



รายงาน : Lab08 - Stack

จัดทำโดย

นายกิตติพิศ หนูทอง รหัสนักศึกษา : 6135512003

นายปฏิภาณ วรรณโก รหัสนักศึกษา : 6135512059

Section : 01

240-207 Programing and Data Structures

“งานทั้งหมดนี้ในรายงานฉบับนี้ล้วนเป็นผลงานของข้าพเจ้า มิได้ลอกหรือสำเนาจากที่อื่นใดในกรณีที่พบว่าเกิดสำเนาด้วยวิธีใดก็ตาม ข้าพเจ้ายินดีไม่ขอรับคะแนนจากรายงานฉบับนี้”

คะแนนที่ได้

ลงชื่อ

กิตติพิศ หนูทอง
(นายกิตติพิศ หนูทอง)

ปฏิภาณ วรรณโก
(นายปฏิภาณ วรรณโก)

Lab08 - Stack

ข้อที่ 1 : Postfix-notation

Postfix-notation

จงรับข้อความซึ่งเป็นการเขียนนิพจน์ทางคณิตศาสตร์แบบ Postfix โดยให้ตั้งสมมติฐานว่า นิพจน์ดังกล่าว จะประกอบด้วย ตัวเลขเฉพาะหลักหน่วย และเครื่องหมาย บวก ลบ และคูณเท่านั้น แต่อาจมีวงเล็บหรือไม่มีวงเล็บก็ได้ หลังจากนั้นทำการคำนวณหาค่าผลลัพธ์โดยใช้ stack ในการประมวลผล

รูปแบบการแสดงผล

```
Enter: <921+->
= 6
```

หมายเหตุ อาจใส่ข้อมูลเป็น 9 2 1 + - ก็ได้

**อนุญาตให้ใช้ array แทน linked-list ได้

Code

lab08-1-1.c

