Assignment 9

สารบัญ

ฟังก์ชัน main

การทำงานของโปรแกรมจะเริ่มต้นทำงานในฟังก์ชัน **main** จากนั้นส่งต่อแยกการทำงานต่าง ๆ ไปในฟังก์ชันอื่น ๆ โดยสามารถอ่านการทำงานโค้ดส่วนฟังก์ชัน **main** ได้ด้านล่างนี้ ซึ่งจะอธิบายตามลำดับ

```
int main ()
{
    printf("Enter input: ");
    gets(input);
    RemoveSpaceAtCorner(input);
    while (strcmp(input, "end")!=0 && strcmp(input, "exit")!=0)
    {
        char temp[100], data[100][40];
        int errorCount = 0, fnCount = 0, opCount = 0, idCount =0, numCount = 0;
        printf("\n**************************************\n");
        printf("Input: %s\n", input);
        strcpy(temp, input); // Copy input to temp for operation
        ToLower(temp); // Set to lower
        printf("To lower: %s\n", temp);
        AddSpace(temp);
        printf("Add space: %s\n", temp);
        SpaceSeparate(temp, data);
        int i;
        printf("\n=== Analyse === \n");
        for (i=0; i < sizeof(data) && strcmp(data[i],"")!=0;i++) {
            if (IsFunction(data[i])) {
                printf(" %s : FUNCTION\n", data[i]);
                fnCount++;
            }
            else if (IsNumber(data[i])) {
                printf(" %s : NUMBER\n", data[i]);
                numCount++;
            }
            else if (IsOperation(data[i])) {
                printf(" %s : OPERATION\n", data[i]);
                opCount++;
            }
            else if (IsValidIndentifier(data[i])) {
                printf(" %s : INDENTIFIER \n", data[i]);
                idCount++;
            }
            else {
                errorCount++;
                printf("%s : ERROR \n", data[i]);
            }
        }
        printf("\n===  RESULT  === \n Number: %d \n Operation: %d \n Function: %d \n Identifier: %d \n Error: %d \nTotal: %d", numCount, opCount, fnCount, idCount, errorCount, numCount + opCount + fnCount + errorCount + idCount);
        printf("\n**************************************\n");
        printf("\n\n");
        printf("Analyse again\n");
        printf("Enter input: ");
        gets(input);
        RemoveSpaceAtCorner(input);
        printf("\n\n\n\n\n");
    }

    printf("\nEnd");

    return 0;
}
```

เริ่มต้นการทำงานด้วยการรับค่า **input**

```
printf("Enter input: ");
gets(input);
```

จากนั้นเรียกใช้ฟังก์ชัน **RemoveSpaceAtCorner** เพื่อลบช่องว่าง(เว้นวรรค) ทั้งหมดบริเวณด้านหน้าและด้านหลัง **input**

```
RemoveSpaceAtCorner(input);
```

ถ้า input ที่ลบช่องว่างออกไปแล้วไม่ใช่ทั้ง exit และ end ให้ทำงานต่อ แต่ถ้าใช้ให้จบการทำงานพร้อมทั้ง print ข้อความ “End”

```
100     while (strcmp(input, "end")!=0 && strcmp(input, "exit")!=0)
101     {…
144     }
145
146     printf("\nEnd");
147
148     return 0;
```

ประกาศจองตัวแปรเพื่อเตรียมใช้งาน

```
char temp[100], data[100][40];
int errorCount = 0, fnCount = 0, opCount = 0, idCount =0, numCount = 0;
```

Print แสดงผลข้อมูล input ที่รับมาจากนั้นก๊อปปี้ Input เก็บแยกไว้ในตัวแปร temp เพื่อใช้คำนวณต่อไป

```
printf("\n**************************************\n");
printf("Input: %s\n", input);
strcpy(temp, input); // Copy input to temp for operation
```

ใช้ฟังก์ชัน ToLower เพื่อทำให้ตัวอักษรเป็นตัวพิมพ์เล็กทั้งหมด จากนั้น print แสดงผล

```
ToLower(temp); // Set to lower
printf("To lower: %s\n", temp);
```

ใช้ฟังก์ชัน AddSpace เพื่อเว้นวรรคที่ operator ต่างๆ จากนั้น print แสดงผล

```
AddSpace(temp);
printf("Add space: %s\n", temp);
```

ใช้ฟังก์ชัน SpaceSeparate เพื่อแยกข้อความเก็บไว้ใน array โดยใช้ช่องว่าง(Space) เป็นตัวแยก

```
SpaceSeparate(temp, data);
```

ลูปตัวตรวจสอบค่าที่อยู่ใน array ถ้าหากค่านั้นมีข้อความไม่เท่ากับ null ให้นำมาวิเคราะห์ข้อมูลโดยใช้ฟังก์ชันต่อไปนี้เพื่อวิเคราะห์หาประเภท

