รายงาน

Stack Program

จัดทำโดย

นายธนรัก ชุ่มสวัสดิ์

รหัสศึกษา 68543210018-6

อาจารย์ผู้สอน

นายปิยพล ยืนยงสถาวร

รายงานฉบับนี้เป็นส่วนหนึ่งของวิชา Data Structures and Algorithms

สาขาวิชาวิศวกรรมซอฟต์แวร์ มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา

ภาคเรียนที่ 2 ปีการศึกษา 2568

Code Program Stack

```c
#include <stdio.h>   // use printf()
#include <termios.h> // for custom getch() on macOS
#include <unistd.h>  // for STDIN_FILENO

#define MaxStack 6   // Set Max Stack

int stack[MaxStack]; // Declare Max Stack 0..5
int x;               // Temporary variable
int SP = 0;          // Initial SP=0
char status = 'N';   // Initial Status = NORMAL
char ch;             // KBD Read variable

// getch() replacement for macOS
int getch(void) {
    struct termios oldattr, newattr;
    int ch;

    tcgetattr(STDIN_FILENO, &oldattr);      // get terminal attributes
    newattr = oldattr;
    newattr.c_lflag &= ~(ICANON | ECHO);    // disable buffered I/O and echo
    tcsetattr(STDIN_FILENO, TCSANOW, &newattr);

    ch = getchar();                         // read character

    tcsetattr(STDIN_FILENO, TCSANOW, &oldattr); // restore settings
    return ch;
}

void push(int x)     // PUSH Function
{
    if (SP == MaxStack - 1)
    { // Check Stack FULL?
        printf("!!!OVER FLOW!!!...\n");
        status = 'O'; // set status = OVER FLOW
```

```
    }
    else
    {
        SP = SP + 1;   // Increase SP
        stack[SP] = x; // Put data into Stack
    }
}

int pop() // POP Function
{
    int x;
    if (SP != 0) // Check Stack NOT EMPTY?
    {
        x = stack[SP]; // Get data from Stack
        stack[SP] = 0; // Clear data (optional)
        SP--;          // Decrease SP
        return (x);    // Return data
    }
    else
    {
        printf("\n!!!UNDER FLOW!!!...\n");
        status = 'U'; // set STATUS = "UNDER FLOW"
        return 0;     // avoid warning
    }
}

void ShowAllStack() // Display Function
{
    int i;                          // Counter variable
    printf(" N : %d\n ", MaxStack - 1); // Display N
    printf("Status : %c\n ", status);   // Display STATUS
    printf("SP : %d\n", SP);            // Display SP
    for (i = 1; i < MaxStack; i++)
    {
        printf("%d:%d ", i, stack[i]); // Display all of data in Stack
```

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

int main()
{
    printf("STACK PROGRAM...\n");
    printf("====================\n");

    while (status == 'N')
    {
        printf("[1=PUSH : 2=POP] : "); // Show MENU

        ch = getch();                  // Use custom getch() for macOS
        printf("%c\n", ch);            // echo input manually

        switch (ch)                    // Check ch
        {
        case '1':
            printf("Enter Number : ");
            scanf("%d", &x); // Read data from KBD
            push(x);         // Call PUSH Function
            ShowAllStack();  // Display all data in Stack
            break;

        case '2':
            x = pop();                 // POP data
            printf("Data : %d\n", x);   // Display it
            ShowAllStack();            // Display all data in Stack
            break;
        }
    }

