รายงาน : Lab04 – Structure & Pointers

จัดทำโดย

นายกิตติพศ หนูทอง รหัสนักศึกษา : 6135512003

นายปฏิภาณ วรรณโก รหัสนักศึกษา : 6135512059

Section : 01

240-207 Programing and Data Structures

“งานทั้งหมดนี้ในรายงานฉบับนี้ล้วนเป็นผลงานของข้าพเจ้า มิได้ลอกหรือสำเนามาจากที่อื่นใด ในกรณีที่พบว่าเกิดสำเนาด้วยวิธีใดก็ตาม ข้าพเจ้ายินดีไม่ขอรับคะแนนจากรายงานฉบับนี้”

คะแนนที่ได้

ลงชื่อ

กิตติพศ หนูทอง ปฏิภาณ วรรณโก

(นายกิตติพศ หนูทอง) (นายปฏิภาณ วรรณโก)

Lab04 – Structure & Pointers

ข้อที่ 1 : Replace Space

Replace Space

เขียนโปรแกรมเพื่อรับข้อความหนึ่งบรรทัด โดยใช้ประโยชน์จากฟังก์ชัน

```
void replace_space(char *s, char c);
```

เพื่อแทนที่ space ด้วยตัวอักษร c

รูปแบบการแสดงผล

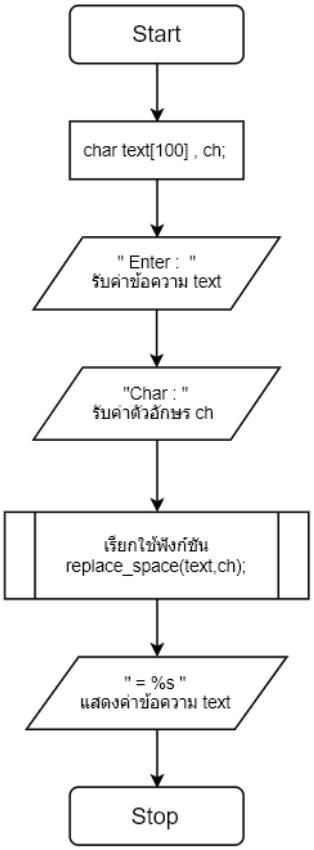
```
Enter: <I love you>
Char: _
= I_love_you
```

Code

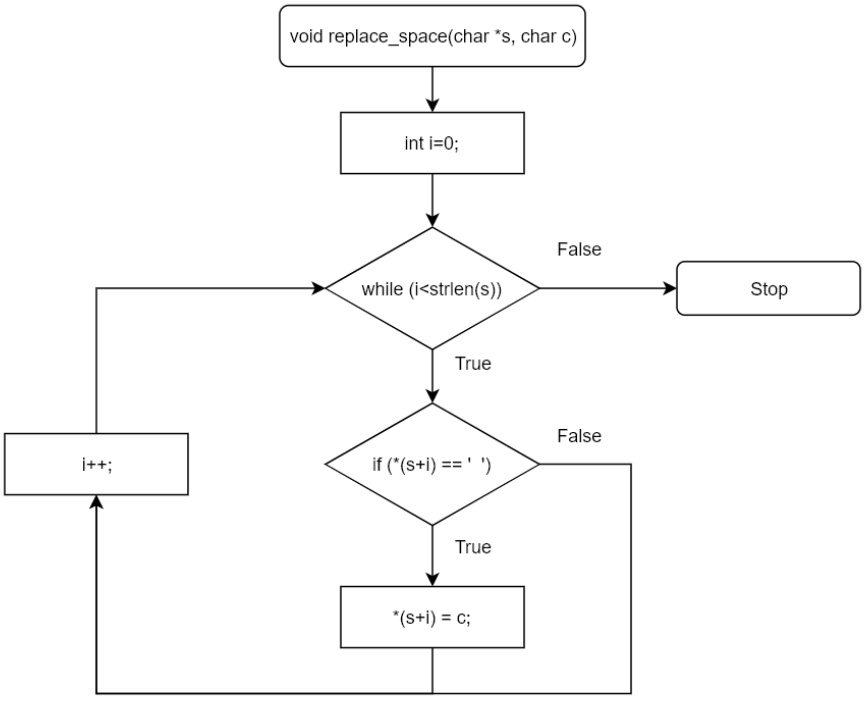
lab04-1.c

```c
#include<stdio.h>
#include<string.h>
void replace_space(char *s, char c);
int main ()
{
    char text[100],ch;
    printf("Enter : ");
    gets(text);
    printf("Char : ");
    scanf("%c",&ch);
    replace_space(text,ch);
    printf("= %s",text);
    return 0;
}
void replace_space(char *s, char c)
{
    int i=0;
    while (i<strlen(s))
    {
        if (*(s+i) == ' ')
        {
            *(s+i)= c;
        }
        i++;
    }
}
```

