5. Array

Lecturer : Watcharin Sarachai, Ph.D.

Email : watcharin_s@mju.ac.th

Facebook : wsarachai

Topic

- Array
- Arrays and for loops
- Selection Sort

Array

- การโปรแกรมที่มีการทำงานกับกลุ่มข้อมูลที่คล้ายกันเกิดขึ้นบ่อย ๆ ในทางปฏิบัติ เช่น
  - การใช้ตัวแปรหนึ่ง ๆ เพื่อจัดเก็บค่า grade ของ student แต่ละคนภายในชั้นเรียน ซึ่งในกรณีนี้อาจมีการประกาศตัวแปรเท่ากับจำนวนของ student โดยมีชื่อตัวแปรที่ใกล้เคียงกัน เช่น grade1, grade2, ..., grade22

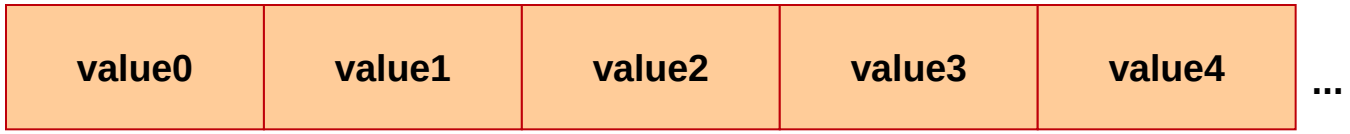
    char grade1, grade2, grade3, ...

- ตัวแปรอะเรย์ยอมให้มีการจัดกลุ่มของตัวแปรที่มีชนิดข้อมูลแบบเดียวกันไว้ด้วยกัน โดยใช้การอ้างถึงโดยใช้ชื่อเดียวกัน

  char grades[22]; /* holds 22 chars */

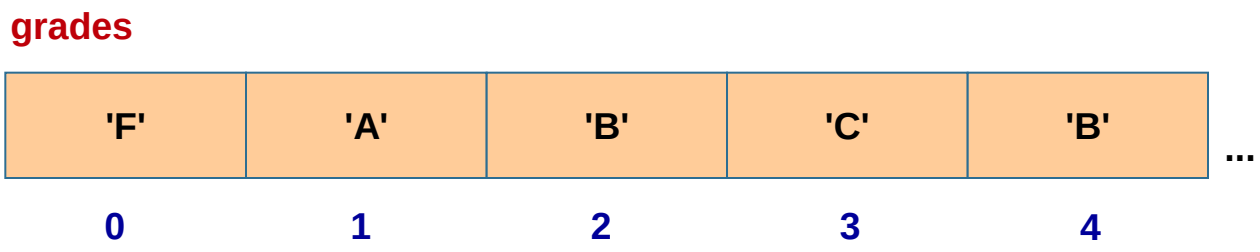
Array

- Variable = เป็นการอ้างถึงตำแหน่งภายในเมมโมรี่โดยอาศัยการใช้ชื่อของตัวแปรเพื่อจัดเก็บค่าเพียงค่าเดียว
- Array = เป็นการอ้างถึงตำแหน่งภายในเมมโมรี่โดยใช้ชื่อของตัวแปรเพื่อจัดเก็บค่าเรียงลำดับกันไปตามจำนวนที่ถูกระบุ โดยมีลักษณะดังต่อไปนี้



Array

- แต่ละ Element จัดเก็บค่าได้เพียง 1 ค่า
- ทุก ๆ Element จะต้องจัดเก็บค่าชนิดเดียวกันเท่านั้น
- ทุก ๆ Element ในกลุ่มเดียวกันสามารถอ้างถึงได้โดยใช้ชื่อเดียวกัน
- การแยกความแตกต่างระหว่างแต่ละ Element สามารถระบุได้โดยใช้ชื่อของอะเรย์ตามด้วยลำดับที่ (index) ที่ถูกระบุ
- ค่าลำดับที่ของอะเรย์เริ่มต้นจาก 0 เสมอ

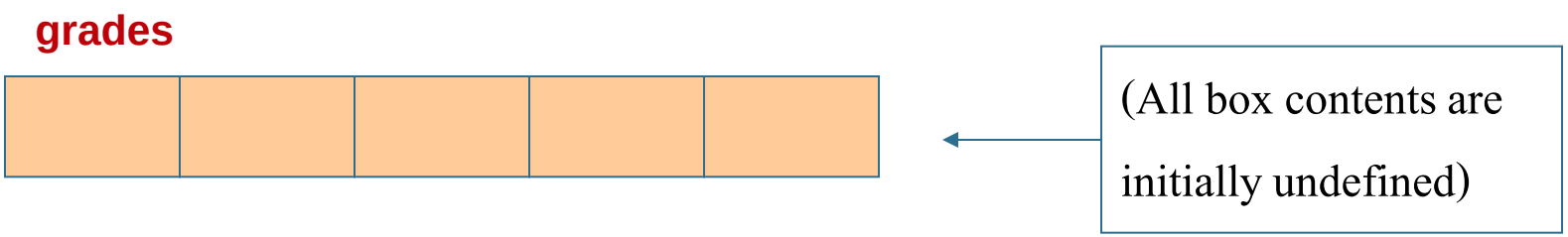


Array

- การสร้างอะเรย์เริ่มต้นจากการประกาศตัวแปรเช่นเดียวกับตัวแปรชนิดอื่น ๆ
- การประกาศตัวแปรเริ่มต้นจากการระบุชนิดข้อมูลภายในอะเรย์ ชื่อของอะเรย์ และจำนวน Element ที่อยู่ภายในอะเรย์ภายใต้เครื่องหมายก้ามปู เช่น

```
char grades[5];
```

- ภายหลังการประกาศตัวแปร



Declaring an array 1

- เป็นกลุ่มของสมาชิก(elements) ที่มีชนิดข้อมูลเป็นแบบเดียวกัน และสามารถอ้างถึงได้โดยใช้ตัวแปรเพียงตัวเดียว
- การประกาศตัวแปรอะเรย์แบบ integers ทำได้ดังนี้

  elementType arrayName [ ] ;

  - **elementType**: ชนิดข้อมูลของสมาชิกภายในอะเรย์
  - **arrayName**: ชื่อของตัวแปรที่ต้องการประกาศ
- การประกาศตัวแปรแบบนี้จะยังไม่ได้มีการจองเมมโมรี่
- การประกาศตัวแปรแบบนี้จะยังไม่ได้กำหนดขนาดของ array
- ชื่อของอะเรย์จะถือเป็น identifier

Arrays

- ชื่ออะเรย์จะเก็บที่อยู่ของอะเรย์จริงๆ อีกที
- ขนาดของอะเรย์ต้องเป็นค่าจำนวนเต็มเสมอ
  - ไม่สามารถใช้ชื่อตัวแปรแทนได้
  - ไม่สามารถเปลี่ยนแปลงขนาดของอะเรย์ในขณะรันไทม์ได้
- ลำดับที่ของสมาชิกภายในอะเรย์เริ่มต้นที่ **0** จนถึง **SIZE - 1**

Initializing an array

- **Option 1:** ในกรณีที่ไม่รู้ค่าของสมาชิกที่แน่นอนภายในอะเรย์ อาจกำหนดไว้เฉพาะจำนวนของสมาชิกภายในอะเรย์ก่อนได้ดังนี้

```
int [5] temp ;    // ค่าของสมาชิกภายในอะเรย์ยังไม่ได้ถูกกำหนดไว้
```