    printf("\n"); // line feed
    return 0;
```

```
}
```

สรุปสิ่งที่แก้ไข

```
int pop() // POP Function
{
   int x;
   if (SP != 0) // Check Stack NOT EMPTY?
   {
      x = stack[SP]; // Get data from Stack
      stack[SP] = 0; // Clear data (optional)
      SP--;          // Decrease SP
      return (x);    // Return data
   }
   else
   {
      printf("\n!!!UNDER FLOW!!!...\n");
      status = 'U'; // set STATUS = "UNDER FLOW"
      return 0;     // avoid warning
   }
}
```

จากเดิมเมื่อ POP จะ ลด SP ลง แต่ค่าที่ Stack[SP] ยังอยู่ (ไม่ถูกลบออก)
ตอนนี้เพิ่มบรรทัดนี้: `stack[SP] = 0`
ทำให้ตัวเลขที่ถูก POP จะหายจาก stack จริงตามที่ต้องการ

อธิบายการทำงานของโปรแกรม

ส่วนประกาศตัวแปรและค่าเริ่มต้น

โปรแกรมเริ่มจากการกำหนดค่า MaxStack = 6 เพื่อสร้าง stack ที่ใช้จริงได้ 5 ช่อง (index 1–5) และประกาศตัวแปรอาร์เรย์ stack[] สำหรับเก็บข้อมูลใน stack รวมถึงตัวแปร SP ที่ทำหน้าที่เป็น Stack Pointer โดยเริ่มต้นถูกตั้งเป็น 0 เพื่อบอกว่า stack ยังว่างอยู่ นอกจากนี้ยังมี status ที่เริ่มต้นเป็น 'N' หมายถึงสถานะปกติ และตัวแปร x สำหรับรับค่าชั่วคราว และ ch สำหรับอ่านคำสั่งจากแป้นพิมพ์ ทั้งหมดนี้เป็นตัวกำหนดสภาวะเริ่มต้นของโปรแกรมก่อนเริ่มใช้งาน stack

```c
#include <stdio.h>   // use printf()
#include <termios.h> // for custom getch() on macOS
#include <unistd.h>  // for STDIN_FILENO

#define MaxStack 6   // Set Max Stack

int stack[MaxStack]; // Declare Max Stack 0..5
int x;               // Temporary variable
int SP = 0;          // Initial SP=0
char status = 'N';   // Initial Status = NORMAL
char ch;             // KBD Read variable
```

ฟังก์ชัน getch()

ฟังก์ชัน getch() ถูกเขียนขึ้นมาเพื่อให้ macOS สามารถรับคีย์จากแป้นพิมพ์ทีละตัวได้ทันทีโดยไม่ต้องกด Enter คล้ายฟังก์ชัน getch() ของ Windows ซึ่งทำโดยการแก้ไขคุณสมบัติของ terminal ชั่วคราว ปิดการ buffering และ echo แล้วอ่านตัวอักษรด้วย getchar() ก่อนจะคืนค่าคุณสมบัติ terminal กลับเหมือนเดิม ฟังก์ชันนี้ช่วยทำให้โปรแกรมตอบสนองต่อการกดปุ่มได้เป็นแบบทันที ทำให้เมนูใช้งานสะดวกขึ้น

```c
int getch(void) {
    struct termios oldattr, newattr;
    int ch;

    tcgetattr(STDIN_FILENO, &oldattr);      // get terminal attributes
    newattr = oldattr;
    newattr.c_lflag &= ~(ICANON | ECHO);    // disable buffered I/O and echo
    tcsetattr(STDIN_FILENO, TCSANOW, &newattr);

    ch = getchar();                         // read character
```

```
    tcsetattr(STDIN_FILENO, TCSANOW, &oldattr); // restore settings
    return ch;
}
```

ฟังก์ชัน PUSH

ฟังก์ชัน push(int x) ทำหน้าที่นำข้อมูลใหม่ใส่ลงบน stack โดยเริ่มจากการตรวจสอบว่าตำแหน่งบนสุด SP ถึงค่าสูงสุดหรือยัง (SP == MaxStack - 1) ซึ่งหมายความว่า stack เต็ม หากเต็มจะเกิด OVERFLOW และเปลี่ยนสถานะ status = 'O' ทำให้โปรแกรมหยุดทำงาน แต่หากยังไม่เต็ม ฟังก์ชันจะเพิ่มค่า SP ขึ้นหนึ่งตำแหน่งและนำค่าที่รับเข้ามา x ไปเก็บไว้ใน stack[SP] ทำให้ข้อมูลใหม่ถูกวางไว้ด้านบนสุดของ stack ตามหลัก LIFO