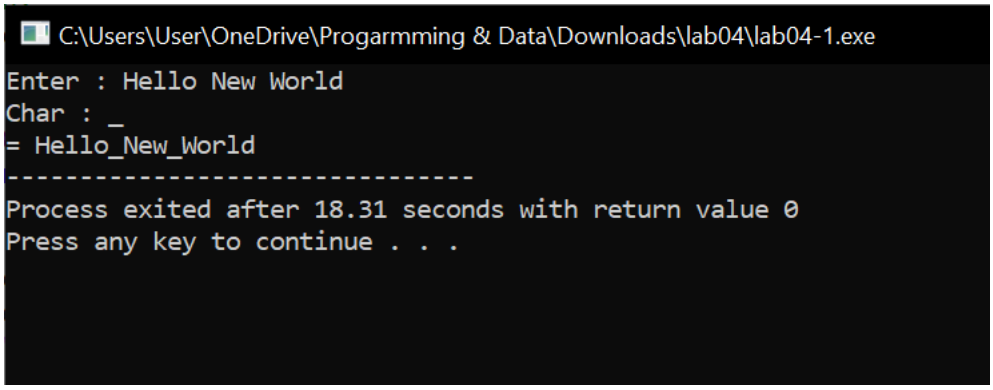
Flowchart



ฟังก์ชัน void replace_space(char *s, char c)



ผลการรันโปรแกรม



อธิบายหลักการทำงาน : เป็นโปรแกรมรับค่าข้อความมาตรวจสอบถ้าเจอ “space bar” ให้เปลี่ยนเป็นค่าตัวอักษรที่รับมา แล้วแสดงข้อความที่ถูกเปลี่ยนออกมาทางหน้าจอ โดยในโปแกรมจะมี 2 ฟังก์ชัน คือ ฟังก์ชัน mian () และ ฟังก์ชัน void replace_space(char *s, char c) มีหลักการทำงานมีดังนี้

ฟังก์ชัน main () จะมีการประกาศตัวแปร char ch และ อาร์เรย์ char text[100] แล้วรับค่าข้อความเก็บใน text และรับค่าตัวอักษรเก็บใน ch ต่อมาเรียกใช้ฟังก์ชัน replace_space(text,ch); ส่งค่า array text และ ch ไปตรวจสอบหลังจากนั้นแสดงค่า text อันใหม่ออกมาทางหน้าจอ

ฟังก์ชัน void replace_space(char *s, char c) ในฟังก์ชัน ค่า address text จะถูกเก็บใน s และค่า ch จะถูกเก็บใน c ต่อมาประกาศตัวแปร int i ให้ค่า i = 0 จากนั้นวนลูปไปเรื่อย ๆ จนกว่า i มีค่ามากกว่าจำนวนตัวอักษรของข้อความในคำสั่ง strlen(s) ภายใต้เงื่อนไข while (i<strlen(s)) ต่อมาใช้เงื่อนไข if (*(s+i) == ' ') ตรวจสอบค่าที่เป็น “space bar” ถ้าใช่ให้ค่า s+i เท่ากับ c ด้วยคำสั่ง *(s+i) = c; หลังจากนั้นค่าใน address text ก็จะถูกเปลี่ยนตามเงื่อนไขในฟังก์ชันนี้

ข้อที่ 2 : Replace_space2

Replace_space2

เขียนโปรแกรมเพื่อรับข้อความหนึ่งบรรทัด โดยใช้ประโยชน์จากฟังก์ชัน

```
char* find_space(char *s);
```

ซึ่งจะทำหน้าที่ ในการคืนค่าพอยเตอร์ไปยังตัวอักษรที่เป็น space และจะคืนค่า NULL หากในข้อความนั้นไม่มี space เพื่อแทนที่ space ด้วยตัวอักษรที่กำหนดจากผู้ใช้

