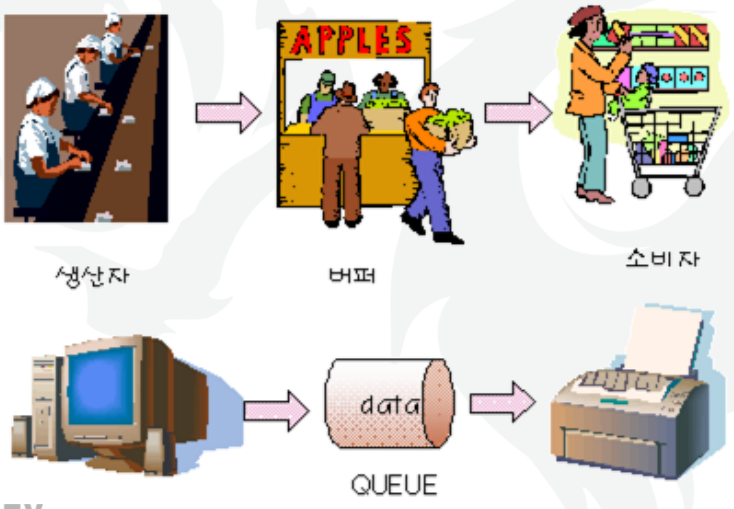

CSE2010 자료구조론

# Week 5: Queue 3

ICT융합학부 한진영

# 큐의 응용: 버퍼

- 서로 다른 속도로 실행되는 두 프로세스 간의 상호 작용을 조화시키는 버퍼 역할을 할 수 있음
  - CPU와 프린터 사이의 프린팅 버퍼, 또는 CPU와 키보드 사이의 키보드 버퍼 등
- 대개 데이터를 생산하는 생산자 프로세스가 있고 데이터를 소비하는 소비자 프로세스가 있으며 이 사이에 큐로 구성되는 버퍼가 존재



# 생산자-소비자 프로세스 알고리즘 예(1)

```
QueueType buffer;

/* 생산자 프로세스 */
producer()
{
    while(1){
        데이터 생산;
        while( lock(buffer) != SUCCESS ) ;
        if( !is_full(buffer) ){
            enqueue(buffer, 데이터);
        }
        unlock(buffer);
    }
}
```

# 생산자-소비자 프로세스 알고리즘 예(2)

```
/* 소비자 프로세스 */
consumer()
{
    while(1){
        while( lock(buffer) != SUCCESS )  ;
        if( !is_empty(buffer) ){
            데이터 = dequeue(buffer);
            데이터 소비;
        }
        unlock(buffer);
    }
}
```

# 큐의 응용: 시뮬레이션

- 큐는 큐잉이론에 따라 시스템의 특성을 시뮬레이션하여 분석하는 데 이용될 수 있음
- 큐잉모델은 고객에 대한 서비스를 수행하는 서버와 서비스를 받는 고객들로 이루어짐
  - 예: 은행에서 고객이 들어와서 서비스를 받고 나가는 과정을 시뮬레이션
    - 고객들이 기다리는 평균시간을 계산

# 은행 시뮬레이션 알고리즘

- 시뮬레이션은 하나의 반복 루프
- 현재 시각을 나타내는 clock이라는 변수를 하나 증가
- is_customer_arrived 함수가 호출되면, is_customer_arrived 함수는 랜덤 숫자를 생성하여 시뮬레이션 파라미터 변수인 arrival_prov와 비교하여 작으면 새로운 고객이 들어왔다고 판단
- 고객의 아이디, 도착시간, 서비스 시간 등의 정보를 만들어 구조체에 복사하고 이 구조체를 파라미터로 하여 큐의 삽입 함수 enqueue()를 호출
- 고객이 필요로 하는 서비스 시간은 역시 랜덤숫자를 이용하여 생성
- 지금 서비스하고 있는 고객이 끝났는지를 검사. 만약 service_time이 0이 아니면 어떤 고객이 지금 서비스를 받고 있는 중임을 의미
- clock이 하나 증가했으므로 service_time을 하나 감소
- 만약 service_time이 0이면 현재 서비스받는 고객이 없다는 것을 의미. 따라서 큐에서 고객 구조체를 하나 꺼내어 서비스를 시작

# 은행 시뮬레이션 프로그램(1)

```
typedef struct
  int id;
  int arrival_time;
  int service_time;
 element;

typedef struct
  element  queue[MAX_QUEUE_SIZE];
  int  front, rear;
 QueueType;
QueueType queue;
```

# 은행 시뮬레이션 프로그램(2)

```
// 0에서 1사이의 실수 난수 생성 함수
double random()
{
  return rand()/(double)RAND_MAX;
}

// 시뮬레이션에 필요한 여러가지 상태 변수
int duration=10; // 시뮬레이션 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;       // 전체고객수
int served_customers;  // 서비스받은 고객수
int waited_time;  // 고객들이 기다린 시간
```

# 은행 시뮬레이션 프로그램(3)

```
// 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.",
        customer.id, customer.arrival_time, customer.service_time);
}
```

# 은행 시뮬레이션 프로그램(4)

```
// 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." ,
          customer.id, clock, clock - customer.arrival_time);
  return service_time;
}
```

# 은행 시뮬레이션 프로그램(5)

```
// 통계치를 출력한다.
print_stat()
{
  printf("서비스받은 고객수 = %d",
                  served_customers);
  printf("전체 대기 시간 = %d분", waited_time);
  printf("1인당 평균 대기 시간 = %f분",
            (double)waited_time/served_customers);
  printf("아직 대기중인 고객수 = %d",
           customers-served_customers);
}
```

# 은행 시뮬레이션 프로그램(6)

```
// 시뮬레이션 프로그램
void 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();
    }
  }
  print_stat();
}
```

```
현재시각=1
고객 0이 1분에 들어옵니다. 서비스시간은 3분입니다.
고객 0이 1분에 서비스를 시작합니다. 대기시간은 0분이었습니다.
현재시각=2
고객 1이 2분에 들어옵니다. 서비스시간은 5분입니다.
현재시각=3
고객 2이 3분에 들어옵니다. 서비스시간은 3분입니다.
현재시각=4
고객 3이 4분에 들어옵니다. 서비스시간은 5분입니다.
고객 1이 4분에 서비스를 시작합니다. 대기시간은 2분이었습니다.
현재시각=5
현재시각=6
현재시각=7
고객 4이 7분에 들어옵니다. 서비스시간은 5분입니다.
현재시각=8
현재시각=9
고객 5이 9분에 들어옵니다. 서비스시간은 2분입니다.
고객 2이 9분에 서비스를 시작합니다. 대기시간은 6분이었습니다.
현재시각=10
고객 6이 10분에 들어옵니다. 서비스시간은 1분입니다.
서비스받은 고객수 = 3
전체 대기 시간 = 8분
1인당 평균 대기 시간 = 2.666667분
아직 대기중인 고객수 = 4
```

# Week 5: Queue 3