```
void push(int x)     // PUSH Function
{
    if (SP == MaxStack - 1)
    { // Check Stack FULL?
        printf("!!!OVER FLOW!!!...\n");
        status = 'O'; // set status = OVER FLOW
    }
    else
    {
        SP = SP + 1;   // Increase SP
        stack[SP] = x; // Put data into Stack
    }
}
```

ฟังก์ชัน POP

ฟังก์ชัน pop() ทำหน้าที่นำข้อมูลบนสุดของ stack ออก โดยตรวจสอบก่อนว่า stack ว่างหรือไม่ (SP == 0) หากว่างจะเกิด UNDERFLOW พร้อมทั้งตั้งสถานะ status = 'U' และทำให้โปรแกรมหยุดทำงาน แต่ถ้าไม่ว่าง ฟังก์ชันจะดึงค่าจาก stack[SP] ออกมาเก็บไว้ในตัวแปรชั่วคราว ลบค่าที่ดึงออกโดยตั้ง stack[SP] = 0 เพื่อให้มองเห็นผลการ pop จริง จากนั้นลดค่า SP ลงหนึ่งตำแหน่ง และคืนค่าที่ pop ออกไป ทำให้เป็นไปตามวิธีทำงานของ stack แบบข้อมูลสุดท้ายเข้า–ออกก่อน (LIFO)

```
int pop() // POP Function
{
    int x;
    if (SP != 0) // Check Stack NOT EMPTY?
```

```
    {
        x = stack[SP]; // Get data from Stack
        stack[SP] = 0; // Clear data (optional)
        SP--;          // Decrease SP
        return (x);    // Return data
    }
    else
    {
        printf("\n!!!UNDER FLOW!!!...\n");
        status = 'U'; // set STATUS = "UNDER FLOW"
        return 0;     // avoid warning
    }
}
```

ฟังก์ชัน ShowAllStack

ฟังก์ชัน ShowAllStack() ทำหน้าที่แสดงสถานะทั้งหมดของ stack เพื่อนำเสนอข้อมูลแก่ผู้ใช้ โดยเริ่มจากการพิมพ์จำนวนตำแหน่งที่สามารถใช้งานได้ (MaxStack - 1), สถานะปัจจุบัน (status), ค่า Stack Pointer (SP) และตามด้วยการแสดงข้อมูลทุกตำแหน่งของ stack ตั้งแต่ index 1 ถึง 5 ทั้งที่มีข้อมูลและไม่มีข้อมูล เพื่อให้ผู้ใช้เห็นโครงสร้างและการเปลี่ยนแปลงของ stack อย่างชัดเจนหลังการ push หรือ pop

```
void ShowAllStack() // Display Function
{
    int i;                          // Counter variable
    printf(" N : %d\n ", MaxStack - 1); // Display N
    printf("Status : %c\n ", status);   // Display STATUS
    printf("SP : %d\n", SP);            // Display SP
    for (i = 1; i < MaxStack; i++)
    {
        printf("%d:%d ", i, stack[i]); // Display all of data in Stack
    }
    printf("\n-----------------------------------------------------------\n");
}
```

ฟังก์ชัน main()

ในฟังก์ชัน main() โปรแกรมเริ่มจากการแสดงหัวข้อ จากนั้นเข้าสู่วงวนที่ทำงานต่อเนื่องตราบเท่าที่สถานะ status == 'N' หมายถึงยังไม่มี overflow หรือ underflow เกิดขึ้น โดยภายในลูปจะแสดงเมนู [1=PUSH : 2=POP] ให้ผู้ใช้เลือก และอ่านคำสั่งแบบตัวอักษรทันทีด้วย getch() หากผู้ใช้กด 1 โปรแกรมจะรับค่าตัวเลขแล้วเรียก push() เพื่อนำข้อมูลใส่ลง stack และแสดงผลด้วย ShowAllStack() แต่ถ้ากด 2

โปรแกรมจะเรียก pop() เพื่อนำข้อมูลออกแล้วแสดงผลเช่นกัน วนไปเรื่อย ๆ จนกว่าจะเกิด overflow หรือ underflow จึงออกจากลูปและจบการทำงาน