รูปแบบการแสดงผล

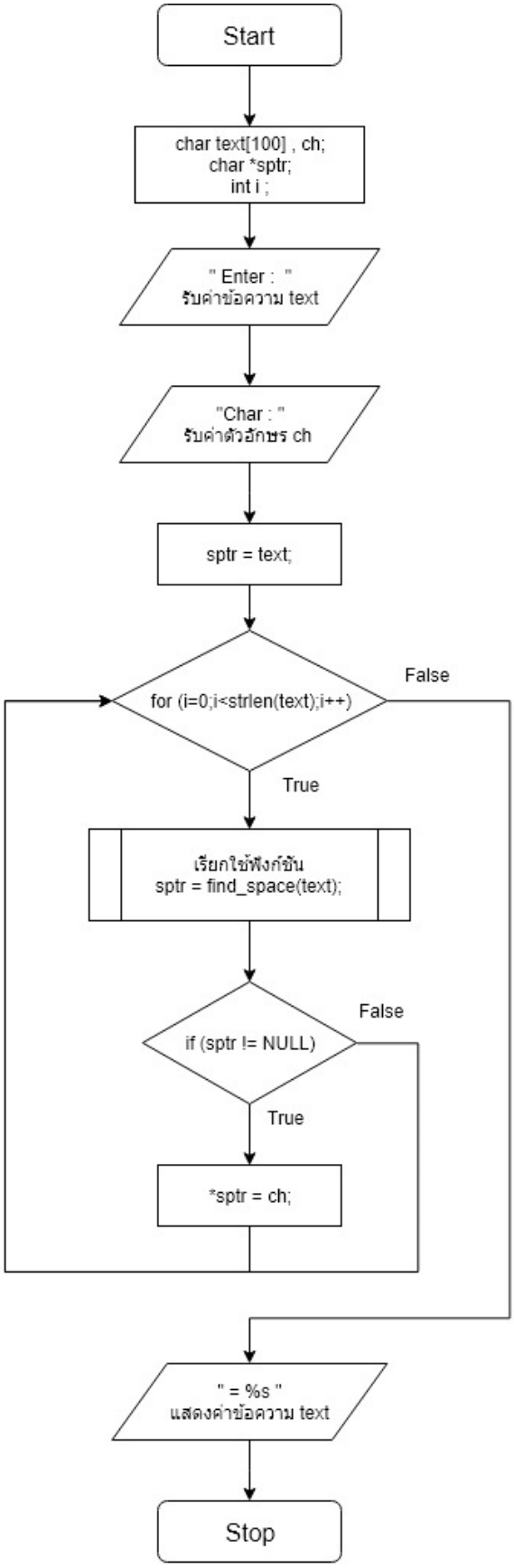
```
Enter: <I love you>
Char: _
= I_love_you
```

Code

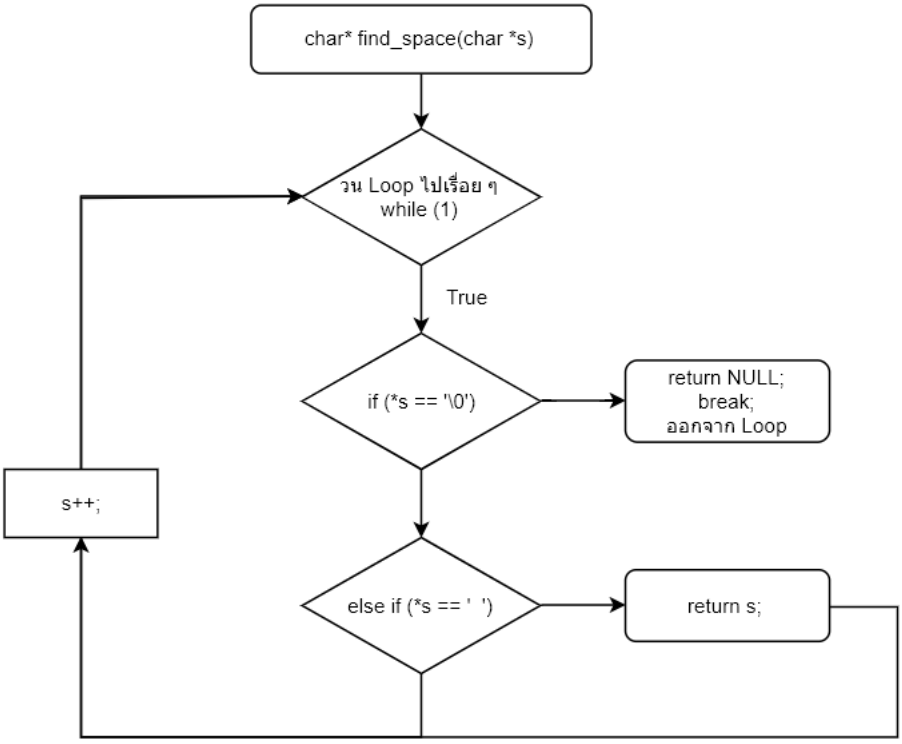
lab04-2.c

```c
#include<stdio.h>
#include<string.h>
char* find_space(char *s);
int main ()
{
    char text[100],ch;
    char *sptr;
    int i;
    printf("Enter : ");
    gets(text);
    printf("Char : ");
    scanf("%c",&ch);
    sptr = text;
    for (i=0;i<strlen(text);i++)
    {
        sptr = find_space(text);
        if (sptr != NULL)
        {
            *sptr = ch;
        }
    }
    printf("= %s",text);
    return 0;
}

char* find_space(char *s)
{
    while (1)
    {
        if (*s == '\0')
        {
            return NULL;
            break;
        }
        else if (*s == ' ')
        {
            return s;
        }
        s++;
    }
}
```

