Data Structure Assignment

2016320205 이지혜
2015410072 김재현
2014250413 이상화

1. 이중 연결 리스트를 활용하여 학생 성적 리스트 만들기 / 이지혜

#include <stdio.h>　　표준 입출력을 위한 헤더
#include <stdlib.h>　　메모리를 할당하는 malloc과 배열 요소개수를 세는 _countof 함수를 사용하기 위한 헤더

typedef struct student {...} Student;
struct student 라는 자료형을 typedef를 사용하여 Student라는 자료형으로 선언하였다.
이름과 학번, 성적을 각각 char 배열과 int형으로 저장한다.

typedef struct studentNode {...} StudentNode;
struct studentNode 라는 자료형을 typedef를 사용하여 StudentNode라는 자료형으로 선언하였다.
Student 자료형과 현재 노드의 앞·뒤를 가리키는 노드의 포인터를 갖는다.

void init_student (Student s[]);
입력받은 Student 배열을 초기화한다. 학번과 이름은 Student_info.txt라는 텍스트 파일로부터 입력받으며, 파일을 읽어오지 못하면 망함을 출력하고 return한다. 이를 확인하기위해 입력받은 값을 출력한다.
또한 성적을 0~100 사이 중에 임의의 정수값으로 저장하고, 이를 출력한다.

void init_list (StudentNode *head_node);
StudentNode를 초기화시키는 함수로서, StudentNode의 포인터를 입력받아, 그 포인터의 노드가 가리키는 앞·뒤 노드가 자기 자신을 가리키게 한다.

void dinsert_node (StudentNode *head_node, StudentNode *new_node);
링크드 리스트에 성적에 따른 내림차순으로 정렬하며 연결하는 함수로, 첫 리스트를 가리키는 StudentNode형의 포인터와 새로 삽입할 노드의 포인터를 입력받는다. 노드가 있는 첫 리스트부터 끝 리스트까지 차례대로 훑으며, 입력받은 new_node의 점수가 알맞은 위치를 찾을 때까지 while문을 반복한 뒤, 해당 위치에 new_node를 삽입한다.

void print_list (StudentNode *head_node);
정렬된 링크드 리스트를 입력받아 상위 10명과 하위 10명을 출력하는 함수로, 첫 리스트를 가리키는 StudentNode형의 포인터를 입력받는다. 노드가 있는 첫 리스트부터 앞·뒤로 각각 학생의 정보다 담긴 10개의 노드를 출력한다.

int main (void) {...}
Student의 정보를 담을 배열을 선언하고, 그 Student를 가리킬 노드의 배열의 포인터와 첫 노드를 선언한다. Student 구조체와 링크드 리스트를 초기화한 뒤, StudentNode 포인터에 메모리를 할당하고 해당 포인터가 각각의 Student를 가리키게 한 뒤, 성적의 내림차순으로 링크드 리스트에 삽입한다. 삽입이 모두 끝난 뒤, 정렬된 리스트에서 성적 상·하위 학생 10명의 정보를 출력한다.

2. 미로찾기 / 이지혜

#include <stdio.h>　　표준 입출력을 위한 헤더
#include <stdlib.h>　　메모리를 할당하는 malloc 함수를 사용하기 위한 헤더

#define TRUE 1　　참이 되는 값
#define FALSE 0　　거짓이 되는 값
#define MAZE_SIZE 10　　미로의 크기

typedef struct point {...} Point;
struct point 라는 자료형을 typedef를 사용하여 Point라는 자료형으로 선언하였다.
좌표의 x, y값을 갖는다.

typedef struct StackNode {...} StackNode;
struct StackNode 라는 자료형을 typedef를 사용하여 StackNode라는 자료형으로 선언하였다.
좌표의 정보와 Stacknode 자료형의 포인터를 갖는다.

typedef struct Stacktype {...} StackType;
struct Stacktype 라는 자료형을 typedef를 사용하여 Stacktype라는 자료형으로 선언하였다.
StackNode의 맨 위를 가리키는 포인터를 갖는다.

void init_stack (StackType *s);
스택을 초기화시키는 함수로, StackType의 포인터를 입력받아 스택의 맨 위가 NULL 값을 가리키게 한다.

void init_path (Point *p, Point entry);
경로의 좌표를 초기화 시키는 함수로, 좌표를 가지는 p의 포인터의 첫 번째 원소가 entry(입구 좌표)를 가리키게 한다.

int is_empty (StackType *s);
스택이 비어있는지 검사하는 함수로, StackType의 포인터를 입력받아 스택의 맨 위가 NULL 값을 가리키면 참, 그렇지 않으면 거짓을 반환한다.

void push (StackType *s, Point *p);
스택에 입력받은 좌표를 넣는 함수로, StackType의 포인터와 입력하려는 좌표를 가리키는 포인터를 입력받는다. 해당 좌표를 스택에 넣기 위해 새 노드를 선언한 뒤, 메모리를 할당하고, 할당이 성공하면 입력받은 좌표의 정보를 갖는 새 노드를 스택에 넣고, 스택의 맨 위가 새 노드를 가리키게 한 후, 입력받은 좌표를 출력한다. 할당이 실패할 경우, 에러 메시지를 출력한다.

