



รายงาน

เรื่อง โปรแกรม SELECTION SORT

จัดทำโดย

นายพงษ์พันธุ์ เลาวพงศ์

รหัสนักศึกษา 66543206019-2

เสนอ

อาจารย์ปิยพล ยืนยงสถาวร

รายงานนี้เป็นส่วนหนึ่งของวิชา

ENGCE124

โครงสร้างข้อมูลและขั้นตอนวิธี

(Data Structures and Algorithms)

หลักสูตร วศ.บ.วิศวกรรมคอมพิวเตอร์

สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา ภาคพายัพ เชียงใหม่

ภาคเรียนที่ 1 ปีการศึกษา 2567

รายงาน

เรื่อง โปรแกรม SELECTION SORT

จัดทำโดย

นายพงษ์พันธุ์ เลาวพงศ์

รหัสนักศึกษา 66543206019-2

เสนอ

อาจารย์ปิยพล ยืนยงสถาวร

รายงานนี้เป็นส่วนหนึ่งของวิชา

ENGCE124

โครงสร้างข้อมูลและขั้นตอนวิธี

(Data Structures and Algorithms)

หลักสูตร วศ.บ.วิศวกรรมคอมพิวเตอร์

สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา ภาคพายัพ เชียงใหม่

ภาคเรียนที่ 1 ปีการศึกษา 2567

คำนำ

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา ENGCE124 โครงสร้างข้อมูลและขั้นตอนวิธี (Data Structures and Algorithms) หลักสูตร วศ.บ. วิศวกรรมคอมพิวเตอร์ สาขาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา ภาคพายัพ เชียงใหม่ ในระดับปริญญาตรีปีที่ 2 โดยมีจุดประสงค์ในการอธิบายโค้ดของโปรแกรม SELECTION SORT รวมถึงอธิบายหลักการทำงานของโปรแกรม SELECTION SORT และอธิบายผลลัพธ์การใช้งานโปรแกรม SELECTION SORT

ผู้จัดทำรายงานหวังว่า รายงานฉบับนี้จะเป็นประโยชน์กับผู้สนใจ หรือนักศึกษาทุกท่านที่กำลังหาศึกษาในหัวข้อของโปรแกรม SELECTION SORT หากมีข้อเสนอแนะหรือข้อผิดพลาดประการใด ผู้จัดทำขอน้อมรับไว้ และขออภัยมา ณ ที่นี้

ผู้จัดทำ

นายพงษ์พันธุ์ เลาวพงศ์

วันที่ 22/09/2567

สารบัญ

	หน้า
คำนำ	ก
สารบัญ	๗
โค้ดของโปรแกรม SELECTION SORT พร้อมคำอธิบาย	1
หลักการทำงานของโปรแกรม SELECTION SORT	5
ผลลัพธ์การใช้งานโปรแกรม SELECTION SORT	12
บรรณานุกรม	14

โค้ดของโปรแกรม SELECTION SORT พร้อมคำอธิบาย

```
#include <stdio.h> // ใช้ printf
#include <conio.h> // ใช้ getch
#include <stdlib.h> // ใช้ random
#include <time.h> // ใช้ time

#define MaxData 100 // กำหนดข้อมูลสูงสุด

int Data[MaxData];

int N;

void PrepareRawData(int N)
{
    int i;

    srand(time(NULL)); // เพื่อให้ได้หมายเลขสุ่มที่แตกต่างใน rand()

    for (i=1;i<=N;i++)

        Data[i]=1+rand() % 99; // หมายเลขสุ่มที่แตกต่างระหว่าง 1..99
}

void DispData(int N,int out) // แสดงข้อมูลในรูปแบบอาร์เรย์ 2
{
    int i;

    for(i=1;i<=N;i++)

    {
        if(out>=i)

            printf("[%2d] ",Data[i]); // แสดง □ หากเป็นผลลัพธ์
```