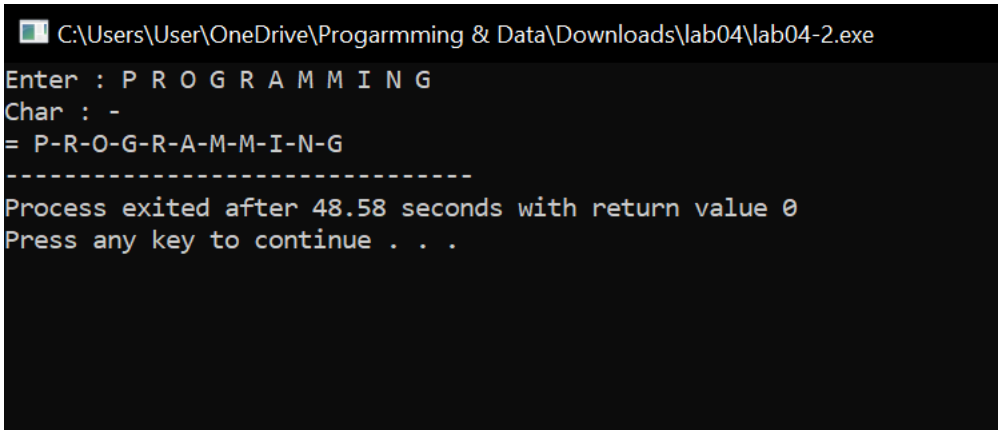
Flowchart


Start
char text[100] , ch;
char *sptr;
int i ;
" Enter : "
รับค่าข้อความ text
"Char : "
รับค่าตัวอักษร ch
sptr = text;
for (i=0;i<strlen(text);i++)
False
True
เรียกใช้ฟังก์ชัน
sptr = find_space(text);
if (sptr != NULL)
False
True
*sptr = ch;
" = %s "
แสดงค่าข้อความ text
Stop

ฟังก์ชัน char* find_space(char *s)



ผลการรันโปรแกรม



อธิบายหลักการทำงาน : เป็นโปรแกรมที่รับค่าตัวอักษรมาเป็นข้อความแล้วให้ตรวจสอบว่ามี “space bar” หรือไม่ ถ้ามีให้ทำการเปลี่ยนเป็นตัวอักษรที่ทำการรับค่ามาและแสดงข้อความที่ถูกเปลี่ยนออกมาทางหน้าจอ แต่ถ้าไม่มี “space bar” ให้คืนค่าเป็น NULL และแสดงค่าข้อความเดิมออกมาทางหน้าจอ โดยในโปรแกรมจะมี 2 ฟังก์ชัน คือ ฟังก์ชัน mian () และ ฟังก์ชัน char* find_space(char *s) มีหลักการทำงานมีดังนี้

ฟังก์ชัน main () จะมีการประกาศตัวแปร int i ไว้วนลูป , ตัวแปรอาร์เรย์ char text [100] , ตัวแปร char ch ไว้รับค่าตัวอักษรมาเปลี่ยนกับค่าตัวอักษร “space bar” และประกาศตัวแปรพอยน์เตอร์ char *sptr เพื่อรับค่า address ของชุดข้อความ โดยเริ่มแรกจะรับค่าข้อความ text และรับค่าตัวอักษร ch แล้วให้พอยน์เตอร์ sptr มีค่า address text ด้วยคำสั่ง sptr = text; จากนั้นวนลูปไปเรื่อย ๆ จนกว่า i มีค่ามากกว่าจำนวนตัวอักษรของข้อความในคำสั่ง strlen(s) ภายใต้เงื่อนไข for (i=0 ; i<strlen(text) ; i++) ในลูปจะมีการเรียกใช้ฟังก์ชันและส่งค่าอาร์เรย์ text ไปในฟังก์ชัน char* find_space(char *s) ด้วยคำสั่ง stpr= find_space(text); ซึ่งจะเก็บค่าที่ return กลับมาไว้ในตัวแปร stpr แล้วนำค่า stpr ในแต่ละตำแหน่งไปตรวจสอบเงื่อนไข if (sptr != NULL) ถ้าค่า stpr ไม่ใช่ค่า NULL ให้เปลี่ยนค่า sptr ตำแหน่งนั้นเป็นค่าของตัวแปร ch ในคำสั่ง *sptr = ch; แต่ถ้าค่า stpr เป็นค่า NULL ให้แสดงข้อความตามเดิม โดยทั้งหมดนี้ค่า text จะถูกเปลี่ยนไปตามค่าของพอยน์เตอร์ sptr หลังจากนั้นแสดงค่าข้อความ text ออกมาทางหน้าจอ