Point pop (StackType *s);
스택의 맨 위의 좌표를 반환하고 해당 스택에서 반환한 노드를 제거하는 함수로, StackType의 포인터를 입력받는다. 만약 해당 스택이 비어있으면 스택이 비어있다는 메시지를 출력한 뒤, 프로그램을 종료시킨다. 스택이 비어있지 않으면, 스택의 맨 위 노드의 좌표를 출력하고, 해당 노드를 제거하며, 스택의 맨 위가 제거된 노드 밑의 노드를 가리키게 한다. 그 후 해당 노드의 좌표를 반환한다.

void find_point (StackType *s, int r, int c);
현재 좌표에서 입력받은 좌표를 탐색할 수 있는지 여부를 찾는 함수로, StackType의 포인터와 탐색하려는 좌표를 입력받는다. 입력받은 좌표가 maze안의 좌표가 아니라면 에러 메시지를 출력한다. 입력받은 좌표가 벽이 아니고, 이미 탐색한 곳이 아니라면, 새 Point 구조체에 해당 좌표를 입력하고, 스택에 해당 좌표를 넣는다.

void print_maze ();
현재 미로의 탐색 상황을 명확하게 확인할 수 있도록 출력하는 함수이다. 벽은 1, 탐색하지 않은 곳은 0, 현재 탐색 중이거나 이미 탐색한 좌표는 8로 나타낸다.

void print_maze (Point *p);
입구부터 출구까지 지금까지 이동한 좌표를 순서대로 출력하는 함수로, 좌표를 저장한 Point 구조체의 포인터를 입력받는다. Point 구조체의 포인터가 가리키는 좌표가 입력하지 않은 좌표가 나올 때까지 루프문을 통해 이동한 좌표를 순차적으로 출력한다.

int maze[MAZE_SIZE][MAZE_SIZE] = {...};
미로의 정보가 담긴 배열이다. 벽은 1, 탐색할 수 있는 곳은 0으로 표현하였다.

int main(void) {...}
입구와 출구의 좌표를 입력하고, 경로를 저장할 Point 구조체 배열을 선언하고 구조체의 원소를 0으로 초기화하였다. 그리고 탐색할 좌표를 담을 StackType을 선언하고, 탐색할 좌표를 나타낼 x, y를 int형으로 선언하였다. 스택과 경로를 초기화 시킨 뒤, 현재 탐색중인 좌표가 출구가 아닐 때까지 루프문을 돌린다. 해당 루프문 안에서 현재 탐색중인 좌표와 탐색한 곳은 8로 표기한다. 탐색은 아래, 위, 왼쪽, 오른쪽 순서대로 스택에 push되며, 만약 스택이 비면 탐색을 실패했다는 메시지를 출력하고 종료한다. 스택이 비어있지 않으면 스택의 맨 위부터 탐색을 진행하고, 탐색한 좌표를 경로에 입력한다. 탐색하는 좌표가 출구를 만나면 루프를 빠져나가 탐색을 성공했다는 메시지를 출력하고 입구부터 출구까지 탐색한 좌표를 순차적으로 출력한다.

3. ZNN.com / 김재현

#include <stdio.h> 표준 입출력을 위한 헤더

#define SIZE_QUEUE 100 news_queue의 최대 사이즈
#define CAPACITY 6 ZNN.com이 초당 처리할 수 있는 용량
#define NUM_USER 20 유저의 수
#define NUM_LOOP 100 루프 횟수
#define CONTENT_VARIETY 3 Znn.com에서 제공하는 콘텐츠의 종류

typedef enum { Video, Image, Text } ContentFidelity;
콘텐츠의 종류의 열거형을 typedef를 이용해 ContentFidelity라는 자료형으로 선언하였다.

char *enumString[] = { "Video", "Image", "Text" };
ContentFidelity에 담긴 열거형(정수)을 문자열로 직접 나타내기 위하여 선언하였다.