โค้ดของโปรแกรม SELECTION SORT พร้อมคำอธิบาย (ต่อ)

```

    else

        printf("%2d  ",Data[i]); // แสดงในรูปแบบปกติหากไม่เป็นผลลัพธ์

    }

    printf("\n");

}

void swap(int a,int b)

{

    int temp;

    temp=Data[a];

    Data[a]=Data[b];

    Data[b]=temp;

}

int Minimum(int j) // ค้นหาค่าต่ำสุดใน Data[] ระหว่าง j..N

{

    int i,temp,Location;

    Location=j; // กำหนดตำแหน่งแรก

    temp=Data[j]; // กำหนดค่าเริ่มต้น

    for(i=j+1;i<=N;i++)

    {

        if(temp>Data[i])

        {

```

โค้ดของโปรแกรม SELECTION SORT พร้อมคำอธิบาย (ต่อ)

```

        temp=Data[i]; // เปลี่ยนเป็นค่าต่ำสุดใหม่

        Location=i; // เก็บตำแหน่งใหม่

    }

}

return(Location); // คืนค่าตำแหน่งของค่าต่ำสุด
} // จบฟังก์ชัน

void SelectionSort(int N)

{

    int i,j,Location;

    printf("-----\n");

    printf(" i LOC ");

    for(i=1;i<=N;i++)

        printf("(%2d) ",i);

    printf("\n      ");

    DispData(N,0); // แสดงทุกขั้นตอนของการเรียงลำดับ

    printf("-----\n");

    for(i=1;i<=N;i++)

    {

        Location=Minimum(i); // ค้นหาตำแหน่งต่ำสุดระหว่าง i..N

        swap(i,Location);

        printf("(%2d) (%2d) ",i,Location); // แสดงตำแหน่งในอาร์เรย์
    }

```

โค้ดของโปรแกรม SELECTION SORT พร้อมคำอธิบาย (ต่อ)

```

        DispData(N,i); // แสดงทุกขั้นตอนของการเรียงลำดับ
    }
} // จบฟังก์ชัน

int main()
{
    printf("ASCENDING SELECTION SORT\n");

    printf("=====\n");

    N=12; // เปลี่ยนจำนวน N ที่นี่

    PrepareRawData(N);

    printf("Raw Data...");

    DispData(N,0);

    printf("Processing Data...\n");

    SelectionSort(N);

    printf("-----\n");

    printf("Sorted Data : ");

    DispData(N,N); // ข้อมูลที่เรียงลำดับแล้ว

    getch();

    return(0);
} // จบฟังก์ชันหลัก

```


หลักการการทำงานของโปรแกรม SELECTION SORT

โปรแกรม SELECTION SORT ใช้เทคนิคการเรียงลำดับแบบเลือก (Selection Sort) เพื่อจัดเรียงข้อมูลในอาร์เรย์ โดยหลักการการทำงานแต่ละฟังก์ชันมีดังนี้

1. การนำเข้าไลบรารี

```
#include <stdio.h> // ใช้ printf
#include <conio.h> // ใช้ getch
#include <stdlib.h> // ใช้ random
#include <time.h> // ใช้ time
```

ในส่วนของการนำเข้าไลบรารี (#include) จะมีรายละเอียดดังนี้

- <stdio.h> : ไลบรารีนี้ใช้สำหรับฟังก์ชันการรับและแสดงผลข้อมูล เช่น printf() ที่ใช้ในการพิมพ์ข้อความออกทางหน้าจอ และ scanf() ที่ใช้สำหรับการรับข้อมูลจากผู้ใช้
- <conio.h> : ไลบรารีนี้ใช้ในการทำงานกับการอินพุตจากคีย์บอร์ดในรูปแบบที่ง่ายขึ้น เช่น getch() ซึ่งใช้เพื่อรอให้ผู้ใช้งานกดปุ่มก่อนที่จะดำเนินการต่อ
- <stdlib.h> : ไลบรารีนี้มีฟังก์ชันที่เกี่ยวข้องกับการจัดการหน่วยความจำ การแปลงค่า และการสุ่ม เช่น rand() ที่ใช้สำหรับสร้างค่าตัวเลขสุ่ม
- <time.h> : ไลบรารีนี้มีฟังก์ชันที่เกี่ยวข้องกับเวลาและวันที่ เช่น time() ที่ใช้เพื่อรับค่าชั่วโมง นาที และวินาทีในรูปแบบ timestamp