- **Option 2:** ในกรณีที่ทราบจำนวนที่แน่นอนของสมาชิกภายในอะเรย์ สามารถกำหนดขนาดและค่าเริ่มต้นของอะเรย์ได้ภายในเครื่องหมายปีกกา:

```
int [5] temp =  {45, 47, 44, 56, 49};
```

- **Option 3:** หากไม่มีการกำหนดจำนวนไว้ แต่มีการกำหนดค่าเริ่มต้นของอะเรย์ไว้ คอมไพเลอร์จะทำการกำหนดค่าขนาดให้โดยอัตโนมัติ

```
int [] temp = {45, 47, 44, 56};
```

Initializing an array

```
int x [ 5 ] = { 1,2,3,4,5 };
```
size 10 bytes

- สร้างอะเรย์ที่ประกอบด้วยสมาชิก 0-4 มีค่าตั้งแต่ 1-5

```
int x [ 5 ] = { 4,3 };
```
size 10 bytes

- สร้างอะเรย์ที่ประกอบด้วยสมาชิก 0-4 มีค่าตั้งแต่ 4,3,?,?,?

```
int x [ ] = { 1,2,3 };
```
size 6 bytes

- สร้างอะเรย์ที่ประกอบด้วยสมาชิก 0-2 มีค่าตั้งแต่ 1,2,3

Initializing an array

- หมายเหตุ เช่นเดียวกับตัวแปรทั่ว ๆ ไป หากไม่มีการกำหนดค่าเริ่มต้นของอะเรย์ไว้ ค่าภายในอะเรย์จะเป็น “garbage values.”
- ในกรณีที่กำหนดขนาดของอะเรย์ไว้ แต่มีจำนวนสมาชิกภายในอะเรย์เกินกว่าขนาดของอะเรย์ สมาชิกส่วนที่เกินจะถูกตัดออกไปโดยอัตโนมัติ
- ขนาดของอะเรย์เมื่อมีการประกาศตัวแปรแล้วจะไม่สามารถเปลี่ยนแปลงได้
- ขนาดของอะเรย์จะต้องไม่เป็นค่าลบ

The array element operator [ ]

- ลำดับสมาชิกภายในอะเรย์:
  - แต่ละ element ของอะเรย์จะมีลักษณะคล้ายกับตัวแปรทั่ว ๆ ไป
- การเข้าถึงค่าสมาชิกภายในอะเรย์สามารถทำได้โดยใช้ array element operator ที่ประกอบไปด้วยชื่อของอะเรย์และลำดับของสมาชิกภายในเครื่องหมายก้ามปู ซึ่งค่านี้จะถูกเรียกว่า index เช่น

array_name[index]

```
#include <stdio.h>
main()
{
    int x[5];
        x[0]=23;  /* valid */
        x[2.3]=5; /* invalid: index is not an int */
}
```

Array Elements

- สมาชิกภายในอะเรย์จะถูกกำหนดลำดับตั้งแต่ ***0*** ไปจนถึง ***N-1***
- การเข้าถึงสมาชิกภายในอะเรย์สามารถทำได้โดยการระบุชื่อของอะเรย์ ตามด้วยเลขลำดับที่อยู่ภายในเครื่องหมายก้ามปู

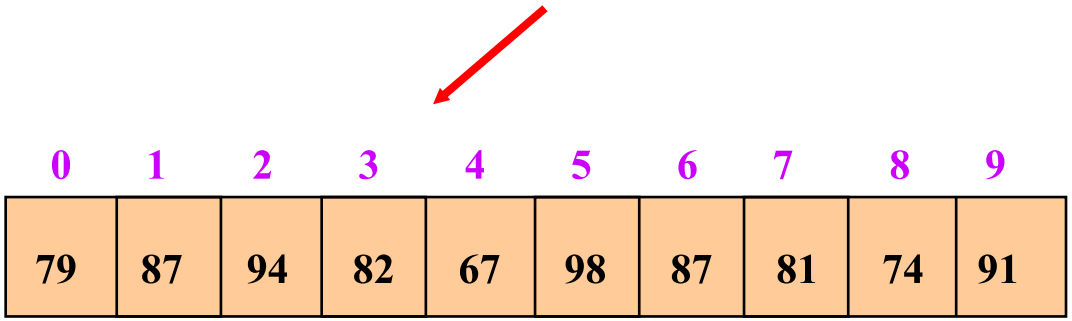
```
int numbers[25];
numbers[0] = 5;
numbers[1] = 11;
int n = 77;
numbers[7] = n + 4000;
n = numbers[1];
numbers[8] = numbers[7];
numbers[n+5] = n;
```

Arrays

- การกำหนดค่า *array* พร้อมกับการประกาศตัวแปร

```
int scores[] = { 79, 87, 94, 82, 67, 98, 87, 81, 74, 91 };
```

แต่ละค่าจะสามารถอ้างถึงได้ตามลำดับของ *index*

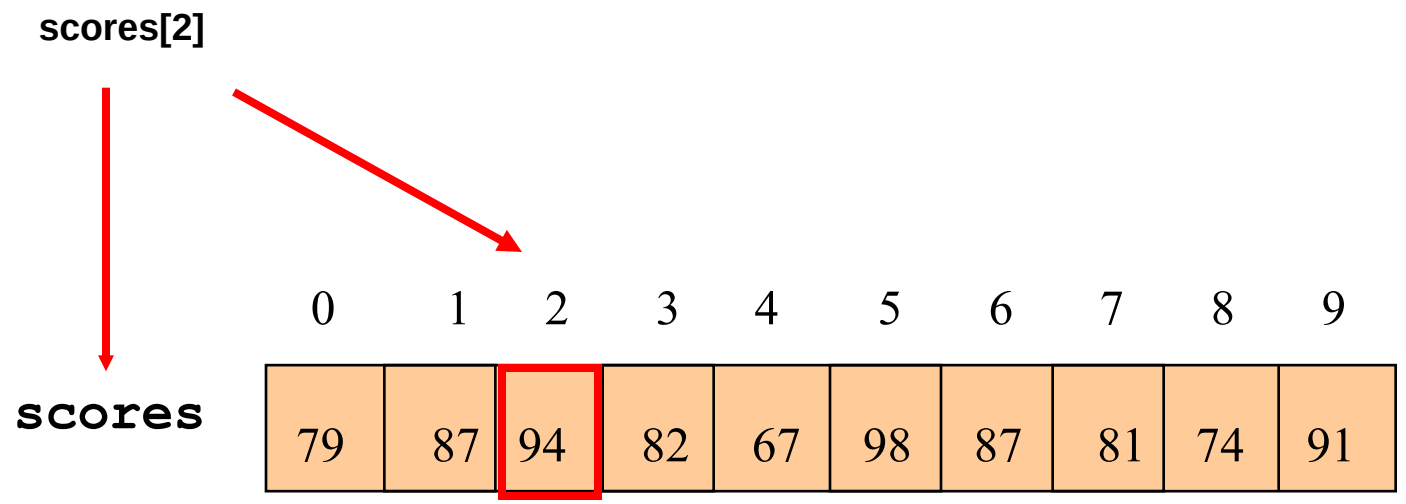


ขนาดของ array ที่มีขนาดเท่ากับ N จะเริ่มต้นจาก 0 ถึง N-1

array เก็บค่า 10 ค่าที่เริ่มต้นจาก 0 ถึง 9

Arrays

- การอ้างค่าภายใน array จะสามารถทำได้โดยอาศัยชื่อของarray ตามด้วยลำดับของ index ที่อยู่ภายในเครื่องหมายก้ามปู
- ตัวอย่างเช่น



จะเป็นการอ้างถึงค่าในลำดับที่ 3 ซึ่งจะได้แก่ 94 (แต่ index จะเป็นลำดับที่ 2)

Array pitfalls

- ผู้ใช้ไม่สามารถกำหนดค่าผ่านชื่อของอะเรย์ทั้งหมดได้:

```
int scores[5];

scores = 18;
```

ERROR!

