4190.103A-001: Programming Practice Lab. 10

Jaewon Choi
Keunsan Park
Haeun Lee
(snucsl.ta@gmail.com)

Systems Software & Architecture Lab.

Seoul National University

Spring 2023

Lab. 9 실습 풀이

실습 1

Q. 간단한 Stack을 구현 해 봅시다.

크기가 100인 배열을 통해 stack을 구현 해 봅시다.

입력: 구현 된 stack에 대하여 일련의 push, pop 명령이 수행 됩니다.
단, stack이 가득 찼을때 push 명령이 들어오면, -1을 출력합니다.

출력: pop명령이 왔을 때, 반환되는 값을 출력합니다.
단, stack이 비어있는경우, -1을 반환합니다.

명령은 아래와 같이 char로 주어집니다.
push: 'i'

```
i 5
```

pop: 'o'
프로그램 종료: 'q'

```c
#include <stdio.h>

int stack[100]={0,};
int top=0;

int push(int data){
    if(top == 100)
        return -1;
    stack[top++] = data;
}
int pop(){
    if(top == 0)
        return -1;
    return stack[--top];
}

int main()
{
    int data,res;
    char command;
    do{
        scanf("%c",&command);
        if(command == 'i'){
            scanf("%d",&data);
            res = push(data);
            if(res == -1)
                printf("%d\n",res);
        }
        else if(command == 'o'){
            res = pop();
            printf("%d\n",res);
        }
    }while(command != 'q');
    return 0;
}
```

실습 2

Q. Stack을 활용하여 청개구리 프로그램을 만들어봅시다.

청개구리 프로그램은 사용자가 입력한 내용을 항상 뒤집어 따라합니다.

청개구리 프로그램은 다음과 같은 조건을 만족합니다!

- 개행문자(엔터 - \n) 가 입력되기 이전까지 사용자가 입력한 내용을 뒤집어서 따라합니다.
- 입력이 단일 'q' 문자로 주어질 때 까지 프로그램을 반복합니다

사용자는 개행문자 이전의 문자를 최대 100개까지만 입력한다고 가정합니다

- 최대 100개까지의 입력을 뒤집어서 따라하는 프로그램을 만드시면 됩니다
- 넘치는 입력은 없으니 넘치는 경우는 고려하지 않으셔도 됩니다

```
while(1){
    do{
        scanf("%c",&c);
        if(c == 'q'){
            if(top != 0){
                push(c);
            }
            else{
                char t;
                scanf("%c",&t);

                if(t == '\n')
                    return 0;
                else{
                    push(c);
                    push(t);
                }
            }
        }
        else if (c != '\n')
            push(c);
    }while(c != '\n');

    printReverse();
    top = 0;
}
```

실습 3

Q. 간단한 Queue를 구현 해 봅시다.

크기가 100인 배열을 통해 Queue를 구현 해 봅시다.

입력: 구현 된 queue에 대하여 일련의 push, pop 명령이 수행 됩니다.
단, queue에 더이상 push가 불가능 할 때 push 명령이 들어오면, -1을 출력합니다.

출력: pop명령이 왔을 때, 반환되는 값을 출력합니다.
단, queue가 비어있는경우, -1을 반환합니다.

명령은 아래와 같이 char로 주어집니다.
push: 'i'

```
i 5
```

pop: 'o'
프로그램 종료: 'q'

```
int push(int data){
    if(tail == 100){
        return -1;
    }

    queue[tail++] = data;
}
int pop(){
    if(head == tail)
        return -1;
    return queue[head++];
}
```

```
int push(int data){
    if(tail == 100){
        if(head == 0)
            return -1;
        else{
            for(int i=head;i<tail;i++){
                queue[i-head] = queue[i];
            }
            head = 0; tail -= head;
        }
    }

    queue[tail++] = data;
}
int pop(){
    if(head == tail)
        return -1;
    return queue[head++];
}
```

실습 4

Q. Queue를 이용해서 다음의 문제를 해결할 수 있는 프로그램을 만들어봅시다.

트럭이 줄줄이 다리를 건너려고 합니다.
다리가 1차선이어서 트럭들이 반드시 순서대로 건너야합니다.
안타깝게도 다리에 무게 제한이 있습니다.
다리의 무게 제한을 지키면서 모든 트럭이 건너는데 걸리는 최소 시간을 구하는 프로그램을 작성해보세요

- 트럭의 개수는 최대 1000개 입니다
- 트럭의 무게, 다리의 길이, 다리의 최대 하중은 모두 정수 입니다
- 건너지 못하는 예시는 없다고 가정합니다(트럭이 다리가 견딜 수 있는 무게보다 무거운 경우)
- 입력은 다음과 같습니다
  - 2줄의 입력이 들어옵니다
  - 첫 줄에는 '[트럭 수] [다리 길이] [다리 최대 하중]'이 입력됩니다(예 : 4 2 10)
  - 두 번째 줄에는 트럭의 수 만큼 무게가 입력됩니다(예 : 7 4 5 6)