2. การกำหนดค่าคงที่

```
#define MaxData 100 // กำหนดข้อมูลสูงสุด
```

ในส่วนของการกำหนดค่าคงที่ จะมีรายละเอียดดังนี้

- #define MaxData 100 : การใช้คำสั่ง #define เพื่อสร้างค่าคงที่ที่ชื่อว่า MaxData ซึ่งมีค่าเท่ากับ 100 โดยที่โปรแกรมจะใช้ค่าตัวนี้ในการกำหนดขนาดสูงสุดของอาร์เรย์ Data ที่จะเก็บข้อมูล โดยการกำหนดค่าคงที่ช่วยให้โปรแกรมมีความยืดหยุ่นและง่ายต่อการปรับเปลี่ยนในอนาคต หากต้องการเปลี่ยนขนาดข้อมูล สามารถเปลี่ยนค่าที่นี้เพียงทีเดียว

3. การประกาศตัวแปร

```
int Data[MaxData];

int N;
```

ในส่วนของการประกาศตัวแปร จะมีรายละเอียดดังนี้

- `int Data[MaxData]` : ประกาศตัวแปรอาร์เรย์ที่ชื่อว่า `Data` ซึ่งมีขนาดสูงสุดตามค่าคงที่ `MaxData` ที่กำหนดไว้ (100) เพื่อเก็บข้อมูลจำนวนเต็ม (integer) ที่ถูกสร้างขึ้นจากฟังก์ชัน `PrepareRawData()`
- `int N` : ประกาศตัวแปร `N` ซึ่งใช้ในการเก็บจำนวนของข้อมูลที่ใช้ต้องการสร้างหรือเรียงลำดับในอาร์เรย์ `Data` โดยทั่วไปแล้ว `N` จะถูกกำหนดในฟังก์ชัน `main()` และสามารถปรับเปลี่ยนได้ตามต้องการ

4. ฟังก์ชัน `PrepareRawData`

```
void PrepareRawData(int N)
{
    int i;

    srand(time(NULL)); // เพื่อให้ได้หมายเลขสุ่มที่แตกต่างใน rand()

    for (i=1;i<=N;i++)

        Data[i]=1+rand() % 99; // หมายเลขสุ่มที่แตกต่างระหว่าง 1..99
}
```

ฟังก์ชัน `PrepareRawData` ทำหน้าที่สร้างข้อมูลดิบในอาร์เรย์ `Data` ขนาด `N` โดยจะทำการสุ่มหมายเลขระหว่าง 1 ถึง 99 ซึ่งขั้นตอนในการทำงานจะเริ่มจากใช้ `srand(time(NULL))` เพื่อกำหนด seed สำหรับการสุ่ม โดย seed จะถูกตั้งค่าตามเวลาปัจจุบันเพื่อให้ได้ผลลัพธ์ที่แตกต่างกันในแต่ละครั้งที่เรียกใช้งานโปรแกรม จากนั้นจะใช้ลูป `for` เพื่อทำการสุ่มและเก็บค่าที่ได้ใน `Data[i]` ตั้งแต่ 1 ถึง `N` โดยใช้ `rand() % 99` เพื่อให้ได้ค่าที่อยู่ในช่วง 0-98 แล้วบวก 1 เพื่อให้อยู่ในช่วง 1-99

