|  |  |  |  |
| --- | --- | --- | --- |
| 2022/2 『자료구조』실습 보고서 | | | |
| 제목 | 5장 실습( O ) 과제( ) | 제출일자 | 2022.  10 .    19 . |
| 학번 | 201911608 | 이름 | 김지환 |

|  |
| --- |
| 1. 프로그램 5.4  #include <stdio.h>  #include <stdlib.h>  #define MAX\_QUEUE\_SIZE 5  typedef int element;  typedef struct {  element data[MAX\_QUEUE\_SIZE];  int front, rear;  } DequeType;  void error(char\* message) {  fprintf(stderr, "%s\n", message);  exit(1);  }  void init\_deque(DequeType\* q) {  q->front = q->rear = 0;  }  int is\_empty(DequeType\* q) {  return (q->front == q->rear);  }  int is\_full(DequeType\* q) {  return ((q->rear + 1) % MAX\_QUEUE\_SIZE == q->front);  }  void deque\_print(DequeType\* q) {  printf("DEQUE(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");  }  void add\_rear(DequeType\* q, element item) {  if (is\_full(q))  error("큐가 포화상태입니다");  q->rear = (q->rear + 1) % MAX\_QUEUE\_SIZE;  q->data[q->rear] = item;  }  element delete\_front(DequeType\* q) {  if (is\_empty(q))  error("큐가 공백상태입니다");  q->front = (q->front + 1) % MAX\_QUEUE\_SIZE;  return q->data[q->front];  }  element get\_front(DequeType\* q) {  if (is\_empty(q)) error("큐가 공백상태입니다");  return q->data[(q->front + 1) % MAX\_QUEUE\_SIZE];  }  void add\_front(DequeType\* q, element val) {  if (is\_full(q))  error("큐가 포화상태입니다");  q->data[q->front] = val;  q->front = (q->front - 1 + MAX\_QUEUE\_SIZE) % MAX\_QUEUE\_SIZE;  }  element delete\_rear(DequeType\* q) {  int prev = q->rear;  if (is\_empty(q))  error("큐가 공백상태입니다");  q->rear = (q->rear - 1 + MAX\_QUEUE\_SIZE) % MAX\_QUEUE\_SIZE;  return q->data[prev];  }  element get\_rear(DequeType\* q) {  if (is\_empty(q))  error("큐가 공백상태입니다");  return q->data[q->rear];  }  int main(void) {  DequeType queue;  init\_deque(&queue);  for (int i = 0; i < 3; i++) {  add\_front(&queue, i);  deque\_print(&queue);  }  for (int i = 0; i < 3; i++) {  delete\_rear(&queue);  deque\_print(&queue);  }  return 0;  }  실행결과    -> add\_front() 와 delete\_rear()로 큐의 앞뒤가 추가, 삭제 됨을 알 수 있다. |
| 2. 프로그램 5.4 QUIZ  01 덱을 큐처럼 사용하려면 enqueue()와 dequeue() 대신에 어떤 연산들을 호출하여야 하는가?  ->  queue 자료구조는 LIFO이므로 enqueue() 연산은 queue의 맨 뒤에 값을 추가하는 연산이다.  dequeue() 연산은 맨 앞에 값을 제거하는 연산이다.  deque에서는 add\_rear() 연산이 맨 뒤에 값을 추가하고 delete\_front()연산이 맨 앞에 값을 제거한다.  enqueue() == add\_rea()  dequeue() == delete\_front()  02. 덱을 스택처럼 사용하려면 push()와 pop() 대신에 어떤 연산들을 호출하여야 하는가?  ->  STACK 자료구조는 FIFO이므로 push() 연산은 맨 뒤에 값을 추가하는 연산이고 pop() 연산은 맨 뒤에 값을 제거하는 연산이다.  deque에서는 add\_rear() 연산이 맨뒤에 값을 추가, delete\_rear() 연산이 맨뒤에 값을 제거한다.  push() == add\_rear()  pop() == delete\_rear() |
| 3. 프프로그램 5.5  # include <stdio.h>  # include <stdlib.h>  #define MAX\_QUEUE\_SIZE 5  typedef struct {  int id;  int arrival\_time;  int service\_time;  } element;  typedef struct {  element data[MAX\_QUEUE\_SIZE];  int front, rear;  } QueueType;  void error(char\* message) {  fprintf(stderr, "%s\n", message);  exit(1);  }  void init\_queue(QueueType\* q) {  q->front = q->rear = 0;  }  int is\_empty(QueueType\* q) {  return (q->front == q->rear);  }  int is\_full(QueueType\* q) {  return ((q->rear + 1) % MAX\_QUEUE\_SIZE == q->front);  }  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");  }  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;  }  element dequeue(QueueType\* q) {  if (is\_empty(q))  error("큐가 공백상태입니다");  q->front = (q->front + 1) % MAX\_QUEUE\_SIZE;  return q->data[q->front];  }  element peek(QueueType\* q) {  if (is\_empty(q))  error("큐가 공백상태입니다");  return q->data[(q->front + 1) % MAX\_QUEUE\_SIZE];  }  int main(void) {  int minutes = 10;  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;  }  실행결과 |
