자료구조 및 알고리즘 레포트 7주차

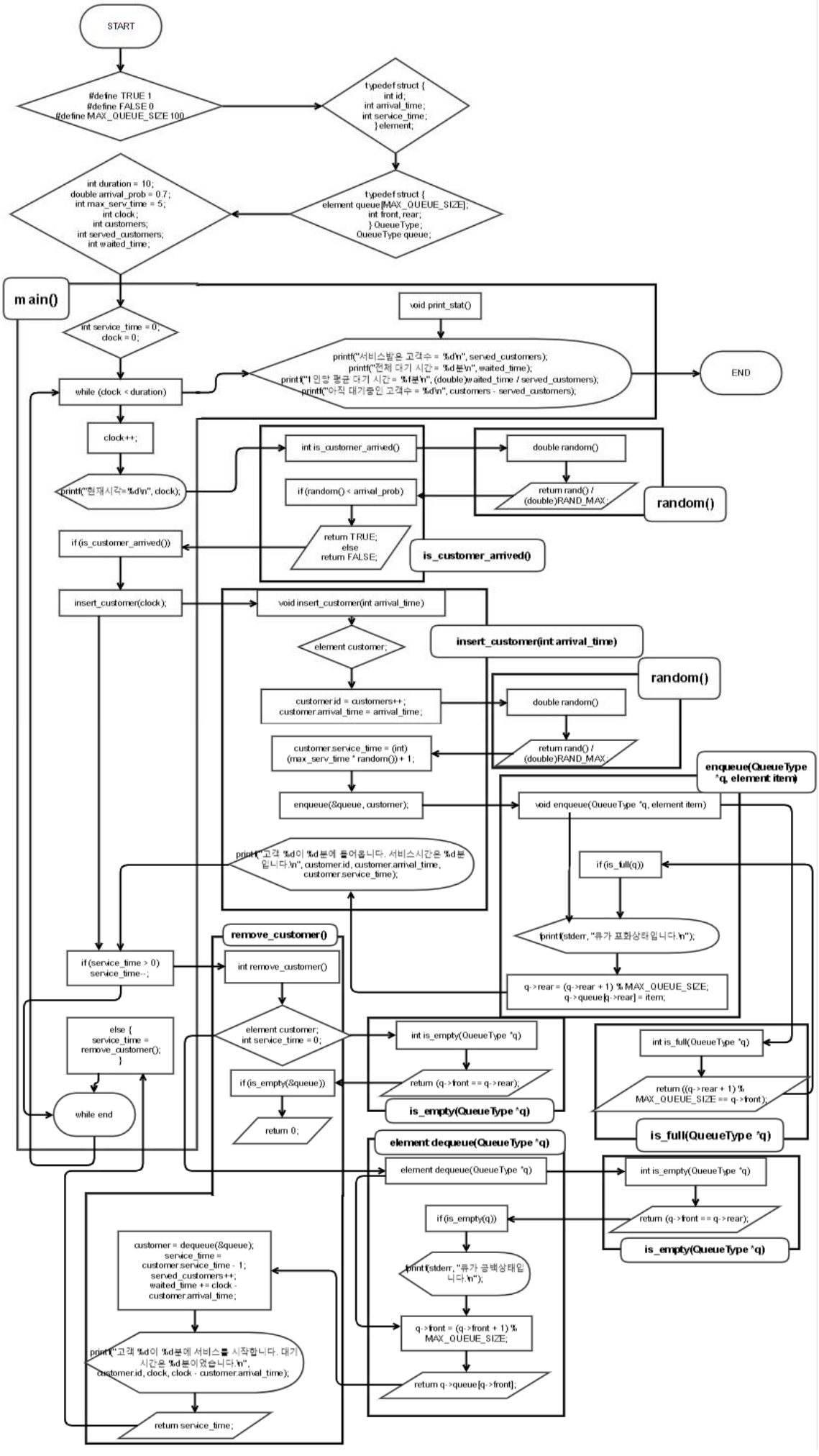
임베디드시스템공학과
2015146003
김기덕

스택을 사용하여 중위 표기식을 후위 표기식으로 변환 하기

1) 프로그램 함수 기능

init 함수 : front와 rear 변수를 0으로 초기화 한다.
is_empty 함수 : 큐가 공백상태라면 1을 아니라면 0을 반환한다.
is_full 함수 : 큐가 포화상태라면 1을 아니라면 0을 반환한다.
enqueue 함수 : 삽입함수로 포화상태가 1이라면 “큐가 포화상태입니다.”라는 문구를 출력하고 큐의 item을 삽입한다.
포화상태가 아니라면 문구출력 없이 삽입한다.
dequeue 함수 : 삭제함수로 공백상태가 1이라면 “큐가 공백상태입니다.”라는 문구를 출력하고 큐 배열의 위치를 +1 한 후 반환한다.
공백상태가 아니라면 문구출력 없이 위치 이동한 후 반환한다.
random 함수 : 0과 1사이의 난수를 반환한다.
is_customer_arrived 함수 : 난수가 하나의 시간 단위에 도착하는 평균 고객의 수인 0.7의 값을 가진 arrival_prob라는 변수보다 작으면 참, 크면 거짓을 반환한다.
insert_customer 함수 : 고객 id를 1씩 증가시키고 도착시간을 입력받고 서비스 시간도 랜덤 값으로 입력받는다. 그 값들을 enqueue 함수로 큐에 삽입시키고 고객의 id와 도착시간, 서비스시간을 출력해준다.
remove_customer 함수 : 큐가 공백이라면 0값을 반환하고 아니라면 dequeue함수로 위치가 이동된 큐 배열의 위치를 입력받는다. served_customers 변수를 +1 추가시키고 현재시간에 고객 도착시간을 빼준 값을 전체 대기시간에 추가한 뒤 고객 id, 현재시간, 대기시간을 출력해준다.
print_stat 함수 : 서비스 받은 고객 수, 전체 대기시간, 1인당 평균 대기시간, 대기중인 고객수를 출력한다.

2) 플로우 차트


START
#define TRUE 1
#define FALSE 0
#define MAX_QUEUE_SIZE 100
typedef struct {
int id;
int arrival_time;
int service_time;
} element;
int duration = 10;
double arrival_prob = 0.7;
int max_serv_time = 5;
int clock;
int customers;
int served_customers;
int waited_time;
typedef struct {
element queue[MAX_QUEUE_SIZE];
int front, rear;
} QueueType;
QueueType queue;
main()
int service_time = 0;
clock = 0;
while (clock < duration)
clock++;
printf("현재시각=%d\n", clock);
if (is_customer_arrived())
insert_customer(clock);
if (service_time > 0)
service_time--;
else {
service_time =
remove_customer();
}
while end
void print_stat()
printf("서비스받은 고객수 = %d\n", served_customers);
printf("전체 대기 시간 = %d분\n", waited_time);