5. ฟังก์ชัน DispData

```
void DispData(int N,int out) // แสดงข้อมูลในรูปแบบอาร์เรย์ 2
{
    int i;

    for(i=1;i<=N;i++)

    {

        if(out>=i)

            printf("[%2d] ",Data[i]); // แสดง [] หากเป็นผลลัพธ์

        else

            printf("%2d  ",Data[i]); // แสดงในรูปแบบปกติหากไม่เป็นผลลัพธ์

    }

    printf("\n");
}
```

ฟังก์ชัน DispData ใช้สำหรับแสดงข้อมูลในอาร์เรย์ Data โดยมีการจัดรูปแบบที่ต่างกันสำหรับค่าที่กำหนดไว้ใน out ซึ่งขั้นตอนในการทำงานจะเริ่มจากใช้ลูป for เพื่อวนผ่านแต่ละค่าในอาร์เรย์ หาก out มีค่ามากกว่าหรือเท่ากับ i จะแสดงค่าด้วยรูปแบบที่มีสัญลักษณ์ [] เพื่อบ่งบอกว่านี่คือค่าที่อยู่ในขั้นตอนการเรียงลำดับ หากไม่ใช่ จะแสดงค่าปกติ จากนั้นจะแสดงผลเป็นบรรทัดใหม่เมื่อเสร็จสิ้นการแสดงผล

6. ฟังก์ชัน swap

```
void swap(int a,int b)
{
    int temp;

    temp=Data[a];

    Data[a]=Data[b];
```

6. ฟังก์ชัน swap (ต่อ)

```
Data[b]=temp;

}
```

ฟังก์ชัน swap ทำหน้าที่สลับค่าระหว่างตำแหน่ง a และ b ในอาร์เรย์ Data ซึ่งขั้นตอนในการทำงานจะเริ่มจากใช้ตัวแปรชั่วคราว temp เพื่อเก็บค่าของ Data[a] จากนั้นทำการสลับค่าของ Data[a] กับ Data[b] โดยใช้ตัวแปรชั่วคราวนี้ ทำให้การเรียงลำดับสามารถดำเนินการได้อย่างถูกต้อง

7. ฟังก์ชัน Minimum

```
int Minimum(int j) // ค้นหาค่าต่ำสุดใน Data[] ระหว่าง j..N

{

    int i,temp,Location;

    Location=j; // กำหนดตำแหน่งแรก

    temp=Data[j]; // กำหนดค่าเริ่มต้น

    for(i=j+1;i<=N;i++)

    {

        if(temp>Data[i])

        {

            temp=Data[i]; // เปลี่ยนเป็นค่าต่ำสุดใหม่

            Location=i; // เก็บตำแหน่งใหม่

        }

    }

    return(Location); // คืนค่าตำแหน่งของค่าต่ำสุด

} // จบฟังก์ชัน
```

7. ฟังก์ชัน Minimum (ต่อ)

ฟังก์ชัน Minimum ทำหน้าที่ค้นหาค่าต่ำสุดในอาร์เรย์ Data ตั้งแต่ตำแหน่ง j จนถึง N ซึ่งขั้นตอนในการทำงานจะเริ่มจากการกำหนดตำแหน่ง Location ให้เท่ากับ j และกำหนดค่าเริ่มต้นให้กับตัวแปร temp ซึ่งเก็บค่าของ Data[j] จากนั้นจะใช้รูป for เพื่อตรวจสอบค่าตั้งแต่ j+1 ถึง N และหาค่าที่ต่ำที่สุด หากพบค่าที่ต่ำกว่า temp จะอัปเดต temp และ Location เพื่อเก็บค่าต่ำสุดและตำแหน่งที่พบ และคืนค่าตำแหน่งของค่าต่ำสุดเมื่อเสร็จสิ้นการตรวจสอบ