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  struct listnode
4  {
5      int data;
6      struct listnode *next;
7  };
8  typedef struct listnode LN;
9  typedef LN *LNP;
10
11 typedef struct
12 {
13     LNP head;
14 }Stack;
15
16 void push(Stack *sptr, int v);
17 int pop(Stack *sptr);
18 void cal(Stack *sptr, char operator);
19
```

```

20 int main()
21 {
22     Stack stack;
23     char s[128];
24     int i,num;
25     printf("Enter: ");
26     gets(s);
27
28     for (i=0;s[i]!='\0';i++)
29     {
30         if(s[i]==' ')
31         {
32             continue;
33         }
34
35         if (s[i] >= '0' && s[i] <= '9')
36         {
37             num = s[i]-48;
38             push(&stack,num);
39         }
40         else
41         {
42             cal(&stack,s[i]);
43         }
44     }
45
46     printf("= %d",pop(&stack));
47     return 0;
48 }
49

```

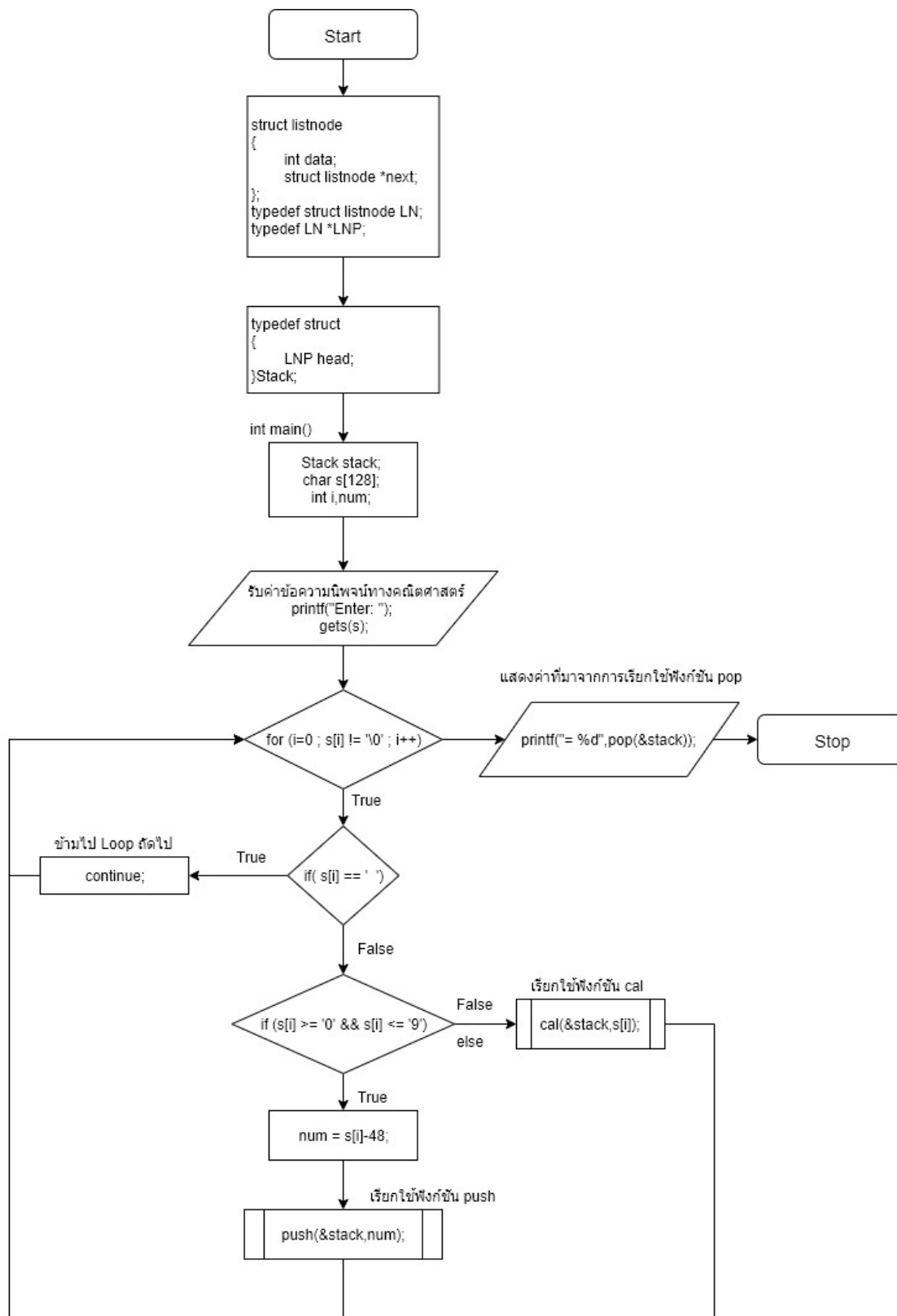
```