printf("1인당 평균 대기 시간 = %f분\n", (double)waited_time / served_customers);
printf("아직 대기중인 고객수 = %d\n", customers - served_customers);
END
int is_customer_arrived()
if (random() < arrival_prob)
return TRUE;
else
return FALSE;
is_customer_arrived()
double random()
return rand() / (double)RAND_MAX;
random()
void insert_customer(int arrival_time)
element customer;
customer.id = customers++;
customer.arrival_time = arrival_time;
customer.service_time = (int)(max_serv_time * random()) + 1;
enqueue(&queue, customer);
printf("고객 %d이 %d분에 들어옵니다. 서비스시간은 %d분입니다.\n", customer.id, customer.arrival_time, customer.service_time);
insert_customer(int arrival_time)
random()
enqueue(QueueType *q, element item)
void enqueue(QueueType *q, element item)
if (is_full(q))
printf(stderr, "큐가 포화상태입니다.\n");
q->rear = (q->rear + 1) % MAX_QUEUE_SIZE;
q->queue[q->rear] = item;
remove_customer()
int remove_customer()
element customer;
int service_time = 0;
if (is_empty(&queue))
return 0;
int is_empty(QueueType *q)
return (q->front == q->rear);
is_empty(QueueType *q)
int is_full(QueueType *q)
return ((q->rear + 1) % MAX_QUEUE_SIZE == q->front);
is_full(QueueType *q)
element dequeue(QueueType *q)
element dequeue(QueueType *q)
if (is_empty(q))
printf(stderr, "큐가 공백상태입니다.\n");
q->front = (q->front + 1) % MAX_QUEUE_SIZE;
return q->queue[q->front];
int is_empty(QueueType *q)
return (q->front == q->rear);
is_empty(QueueType *q)
customer = dequeue(&queue);
service_time = customer.service_time - 1;
served_customers++;
waited_time += clock - customer.arrival_time;
printf("고객 %d이 %d분에 서비스를 시작합니다. 대기시간은 %d분이었습니다.\n", customer.id, clock, clock - customer.arrival_time);
return service_time;

