



รายงาน

เรื่อง โปรแกรม BINARY SEARCH

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 BINARY SEARCH

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 BINARY SEARCH รวมถึงอธิบายหลักการทำงานของโปรแกรม BINARY SEARCH และอธิบายผลลัพธ์การใช้งานโปรแกรม BINARY SEARCH

ผู้จัดทำรายงานหวังว่า รายงานฉบับนี้จะเป็นประโยชน์กับผู้สนใจ หรือนักศึกษาทุกท่านที่กำลังหาศึกษาในหัวข้อของโปรแกรม BINARY SEARCH หากมีข้อเสนอแนะหรือข้อผิดพลาดประการใด ผู้จัดทำขอน้อมรับไว้ และขออภัยมา ณ ที่นี้

ผู้จัดทำ

นายพงษ์พันธุ์ เลาหวงศ์

วันที่ 28/09/2567

สารบัญ

	หน้า
คำนำ	ก
สารบัญ	๗
โค้ดของโปรแกรม BINARY SEARCH พร้อมคำอธิบาย	1
หลักการทำงานของโปรแกรม BINARY SEARCH	7
ผลลัพธ์การใช้งานโปรแกรม BINARY SEARCH	16
บรรณานุกรม	18

โค้ดของโปรแกรม BINARY SEARCH พร้อมคำอธิบาย

```
#include <stdio.h> // ใช้ฟังก์ชัน printf
#include <conio.h> // ใช้ฟังก์ชัน getch
#include <stdlib.h> // ใช้ฟังก์ชันสุ่ม random
#include <time.h> // ใช้ฟังก์ชันจัดการเวลา time
#include <windows.h> // ใช้ฟังก์ชันเสียง Sound

#define MaxData 100 // กำหนดจำนวนข้อมูลสูงสุด

int Data[MaxData];

int N,key,Times;

bool result;

bool Duplicate(int i,int Data1) // ตรวจสอบข้อมูลซ้ำ
{
    int j;

    for(j=1;j<=i;j++)
    {
        if(Data1==Data[j])
            return(true);
    }

    return(false);
}

void PrepareRawKey(int N)
{

```

โค้ดของโปรแกรม BINARY SEARCH พร้อมคำอธิบาย (ต่อ)

```

int i,j,temp;

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

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

{

    temp=(rand() % 89)+10; // สุ่มค่าตัวเลข 10..99

    while(Duplicate(i-1,temp)) // ลูปหากยังซ้ำอยู่

        temp=(rand() % 89)+10; // สุ่มอีกครั้ง

    Data[i]=temp; // เก็บหมายเลขใหม่

} // จบ for

} // จบฟังก์ชัน

void DispKey(int N)

{

    int i;

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

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

    printf("\n");

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

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

    printf("\n");

}

```

โค้ดของโปรแกรม BINARY SEARCH พร้อมคำอธิบาย (ต่อ)

```

void BubbleSort(int N) // จัดเรียงข้อมูลจากน้อยไปมาก
{
    int i,j,temp;
    for(i=1;i<=N-1;i++) // ลูปไปข้างหน้า
    {
        if(Data[i]>Data[i+1]) // ถ้าข้อมูลไม่อยู่ในตำแหน่งที่ถูกต้อง
        {
            j=i+1; // ลูปถอยหลัง
            while(Data[j]<Data[j-1]) // ลูปหากยังมีการสลับข้อมูล
            {
                temp=Data[j-1]; // สลับข้อมูล
                Data[j-1]=Data[j];
                Data[j]=temp;
                j--; // นับ j ลง
            } // จบ while
        } // จบ if
    } // จบ for

} // จบฟังก์ชัน

bool BinarySearch(int Key1)
{
    int L,R,Mid;

```

โค้ดของโปรแกรม BINARY SEARCH พร้อมคำอธิบาย (ต่อ)

```
L=1;

R=N;

Times=0; // กำหนดค่าเริ่มต้นสำหรับการค้นหา

while(L<=R)

{

    Mid=(L+R)/2; // คำนวณค่ากลาง

    Times++; // นับจำนวนครั้งที่ค้นหา

    printf("L : %2d ",L);

    printf("R : %2d ",R);

    printf("Mid : %2d ",Mid);

    printf("Searching Time : %d\n",Times);

    if(Key1==Data[Mid])

        return(true); // ถ้าพบ

    else

    {

        if(Key1<Data[Mid])

            R=Mid-1; // ย้าย R

        else

            L=Mid+1; // ย้าย L

    }

} // จบ while
```