- ผู้ใช้ไม่สามารถเปลี่ยนแปลงขนาดของอะเรย์ได้:

```
int size;
int scores[size];
printf("Type array size: ");

scanf("%d", &size);
```

ERROR!

Array pitfalls

- ผู้ใช้สามารถเข้าถึงลำดับของสมาชิกได้ไม่จำกัด แม้ว่าจะไม่ถูกต้องก็ตาม ภาษาซีจะไม่มีการตรวจสอบความถูกต้องในการกำหนดลำดับที่ของสมาชิกภายในอะเรย์

```
int scores[5];
scores[-1] = 18;
```

CAREFUL!

- การกำหนดค่าระหว่างอะเรย์ไม่สามารถทำได้โดยตรง:

```
int scores[30], grades[30];
scores = grades
```

ERROR!

- ในกรณีของการกำหนดค่าสามารถทำได้ทีละค่าผ่านการทำงานของ Loop

Arrays and for loops

- เนื่องจากข้อมูลภายในอะเรย์มีลักษณะเรียงลำดับกันไป ดังนั้นจึงเหมาะสมที่จะใช้งานร่วมกับคำสั่ง for loops ในการเข้าถึงข้อมูลภายในอะเรย์
  - ตัวอย่างเช่น การพิมพ์ค่าสมาชิกภายในอะเรย์

```
for ( i = 0; i < 8; i++) {
    printf(“  “, numbers[i]);
}
printf(“\n“);   // end the line of output
```

0	1	2	3	4	5	6	7
0	4	11	0	44	0	0	2

  - Output

    0 4 11 0 44 0 0 2

How Arrays and for loops work

กำหนดค่าเริ่มต้นให้แก่ loop

```
for ( i = 0; i < 8; i++) {
        printf(“  “, numbers[i]);
}
printf(“\n“);
```

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*
0	4	11	0	44	0	0	2

i = 0

How Arrays and for loops work

i <= 8? [true]

```
for ( i = 0; i < 8; i++) {
        printf(“  “, numbers[i]);
}
printf(“\n“);
```

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*
0	4	11	0	44	0	0	2

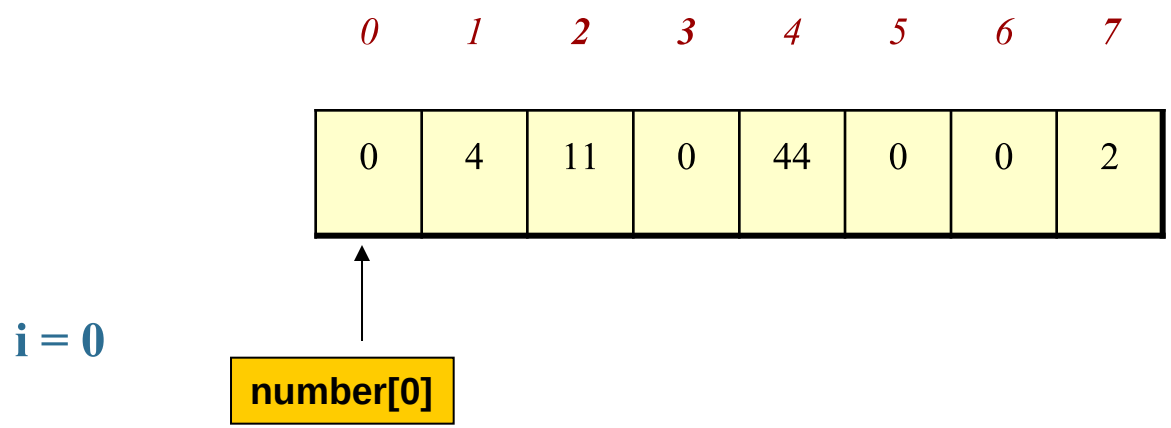
i = 0

How Arrays and for loops work

แสดงผล number[0]

0

```
for ( i = 0; i < 8; i++) {
        printf(“  “, numbers[i]);
}
printf(“\n“);
```

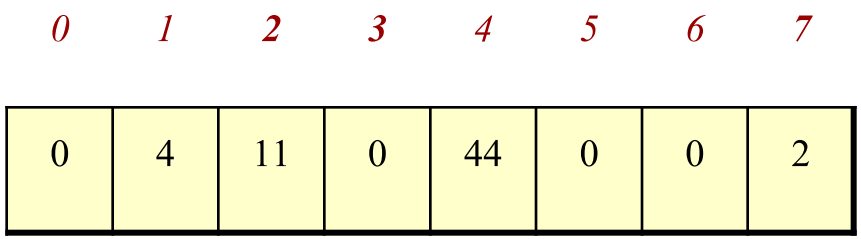


How Arrays and for loops work

เพิ่มค่า i เป็น 1

0

```
for ( i = 0; i < 8; i++) {
        printf(“  “, numbers[i]);
}
printf(“\n“);
```



i = 1

How Arrays and for loops work

i <= 8? [true]

0

```
for ( i = 0; i < 8; i++) {
        printf(“  “, numbers[i]);
}
printf(“\n“);
```

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*
0	4	11	0	44	0	0	2

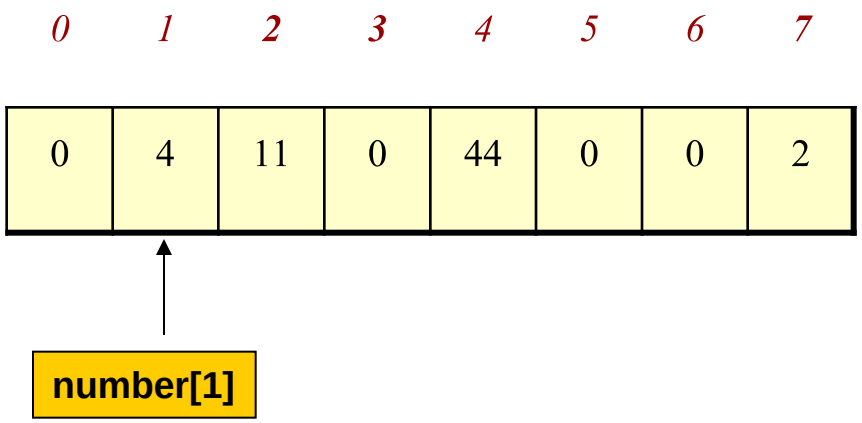
i = 1

How Arrays and for loops work

แสดงผล number[1]

0 4

```
for ( i = 0; i < 8; i++) {
        printf(“  “, numbers[i]);
}
printf(“\n“);
```



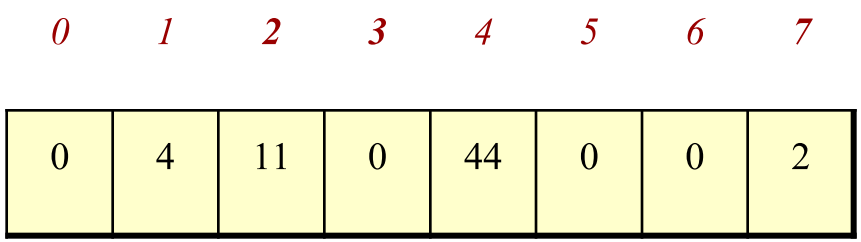
i = 1

How Arrays and for loops work

เพิ่มค่า i เป็น 2

0 4

```
for ( i = 0; i < 8; i++) {
        printf(“  “, numbers[i]);
}
printf(“\n“);
```