8. ฟังก์ชัน SelectionSort

```
void SelectionSort(int N)
{
    int i,j,Location;

    printf("-----\n");

    printf(" i LOC ");

    for(i=1;i<=N;i++)

        printf("(%2d) ",i);

    printf("\n      ");

    DispData(N,0); // แสดงทุกขั้นตอนของการเรียงลำดับ

    printf("-----\n");

    for(i=1;i<=N;i++)
    {

        Location=Minimum(i); // ค้นหาตำแหน่งต่ำสุดระหว่าง i..N

        swap(i,Location);

        printf("(%2d) (%2d) ",i,Location); // แสดงตำแหน่งในอาร์เรย์

        DispData(N,i); // แสดงทุกขั้นตอนของการเรียงลำดับ
```

8. ฟังก์ชัน SelectionSort (ต่อ)

```

    }

    } // จบฟังก์ชัน

```

ฟังก์ชัน SelectionSort เป็นฟังก์ชันหลักสำหรับการเรียงลำดับ โดยใช้เทคนิคการเรียงลำดับแบบเลือก ซึ่งขั้นตอนในการทำงานจะเริ่มจาก แสดงหัวตารางด้วยการพิมพ์ตำแหน่งและค่าที่จะเรียงลำดับ จากนั้นจะแสดงข้อมูลก่อนการเรียงลำดับ โดยจะใช้ลูป for เพื่อทำการเรียงลำดับตั้งแต่ 1 ถึง N โดยในแต่ละรอบจะเรียกใช้ฟังก์ชัน Minimum(i) เพื่อค้นหาตำแหน่งของค่าต่ำสุด จากนั้นทำการสลับค่าระหว่างตำแหน่งปัจจุบัน i และตำแหน่งที่พบค่าต่ำสุด และแสดงข้อมูลหลังจากทำการสลับในแต่ละขั้นตอน

9. ฟังก์ชัน main

```

int main()

{

    printf("ASCENDING SELECTION SORT\n");

    printf("=====\n");

    N=12; // เปลี่ยนจำนวน N ที่นี่

    PrepareRawData(N);

    printf("Raw Data...");

    DispData(N,0);

    printf("Processing Data...\n");

    SelectionSort(N);

    printf("-----\n");

    printf("Sorted Data : ");

    DispData(N,N); // ข้อมูลที่เรียงลำดับแล้ว

```

9. ฟังก์ชัน main (ต่อ)

```
    getch();  
  
    return(0);  
  
} // จบฟังก์ชันหลัก
```

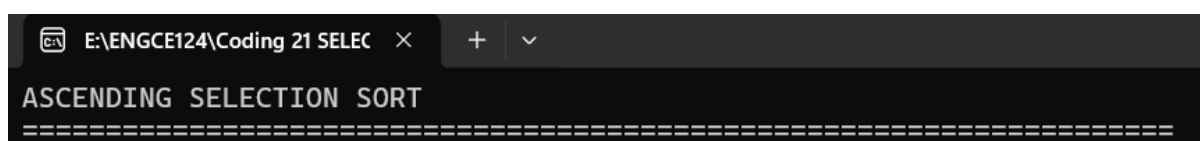
ฟังก์ชัน main เป็นจุดเริ่มต้นของโปรแกรม ซึ่งขั้นตอนในการทำงานจะเริ่มจากแสดงชื่อโปรแกรมและแบ่งหัวข้อออกเป็นบรรทัด จากนั้นกำหนดจำนวนข้อมูล N ที่จะใช้ และเรียกใช้ฟังก์ชัน PrepareRawData(N) เพื่อสร้างข้อมูลดิบ และแสดงข้อมูลดิบก่อนการเรียงลำดับ จากนั้นก็จะเรียกใช้ฟังก์ชัน SelectionSort(N) เพื่อทำการเรียงลำดับข้อมูล และแสดงผลลัพธ์สุดท้ายของข้อมูลที่เรียงลำดับแล้ว

ผลลัพธ์การใช้งานโปรแกรม SELECTION SORT

โปรแกรมการเรียงลำดับแบบเลือกจากน้อยไปมากนี้ถูกออกแบบมาเพื่อให้ผู้ใช้สามารถสร้างข้อมูลสุ่มและทำการเรียงลำดับข้อมูล ในส่วนนี้จะอธิบายการทำงานและการแสดงผลลัพธ์ของโปรแกรมอย่างละเอียด