50 void push(Stack *sptr, int v)
51 {
52     LN *new_node = NULL;
53     new_node=(LN*)malloc(sizeof(LN));
54     new_node->data = v;
55     new_node->next = sptr->head ;
56     sptr->head = new_node;
57 }
58
59 int pop(Stack *sptr)
60 {
61     int d;
62     d = sptr->head->data;
63     sptr->head = sptr->head->next;
64     return d;
65 }
66 void cal(Stack *sptr, char operator)
67 {
68     int a,b;
69     a = pop(sptr);
70     b = pop(sptr);
71     switch(operator)
72     {
73         case '+' : push(sptr,b+a);break;
74         case '-' : push(sptr,b-a);break;
75         case '*' : push(sptr,b*a);break;
76     }
77 }

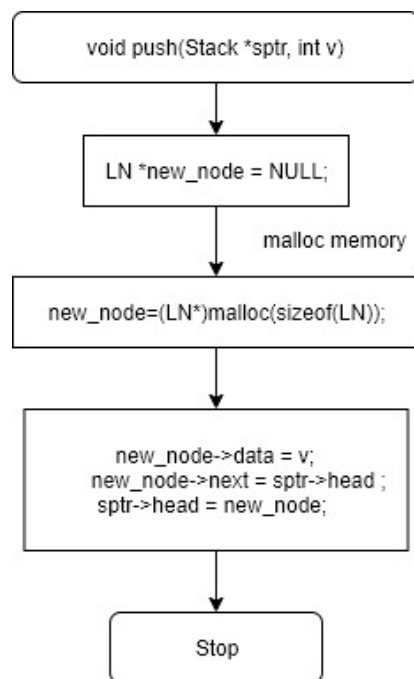
```

Flowchart

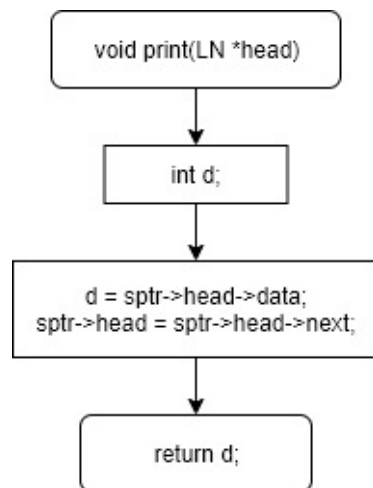
ฟังก์ชัน main()



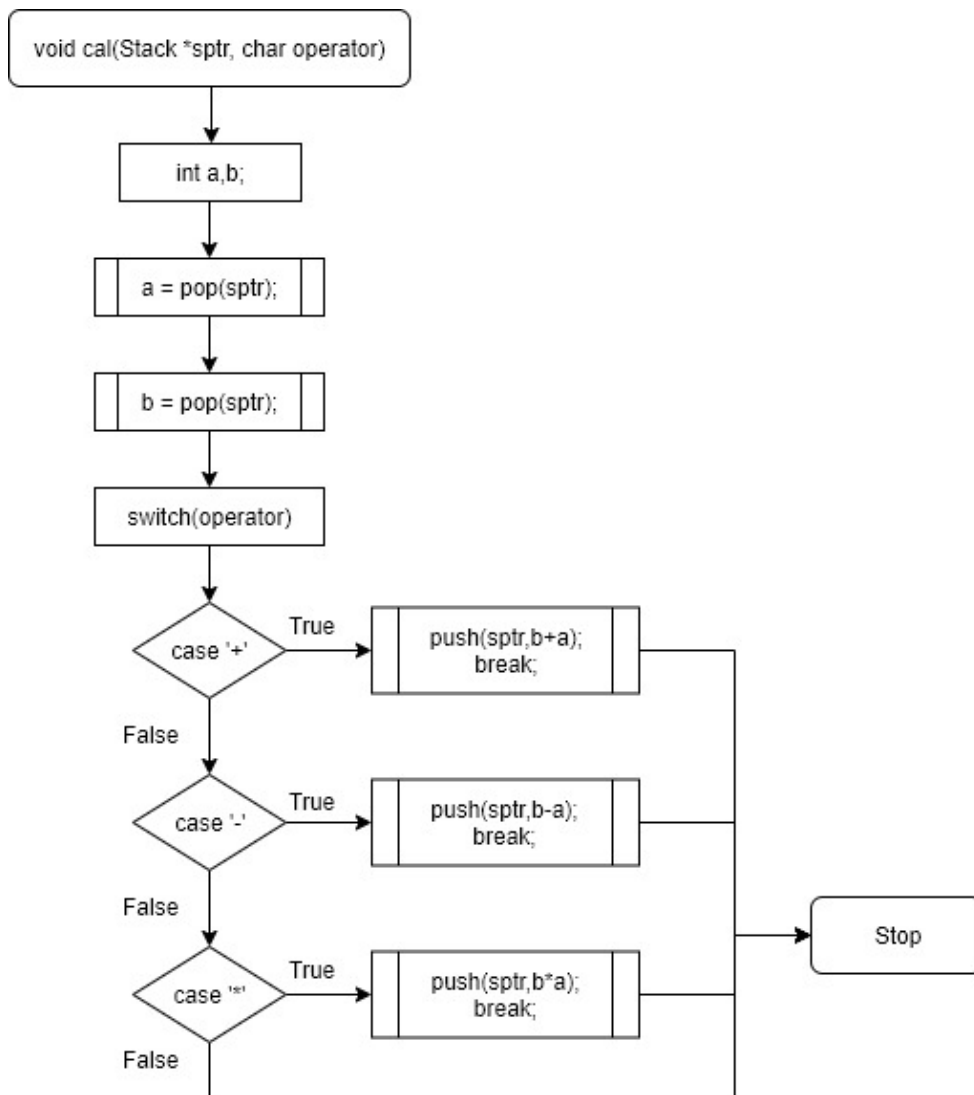
ฟังก์ชัน void push(Stack *sptr, int v)



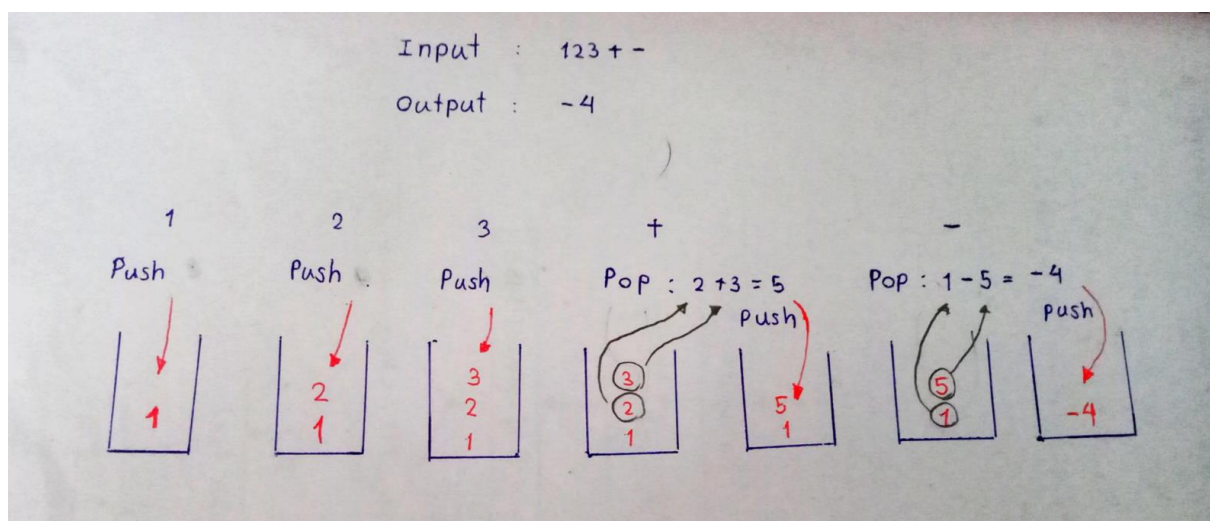
ฟังก์ชัน int pop(Stack *sptr)



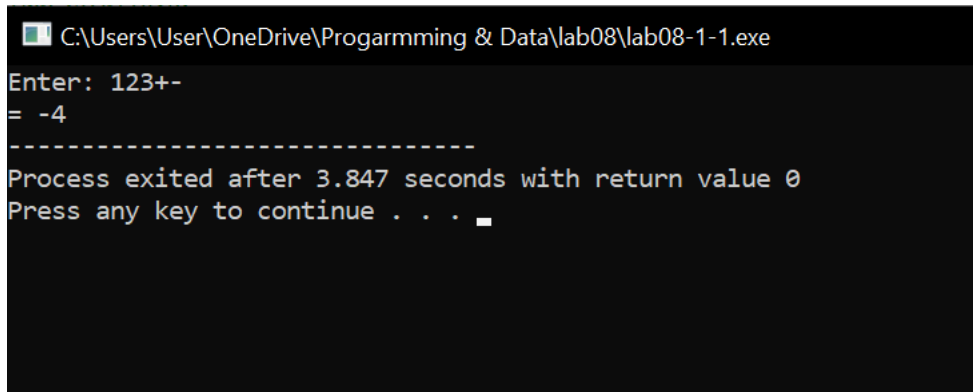
ฟังก์ชัน void cal(Stack *sptr, char operator)



แผนภาพการทำ Postfix-notation



ผลการรันโปรแกรม