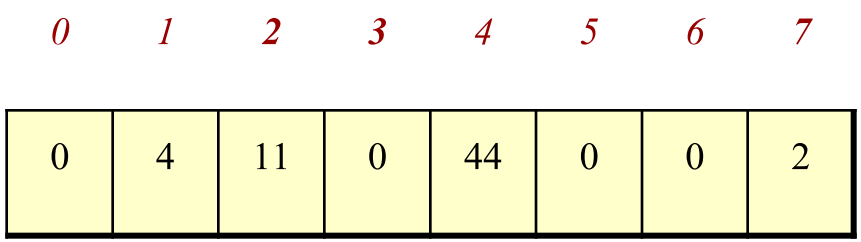
i = 2

How Arrays and for loops work

0 4

i <= 8? [true]

```
for ( i = 0; i < 8; i++) {
        printf(“  “, numbers[i]);
}
printf(“\n“);
```



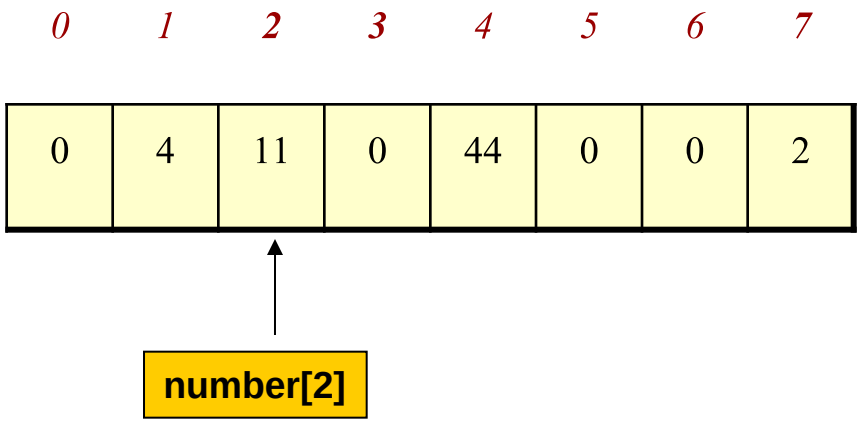
i = 2

How Arrays and for loops work

แสดงผล number[2]

0 4 11

```
for ( i = 0; i < 8; i++) {
        printf(“  “, numbers[i]);
}
printf(“\n“);
```



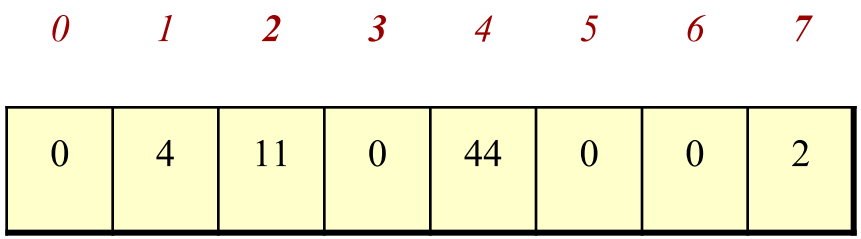
i = 2

How Arrays and for loops work

เพิ่มค่า i เป็น 3

0 4 11

```
for ( i = 0; i < 8; i++) {
        printf(“  “, numbers[i]);
}
printf(“\n“);
```



i = 3

How Arrays and for loops work

i <= 8? [true]

0 4 11

```
for ( i = 0; i < 8; i++) {
        printf("  ", numbers[i]);
}
printf("\n");
```

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*
0	4	11	0	44	0	0	2

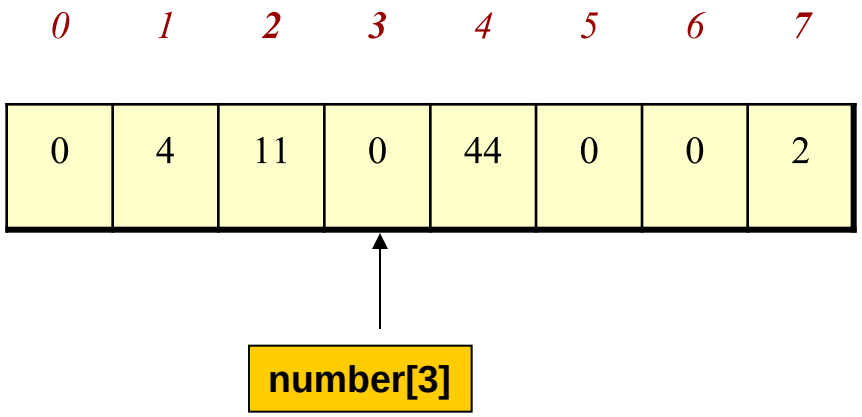
i = 3

How Arrays and for loops work

แสดงผล number[3]

0 4 11 0

```
for ( i = 0; i < 8; i++) {
        printf(“  “, numbers[i]);
}
printf(“\n“);
```



i = 3

How Arrays and for loops work

เพิ่มค่า i เป็น 7

0 4 11 0 44 0 0

```
for ( i = 0; i < 8; i++) {
        printf(“  “, numbers[i]);
}
printf(“\n“);
```

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*
0	4	11	0	44	0	0	2

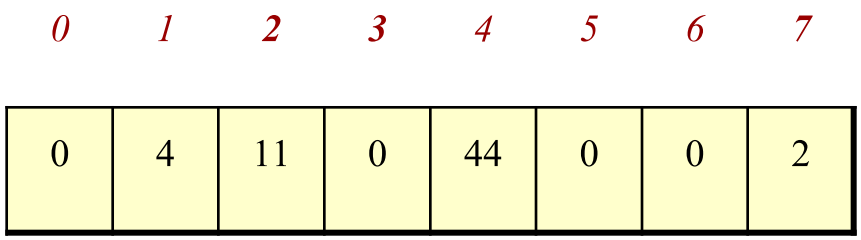
i = 7

How Arrays and for loops work

i <= 8? [true]

0 4 11 0 44 0 0

```
for ( i = 0; i < 8; i++) {
        printf(“  “, numbers[i]);
}
printf(“\n“);
```



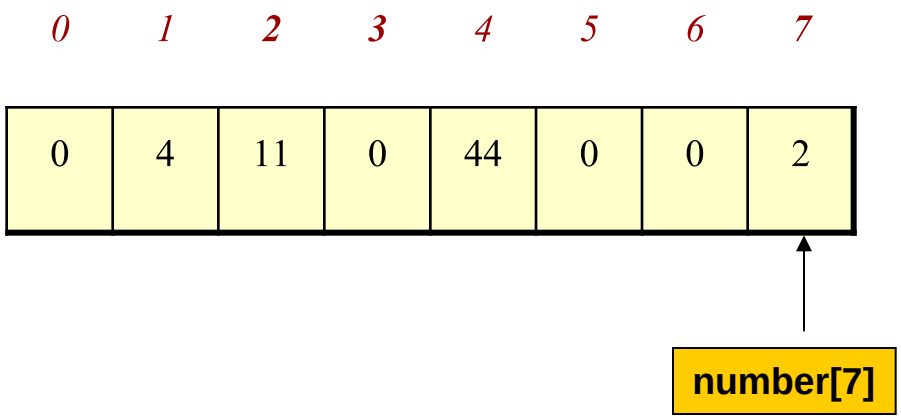
i = 7

How Arrays and for loops work

แสดงผล number[7]