1. การเริ่มต้นโปรแกรม

เมื่อผู้รันโปรแกรม จะมีการแสดงข้อความ "ASCENDING SELECTION SORT" บนหน้าจอ ซึ่งเป็นชื่อของโปรแกรม ต่อมาโปรแกรมจะแสดงเส้นแบ่งด้วยเครื่องหมาย "=" เพื่อแยกส่วนของข้อความแนะนำออกจากผลลัพธ์ที่ตามมา



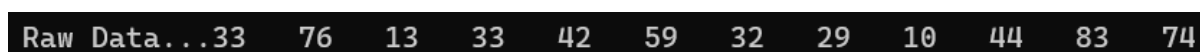
```
E:\ENGCE124\Coding 21 SELEC  x  +  v
ASCENDING SELECTION SORT
=====
```

2. การกำหนดขนาดของข้อมูล

โปรแกรมกำหนดค่าของ N ซึ่งบ่งบอกจำนวนข้อมูลที่ต้องการสร้าง โดยในกรณีนี้มีการกำหนด N=12 หมายความว่า จะมีการสุ่มสร้างข้อมูล 12 ชุด

3. การสร้างและแสดงข้อมูลดิบ

โปรแกรมเรียกใช้ฟังก์ชัน PrepareRawData(N) เพื่อสุ่มหมายเลขระหว่าง 1 ถึง 99 และเก็บในอาร์เรย์ Data หลังจากนั้นจะมีการแสดงผลข้อความ "Raw Data..." เพื่อบ่งบอกว่าข้อมูลดิบที่สุ่มได้จะแสดงต่อไป โดยโปรแกรมจะแสดงผลข้อมูลในรูปแบบของอาร์เรย์ โดยใช้ฟังก์ชัน DispData(N, 0) ซึ่งจะแสดงข้อมูลทั้งหมดในอาร์เรย์ Data โดยไม่มีสัญลักษณ์พิเศษ เนื่องจากข้อมูลเหล่านี้ยังไม่ได้ถูกเรียงลำดับ



```
Raw Data...33 76 13 33 42 59 32 29 10 44 83 74
```

4. การประมวลผลข้อมูล

โปรแกรมจะแสดงข้อความ "Processing Data..." เพื่อบ่งบอกว่ากำลังดำเนินการเรียงลำดับข้อมูล ซึ่งเรียกใช้ฟังก์ชัน SelectionSort(N) ซึ่งจะทำการเรียงลำดับข้อมูลในอาร์เรย์ Data โดยใช้วิธีการเลือก ในระหว่างที่โปรแกรมทำการเรียงลำดับ ข้อมูลจะถูกแสดงออกมาหลังจากแต่ละขั้นตอน โดยโปรแกรมจะพิมพ์ตารางระบุหมายเลขตำแหน่งในอาร์เรย์และตำแหน่งที่ต่ำสุดที่ค้นพบในแต่ละรอบ จากนั้นข้อมูลจะถูกอัปเดตในแต่ละรอบของการเรียงลำดับ ซึ่งจะมีการแสดงข้อมูลที่ถูกสลับพร้อมกับหมายเลขตำแหน่งที่ได้ทำการเปลี่ยนแปลง