ฟังก์ชัน char* find_space(char *s) จะรับค่าอาร์เรย์ text เก็บในตัวแปรพอยน์เตอร์ s ซึ่งจะวนลูปต่อเนื่องโดยมีเงื่อนไขเป็นจริงตลอด while (1) และในลูปจะมีเงื่อนไข if (*s == '\0') ถ้าค่าพอยน์เตอร์ s มีค่าเท่ากับ “\0” ให้ return NULL และสั่ง break; หยุดลูป แต่ถ้าไม่ใช่จะตรวจสอบค่าพอยน์เตอร์ s ถ้ามีค่าเท่ากับ “space bar” ในเงื่อนไข else if (*s = = ' ') ให้ return s จากนั้นเลื่อนพอยน์เตอร์ s ไปเป็นตำแหน่งถัดไปในคำสั่ง s++ และวนลูปไปเรื่อย ๆ จนกว่าจะเจอคำสั่งหยุดลูป

ข้อที่ 3 : V Processing

V Processing

เขียนโปรแกรม เพื่อรับข้อมูลเข้าสู่ `struct v` แล้วประมวลผลหาค่า f ที่ต้องการตามสูตรที่กำหนดให้

```
f = v.k + (v.set.factor * sum(v.set.values))
```

`v.k` คือ ค่า `k` ของ `v`

`v.set.factor` คือ ค่า factor ของ set ของ `v`

`sum(v.set.values)` คือ ผลรวมของ element ทั้งหมดในอาเรย์ values ของ set ของ `v` ซึ่งจะมีจำนวนตามที่ผู้ใช้ระบุ

รูปแบบการแสดงผล

```
k = <12>
set.factor = <1.5>
set.amount = <5>
set.values = <3 1 4 5 7>
f = 42.0
```