IsFunction: ตรวจสอบว่าข้อความนั้นเป็นฟังก์ชันหรือไม่

IsNumber: ตรวจสอบว่าข้อความนั้นเป็นตัวเลขหรือไม่

IsOperation: ตรวจสอบว่าข้อความนั้นเป็น Operator หรือไม่

IsValidIndentifer: ตรวจสอบว่าข้อความนั้นเป็น Indentifier ชื่อที่เขียนถูกต้องตามหลักภาษาซีหรือไม่

หากไม่เข้าเงื่อนไขใดๆ เลยข้อความนั้นเป็น error

เมื่อตรวจสอบทราบแล้วว่าข้อความนั้นเป็นอะไรก็ทำการ print แสดงผลบอกว่าข้อความนี้คืออะไร

และนับจำนวนของประเภทนั้นเพิ่มขึ้น

```
for (i=0; i < sizeof(data) && strcmp(data[i],"")!=0;i++) {
    if (IsFunction(data[i])) {
        printf(" %s : FUNCTION\n", data[i]);
        fnCount++;
    }
    else if (IsNumber(data[i])) {
        printf(" %s : NUMBER\n", data[i]);
        numCount++;
    }
    else if (IsOperation(data[i])) {
        printf(" %s : OPERATOR\n", data[i]);
        opCount++;
    }
    else if (IsValidIndentifier(data[i])) {
        printf(" %s : INDENTIFIER \n", data[i]);
        idCount++;
    }
    else {
        errorCount++;
        printf("%s : ERROR \n", data[i]);
    }
}
```

เมื่อวิเคราะห์ข้อความทั้งหมดเสร็จแล้วก็ปริ้นแสดงผลว่ามีอะไรจำนวนเท่าไร เป็นอันจบการทำงานครั้งนี้

```
printf("\n===  RESULT  === \n Number: %d \n Operation: %d \n Function: %d \n Identifier: %d \n Error: %d \nTotal: %d",
 numCount, opCount, fnCount, idCount, errorCount, numCount + opCount + fnCount + errorCount + idCount);
```

จากนั้นแสดงข้อความให้ผู้ใช้กรอกข้อมูล input สำหรับการทำงานครั้งใหม่ พร้อมทั้งใช้ฟังก์ชัน RemoveSpaceAtCorner

เพื่อลบช่องว่างด้านหน้าและด้านหลังของ input

```
printf("\n**********************************\n");
printf("\n\n");
printf("Analyse again\n");
printf("Enter input: ");
gets(input);
RemoveSpaceAtCorner(input);
printf("\n\n\n\n\n");
```

หลังจากนั้นจะจบลูปของ while และกับไปเช็คเงื่อนไขของ while ใหม่อีกครั้งเพื่อตรวจสอบว่า input ที่ได้รับมาไม่ใช่ทั้ง end และ exit เพื่อทำงานต่อไป

ฟังก์ชัน IsValidIndentifier

ใช้สำหรับตรวจสอบว่าข้อความ (string) ที่รับมาว่าเป็นชื่อตัวแปรและเขียนถูกต้องตามหลักของภาษาซีหรือไม่

Return 1 เมื่อเป็นชื่อตัวแปรและถูกต้องตามหลัก

Return 0 เมื่อไม่ได้เป็นชื่อตัวแปรหรือไม่ถูกต้องตามหลัก

```
int IsValidIndentifier (char *s) {
    if (!isalpha(s[0]) && s[0] != '_')
        return 0;
    else {
        int i;
        for (i = 1; i < strlen(s);i++) {
            if (!isalpha(s[i]) && s[i] != '_' && isdigit(s[i]) == 0)
                return 0;
        }
    }
    return 1;
}
```

ตรวจสอบว่า string ที่รับมา ถ้าหากขึ้นต้นตัวแรกไม่ใช่ตัวอักษรและไม่ใช่ _ ซึ่งผิดหลักของภาษาซี แสดงว่าข้อความนี้ไม่ใช่ indentifier จึง return 0

```
if (!isalpha(s[0]) && s[0] != '_')
        return 0;
```

เมื่อผ่านเงื่อนไขด้านบนมาแล้วก็เช็คต่อด้วยว่า ที่เหลือเป็นตัวอักษรหรือเป็นตัวเลขหรือเป็นเครื่องหมาย _ หรือไม่

ถ้าหากเป็นนอกเหนือจากนี้ไม่ถูกหลักของภาษาซี จึง return 0

```
int i;
    for (i = 1; i < strlen(s);i++) {
        if (!isalpha(s[i]) && s[i] != '_' && isdigit(s[i]) == 0)
            return 0;
    }
```