โค้ดของโปรแกรม BINARY SEARCH พร้อมคำอธิบาย (ต่อ)

```

    return(false); // ถ้าไม่พบ

} // จบฟังก์ชัน

int main()

{

    printf("BINARY SEARCH\n");

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

    N=32;

    PrepareRawKey(N);

    printf("Raw key :\n");

    DispKey(N); // แสดงข้อมูลดิบ

    BubbleSort(N);

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

    while(key!=-999)

    {

        printf("Sorted Key :\n");

        DispKey(N); // แสดงข้อมูลที่จัดเรียงแล้ว

        printf("\nEnter Key for Search(-999 for EXIT) = ");

        scanf("%d",&key); // อ่านค่าคีย์จากคีย์บอร์ด

        if(key!=-999)

        {

            result=BinarySearch(key); // เรียกฟังก์ชัน Binary Search

```

โค้ดของโปรแกรม BINARY SEARCH พร้อมคำอธิบาย (ต่อ)

```
if(result)

    printf("Result...FOUND\n"); // ถ้าพบ

else

{

    Beep(600,600);

    printf("Result...NOT FOUND!!\n"); // ถ้าไม่พบ

}

//printf("Searching Time : %d\n",Times);

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

} // จบ if

} // จบ while

return(0);

} // จบ main
```

หลักการทำงานของโปรแกรม BINARY SEARCH

โปรแกรมข้างต้นใช้งานอัลกอริทึม Binary Search (การค้นหาแบบทวิภาค) ในการค้นหาข้อมูลในอาร์เรย์ที่ถูกจัดเรียง (เรียงจากน้อยไปมากหรือมากไปน้อย) โดยอัลกอริทึมนี้มีความรวดเร็วและมีประสิทธิภาพสูงกว่าการค้นหาแบบปกติ (Linear Search) เพราะมันใช้หลักการแบ่งครึ่งขอบเขตของการค้นหาในแต่ละรอบ ทำให้จำนวนข้อมูลที่ต้องตรวจสอบลดลงอย่างมากในแต่ละขั้นตอน

1. การประกาศและใช้งานไลบรารี

```
#include <stdio.h> // ใช้ฟังก์ชัน printf
#include <conio.h> // ใช้ฟังก์ชัน getch
#include <stdlib.h> // ใช้ฟังก์ชันสุ่ม random
#include <time.h> // ใช้ฟังก์ชันจัดการเวลา time
#include <windows.h> // ใช้ฟังก์ชันเสียง Sound
```