3) 전체 프로그램 코드 및 실행화면

```
#define _CRT_SECURE_NO_WARNINGS
#include "stdlib.h"
#include "stdio.h"
#include "math.h"
#include <Windows.h>
#define TRUE 1
#define FALSE 0
#define MAX_QUEUE_SIZE 100
typedef struct {
	int id;
	int arrival_time;
	int service_time;
} element;
typedef struct {
	element queue[MAX_QUEUE_SIZE];
	int front, rear;
} QueueType;
QueueType queue;
// 시뮬레이션에 필요한 여러 가지 상태 변수
int duration = 10; // 시물레이션 시간
double arrival_prob = 0.7; // 하나의 시간 단위에 도착하는 평균 고객의 수
int max_serv_time = 5; // 하나의 고객에 대한 최대 서비스 시간
int clock;
// 시뮬레이션의 결과
int customers; // 전체 고객 수
int served_customers; // 서비스 받은 고객 수
int waited_time; // 고객들이 기다린 시간
// 초기화 함수
void init(QueueType *q)
{
	q->front = q->rear = 0;
}
// 공백 상태 검출 함수
int is_empty(QueueType *q)
{
	return (q->front == q->rear);
}
// 포화 상태 검출 함수
int is_full(QueueType *q)
```

```
{
        return ((q->rear + 1) % MAX_QUEUE_SIZE == q->front);
}
// 삽입 함수
void enqueue(QueueType *q, element item)
{
        if (is_full(q)) {
                fprintf(stderr, "큐가 포화상태입니다.\n");
                return;
        }
        q->rear = (q->rear + 1) % MAX_QUEUE_SIZE;
        q->queue[q->rear] = item;
}
// 삭제 함수
element dequeue(QueueType *q)
{
        if (is_empty(q)) {
                fprintf(stderr, "큐가 공백상태입니다.\n");
                exit(1);
        }
        q->front = (q->front + 1) % MAX_QUEUE_SIZE;
        return q->queue[q->front];
}
// [0..1] 사이의 난수 생성 함수
double random()
{
        return rand() / (double)RAND_MAX;
}
// 랜덤 숫자를 생성하여 고객이 도착했는지 도착하지 않았는지를 판단
int is_customer_arrived()
{
        if (random() < arrival_prob)
                return TRUE;
        else
                return FALSE;
}
// 새로 도착한 고객을 큐에 삽입
void insert_customer(int arrival_time)
{
        element customer;
        customer.id = customers++;
```

```
        customer.arrival_time = arrival_time;
        customer.service_time = (int)(max_serv_time * random()) + 1;
        enqueue(&queue, customer);
        printf("고객 %d이 %d분에 들어옵니다. 서비스시간은 %d분입니다.\n",
customer.id, customer.arrival_time, customer.service_time);
}
// 큐에서 기다리는 고객을 꺼내어 고객의 서비스 시간을 반환한다.
int remove_customer()
{
        element customer;
        int service_time = 0;
        if (is_empty(&queue))
                return 0;
        customer = dequeue(&queue);
        service_time = customer.service_time - 1;
        served_customers++;
        waited_time += clock - customer.arrival_time;
        printf("고객 %d이 %d분에 서비스를 시작합니다. 대기시간은 %d분이었습니다.\n",
                customer.id, clock, clock - customer.arrival_time);
        return service_time;
}
// 통계치를 출력한다.
void print_stat()
{
        printf("서비스받은 고객수 = %d\n", served_customers);
        printf("전체 대기 시간 = %d분\n", waited_time);
        printf("1인당 평균 대기 시간 = %f분\n", (double)waited_time /
served_customers);
        printf("아직 대기중인 고객수 = %d\n", customers - served_customers);
}
// 시뮬레이션 프로그램
void main()
{
        system("title Kimkideok");

        int service_time = 0;
        clock = 0;
        while (clock < duration) {
                clock++;
                printf("현재시각=%d\n", clock);
                if (is_customer_arrived()) {
```

```
                insert_customer(clock);
            }
            if (service_time > 0)
                service_time--;
            else {
                service_time = remove_customer();
            }
        }
        print_stat();
}
```