0 4 11 0 44 0 0 2

```
for ( i = 0; i < 8; i++) {
        printf(“  “, numbers[i]);
}
printf(“\n“);
```



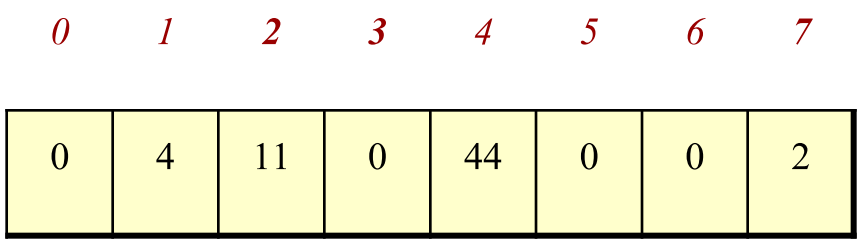
i = 7

How Arrays and for loops work

เพิ่มค่า i เป็น 8

0 4 11 0 44 0 0 2

```
for ( i = 0; i < 8; i++) {
        printf("  ", numbers[i]);
}
printf("\n");
```



i = 8

How Arrays and for loops work

i <= 8? [false]

0 4 11 0 44 0 0 2

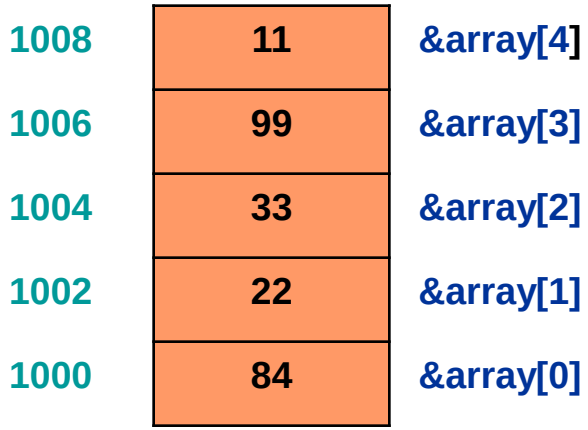
```
for ( i = 0; i < 8; i++) {
        printf("  ", numbers[i]);
}
printf("\n");
```

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*
0	4	11	0	44	0	0	2

i = 8

Array Elements & For Loop

- เนื่องจากอะเรย์เป็นกลุ่มของข้อมูลที่เรียงลำดับกัน ดังนั้นจึงเหมาะสมอย่างยิ่งกับการเข้าถึงข้อมูลโดยใช้ For Loop




```
#include<stdio.h>
main() {
      int array[] = { 84, 22, 33, 99, 11 };
      int i, sum ;
      for ( i = 0; i < 5; i++) {
            printf("\t%d.", array[i]);
            sum = sum+ array[i];
      }
}
```

Output: 84 22 33 99 11

Bounds Checking

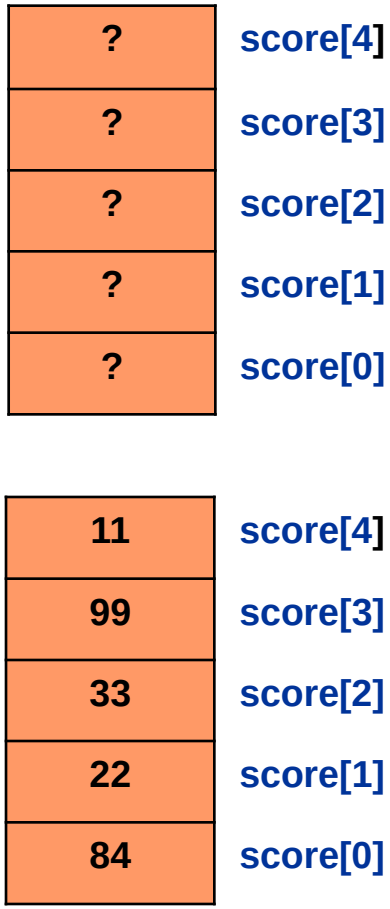
- เมื่ออะเรย์ถูกสร้างขึ้นจะมีขนาดคงที่เสมอ
- index จะถูกใช้ในการอ้างถึงอะเรย์ที่ต้องระบุค่าที่ถูกต้อง
- นั่นคือค่า index จะต้องอยู่ในขอบเขตที่กำหนดไว้ (0 ถึง N-1) เท่านั้น
- ตัวอย่างเช่น หากอะเรย์ code มีค่าเท่ากับ 100 ค่า ค่า index ที่สามารถใช้ในการอ้างอิงจะเริ่มตั้งแต่ 0 ถึง 99

problem

```
for (index=0; index <= 100; index++) {
    codes[index] = index*5 + temp;
}
```

A simple example

```
#include <stdio.h>                    //61
main() {
      int i , score[5] , total = 0 ;
      for ( i = 0 ; i < 5 ; i++ ) {
         printf( “Enter score %d” , i );
         scanf ( “%d” , &score[ i ] );
      }
      for ( i = 0 ; i < 5 ; i++ ) {
         total = total + score[i];
      }
      printf ( “The average score is %d” , total / 5 );

}
```

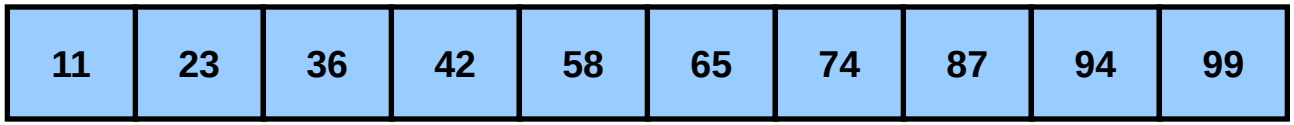


Example : Find Minimum

```
#include <stdio.h>                    //62
main()
{
    int  array [] = { 42, 23, 74, 11, 65, 58, 94, 36, 99, 87 } ;
    int  i, min;
    min = 100;

     for ( i = 0 ; i < 10 ; i++ )  {
         if  ( min > array[i]) {
             min = array[i] ;
         }
         printf("Minimum number = %d", min);
     }
}
```

Example : Selection Sort

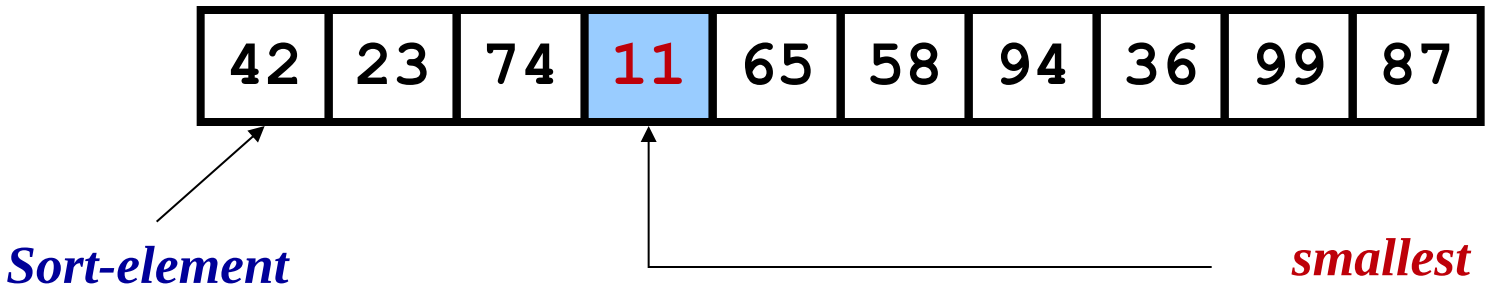
```
#include <stdio.h>                    //63
main()
{
   int  array [] = { 42, 23, 74, 11, 65, 58, 94, 36, 99, 87 } ;
   int  i, j, temp;

   for ( i = 0 ; i < 10 ; i++ ) {
      for ( j = i ; j < 10 ; j++ ) {
         if  ( array[i] > array[j] ) {
            temp =  array[j] ;
            array[j] = array[i];
            array[i] = temp;
         }
      }
      printf("%d ", array[i]);
    }
}
```