ในส่วนการประกาศและใช้งานไลบรารี มีรายละเอียดดังนี้

- `#include <stdio.h>` : ไลบรารี `<stdio.h>` เป็นไลบรารีมาตรฐานของภาษา C ที่เกี่ยวข้องกับการจัดการอินพุตและเอาต์พุต (I/O) ซึ่งฟังก์ชันที่ใช้จากไลบรารีนี้คือ `printf()` ซึ่งใช้ในการพิมพ์ข้อความหรือแสดงข้อมูลออกทางหน้าจอคอนโซล
- `#include <conio.h>` : ไลบรารี `<conio.h>` เป็นไลบรารีที่ใช้ในระบบปฏิบัติการ Windows ซึ่งจัดการการรับอินพุตจากคีย์บอร์ดโดยตรง ซึ่งฟังก์ชันที่ใช้จากไลบรารีนี้คือ `getch()` ซึ่งใช้ในการรอรับการกดปุ่มจากผู้ใช้ โดยไม่ต้องแสดงตัวอักษรที่กดบนหน้าจอ (ใช้มักใช้หยุดรอให้ผู้ใช้กดคีย์)
- `#include <stdlib.h>` : ไลบรารี `<stdlib.h>` เป็นไลบรารีมาตรฐานที่มีฟังก์ชันต่าง ๆ มากมาย รวมถึงฟังก์ชันสุ่มตัวเลขเช่น `rand()` และการจองหน่วยความจำ ซึ่งฟังก์ชันที่ใช้จากไลบรารีนี้คือ `rand()` ซึ่งใช้ในการสร้างตัวเลขสุ่ม นอกจากนี้ยังสามารถใช้ฟังก์ชันอื่น ๆ เช่น `exit()` เพื่อออกจากโปรแกรม
- `#include <time.h>` : ไลบรารี `<time.h>` ใช้สำหรับจัดการเกี่ยวกับเวลา โดยเฉพาะการใช้เวลาเป็นค่าอ้างอิงในการสุ่มตัวเลขให้แตกต่างกันในแต่ละครั้งที่รันโปรแกรม ซึ่งฟังก์ชันที่ใช้จากไลบรารีนี้คือ `time()` ซึ่งใช้ในการดึงค่าเวลาปัจจุบันจากระบบมาใช้เป็นค่า seed ในการสุ่มตัวเลข เพื่อให้ตัวเลขที่สุ่มจาก `rand()` ไม่ซ้ำกันในการรันแต่ละครั้ง

- `#include <windows.h>` : ไลบรารี `<windows.h>` เป็นไลบรารีของระบบปฏิบัติการ Windows ที่ให้ฟังก์ชันการทำงานหลากหลาย รวมถึงการจัดการกับระบบเสียง การแสดงผล กราฟิก และระบบไฟล์ ซึ่งฟังก์ชันที่ใช้จากไลบรารีนี้ในโค้ดคือ `Beep()` ซึ่งเป็นฟังก์ชันที่ใช้ส่งเสียงเตือนผ่านลำโพงคอมพิวเตอร์ โดยกำหนดความถี่และระยะเวลาของเสียงได้

2. การประกาศค่าคงที่ (Constant Definitions)

```
#define MaxData 100 // กำหนดจำนวนข้อมูลสูงสุด
```

ในส่วนการประกาศค่าคงที่มีรายละเอียดดังนี้

- `#define MaxData 100` : คำสั่ง `#define` ใช้ในการกำหนดค่าคงที่ในโปรแกรม ซึ่งในที่นี้กำหนด `MaxData` มีค่าเท่ากับ 100 หมายถึงอาร์เรย์ `Data[]` จะมีขนาด 100 ตำแหน่ง ซึ่งเป็นจำนวนข้อมูลสูงสุดที่โปรแกรมสามารถจัดการได้ เมื่อกำหนดค่าคงที่แบบนี้แล้ว ทุกครั้งที่ใช้ `MaxData` ในโปรแกรม จะหมายถึงค่าที่กำหนดไว้คือ 100

3. การประกาศตัวแปร

```
int Data[MaxData];

int N,key,Times;

bool result;
```

ในส่วนการประกาศตัวแปร มีรายละเอียดดังนี้

- `int Data[MaxData]` : ประกาศตัวแปร `Data[]` ซึ่งเป็นอาร์เรย์ที่มีขนาดตามค่าคงที่ `MaxData` หรือ 100 ตำแหน่ง โดยตัวแปรนี้จะถูกใช้เก็บข้อมูลที่ส่งเข้ามาในโปรแกรมเพื่อทำการจัดเรียงและค้นหาภายหลัง
- `int N,key,Times` : ประกาศตัวแปรตัวเลขจำนวนเต็ม (Integer) 3 ตัว ได้แก่
 - `N`: ใช้เก็บจำนวนข้อมูลที่ถูกสร้างหรือใช้งานในโปรแกรม (ไม่จำเป็นต้องใช้เต็มจำนวน 100)
 - `key`: ใช้เก็บค่าคีย์ (หรือข้อมูล) ที่ผู้ใช้ป้อนเข้ามาเพื่อทำการค้นหา
 - `Times`: ใช้เก็บจำนวนครั้งที่ใช้ในการค้นหาในกระบวนการ Binary Search เพื่อแสดงผลให้ผู้ใช้งานเห็นว่าต้องใช้จำนวนครั้งในการค้นหามากน้อยแค่ไหน