i	LOC	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
		33	76	13	33	42	59	32	29	10	44	83	74
(1)	(9)	[10]	76	13	33	42	59	32	29	33	44	83	74
(2)	(3)	[10]	[13]	76	33	42	59	32	29	33	44	83	74
(3)	(8)	[10]	[13]	[29]	33	42	59	32	76	33	44	83	74
(4)	(7)	[10]	[13]	[29]	[32]	42	59	33	76	33	44	83	74
(5)	(7)	[10]	[13]	[29]	[32]	[33]	59	42	76	33	44	83	74
(6)	(9)	[10]	[13]	[29]	[32]	[33]	[33]	42	76	59	44	83	74
(7)	(7)	[10]	[13]	[29]	[32]	[33]	[33]	[42]	76	59	44	83	74
(8)	(10)	[10]	[13]	[29]	[32]	[33]	[33]	[42]	[44]	59	76	83	74
(9)	(9)	[10]	[13]	[29]	[32]	[33]	[33]	[42]	[44]	[59]	76	83	74
(10)	(12)	[10]	[13]	[29]	[32]	[33]	[33]	[42]	[44]	[59]	[74]	83	76
(11)	(12)	[10]	[13]	[29]	[32]	[33]	[33]	[42]	[44]	[59]	[74]	[76]	83
(12)	(12)	[10]	[13]	[29]	[32]	[33]	[33]	[42]	[44]	[59]	[74]	[76]	[83]

5. การแสดงผลข้อมูลที่เรียงลำดับ

หลังจากที่โปรแกรมได้ทำการเรียงลำดับเสร็จสิ้น จะมีการแสดงเส้นแบ่งด้วยเครื่องหมาย "-" จากนั้นโปรแกรมจะแสดงข้อความ "Sorted Data :." เพื่อบ่งบอกว่าข้อมูลที่แสดงต่อไปนี้เป็นผลลัพธ์ที่เรียงลำดับแล้ว โปรแกรมจะแสดงข้อมูลที่เรียงลำดับในอาร์เรย์ Data โดยใช้ฟังก์ชัน DispData(N, N) ซึ่งจะทำให้ค่าที่อยู่ในตำแหน่งสุดท้ายมีสัญลักษณ์ □ เพื่อบ่งบอกว่าข้อมูลเหล่านี้คือผลลัพธ์สุดท้ายที่ถูกเรียงลำดับ

```

E:\ENGCE124\Coding 21 SELEC  x  +  v
ASCENDING SELECTION SORT
=====
Raw Data...33  76  13  33  42  59  32  29  10  44  83  74
Processing Data...
-----
i LOC ( 1) ( 2) ( 3) ( 4) ( 5) ( 6) ( 7) ( 8) ( 9) (10) (11) (12)
      33  76  13  33  42  59  32  29  10  44  83  74
-----
( 1) ( 9) [10] 76  13  33  42  59  32  29  33  44  83  74
( 2) ( 3) [10] [13] 76  33  42  59  32  29  33  44  83  74
( 3) ( 8) [10] [13] [29] 33  42  59  32  76  33  44  83  74
( 4) ( 7) [10] [13] [29] [32] 42  59  33  76  33  44  83  74
( 5) ( 7) [10] [13] [29] [32] [33] 59  42  76  33  44  83  74
( 6) ( 9) [10] [13] [29] [32] [33] [33] 42  76  59  44  83  74
( 7) ( 7) [10] [13] [29] [32] [33] [33] [42] 76  59  44  83  74
( 8) (10) [10] [13] [29] [32] [33] [33] [42] [44] 59  76  83  74
( 9) ( 9) [10] [13] [29] [32] [33] [33] [42] [44] [59] 76  83  74
(10) (12) [10] [13] [29] [32] [33] [33] [42] [44] [59] [74] 83  76
(11) (12) [10] [13] [29] [32] [33] [33] [42] [44] [59] [74] [76] 83
(12) (12) [10] [13] [29] [32] [33] [33] [42] [44] [59] [74] [76] [83]
-----
Sorted Data : [10] [13] [29] [32] [33] [33] [42] [44] [59] [74] [76] [83]
-----
Process exited after 1.612 seconds with return value 0
Press any key to continue . . . |

```

บรรณานุกรม

ChatGPT. (-). Exploring Selection Sort: Functionality and Results. สืบค้น 22 กันยายน 2567,
จาก <https://chatgpt.com/>