```c
int main()
{
    printf("STACK PROGRAM...\n");
    printf("====================\n");

    while (status == 'N')
    {
        printf("[1=PUSH : 2=POP] : "); // Show MENU

        ch = getch();                  // Use custom getch() for macOS
        printf("%c\n", ch);            // echo input manually

        switch (ch)                    // Check ch
        {
        case '1':
            printf("Enter Number : ");
            scanf("%d", &x); // Read data from KBD
            push(x);         // Call PUSH Function
            ShowAllStack();  // Display all data in Stack
            break;

        case '2':
            x = pop();                 // POP data
            printf("Data : %d\n", x);   // Display it
            ShowAllStack();            // Display all data in Stack
            break;
        }
    }

    printf("\n"); // line feed
    return 0;
}
```

ผลลัพธ์ของโปรแกรม

การ PUSH

```
STACK PROGRAM...
==================
[1=PUSH : 2=POP] : 1
Enter Number : 10
 N : 5
 Status : N
 SP : 1
1:10 2:0 3:0 4:0 5:0
------------------------------------------------------------
[1=PUSH : 2=POP] :

[1=PUSH : 2=POP] : 1
Enter Number : 20
 N : 5
 Status : N
 SP : 2
1:10 2:20 3:0 4:0 5:0
------------------------------------------------------------
[1=PUSH : 2=POP] :

[1=PUSH : 2=POP] : 1
Enter Number : 30
 N : 5
 Status : N
 SP : 3
1:10 2:20 3:30 4:0 5:0
------------------------------------------------------------
[1=PUSH : 2=POP] :

[1=PUSH : 2=POP] : 1
Enter Number : 50
 N : 5
 Status : N
 SP : 4
1:10 2:20 3:30 4:50 5:0
------------------------------------------------------------
[1=PUSH : 2=POP] :

[1=PUSH : 2=POP] : 1
Enter Number : 10
 N : 5
 Status : N
 SP : 5
1:10 2:20 3:30 4:50 5:10
------------------------------------------------------------
[1=PUSH : 2=POP] :

[1=PUSH : 2=POP] : 1
Enter Number : 10
!!!OVER FLOW!!!...
 N : 5
 Status : O
 SP : 5
1:10 2:20 3:30 4:50 5:10
------------------------------------------------------------
```

การ POP

```
STACK PROGRAM...
==================
[1=PUSH : 2=POP] : 1
Enter Number : 10
 N : 5
 Status : N
 SP : 1
1:10 2:0 3:0 4:0 5:0
------------------------------------------------------------
[1=PUSH : 2=POP] :

[1=PUSH : 2=POP] : 1
Enter Number : 20
 N : 5
 Status : N
 SP : 2
1:10 2:20 3:0 4:0 5:0
------------------------------------------------------------
[1=PUSH : 2=POP] :

[1=PUSH : 2=POP] : 2
Data : 20
 N : 5
 Status : N
 SP : 1
1:10 2:0 3:0 4:0 5:0
------------------------------------------------------------
[1=PUSH : 2=POP] : 2
Data : 10
 N : 5
 Status : N
 SP : 0
1:0 2:0 3:0 4:0 5:0
------------------------------------------------------------
[1=PUSH : 2=POP] : 2

!!!UNDER FLOW!!!...
Data : 0
 N : 5
 Status : U
 SP : 0
1:0 2:0 3:0 4:0 5:0
------------------------------------------------------------
```