- bool result : ประกาศตัวแปร result ซึ่งเป็นตัวแปรแบบ Boolean ที่มีค่าเป็น true หรือ false โดยตัวแปรนี้จะใช้เก็บผลลัพธ์จากการค้นหาในฟังก์ชัน Binary Search ว่าค้นหาข้อมูลสำเร็จหรือไม่ ถ้าหากพบข้อมูลที่ต้องการค้นหา result จะมีค่าเป็น true แต่ถ้าไม่พบ result จะมีค่าเป็น false

4. ฟังก์ชัน Duplicate

```
bool Duplicate(int i,int Data1) // ตรวจสอบข้อมูลซ้ำ
{
    int j;

    for(j=1;j<=i;j++)
    {
        if(Data1==Data[j])

            return(true);
    }

    return(false);
}
```

ฟังก์ชัน Duplicate มีหน้าที่ในการตรวจสอบว่ามีข้อมูลซ้ำหรือไม่ โดยจะตรวจสอบว่าเลขที่สุ่มได้ (Data1) มีอยู่ในอาร์เรย์ Data[] ในตำแหน่งก่อนหน้า (i) หรือไม่ ถ้ามีข้อมูลซ้ำ ฟังก์ชันจะส่งค่ากลับเป็น true เพื่อให้ทราบว่าเกิดการซ้ำ ถ้าไม่ซ้ำจะส่งค่ากลับเป็น false โดยหลักการทำงานเริ่มต้นจาก วนลูปผ่านอาร์เรย์ Data[] จากตำแหน่งที่ 1 ถึงตำแหน่ง i เพื่อเปรียบเทียบว่า Data1 มีค่าเท่ากับค่าที่เก็บใน Data[j] หรือไม่ ถ้าพบว่ามีค่าซ้ำกัน ฟังก์ชันจะส่งค่ากลับเป็น true เพื่อบอกว่ามีข้อมูลซ้ำ แต่ถ้าไม่พบค่าซ้ำ ฟังก์ชันจะวนลูปจนจบและส่งค่ากลับเป็น false

5. ฟังก์ชัน PrepareRawKey

```
void PrepareRawKey(int N)
{
    int i,j,temp;
```

5. ฟังก์ชัน PrepareRawKey (ต่อ)

```

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

for (i=1;i<=N;i++)
{
    temp=(rand() % 89)+10; // สุ่มค่าตัวเลข 10..99

    while(Duplicate(i-1,temp)) // ลูปหากยังซ้ำอยู่

    temp=(rand() % 89)+10; // สุ่มอีกครั้ง

    Data[i]=temp; // เก็บหมายเลขใหม่
} // จบ for
} // จบฟังก์ชัน

```

ฟังก์ชัน PrepareRawKey ทำหน้าที่สร้างข้อมูลสุ่มที่ไม่ซ้ำกัน (ค่าในช่วง 10 ถึง 99) ลงในอาร์เรย์ Data[] โดยจำนวนข้อมูลจะขึ้นอยู่กับค่าของ N ที่ส่งเข้ามา โดยการทำงานของฟังก์ชันนี้มีการใช้ฟังก์ชัน Duplicate() เพื่อให้แน่ใจว่าไม่มีการสุ่มตัวเลขซ้ำ โดยหลักการทำงานเริ่มจากใช้ฟังก์ชัน srand(time(NULL)) เพื่อทำให้การสุ่มข้อมูลใน rand() ได้ผลลัพธ์ที่แตกต่างกันในแต่ละครั้งที่เรียกใช้โปรแกรม จากนั้นจะวนลูปตั้งแต่ 1 ถึง N เพื่อสุ่มตัวเลขที่แตกต่างกัน โดยจะสุ่มตัวเลขในช่วง 10 ถึง 99 ด้วยการใช้ rand() % 89 + 10 เมื่อสุ่มตัวเลขเสร็จจะมีการตรวจสอบว่าตัวเลขที่สุ่มได้ซ้ำกับข้อมูลที่สุ่มมาก่อนหน้านี้หรือไม่ โดยใช้ฟังก์ชัน Duplicate() โดยถ้าตัวเลขซ้ำจะสุ่มใหม่จนกว่าจะได้ตัวเลขที่ไม่ซ้ำแล้วเก็บไว้ในอาร์เรย์ Data[]