주어진 입력에 따라서 트럭이 다리를 모두 건너는데 걸리는 최소 시간을 계산하여 출력해보세요

```
while(1)
{
    time++; // 매 반복마다 time을 증가시킵니다.

    /* Write your code here */
    // step 1: 트럭들이 다리에 있었던 시간을 관리하고, 시간이 충족된 트럭을 탈출시킴
    if(queue[front+1] >= bridge_len){
        total_weight -= truck[front+1];
        front++;
    }

    for(int i=front+1;i<=rear;i++){
        queue[i] += 1;
    }
    // step 2: 무게를 넘지 않는 조건에서 트럭을 다리에 진입시킴
    if(rear != truck_num-1){
        total_weight += truck[rear+1];
        if(total_weight > limit)
            total_weight -= truck[rear+1];
        else
            queue[++rear] = 1;
    }
    // step 3: 마지막 트럭이 다리에 진입하였는지 확인
    if(front != -1 && front == rear)
        break;

    // printf("front : %d, rear = %d\nqueue : ",front,rear);
    // for(int i=front+1;i<=rear;i++)
    //     printf("%d ",queue[i]);
    // printf("\n");

}
```

Lab. 9 과제 풀이

과제 1

Q. 괄호를 맞게 코딩했는지 확인해 봅시다.

코드를 짜다보면, 괄호를 잘못사용하여 에러가 발생하는 경우가 종종 있습니다.
코드를 입력 받고, 괄호를 맞게 작성하였는지 확인하는 프로그램을 작성 해 봅시다.

- 괄호를 제외한 다른 문법적 내용은 전혀 고려하지 않습니다.

입력으로는,

- 첫째 줄에 양의 정수 n을 입력 받습니다.
- 둘째 줄부터 n개의 줄로 이루어진 코드를 입력 받습니다.
- 각 줄은 개행문자('\n')로 구분되어 있습니다.
- 괄호 쌍은 100개를 넘지 않습니다.

출력으로는,

- 괄호가 맞게 작성 된 경우 'Y',
- 틀린경우 'N'을 출력합니다.

```
int main()
{
    int n;
    char pt;

    scanf("%d\n", &n);
    for(int i=0;i<n;i++){
        do{
            scanf("%c",&pt);
            if(pt == '{' || pt == '(')
                push(pt);
            else if(pt == '}' || pt == ')')
                pop(pt);
        }while(pt != '\n');
    }
    if(top == -1)
        printf("Y\n");
    else
        printf("N\n");
    return 0;
}
```

과제 2

Q. 놀이공원을 시뮬레이션 해 봅시다.

10 명(0번부터 9번)의 사람들이 놀이공원에 갔습니다.
놀이공원에는 3개(0번 부터 2번)의 놀이기구가 있습니다.
놀이기구를 타기위해서는 각 놀이기구에 대기를 걸어야하는데,
줄서는건 귀찮으므로, 각 놀이기구에서 대기번호를 받으면 됩니다.
하지만 순서가 왔을때 다른 놀이기구를 타고 있었다면, 기회는 사라지고 다음 순번의 사람이 탑승합니다.

각 놀이기구에 총 몇명의 사람들이 탔는지 출력하시오 (중복 포함)

- 한 놀이기구에서 다른 놀이기구를 타러가는 시간은 '0'입니다.
- 동시에 두 놀이기구에 탈 기회가 온 경우, 작은 번호의 놀이기구에 탑승하게 됩니다.
- 중복을 포함하며, 0번 놀이기구를 1번 사람이 2번, 4번 사람이 3번 탔다면 총 5명이 탑승한것으로 계산합니다.

```
// Write your code here
for(int i=0;i<MAX_PEOPLE;i++)
    ride[i] = -1;
while(1){
    // 1. 시간 측정
    time++;

    // 1-1. 주행시간이 종료된 놀이기구의 현재 탑승인원 하차
    for(int i=0;i<MAX_RIDE;i++){
        if(time % runTime[i] == 0){
            for(int j=0;j<MAX_PEOPLE;j++){
                if(ride[j] == i)
                    ride[j] = -1;
            }
            // 2. 3개의 놀이기구에 대하여 loop
            int board = 0;
            do{
            // 2-1. 대기열에서 pop
                int idx_person = pop(i);
                if(idx_person == -1)
                    break;
            // 2-2. 다른 놀이기구에 탑승하였는지 확인
                if(ride[idx_person] != -1)
                    continue;
            // 2-3. 탑승 가능인원인 경우 count check
                ride[idx_person] = i;
                board++;
            }while(board < limit[i]);
            cnt[i] += board;
        }
    }
    // 3. 종료 조건 확인
    int flag=0;
    for(int i=0;i<MAX_PEOPLE;i++){
        if(ride[i] != -1){ //1명이라도 타고 있으면
            flag = 1;
            break;
        }
    }
    if(!flag)
        break;
```