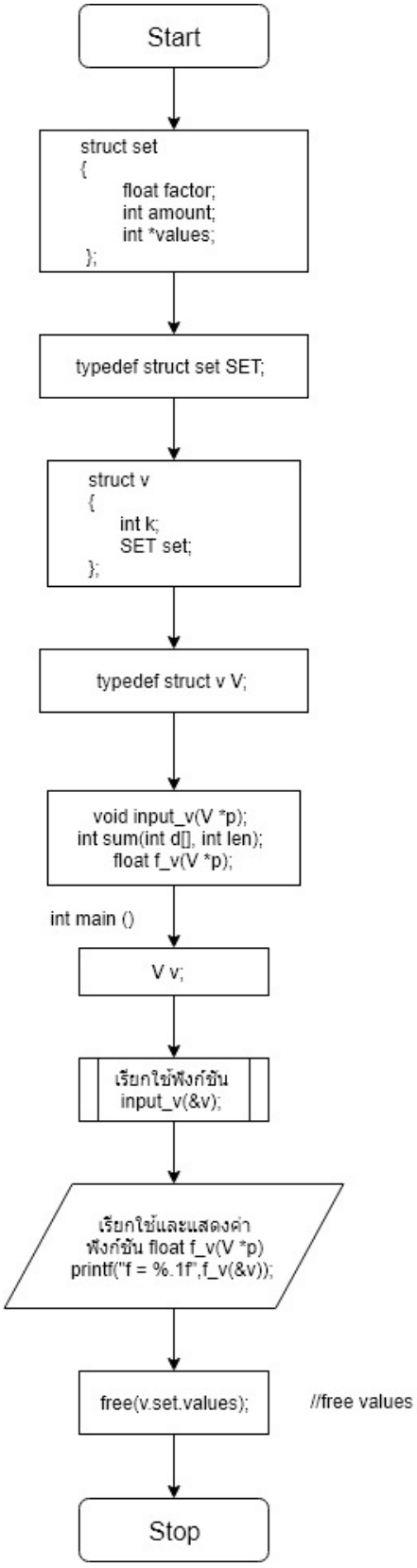
Code

lab04-3.c

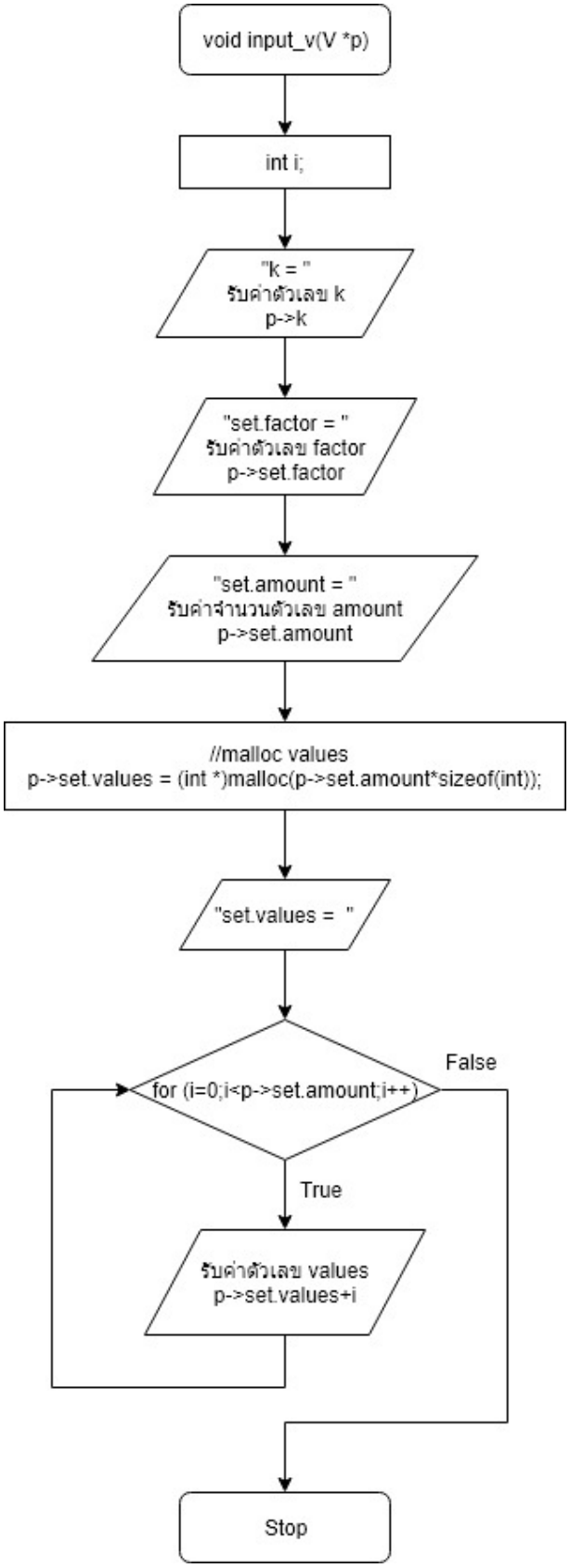
```c
#include<stdio.h>
#include<stdlib.h>

struct set
{
    float factor;
    int amount;
    int *values;
};
typedef struct set SET;

struct v
{   int k;
    SET set;
};
typedef struct v V;

void input_v(V *p);
int sum(int d[], int len);
float f_v(V *p);

```

```c
int main()
{
    V v;
    input_v(&v);
    printf("f = %.1f",f_v(&v));
    //free values
    free(v.set.values);
    return 0;
}
void input_v(V *p)
{
    int i;
    printf("k = ");
    scanf("%d",&p->k);
    printf("set.factor = ");
    scanf("%f",&p->set.factor);
    printf("set.amount = ");
    scanf("%d",&p->set.amount);
    //malloc values
    p->set.values = (int *)malloc(p->set.amount*sizeof(int));
    printf("set.values = ");
    for (i=0;i<p->set.amount;i++)
    {
        scanf("%d",p->set.values+i);
    }
}

int sum(int d[], int len)
{
    int ssum=0,i;
    for (i=0;i<len;i++)
    {
        ssum += d[i];
    }
    return ssum;

}

float f_v(V *p)
{
    float ans;
    int total;

    total = sum(p->set.values,p->set.amount);
    ans = p->k + (p->set.factor*(total));
    return ans;
}
```