6. ฟังก์ชัน DispKey

```

void DispKey(int N)
{
    int i;

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

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

    printf("\n");
}

```

6. ฟังก์ชัน DispKey (ต่อ)

```

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

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

printf("\n");

}

```

ฟังก์ชัน DispKey ทำหน้าที่แสดงข้อมูลในอาร์เรย์ Data[] พร้อมกับแสดงลำดับของข้อมูลในรูปแบบตาราง โดยจะแสดงทั้งลำดับ (Subscript) และค่าที่เก็บในแต่ละช่องของอาร์เรย์ โดยหลักการทำงานเริ่มต้นโดยการวนลูปแสดงลำดับของข้อมูลจาก 1 ถึง N จากนั้นจะวนลูปแสดงค่าของข้อมูลในอาร์เรย์ Data[] ที่ตรงกับลำดับนั้น ๆ และขึ้นบรรทัดใหม่หลังจากแสดงข้อมูลทั้งหมด

7. ฟังก์ชัน BubbleSort

```

void BubbleSort(int N) // จัดเรียงข้อมูลจากน้อยไปมาก

{

    int i,j,temp;

    for(i=1;i<=N-1;i++) // ลูปไปข้างหน้า

    {

        if(Data[i]>Data[i+1]) // ถ้าข้อมูลไม่อยู่ในตำแหน่งที่ถูกต้อง

        {

            j=i+1; // ลูปถอยหลัง

            while(Data[j]<Data[j-1]) // ลูปหากยังมีการสลับข้อมูล

            {

                temp=Data[j-1]; // สลับข้อมูล

                Data[j-1]=Data[j];

                Data[j]=temp;

```

7. ฟังก์ชัน BubbleSort (ต่อ)

```

        j--; // นับ j ลง

    } // จบ while

    } // จบ if

} // จบ for

} // จบฟังก์ชัน

```

ฟังก์ชัน BubbleSort ใช้วิธีการ Bubble Sort เพื่อจัดเรียงข้อมูลในอาร์เรย์ Data[] จากน้อยไปมาก โดยการ ทำงานของ Bubble Sort จะเปรียบเทียบข้อมูลที่อยู่ติดกัน และสลับตำแหน่งหากข้อมูลอยู่ผิดลำดับ จนกว่า ข้อมูลทั้งหมดจะถูกจัดเรียงเรียบร้อย โดยหลักการทำงานเริ่มต้นโดยวนลูปจากตำแหน่งที่ 1 ถึง N-1 ซึ่งในแต่ละ รอบของลูป จะตรวจสอบว่าข้อมูลในตำแหน่ง Data[i] มากกว่าข้อมูลในตำแหน่ง Data[i+1] หรือไม่ ถ้า มากกว่าให้สลับข้อมูลกัน หลังจากสลับข้อมูลแล้ว จะวนลูปถอยหลังเพื่อตรวจสอบและสลับข้อมูลที่ยังอยู่ผิด ตำแหน่งต่อไปจนกว่าจะจัดเรียงเสร็จ

8. ฟังก์ชัน BinarySearch

```

bool BinarySearch(int Key1)

{

    int L,R,Mid;

    L=1;

    R=N;

    Times=0; // กำหนดค่าเริ่มต้นสำหรับการค้นหา

    while(L<=R)

    {

        Mid=(L+R)/2; // คำนวณค่ากลาง

        Times++; // นับจำนวนครั้งที่ค้นหา
    }
}

```


8. ฟังก์ชัน BinarySearch (ต่อ)

```

printf("L : %2d ",L);

printf("R : %2d ",R);

printf("Mid : %2d ",Mid);

printf("Searching Time : %d\n",Times);

if(Key1==Data[Mid])

    return(true); // ถ้าพบ

else

{

    if(Key1<Data[Mid])

        R=Mid-1; // ย้าย R

    else

        L=Mid+1; // ย้าย L

}

} // จบ while

return(false); // ถ้าไม่พบ

} // จบฟังก์ชัน