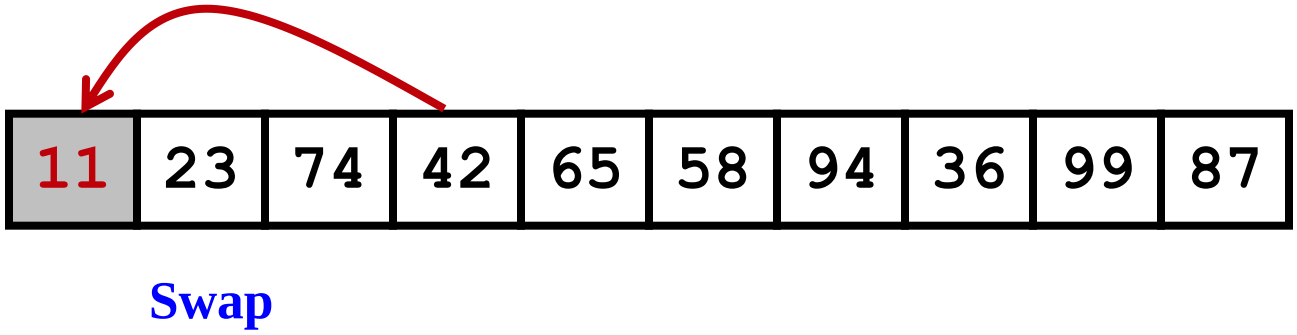


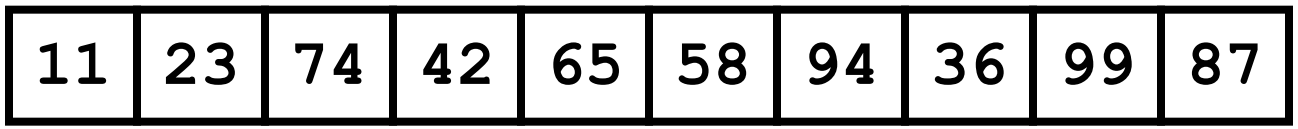
Selection Sort

- เป็นอัลกอลิทึมส์แบบง่าย ๆ ในการเรียงลำดับ
- วิธีการนี้ถูกเรียกว่า Selection Sort เนื่องจากมีการทำงานซ้ำ ๆ กันในการ ‘ selecting ‘ ค่าน้อยที่สุดที่เหลืออยู่นั่นเอง
- **การทำงาน**
  - เริ่มต้นจากตำแหน่งแรกและตำแหน่งถัดไปที่เหลือของอะเรย์ทั้งหมด เพื่อหาค่าที่น้อยที่สุด
  - นำมาสลับกับตำแหน่งแรก
  - ทำซ้ำกับค่าในตำแหน่งถัดไปภายในอะเรย์ จากตำแหน่งที่สองไปจนถึงตำแหน่งสุดท้าย

