/\*

240403 큐의 응용프로그램

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

// ===== 원형큐 코드 시작 ======

#define MAX\_QUEUE\_SIZE 5

typedef int element;

typedef struct { // 큐 타입

element data[MAX\_QUEUE\_SIZE];

int front, rear;

} QueueType;

// 오류 함수

void error(const char\* message)

{

fprintf(stderr, "%s\n", message);

exit(1);

}

// 공백 상태 검출 함수

void init\_queue(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);

}

// 원형큐 출력 함수

void queue\_print(QueueType\* q)

{

printf("QUEUE(front=%d rear=%d) = ", q->front, q->rear);

if (!is\_empty(q)) {

int i = q->front;

do {

i = (i + 1) % (MAX\_QUEUE\_SIZE);

printf("%d | ", q->data[i]);

if (i == q->rear)

break;

} while (i != q->front);

}

printf("\n");

}

// 삽입 함수

void enqueue(QueueType\* q, element item)

{

if (is\_full(q))

error("큐가 포화상태입니다");

q->rear = (q->rear + 1) % MAX\_QUEUE\_SIZE;

q->data[q->rear] = item;

}

// 삭제 함수

element dequeue(QueueType\* q)

{

if (is\_empty(q))

error("큐가 공백상태입니다");

q->front = (q->front + 1) % MAX\_QUEUE\_SIZE;

return q->data[q->front];

}

// 삭제 함수

element peek(QueueType\* q)

{

if (is\_empty(q))

error("큐가 공백상태입니다");

return q->data[(q->front + 1) % MAX\_QUEUE\_SIZE]; //

(이 코드는 큐에서 front 다음 위치에 있는 데이터를 반환하는 연산을 수행합니다.)

}

// ===== 원형큐 코드 끝 ======

int main(void)

{

QueueType queue;

init\_queue(&queue);

srand(time(NULL));

for (int i = 0; i < 100; i++) {

if (rand() % 5 == 0) { // 5로 나누어 떨어지면

enqueue(&queue, rand() % 100);

}

queue\_print(&queue);

if (rand() % 10 == 0) { // 10로 나누어 떨어지면

int data = dequeue(&queue);

}

queue\_print(&queue);

}

return 0;

}

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

# include <time.h>

// 프로그램 5.2에서 다음과 같은 부분을 복사한다.

// ================ 원형큐 코드 시작 =================

typedef struct { // 요소 타입

int id;

int arrival\_time;

int service\_time;

} element; // 교체!

// ================ 원형큐 코드 종료 =================

// ===== 원형큐 코드 시작 ======

#define MAX\_QUEUE\_SIZE 5

typedef struct { // 큐 타입

element data[MAX\_QUEUE\_SIZE];

int front, rear;

} QueueType;

// 오류 함수

void error(const char\* message)

{

fprintf(stderr, "%s\n", message);

exit(1);

}

// 공백 상태 검출 함수

void init\_queue(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);

}

// 원형큐 출력 함수

void queue\_print(QueueType\* q)

{

printf("QUEUE(front=%d rear=%d) = ", q->front, q->rear);

if (!is\_empty(q)) {

int i = q->front;

do {

i = (i + 1) % (MAX\_QUEUE\_SIZE);

printf("%d | ", q->data[i].arrival\_time);

if (i == q->rear)

break;

} while (i != q->front);

}

printf("\n");

}

// 삽입 함수

void enqueue(QueueType\* q, element item)

{

if (is\_full(q))

error("큐가 포화상태입니다");

q->rear = (q->rear + 1) % MAX\_QUEUE\_SIZE;

q->data[q->rear] = item;

}

// 삭제 함수

element dequeue(QueueType\* q)

{

if (is\_empty(q))

error("큐가 공백상태입니다");

q->front = (q->front + 1) % MAX\_QUEUE\_SIZE;

return q->data[q->front];

}

// 삭제 함수

element peek(QueueType\* q)

{

if (is\_empty(q))

error("큐가 공백상태입니다");

return q->data[(q->front + 1) % MAX\_QUEUE\_SIZE];

}

// ===== 원형큐 코드 끝 ======

int main(void)

{

int minutes = 60;

int total\_wait = 0;

int total\_customers = 0;

int service\_time = 0;

int service\_customer;

QueueType queue;

init\_queue(&queue);

srand(time(NULL));

for (int clock = 0; clock < minutes; clock++) {

printf("현재시각=%d\n", clock);

if ((rand() % 10) < 3) {

element customer;

customer.id = total\_customers++;

customer.arrival\_time = clock;

customer.service\_time = rand() % 3 + 1;

enqueue(&queue, customer);

printf("고객 %d이 %d분에 들어옵니다. 업무처리시간= %d분\n",

customer.id, customer.arrival\_time, customer.service\_time);

}

if (service\_time > 0) {

printf("고객 %d 업무처리중입니다. \n", service\_customer);

service\_time--;

}

else {

if (!is\_empty(&queue)) {

element customer = dequeue(&queue);

service\_customer = customer.id;

service\_time = customer.service\_time;

printf("고객 %d이 %d분에 업무를 시작합니다. 대기시간은 %d분이었습니다.\n",

customer.id, clock, clock - customer.arrival\_time);

total\_wait += clock - customer.arrival\_time;

}

}

}

printf("전체 대기 시간=%d분 \n", total\_wait);

return 0;

}