```

ฟังก์ชัน BinarySearch ทำหน้าที่ค้นหาค่าที่ผู้ใช้ระบุ (คีย์) ในอาร์เรย์ Data[] ที่ถูกจัดเรียงแล้ว โดยใช้วิธี Binary Search วิธีนี้มีการแบ่งครึ่งอาร์เรย์ซ้ำ ๆ เพื่อลดจำนวนการเปรียบเทียบ ทำให้สามารถค้นหาได้อย่างรวดเร็ว โดยหลักการทำงานเริ่มต้นกำหนดตัวแปร L และ R เพื่อเก็บขอบเขตของการค้นหา โดย L เริ่มต้นที่ 1 และ R เริ่มต้นที่ N จากนั้นจะวนลูปจนกว่า L จะมากกว่า R ซึ่งเป็นเงื่อนไขที่บ่งบอกว่าค้นหาจนครบแล้ว โดยในแต่ละรอบของลูป คำนวณหาตำแหน่งตรงกลางของอาร์เรย์ (Mid) โดยใช้ $(L+R)/2$ จากนั้นจะตรวจสอบว่าค่าของคีย์ที่ต้องการค้นหาตรงกับข้อมูลในตำแหน่ง Mid หรือไม่ ถ้าตรงกัน จะส่งค่ากลับเป็น true และสิ้นสุดการทำงาน แต่ถ้าไม่ตรงกัน จะเปรียบเทียบว่าคีย์ที่ต้องการค้นหาน้อยกว่าหรือมากกว่าข้อมูลในตำแหน่ง Mid

ถ้าคิยน้อยกว่า จะย้ายขอบเขตการค้นหาด้านขวา (R) ไปที่ Mid-1 แต่ถ้าคิยมากกว่า จะย้ายขอบเขตด้านซ้าย (L) ไปที่ Mid+1 แต่สุดท้ายถ้าค้นหาไม่พบ จะส่งค่ากลับเป็น false

9. ฟังก์ชัน main

```
int main()

{

    printf("BINARY SEARCH\n");

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

    N=32;

    PrepareRawKey(N);

    printf("Raw key :\n");

    DispKey(N); // แสดงข้อมูลดิบ

    BubbleSort(N);

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

    while(key!=-999)

    {

        printf("Sorted Key :\n");

        DispKey(N); // แสดงข้อมูลที่จัดเรียงแล้ว

        printf("\nEnter Key for Search(-999 for EXIT) = ");

        scanf("%d",&key); // อ่านค่าคีย์จากคีย์บอร์ด

        if(key!=-999)

        {

            result=BinarySearch(key); // เรียกฟังก์ชัน Binary Search
```

9. ฟังก์ชัน main (ต่อ)

```

    if(result)

        printf("Result...FOUND\n"); // ถ้าพบ

    else

    {

        Beep(600,600);

        printf("Result...NOT FOUND!\n"); // ถ้าไม่พบ

    }

    //printf("Searching Time : %d\n",Times);

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

} // จบ if

} // จบ while

return(0);

} // จบ main

```

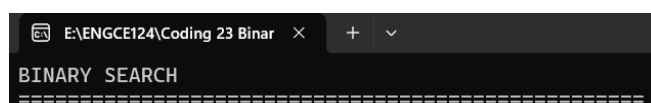
ฟังก์ชัน main เป็นฟังก์ชันหลัก เปรียบเสมือนเป็นจุดเริ่มต้นของการทำงานของโปรแกรม โดยจะเรียกใช้ฟังก์ชันต่าง ๆ ที่อธิบายไปก่อนหน้านี้เพื่อตั้งค่าและจัดการการค้นหาข้อมูล โดยหลักการทำงานของโปรแกรมเริ่มต้นโดยการแสดงข้อความบอกว่าเป็นโปรแกรม Binary Search จากนั้นกำหนดค่าตัวแปร N เป็น 32 ซึ่งเป็นจำนวนข้อมูลที่ต้องการสุ่มและจัดเรียง ต่อมาจะเรียกฟังก์ชัน PrepareRawKey() เพื่อสุ่มข้อมูลในอาร์เรย์ Data[] และแสดงข้อมูลที่สุ่มได้ผ่านฟังก์ชัน DispKey() จากนั้นจะเรียกฟังก์ชัน BubbleSort() เพื่อจัดเรียงข้อมูลจากน้อยไปมากและวนลูปเพื่อให้ผู้ใช้ใส่คีย์ที่ต้องการค้นหา โดยหากคีย์ที่ใส่ไม่เท่ากับ -999 จะเรียกฟังก์ชัน BinarySearch() เพื่อทำการค้นหา เมื่อค้นหาเสร็จจะแสดงผลว่าพบข้อมูลหรือไม่ ถ้าพบจะแสดงข้อความว่า "FOUND" ถ้าไม่พบจะแสดงข้อความว่า "NOT FOUND" พร้อมเสียงเตือน โดยสุดท้ายจะวนลูปจนกว่าผู้ใช้จะใส่ -999 เพื่อออกจากโปรแกรม