typedef struct {...} news;
새로운 구조체를 typedef를 이용해 news라는 자료형으로 선언하였다.
콘텐츠를 요청한 유저와 요청한 콘텐츠의 종류를 갖는다.

typedef struct {...} Queue;
새로운 구조체를 typedef를 이용해 Queue라는 자료형으로 선언하였다.
news의 배열과 배열의 시작과 끝을 가리키는 변수, 그리고 요청된 콘텐츠의 수를 갖는다.

typedef struct {...} User;
새로운 구조체를 typedef를 이용해 User라는 자료형으로 선언하였다.
유저의 정보를 담고 있다. 유저가 요청해서 받은 콘텐츠의 수, 받은 콘텐츠의 품질에 따른 점수, 요청한 콘텐츠를 받을 때까지 기다린 시간, 기다린 시간의 합, 그리고 현재 콘텐츠를 요청하였는지 여부에 대한 정보를 갖는다.

Queue new_queue;
Queue 구조체를 선언한다.

ContentFidelity currentFidelity = Video;
현재 콘텐츠의 품질을 나타내는 ContentFidelity 자료형의 변수를 선언한다. 초기 콘텐츠의 품질을 Video이다.

User userInfo[NUM_USER];
유저들의 정보를 담기위한 User 자료형의 배열을 선언한다.

void request () {...}
유저들이 콘텐츠를 요청하는 함수이다. 만약 유저가 이미 요청 중이라면 콘텐츠를 다시 요청하지 않으며, 요청 중이지 않으면 50% 확률로 콘텐츠를 요청한다. 만약 유저가 콘텐츠를 요청하면, Queue 구조체의 queue 배열 제일 뒤에 요청한 유저의 번호와 요청한 콘텐츠의 품질을 저장하고, 해당 User의 배열에 현재 요청 중임을 표시한다. Queue 구조체의 queue 배열이 끝에 도달하면 다시 첫 번째 배열을 가리키는 순환 queue로 구현하였다. 그리고 현재 요청 중인 유저의 수를 가리키는 변수를 증가시킨다.

void setFidelity () {...}
현재 유저들의 콘텐츠 요청상황에 맞춰 콘텐츠의 품질을 조정하는 함수이다. 현재 대기 중인 유저들의 대기시간을 다 더한 뒤, 대기 중인 유저 수로 나눈 평균 대기시간이 3초 이상이면 콘텐츠의 품질을 한 단계 낮추고, 2초 이하이면 콘텐츠의 품질을 한 단계 높인다. 그 후 현재 평균 대기시간과 현재 콘텐츠의 품질을 출력한다.

void provide () {...}
콘텐츠를 요청한 유저들에게 콘텐츠를 제공하는 함수이다. ZNN.com이 초당 처리할 수 있는 용량만큼 유저들이 요청한 콘텐츠를 제공한다. 콘텐츠를 제공하면, 해당 User의 배열에 제공받은 콘텐츠의 품질에 따른 점수, 제공받은 콘텐츠의 수, 대기시간을 입력하고, 현재 요청을 하지 않은 상태로 바꾼다. 그리고 현재 요청 중인 유저의 수를 가리키는 변수를 감소시킨 뒤, 아직 콘텐츠를 제공받지 못한 유저들의 대기시간을 증가시킨다.

void printQueue () {...}
현재 콘텐츠를 요청한 유저들의 대기 줄을 출력하는 함수이다. Queue 구조체의 queue배열에 담긴(현재 콘텐츠를 제공받기를 기다리는) 유저와 요청한 콘텐츠의 종류를 각각 출력한다.

void printResult () {...}
100번의 루프가 끝난 뒤 각 유저들과 해당 유저가 기다린 시간, 제공받은 콘텐츠 점수들의 평균을 출력하는 함수이다. 만약 유저가 어떠한 콘텐츠도 제공받지 못 했으면 대기시간과 콘텐츠 점수는 0이 되며, 적어도 하나의 콘텐츠를 제공받았을 경우 평균 대기시간은 (총 대기시간 / 제공받은 콘텐츠의 수)이고, 평균 콘텐츠 점수는 (총 콘텐츠의 점수 / 제공받는 콘텐츠의 수)가 된다.

int main() {...}
100번 동안(설정상 100초 동안) 루프를 반복시키며 2초에 한 번씩 콘텐츠의 품질을 설정하고 유저들이 콘텐츠를 요청하고, 1초에 한 번씩 대기 중인 유저들의 정보가 담긴 queue의 내용을 출력하고 유저에게 콘텐츠를 제공한다. 루프가 모두 끝난 뒤 모든 유저들과 각 유저의 평균 대기시간과 평균 콘텐츠 점수를 출력한다.