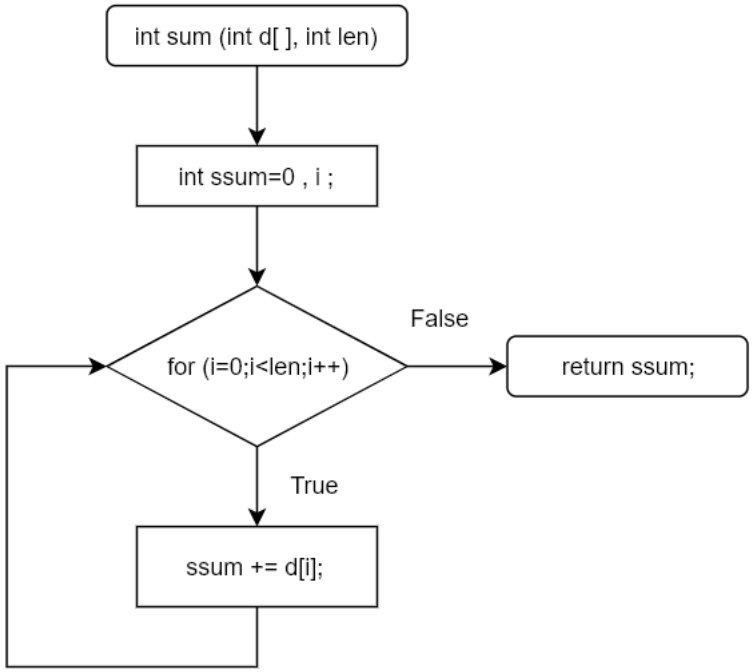
Flowchart


Start
struct set
{
float factor;
int amount;
int *values;
};
typedef struct set SET;
struct v
{
int k;
SET set;
};
typedef struct v V;
void input_v(V *p);
int sum(int d[], int len);
float f_v(V *p);
int main ()
V v;
เรียกใช้ฟังก์ชัน
input_v(&v);
เรียกใช้และแสดงค่า
ฟังก์ชัน float f_v(V *p)
printf("f = %.1f",f_v(&v));
free(v.set.values);
//free values
Stop

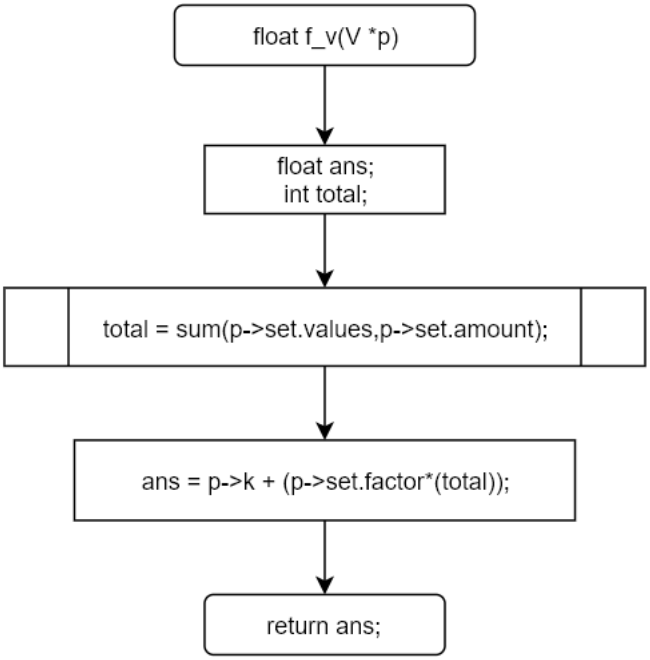
ฟังก์ชัน void input_v(V *p)



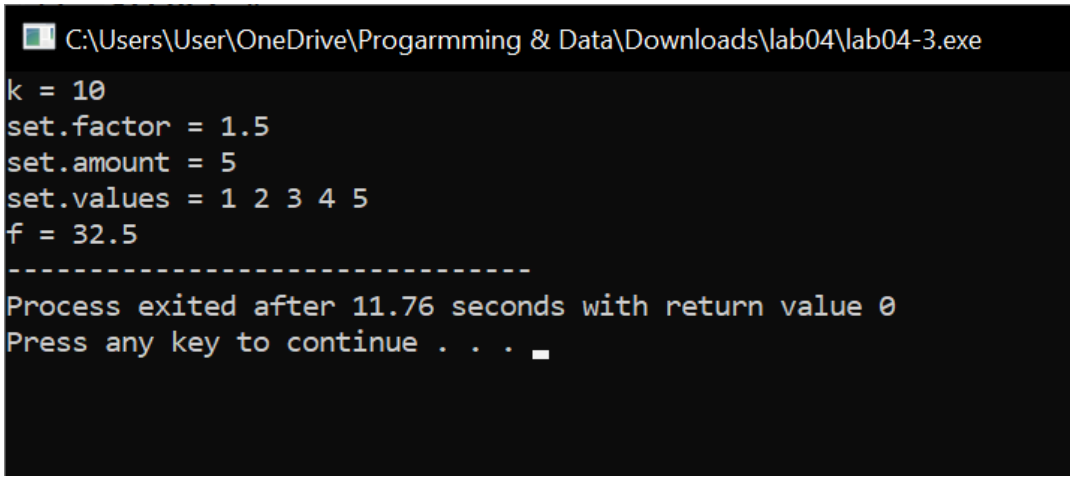
ฟังก์ชัน int sum (int d[ ], int len)