ผลลัพธ์การใช้งานโปรแกรม BINARY SEARCH

โปรแกรม BINARY SEARCH เป็นโปรแกรมที่ใช้สำหรับ ค้นหาข้อมูลแบบทวิภาค (Binary Search) ซึ่งทำการสุ่มตัวเลขเข้ามาในอาร์เรย์ จากนั้นจัดเรียงข้อมูล และให้ผู้ใช้สามารถค้นหาตัวเลขในอาร์เรย์ได้โดยใช้การค้นหาแบบทวิภาค การแสดงผลของโปรแกรมจะแสดงรายละเอียดในแต่ละขั้นตอนเพื่อให้ผู้ใช้เห็นว่ามี การค้นหาอย่างไร และโปรแกรมจะบอกผลลัพธ์ว่าค้นพบข้อมูลหรือไม่

1. การเริ่มต้นโปรแกรม

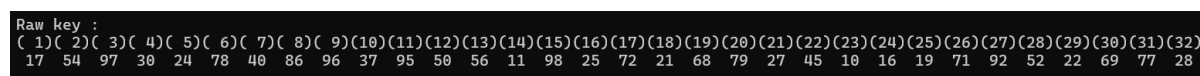
เมื่อรันโปรแกรม ผู้ใช้จะเห็นข้อความหัวข้อของโปรแกรมที่มีชื่อว่า “BINARY SEARCH” แสดงออกมา บนหน้าจอคอนโซล



```
E:\ENGCE124\Coding 23 Binar  X  +  v
BINARY SEARCH
=====
```

2. การสุ่มตัวเลข

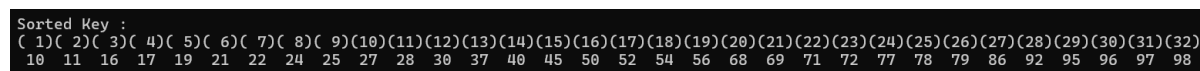
โปรแกรมจะทำการสุ่มตัวเลข 32 ตัวให้อยู่ในช่วงระหว่าง 10 ถึง 99 และเก็บตัวเลขที่สุ่มได้ไว้ในอาร์เรย์ จากนั้นจะแสดงข้อมูลที่สุ่มขึ้นมาในรูปแบบดิบ (ยังไม่ได้จัดเรียง) โดยแสดงตัวเลขทั้งหมดพร้อมกับตำแหน่งของตัวเลขในอาร์เรย์



```
Raw key :
( 1)( 2)( 3)( 4)( 5)( 6)( 7)( 8)( 9)(10)(11)(12)(13)(14)(15)(16)(17)(18)(19)(20)(21)(22)(23)(24)(25)(26)(27)(28)(29)(30)(31)(32)
17 54 97 30 24 78 40 86 96 37 95 50 56 11 98 25 72 21 68 79 27 45 10 16 19 71 92 52 22 69 77 28
=====
```