ถ้าหากเช็คเงื่อนไขด้านบนทั้งสองแล้วไม่มีการ return 0 ออกไปก่อนแสดงว่าข้อความนี้ถูกต้องตามหลักจึง return 1

```
return 1;
```

ฟังก์ชัน IsNumber

ใช้สำหรับตรวจสอบข้อความ(string)ที่รับมาว่าเป็นตัวเลขหรือไม่

Return 1 เมื่อเป็นตัวเลข

Return 0 เมื่อไม่ใช่ตัวเลข

```
int IsNumber(char *s) {
    double num;
    int i;
    for (i=0; i < strlen(s); i++) {
        if (isdigit(s[i])==0 && s[i] != '.')
            return 0;
    }
    return IsValidNumberFormat(s); // If c1 and c2 == null that mean no any character then check number format
}
```

ตรวจสอบ char แต่ละตัวหากไม่ใช่ตัวเลขและไม่ใช่ . (จุด) แสดงว่าไม่ใช่ตัวเลข

```
for (i=0; i < strlen(s); i++) {
        if (isdigit(s[i])==0 && s[i] != '.')
            return 0;
    }
```

หลังจากเช็คว่าเป็นตัวเลขแล้วก็ส่งต่อข้อความไปยังฟังก์ชัน IsValidNumberFormat เพื่อตรวจสอบฟอร์แมตของตัวเลขว่าเรียงถูกต้องหรือไม่ ข้อความนี้จะเป็นตัวเลขหรือไม่เป็นขึ้นอยู่กับฟอร์แมตว่าถูกต้องหรือไม่ ดังนั้นจึง return ด้วยผลลัพธ์จากฟังก์ชัน IsValidNumberFormat

```
return IsValidNumberFormat(s);
```

ฟังก์ชัน IsValidNumberFormat

ฟังก์ชันสำหรับตรวจสอบรูปแบบฟอร์แมตของตัวเลขว่าถูกต้องตามหลักหรือไม่ เช่น 1 3.25 4.00 เป็นรูปแบบที่ถูกต้อง

ส่วนรูปแบบที่ไม่ถูกต้องได้แก่ 2.5.1เป็นต้น โดยฟังก์ชันนี้จะนับว่ามีจุดหากมีจุดมากกว่า 1 จุดแสดงว่าไม่ถูกต้อง

Return 1 เมื่อฟอร์แมตถูกต้อง

Return 0 เมื่อฟอร์แมตไม่ถูกต้อง

```c
int IsValidNumberFormat (char *s) {
    int dotCount = 0, i = 0;
    for (i = 0; i < strlen(s); i++) {
        if (s[i] == '.') dotCount++;
    }
    if (dotCount < 2) return 1;
    else return 0;
}
```

ฟังก์ชัน IsFunction

ใช้ตรวจสอบว่าข้อความ(string) ที่รับมาเป็น function หรือไม่

Return 1 เมื่อเป็นฟังก์ชัน

Return 0 เมื่อไม่เป็นฟังก์ชัน

```
int IsFunction (char *s) {
    const char fnList[10][10] = {"sin", "cos", "tan", "asin", "acos", "atan", "sqrt", "log", "exp", "pow"};
    int i;
    for (i=0;i<10;i++) {
        if (strcmp(s, fnList[i])==0)
            return 1;  // If string is equal function name return 1(TRUE)
    }
    return 0;
}
```

กำหนด list ชื่อฟังก์ชันเอาไว้ใช้ตรวจสอบ

```
const char fnList[10][10] = {"sin", "cos", "tan", "asin", "acos", "atan", "sqrt",
"log", "exp", "pow"};
```

ตรวจสอบใน list รายชื่อฟังก์ชันว่าตรงกับชื่อไหนหรือไม่หากตรงแสดงว่าเป็นฟังก์ชัน จึง return 1

แต่ถ้าหากเช็คจนหมด list ไม่ตรงสักชื่อแสดงว่าไม่ใช่ฟังก์ชัน return 0

```
for (i=0;i<10;i++) {
        if (strcmp(s, fnList[i])==0)
            return 1;  // If string is equal function name return 1(TRUE)
    }
    return 0;
```

ฟังก์ชัน IsOperation

ใช้ตรวจสอบว่าข้อความ(string) ที่รับมาเป็น Operator(,+-*/^()) หรือไม่

Return 1 เมื่อเป็น

Return 0 เมื่อไม่เป็น

```c
int IsOperation (char *s) {
    int i;
    for (i = 0; i < strlen(s); i++) {
        if (strchr(",+-*/^()", s[i]) != NULL)
            return 1;
    }
    return 0;
}
```

