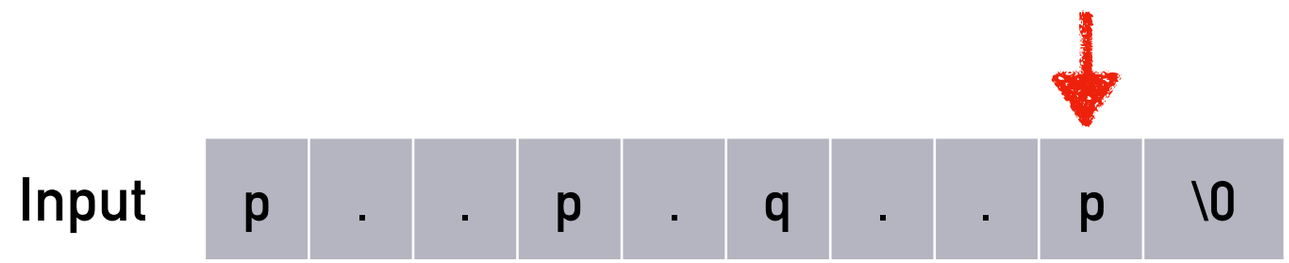



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
        Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
            Paired++;
}
```

Paired =	1
i =	8

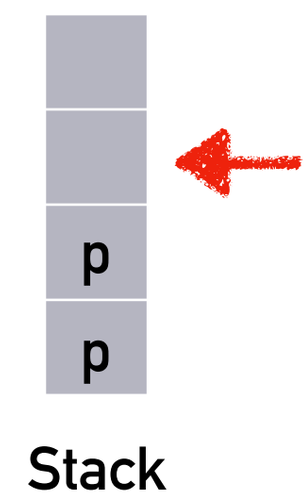


Code Demo

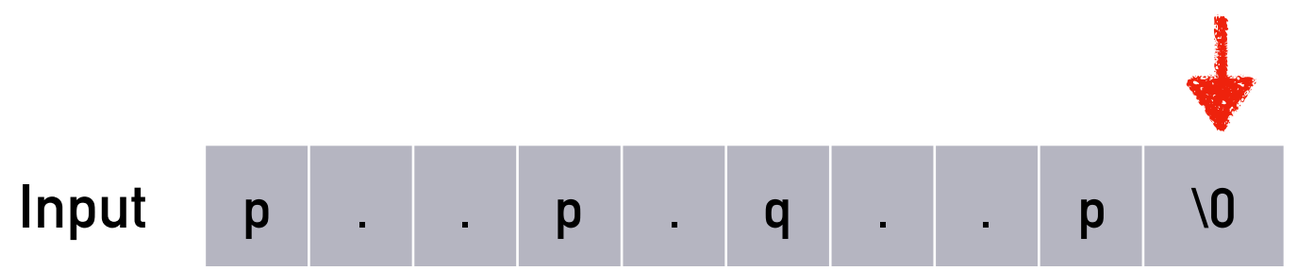



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
        Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
            Paired++;
}
```

Paired =	1
i =	8

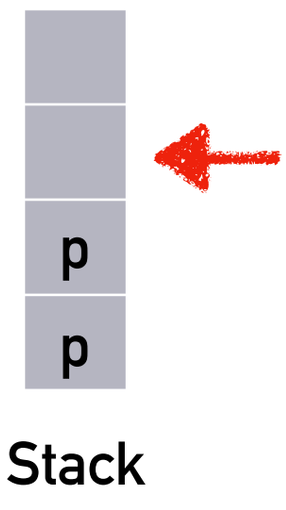


Code Demo

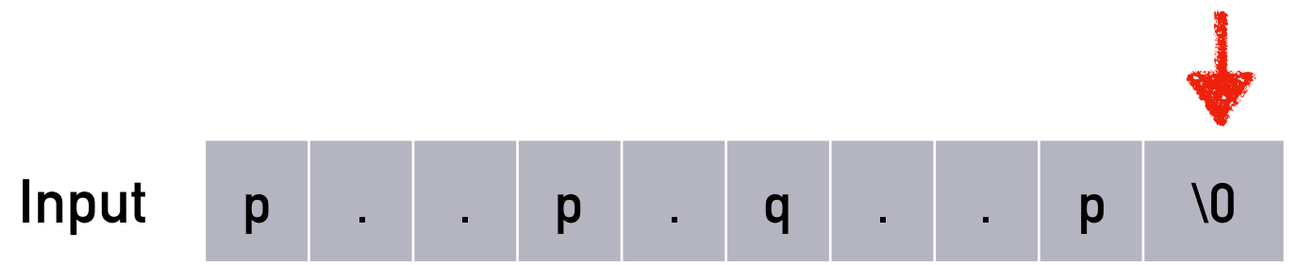



```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
        Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
            Paired++;
}
```

Paired =	1
i =	9

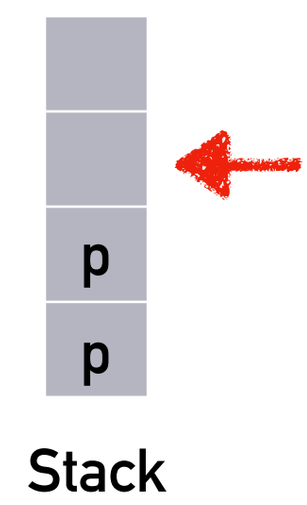


Code Demo




```
int Paired = 0;

for(i = 0; i < strlen(input) ; i++)
{
    if (input[i] == ‘p’)
         Push(P);
    else if(input == ‘q’)
        if(Pop() == true)
           Paired++;
}
```

Paired =	1
i =	9



測試資料長度：
前三筆必小於10^3
第四筆必小於10^5
最後一筆則必小於10^7

可是測資長度很大（~10^7）
我也要開一個10^7的陣列跟Stack?!

當然不用。

更快的做法

- 測試資料如何處理？
- 只用一個參數模擬Stack的操作

Think!

Stack的其他應用

- 括號配對 (Uva673)
- 中序式轉換 (d016)
- 遞迴演算法的改寫（Ex: Depth-First Search）