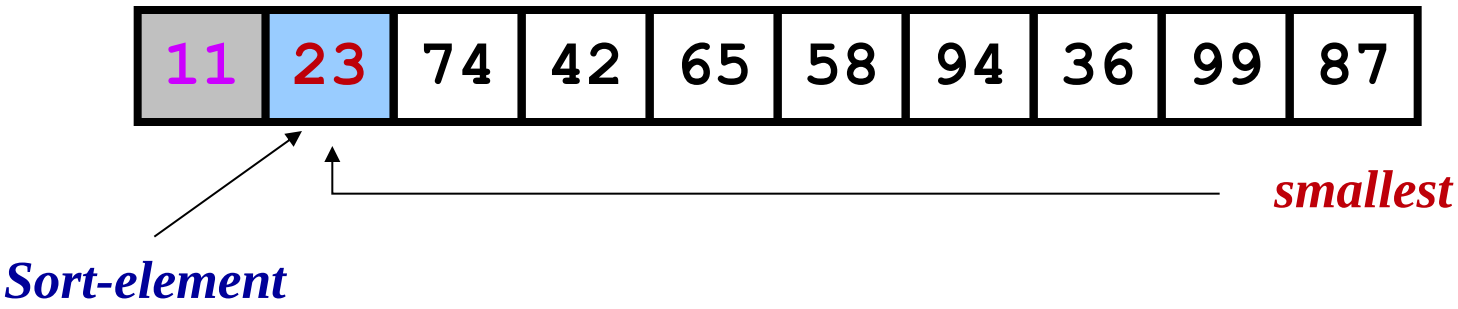
Selection Sort

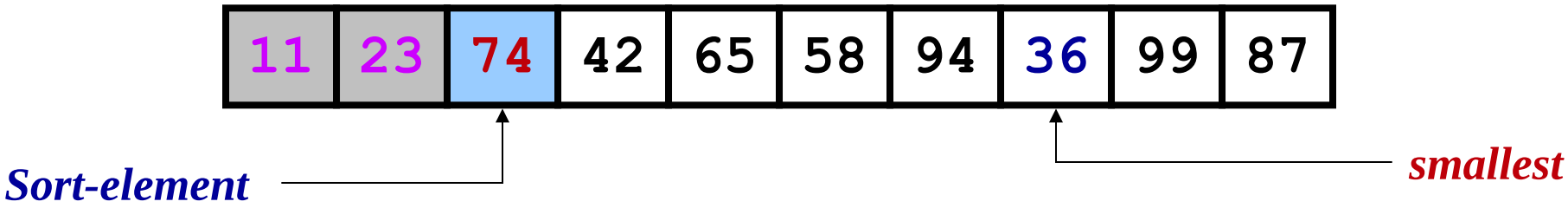


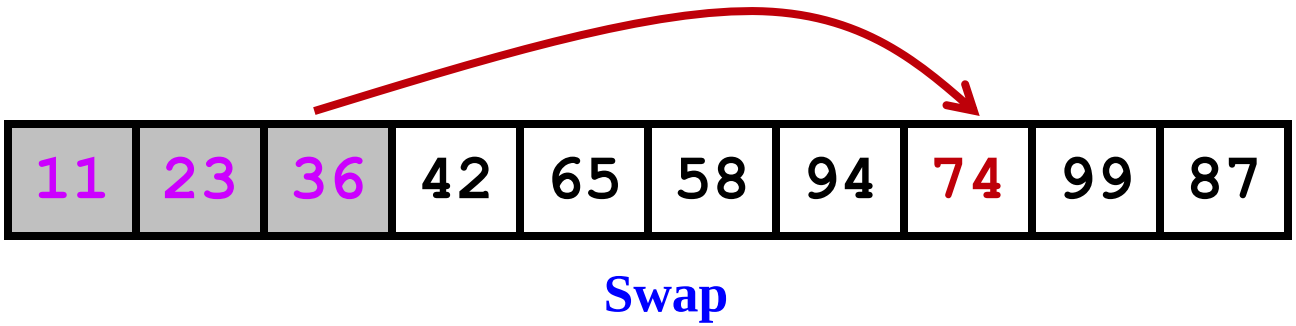




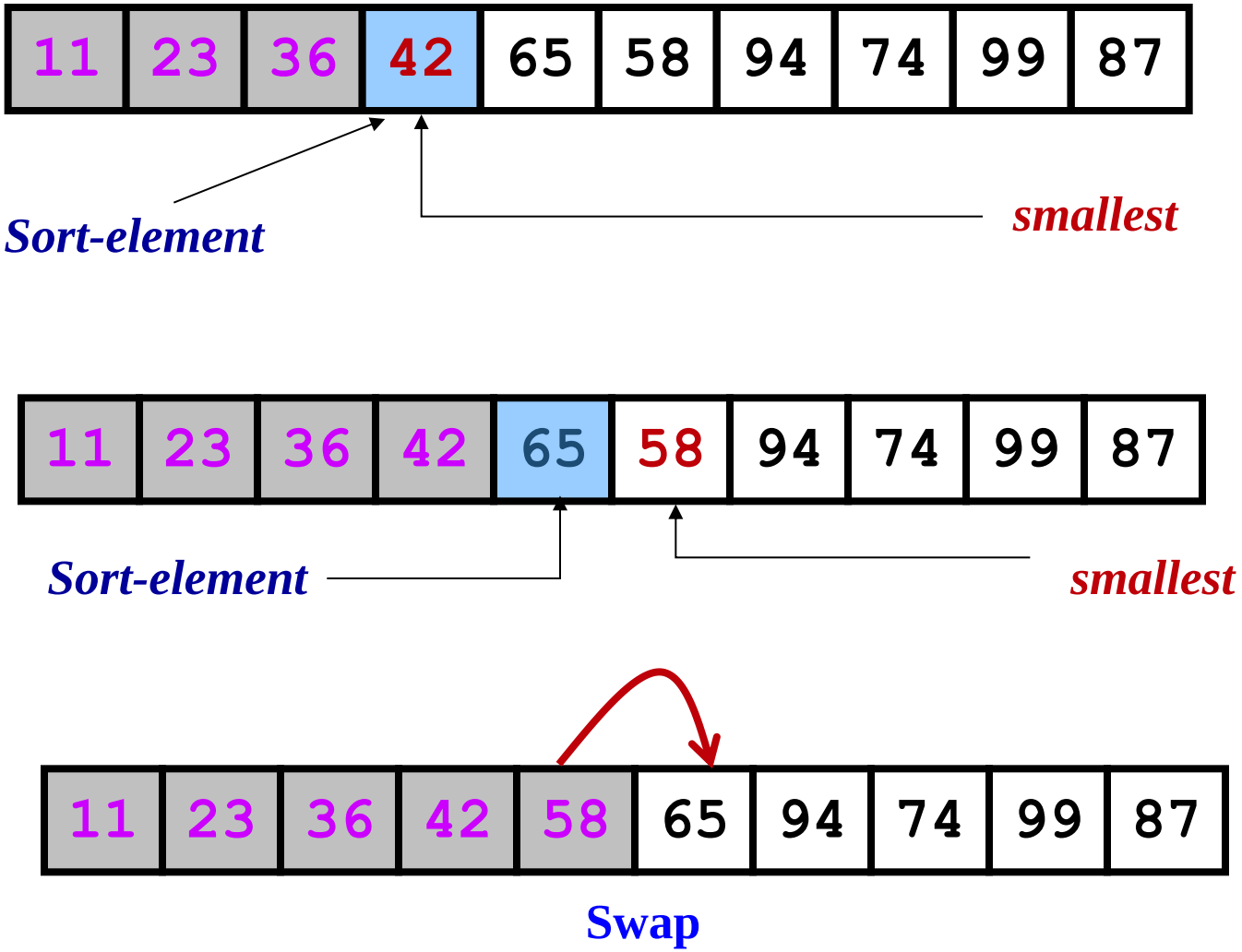
Pictorial representation of Selection Sort