ฟังก์ชัน float f_v (V *p)



ผลการรันโปรแกรม



อธิบายหลักการทำงาน : เป็นโปรแกรมที่รับค่าตัวเลขมาประมวลผลคำนวณหาค่า f โดยมีสูตรดังนี้

f = v.k + (v.set.factor * sum (v.set.values)) ในโปรมแกรมจะมีการประกาศโครงสร้างข้อมูล

- struct set เก็บข้อมูลตัวแปร float factor , int amount และพอยน์เตอร์ int *values และประกาศตัวแปรชนิด SET ในคำสั่ง typedef struct set SET;

- struct v เก็บข้อมูลตัวแปร int k , SET set และประกาศตัวแปรชนิด V ในคำสั่ง typedef struct v V;

โปรแกรมจะมีหลักการทำงานดังนี้ ในฟังก์ชัน main () จะมีการประกาศตัวแปร V v มาเป็นโครงสร้างของจำนวนตัวเลขต่าง ๆ เพื่อมาหาค่าตามสูตรซึ่งมีการใช้ฟังก์ชัน input_v(&v); โดยส่งค่า address ของ v ไปทำการรับค่าตัวเลข จากนั้นแสดงค่าของ “f = %.1f” โดยค่าที่จะแสดงออกมาจะเป็นค่าที่มาจากการเรียกใช้ฟังก์ชัน f_v(&v)โดยส่งค่า address ของ v เพื่อนำค่าที่รับมาไปคำนวณตามสูตรข้างต้น หลังจากนั้นทำการคืนค่า memory ที่ขอด้วยคำสั่ง free(v.set.values);

ฟังก์ชัน void input_v (V *p) จะรับค่า address v จากฟังก์ชัน main () เก็บในตัวแปรพอยน์เตอร์ p และประกาศตัวแปร int i จากนั้นทำการรับค่าตัวเลขจำนวนเต็มเก็บในตัวแปร k (p->k), รับค่าตัวเลขทศนิยมเก็บในตัวแปร factor (p->set.factor) และรับค่าจำนวนตัวเลขที่จะนำมาหาผลรวมเก็บในตัวแปร amount (p->set.amount) จากนั้นขอmemory ที่ว่างไว้ให้กับตัวแปรพอยน์เตอร์ values โดยการ malloc และวนลูปรับค่าของชุดเลขจำนวนเต็มตามจำนวนของตัวแปร amount ภายใต้เงื่อนไข for (i=0 ; i <p->set.amount ; i++) เก็บค่าไว้ในตัวแปรพอยน์เตอร์ values (p->set.values+i)

ฟังก์ชัน f_v (*p) ประกาศตัวแปร float ans เพื่อมารับค่าจากสูตร ans = v.k + (v.set.factor * ผลรวมของจำนวนในอาเรย์) และตัวแปร int total รับค่าจากการเรียกใช้ฟังก์ชัน int sum (int d[ ], int len) ประกาศให้ค่าของ total = sum (p->set.values , p->set.amount); โดยส่งค่า values และค่า amount ต่อมาทำการประมวลผลคำนวณสูตรซึ่งประกาศให้ค่าของ ans = p->k + (p->factor * total); และคืนค่า ans กลับฟังก์ชัน main()

ฟังก์ชัน sum(int d[] , int len); จะรับค่า values เก็บในตัวแปรอาร์เรย์ d และรับค่า amount เก็บในตัวแปร len จากนั้นประกาศค่าตัวแปร i เพื่อวนลูป และให้ค่าตัวแปร ssum = 0 เพื่อเก็บค่าการบวกจำนวนเต็มอาเรย์ ต่อมาทำการวนลูป for (i=0;i<len;i++) เพื่อเก็บค่าผลบวกของแต่ละตำแหน่งอาร์เรย์เก็บในตัวแปร ssum ด้วยคำสั่ง ssum += d[i] ; และคืนค่า ssum กลับฟังก์ชัน f_v (*p)

ความรู้จากการทำ Lab : การเรียกใช้ฟังก์ชันแบบ call by reference คือการส่งค่าไปทั้งอาร์เรย์ และการคืนค่า NULL รวมถึงการใช้คำสั่ง malloc ในการขอ memory ให้กับตัวแปรพอยน์เตอร์และการใช้คำสั่ง free ในการคืนค่า memory กลับ