3. การจัดเรียงข้อมูล

หลังจากแสดงข้อมูลที่สุ่มขึ้นมาแล้ว โปรแกรมจะจัดเรียงตัวเลขเหล่านั้นจากน้อยไปมากโดยใช้ Bubble Sort ซึ่งเป็นขั้นตอนที่ทำให้ข้อมูลพร้อมสำหรับการค้นหาแบบ Binary Search จากนั้นจะแสดงผลลัพธ์ของการจัดเรียงข้อมูลให้ผู้ใช้เห็น



```
Sorted Key :
( 1)( 2)( 3)( 4)( 5)( 6)( 7)( 8)( 9)(10)(11)(12)(13)(14)(15)(16)(17)(18)(19)(20)(21)(22)(23)(24)(25)(26)(27)(28)(29)(30)(31)(32)
10 11 16 17 19 21 22 24 25 27 28 30 37 40 45 50 52 54 56 68 69 71 72 77 78 79 86 92 95 96 97 98
```

4. การป้อนค่าคีย์สำหรับค้นหา

โปรแกรมจะขอให้ผู้ใช้ป้อนตัวเลขที่ต้องการค้นหาในอาร์เรย์ที่ถูกจัดเรียงแล้ว โดยผู้ใช้สามารถป้อนตัวเลขที่ต้องการค้นหาได้ ถ้าต้องการหยุดโปรแกรมสามารถป้อน -999



```
Enter Key for Search(-999 for EXIT) = |
```

5. การทำงานของ Binary Search

เมื่อผู้ใช้ป้อนตัวเลขที่ต้องการค้นหา โปรแกรมจะทำการค้นหาแบบทวิภาค (Binary Search) โดยจะแสดงรายละเอียดของขั้นตอนการค้นหาในแต่ละรอบ ได้แก่ค่าของตัวชี้ตำแหน่งซ้าย (L), ขวา (R), ตำแหน่งกึ่งกลาง (Mid), และจำนวนครั้งที่ทำการค้นหา (Times) โดยโปรแกรมจะแบ่งครึ่งอาร์เรย์แล้วค้นหาข้อมูลตามตำแหน่งกึ่งกลางในแต่ละรอบ ในการค้นหาข้อมูลถ้าค้นพบข้อมูลที่ต้องการ โปรแกรมจะแสดงข้อความ “Result...FOUND” แต่ถ้าค้นหาแล้วไม่พบ โปรแกรมจะแสดงข้อความ “Result...NOT FOUND!!” พร้อมทั้งส่งเสียงเตือนผ่านฟังก์ชัน Beep() เพื่อแจ้งเตือนผู้ใช้ว่าข้อมูลที่ค้นหาไม่มีในอาร์เรย์

```
Enter Key for Search(-999 for EXIT) = 10
L : 1 R : 32 Mid : 16 Searching Time : 1
L : 1 R : 15 Mid : 8 Searching Time : 2
L : 1 R : 7 Mid : 4 Searching Time : 3
L : 1 R : 3 Mid : 2 Searching Time : 4
L : 1 R : 1 Mid : 1 Searching Time : 5
Result...FOUND
-----Searching Finished
```

```
Enter Key for Search(-999 for EXIT) = 8
L : 1 R : 32 Mid : 16 Searching Time : 1
L : 1 R : 15 Mid : 8 Searching Time : 2
L : 1 R : 7 Mid : 4 Searching Time : 3
L : 1 R : 3 Mid : 2 Searching Time : 4
L : 1 R : 1 Mid : 1 Searching Time : 5
Result...NOT FOUND!!
-----Searching Finished
```

6. การค้นหาเพิ่มเติม และจบการทำงานของโปรแกรม

หลังจากแสดงผลการค้นหาแล้ว โปรแกรมจะกลับไปขั้นตอนขอให้ผู้ใช้ป้อนค่าคีย์ใหม่ เพื่อค้นหาข้อมูลตัวอื่นได้อีก แต่ผู้ใช้สามารถป้อนค่าคีย์ใหม่หรือเลือกป้อน -999 เพื่อออกจากโปรแกรม

```
Enter Key for Search(-999 for EXIT) = -999
-----
Process exited after 665 seconds with return value 0
Press any key to continue . . . |
```

บรรณานุกรม

ChatGPT. (-). Efficient Data Lookup with Binary Search. สืบค้น 28 กันยายน 2567,
จาก <https://chatgpt.com/>