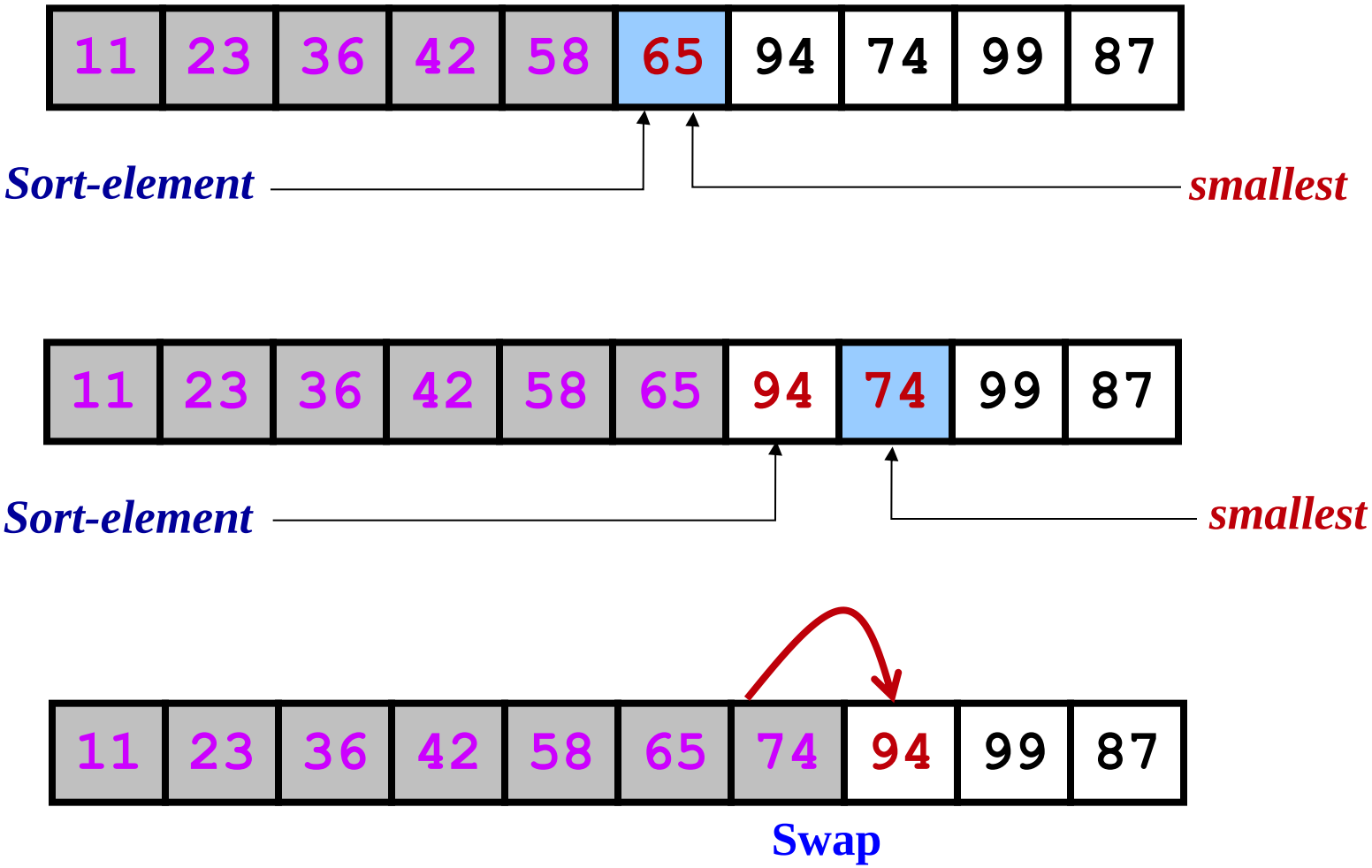




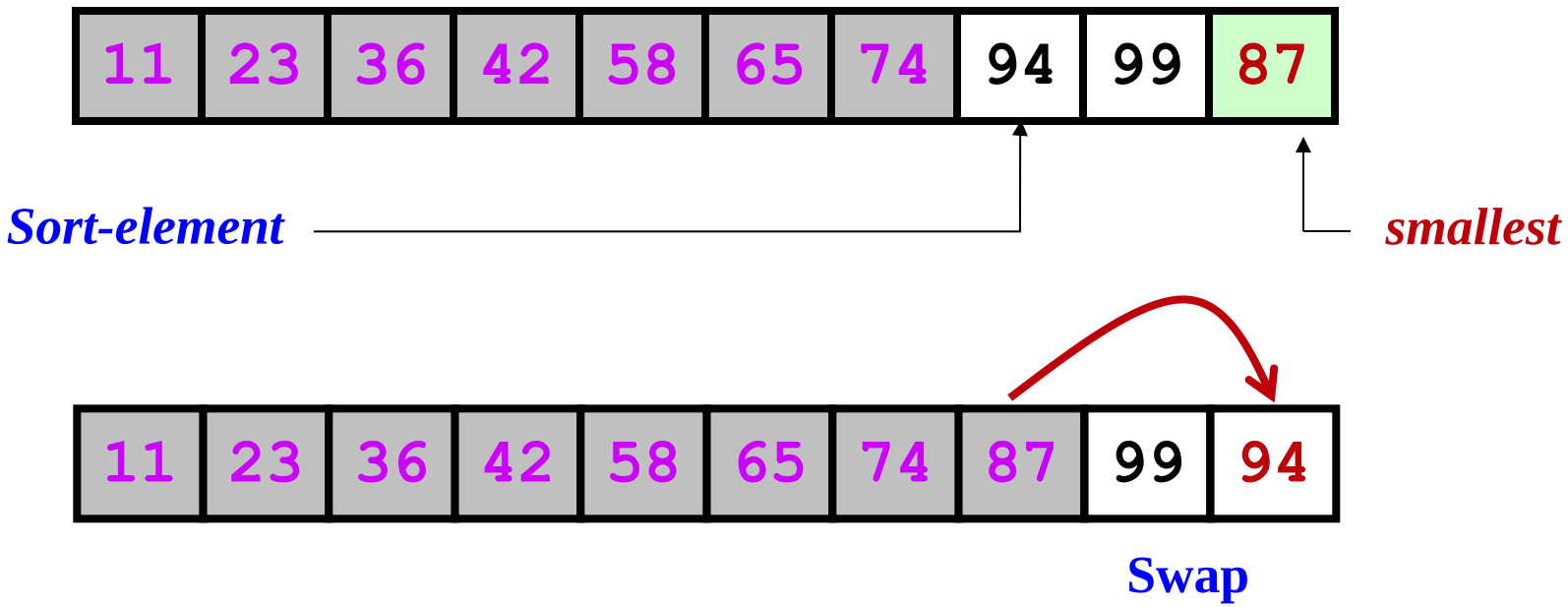
Pictorial representation of Selection Sort



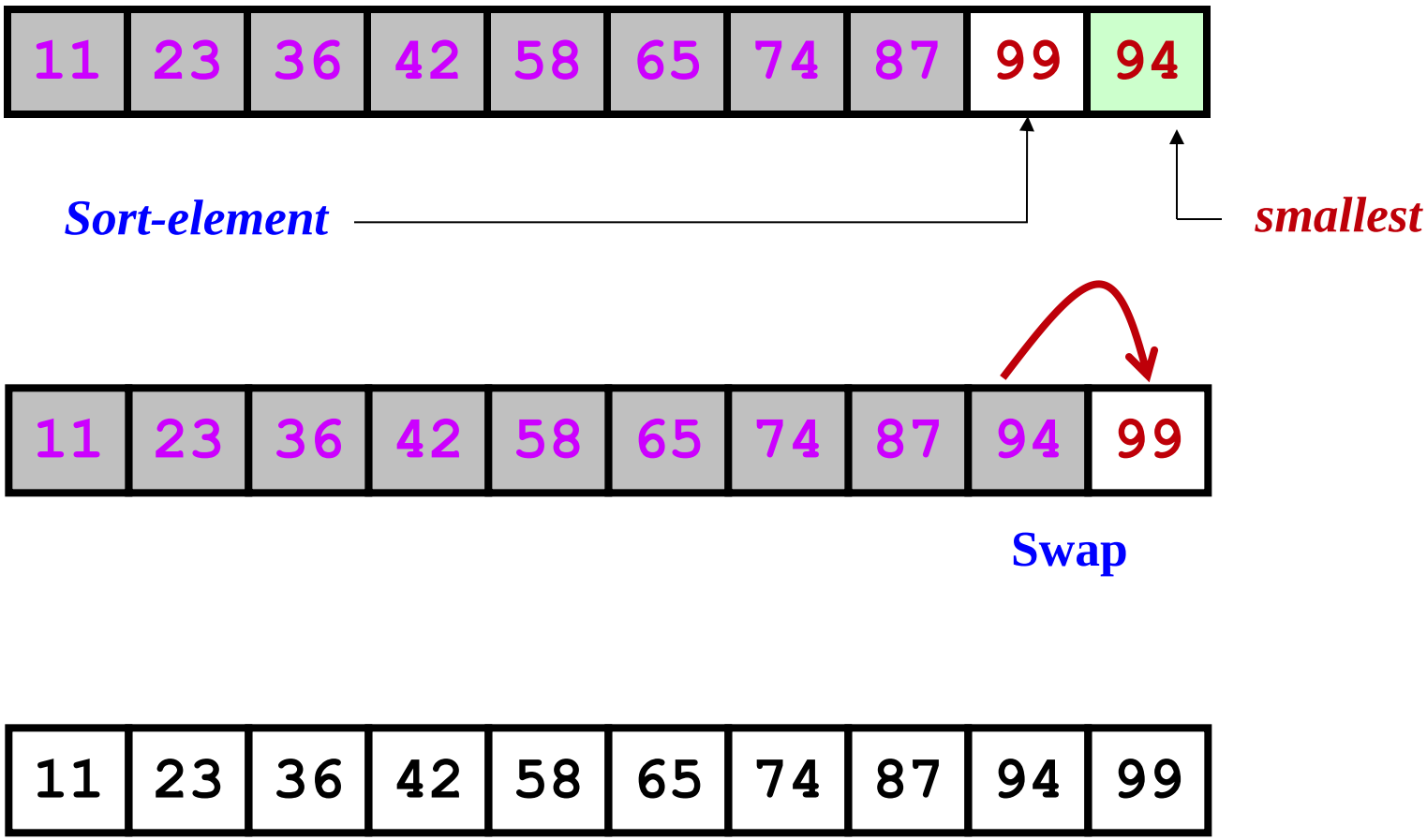
Pictorial representation of Selection Sort



Pictorial representation of Selection Sort



Pictorial representation of Selection Sort



Question