실습 & 과제

실습

실습 1 - 2의 보수

Step 1) N을 이진수로 바꾸어 bit[ ] 배열에 저장.
e.g) 10 = 1010 –> bit[0]에 0 저장, … bit[3]에 1 저장

Step 2) bit[ ] 배열과 1과 0을 반대로 bitReversed[ ] 배열에 저장한 후,
1을 더함. 이 때 2가 된 경우 올림 연산 수행.
e.g.) bitReversed[0]에 1을 더한 후, 0번째 자리 ~ 31번째 자리까지를 확인

Step 3) bit[ ] 배열과 bitReversed[ ] 배열에서 서로 다른 원소의 갯수를 카운트.

실습 2 - 2의 제곱

Method) 모듈러 연산(%)을 사용하는 대신에
주어진 mask 변수와 left shift(<<) 연산을 활용해 코드 작성.

실습 3 - 두 시간대의 차이

Method 1) scanf("%02d : %02d : %02d", &hour, &minutes, &seconds);
형식으로 입력 받을 수 있다. **(간편)**

Method 2) 입력을 string으로 받은 경우 , string -> int로 변환 필요
a) scanf("%[^₩n]s", str); // e.g) "abcd₩0" (버퍼에 ₩n 남음)
b) gets(str); // e.g) "abcd₩0" (버퍼에 ₩n 안남음)
c) fgets(str, size, stdin); // e.g) "abcd₩n₩0" (버퍼에 ₩n 안남음)
-> size가 입력된 크기보다 1만 클땐 "abcd₩0" (버퍼에 ₩n 남음)

```
char *fgets(char *str, int n, FILE *stream)
```

```
char * gets(char * str)
```

Method 2-1) sscanf(), atoi() 등등을 활용 가능.

실습 4 - 1과0

Method 1) N명의 사람들에 대해 N x (N−1) / 2 만큼 연산을 해준다.

Method 2) 이마의 숫자를 모두 이진수로 바꾼 후, 각 자리수를 기준으로 계산해준다.

```
1 0 0 1 1 = 19
0 1 0 1 0 = 10
---------------
1 1 0 0 1 = 25
```

e.g) 마을 사람의 숫자 모두에서
4번째 자리(2^3)에 위치한 1의 갯수 a 와 0의 갯수 b를 세면,
a*b * 2^3 을 한 값이 4번째 자리에 대한 총 합이됨.
(xor연산은 서로 다를 경우 1이 되므로)
-> array[32]를 선언해 1의 갯수 카운트
-> N − array[i] 는 0의 갯수가 됨

과제

과제 1 - 비밀번호 찾기

Method) i 번째 행 j 번째 열의 원소는 배열의 i 번째 원소와 j 번째 연산의 And(&) 연산이므로, i 번째 배열의 원소값의 최소값을 구하기 위해 어떤 연산을 해야할 지 생각해본다.

과제 2 - N개의 보트

비트 마스킹이란?

- 컴퓨터는 내부적으로 모든 자료를 이진수로 표현.
  -> 이와 같은 특성을 이용해 정수의 **이진수 표현을 자료구조**로 쓰는 기법을 비트 마스크라고 한다.

- 비트 마스크를 이용하면 더 빠른 수행 시간, 더 간결한 코드, 더 적은 메모리 사용이라는 효과를 얻을 수 있다.

과제 2 - N개의 보트

비트 마스킹으로 풀 경우

Step 1) M번의 명령을 받아 번호에 따라 맞게 수행.
이 때 shift 연산(<<, >>)과 and 연산(&) or 연산(|) not 연산(~)을 활용할 수 있다.
e.g) 2번째 보트 3번째에 탑승 : boat[2] |= (1 << 3);
2번째 보트 3번째 하차 : boat[2] &= ~(1 << 3);

Step 2) 보트에 타고 있는 승객 패턴에서 중복되는 패턴을 제외하고 그 수를 세어 준다.
boat[ ] 배열에 각 원소는 정수값이므로, 정렬을 한 후 세 주거나,
check 배열를 만들어 세 주는 등, 다양한 방법이 있다.

3	8	2

3 = ..0011 이므로 1번째 칸과 2번째칸에 탑승해 있는 상태

실습 / 과제

- 과제 내용
  - 당일 진행한 실습과 관련된 문제
- 제출 기한
  - 실습 : 당일 자정까지, 다음날 자정까지 지각제출 허용 (단, 점수 -30%)
  - 과제 : 다음주 수요일 23:59 까지, 다음날 자정까지 지각제출 허용 (단, 점수 -30%)
- 제출 방법
  - Elice의 Submit 기능 활용