```
C:\Users\User\OneDrive\Programming & Data\lab08\lab08-1-1.exe
Enter: 123+-
= -4
-----
Process exited after 3.847 seconds with return value 0
Press any key to continue . . .
```

อธิบายหลักการทำงาน : โปรแกรมนี้เป็นรับข้อความซึ่งเป็นการเขียนนิพจน์ทางคณิตศาสตร์แบบ Postfix เพื่อนำมาบวกลบคูณกันตามอักขระเครื่องหมายที่ได้กำหนดไว้ในการรับค่า

ตัวอย่าง นิพจน์ทางคณิตศาสตร์แบบ Postfix จะเป็นการเขียนแบบ 12- ซึ่งหมายความว่า 1-2 โดยในที่นี้จะใช้ stack รูปแบบเพื่อที่จะใส่ค่าและดึงค่าออกมาใช้งาน จากผลรันโปรแกรมจะให้ 123+- คือการ push ค่า 1 2 3 ลงใน stack เมื่อเจอ “+” จะเป็นตัว pop ค่า 3 และ 2 มาบวกกันจะได้ $2 + 3 = 5$ จากนั้น push ค่าตอบที่เป็น 5 ลงใน stack ต่อมาเมื่อเจอ “-” เป็นการ pop ค่า 5 และ 1 ออกมาลบกัน จะได้ $1 - 5 = -4$ จากนั้น push -4 ลงใน stack หลังจากนั้นทำการแสดงค่า top ของ stack (ค่าของ node ตัวสุดท้ายที่ทำการรับค่าจากการpush) ออกมาทางหน้าจอ ในโปรแกรมจะมีการสร้างโครงสร้างข้อมูล struct listnode ที่ประกอบไปด้วยตัวแปร int data และ pointer listnode *next เพื่อชี้ node ถัดไป และทำการ typedef เป็น *LNP จากนั้นสร้างโครงสร้างข้อมูล struct stack เพื่อนำมาทำ stack ซึ่งประกอบไปด้วย LNP head เพื่อชี้ node และทำการ typedef เป็น Stack

ฟังก์ชัน main() หรือฟังก์ชันหลัก ในฟังก์ชันจะมีการประกาศตัวแปร Stack stack , char s[128] เพื่อรับข้อมูล input , int l เพื่อวน loop และ int num จะเป็นการแปลงค่าตัวอักษรใน char ให้เป็นเลขจำนวนเต็ม โดยการนำค่าตัวอักษรมาลบกับ 48 (ตามรหัส ascii) การทำงานจะเริ่มจากการรับค่าข้อมูล เก็บในตัวแปร s จากนั้นจะเป็นการใช้ loop for (i=0 ; s[i]!='\n' ; i++) ใน loop จะมีการใช้เงื่อนไข if else โดยเงื่อนไขแรก if(s[i]==' ') ตรวจสอบว่าเจอ space bar หรือไม่ ถ้าใช่ให้ใช้คำสั่ง continue เพื่อข้ามไป loop ถัดไป ถ้าไม่ใช่ให้ทำเงื่อนไขถัดไป if (s[i] >= '0' && s[i] <= '9') ตรวจสอบว่าเป็นค่าตัวอักษรเป็นตัวเลข 0-9 หรือไม่ ถ้าใช่ให้ num = s[i]-48; จากนั้นทำการเรียกใช้ฟังก์ชัน push(&stack,num); โดยการส่งค่า &stack และค่าของ num ไป ถ้าไม่ตรงเงื่อนไขนี้ให้ทำการเรียกใช้ฟังก์ชัน cal(&stack,s[i]); โดยการส่งค่า &stack และค่าของ s[i] ไป สุดท้ายจะทำการแสดงค่าที่คืนมาจากการเรียกใช้ฟังก์ชัน int pop(&stack) ออกมาทางหน้าจอ

ฟังก์ชัน `void push(Stack *sptr, int v)` เป็นโปรแกรมที่ใส่ค่า `v` ลงใน stack โดยจะมีการประกาศตัวแปรใหม่ชื่อ `LN *new_node = NULL` เพื่อจะเป็น node ใหม่ และทำการ `malloc` memory ขณะรันโปรแกรม จากนั้นให้ค่าของ `new_node->data=v`; และค่า `new_node->next = sptr->head`; และสุดท้ายให้ `sptr->head = new_node`; จะทำให้ค่าของ `sptr->head` เปลี่ยนไปตาม `new_node` ทำให้ node ถัดไปมาต่อ node แรกซึ่งเป็นการ `insert at front`

ฟังก์ชัน `int pop(Stack *sptr)` เป็นฟังก์ชันที่ใช้ในการดึงค่า `top` ของ stack ออกมาจากฟังก์ชัน ซึ่งจะมีการประกาศตัวแปรใหม่ `int d` มารับค่า `top` ของ stack โดยให้ค่า `d = sptr->head->data`; และทำการเลื่อน stack โดยการ `sptr->head = sptr->head->next`; จากนั้นคืนค่า `d` กลับ

ฟังก์ชัน `void cal(Stack *sptr, char operator)` เป็นฟังก์ชันบวกลบคูณตามค่าของ `operator` ในฟังก์ชันจะมีการประกาศตัวแปร `int a` และ `b` เพื่อรับค่าที่คืนมาจากการเรียกใช้ฟังก์ชัน `pop()` โดยให้ `a` และ `b` เรียกใช้ฟังก์ชัน `pop(sptr)`; โดยส่งค่าของ `sptr` ไป จากนั้นทำการใช้เงื่อนไข `switch(operator)` โดยแบ่งเป็น 3 case ดังนี้

case '+': `push(sptr,b+a);break`; เป็นการส่งผล `b+a` ให้ฟังก์ชัน `push()` และจบฟังก์ชัน

case '-': `push(sptr,b-a);break`; เป็นการส่งผล `b-a` ให้ฟังก์ชัน `push()` และจบฟังก์ชัน

case '*': `push(sptr,b*a);break`; เป็นการส่งผล `b*a` ให้ฟังก์ชัน `push()` และจบฟังก์ชัน

ความรู้จากการทำ Lab Postfix-notation : ได้ความรู้ในการใช้ stack ให้เกิดประโยชน์ ซึ่งในการ `push` จะใช้วิธีการ `Linked List` ในรูปแบบ `insert at front` ที่เป็นการรับข้อมูลมา `insert` ไว้ทางด้านหน้าหรือ `top` ของ stack และในการ `pop` จะเป็นการนำข้อมูลที่อยู่ทางด้านหน้าสุดออกมาใช้ก่อนจากนั้นก็เลื่อนเป็น node ถัดไป