ตรวจสอบว่าตรงกับ ,+-*/^() ตัวใดตัวหนึ่งหรือไม่ ถ้าตรงแสดงว่าเป็น operator จึง return 1

ถ้าตรวจสอบทั้งหมดแล้วไม่ตรงแสดงว่าไม่ได้เป็น operator จึง return 0

ฟังก์ชัน AddSpace

ใช้เพิ่มช่องว่าง(เว้นวรรค) หน้าและหลัง operator(,+-*/^())

```
void AddSpace (char *s) {
    int i,j;
    char temp[200] = "", old[200] = "";
    for (i=0,j=strlen(s); i < j;i++) {
        if (strchr(",+-*/^()", s[i]) != NULL)
            sprintf(temp,"%s %c ", old, s[i]);
        else
            sprintf(temp,"%s%c",old, s[i]);
        strcpy(old, temp);
    }
    strcpy(s,temp);
}
```

โดยกำหนดตัวแปร array คือ temp และ old ไว้ใช้สำหรับเพิ่มช่องว่าง

```
char temp[200] = "", old[200] = "";
```

โดยจะเช็ค char ทีละตัว หากเป็น operator จะนำ old ซึ่งคือข้อความก่อนหน้ามาและเว้นวรรคไว้ต่อด้วย char ของ operator โดยทั้งหมดนี้จะเซฟเก็บไว้ที่ temp

```
if (strchr(",+-*/^()", s[i]) != NULL)
            sprintf(temp,"%s %c ", old, s[i]);
```

ถ้าไม่ใช่ operator ก็นำไปเรียงต่อกันปกติและเก็บไว้ที่ temp

```
else
            sprintf(temp,"%s%c",old, s[i]);
```

เสร็จแล้วก็ก๊อป temp ไปใส่ old เพื่อจะใช้แทรกไว้หน้า char ตัวต่อไป

```
strcpy(old, temp);
```

หลังจากจบลูป char ทั้งหมดแล้วก็ก๊อปปี้ค่า temp ไปทับข้อความเดิม

```
strcpy(s,temp);
```

ฟังก์ชัน SpaceSeparate

ใช้สำหรับแบ่งข้อความด้วย ช่องว่าง(เว้นวรรค) มาเก็บแยกไว้ใน array

```
void SpaceSeparate (char *s, char data[][40]) {
    char * word;
    int count = 0;
    word = strtok(s, " ");
    while (word != NULL) {
        strcpy(data[count++], word);
        word = strtok(NULL, " ");
    }
    data[count][0] = '\0';
}
```

ตรวจหาว่ามีช่องว่าง(เว้นวรรค) หรือไม่ถ้ามีก็นำข้อความที่แบ่งได้เก็บไว้และค้นหาเว้นวรรคตัวถัดไปจนกว่าจะค้นไม่ได้

เมื่อจบการค้นหาแล้วก็ปิด array ตัวสุดท้ายด้วยเพื่อบอกว่าข้อความ array มีถึงแค่นี้

```
data[count][0] = '\0';
```

ฟังก์ชัน ToLower

ใช้สำหรับทำใช้ข้อความ(string) กลายเป็นอักษรภาษาอังกฤษพิมพ์เล็กทั้งหมด

โดยจะไล่เปลี่ยน char ใน array ทีละตัว

```
void ToLower(char *s) {
    int i;
    for (i = 0;s[i];i++) s[i] = tolower(s[i]);
}
```

ฟังก์ชัน RemoveSpaceAtCorner

ใช้สำหรับลบช่องว่าง(เว้นวรรค) ทั้งหมดที่อยู่ด้านหน้าและด้านหลังของข้อความ(string)

```
void RemoveSpaceAtCorner (char *s) {
    while (s[strlen(s)-1] == ' ') input[strlen(s) - 1] = '\0'; /
    while (s[0] == ' ')
        strcpy(s, s + 1); // If first character is space then re
}
```

ตรวจสอบว่าอักษรก่อนตัวสุดท้ายยังเป็นเว้นวรรคอยู่หรือไม่ถ้าเป็นให้เปลี่ยนจากเว้นวรรคเป็น \0 แทนเพื่อนปิดข้อความ ทำแบบนี้ซ้ำไปเลื่อยๆ จนกว่าก่อนตัวสุดท้ายจะไม่ใช่เว้นวรรค

```
while (s[strlen(s)-1] == ' ') input[strlen(s) - 1] = '\0';
```

ตรวจสอบว่าตัวแรกเป็นเว้นวรรคหรือไม่ ถ้าเป็นไม่เป็นให้ลบออก ด้วยการเลื่อนข้อความมาทางซ้าย

```
while (s[0] == ' ')
        strcpy(s, s + 1);
```