| 4. 프로그램 5.5 도전문제  위의 소스에서는 은행원이 한 명이라고 가정하고 있다. 만약 은행의 행원이 2명이라면 위의 소스를 어떻게 수정하여야 하는가?  # include <stdio.h>  # include <stdlib.h>  #define MAX\_QUEUE\_SIZE 5  typedef struct {  int id;  int arrival\_time;  int service\_time;  } element;  typedef struct {  element data[MAX\_QUEUE\_SIZE];  int front, rear;  } QueueType;  void error(char\* message) {  fprintf(stderr, "%s\n", message);  exit(1);  }  void init\_queue(QueueType\* q) {  q->front = q->rear = 0;  }  int is\_empty(QueueType\* q) {  return (q->front == q->rear);  }  int is\_full(QueueType\* q) {  return ((q->rear + 1) % MAX\_QUEUE\_SIZE == q->front);  }  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");  }  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;  }  element dequeue(QueueType\* q) {  if (is\_empty(q))  error("큐가 공백상태입니다");  q->front = (q->front + 1) % MAX\_QUEUE\_SIZE;  return q->data[q->front];  }  element peek(QueueType\* q) {  if (is\_empty(q))  error("큐가 공백상태입니다");  return q->data[(q->front + 1) % MAX\_QUEUE\_SIZE];  }  int main(void) {  int minutes = 10;  int total\_wait = 0;  int total\_customers = 0;  int service\_time1 = 0;  int service\_time2 = 0;  int service\_customer;  QueueType queue1;  init\_queue(&queue1);  QueueType queue2;  init\_queue(&queue2);  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(&queue1, customer);  printf("직원 1의 고객 %d이 %d분에 들어옵니다. 업무처리시간= %d분\n",  customer.id, customer.arrival\_time, customer.service\_time);  }  if (service\_time1 > 0) {  printf("직원 1의 고객 %d 업무처리중입니다. \n", service\_customer);  service\_time1--;  }  else  if (!is\_empty(&queue1)) {  element customer = dequeue(&queue1);  service\_customer = customer.id;  service\_time1 = customer.service\_time;  printf("직원 1의 고객 %d이 %d분에 업무를 시작합니다. 대기시간은 %d분이었습니다.\n",  customer.id, clock, clock - customer.arrival\_time);  total\_wait += clock - customer.arrival\_time;  }  if ((rand() % 10) < 3) {  element customer;  customer.id = total\_customers++;  customer.arrival\_time = clock;  customer.service\_time = rand() % 3 + 1;  enqueue(&queue2, customer);  printf("직원 2의 고객 %d이 %d분에 들어옵니다. 업무처리시간= %d분\n",  customer.id, customer.arrival\_time, customer.service\_time);  }  if (service\_time2 > 0) {  printf("직원 2의 고객 %d 업무처리중입니다. \n", service\_customer);  service\_time2--;  }  else  if (!is\_empty(&queue2)) {  element customer = dequeue(&queue2);  service\_customer = customer.id;  service\_time2 = customer.service\_time;  printf("직원 2의 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;  }  실행결과 |
| 5. 연습문제 11  회문(palindrome)이란 앞뒤 어느 쪽에서 읽어도 같은 말 , 구, 문 등을 의미한다. 예를 들면 “eye”, “madam”, “radar” 등이다. 여기서 물론 구두점이나 스페이스, 대소문자 등은 무시하여야한다. 덱을 이용하여 주어진 문자열이 회문인지 아닌지를 결정하는 프로그램을 작성하라. 다음 그림을 참조한다.  #include <stdio.h>  #include <stdlib.h>  #define MAX\_QUEUE\_SIZE 10  typedef char element;  typedef struct {  element data[MAX\_QUEUE\_SIZE];  int front, rear;  } DequeType;  void error(char\* message) {  fprintf(stderr, "%s\n", message);  exit(1);  }  void init\_deque(DequeType\* q) {  q->front = q->rear = 0;  }  int is\_empty(DequeType\* q) {  return (q->front == q->rear);  }  int is\_full(DequeType\* q) {  return ((q->rear + 1) % MAX\_QUEUE\_SIZE == q->front);  }  void deque\_print(DequeType\* q) {  printf("DEQUE(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");  }  void add\_rear(DequeType\* q, element item) {  if (is\_full(q))  error("큐가 포화상태입니다");  q->rear = (q->rear + 1) % MAX\_QUEUE\_SIZE;  q->data[q->rear] = item;  }  element delete\_front(DequeType\* q) {  q->front = (q->front + 1) % MAX\_QUEUE\_SIZE;  return q->data[q->front];  }  element get\_front(DequeType\* q) {  if (is\_empty(q)) error("큐가 공백상태입니다");  return q->data[(q->front + 1) % MAX\_QUEUE\_SIZE];  }  void add\_front(DequeType\* q, element val) {  if (is\_full(q))  error("큐가 포화상태입니다");  q->data[q->front] = val;  q->front = (q->front - 1 + MAX\_QUEUE\_SIZE) % MAX\_QUEUE\_SIZE;  }  element delete\_rear(DequeType\* q) {  int prev = q->rear;  if (is\_empty(q))  error("큐가 공백상태입니다");  q->rear = (q->rear - 1 + MAX\_QUEUE\_SIZE) % MAX\_QUEUE\_SIZE;  return q->data[prev];  }  element get\_rear(DequeType\* q) {  if (is\_empty(q))  error("큐가 공백상태입니다");  return q->data[q->rear];  }  int main(void) {  DequeType queue;  init\_deque(&queue);    char str[101];  printf("회문을 입력하시오: ");  gets(str);  for (int i = 0; i < strlen(str); i++) {  char c = str[i];  add\_rear(&queue, c);  }  while (queue.rear >= queue.front) {  char r = delete\_rear(&queue);  char f = delete\_front(&queue);  if (f != r) {  printf("회문이 아닙니다.\n");  exit(0);  }  }  printf("회문입니다.\n");  return 0;  }  실행결과 |