자료구조 과제#1

다항식 구현

충북대학교

CHUNGBUK NATIONAL UNIVERSITY

과목: 자료구조
교수: 이의종
학번: 2022042011
이름: 문서영

1. 코드 설명 //주석으로 설명

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include<stdlib.h>
#include <time.h>

// malloc 매크로 정의//동적할당
#define MALLOC(p,s) \
   if(!((p)=malloc(s))){\
      fprintf(stderr,"Improper value of n \n");\
      exit(EXIT_FAILURE);\
   }
//MAX 매크로 정의
#define MAX(a,b) ((a) > (b) ? (a) : (b))
typedef struct {//지수, 계수 구분하여 저장하기 위한 구조체 지정
    int coef;
    int exp;
}Term;
typedef struct polyNode {
    int coef;
    int exp;
    struct polyNode*link;
}polyNode;

//배열_초기 방법으로 배열을 계산하는 함수
int*make_array1(FILE*fp, int num_terms) {
    // 파일에서 다항식 항들을 읽고,
    //계수를 지수 내림차순으로 정렬한 배열을 생성하여 반환하는 함수

    Term*terms;//계수와 지수를 함께 저장할 구조체

    //배열에 동적할당
    MALLOC(terms, num_terms *sizeof(Term));

    int max_exp =0;//최대 지수값 저장할 변수

    for (int i =0; i <num_terms; i++) {//구조체 형식으로 배열에 지수, 계수 저
장
        fscanf(fp, "%d\t%d", &terms[i].coef, &terms[i].exp);
        if (terms[i].exp >max_exp)//최대 지수 갱신
            max_exp =terms[i].exp;
    }
    //배열의 크기: 최대 지수+2
    int*arr;//배열 동적할당
    MALLOC(arr, (max_exp +2) *sizeof(int));

    arr[0] =max_exp;//배열 첫부분에 최대 지수 저장
```

```
    //계수 부분은 우선 초기화
    for (int i =1; i <=max_exp +1; i++) {
        arr[i] =0;
    }
    //구조체 배열을 순회하면서, 해당 지수 위치에 계수 저장
    //배열의 인덱스를 기준으로 지수가 저장되므로 자동 내림차순 정렬
    for (int i =0; i <num_terms; i++) {
        int pos =max_exp -terms[i].exp +1;
        arr[pos] =terms[i].coef;
    }
    //임시로 쓴 구조체 배열 메모리 해제
    free(terms);
    //생성한 배열 반환
    return arr;
}
//배열_초기 텍스트 파일로 출력하는 함수
void write_poly(FILE*fp, int*poly) {
    int max_deg =poly[0];
    int firstTerm =1;
    // 최대 지수부터 0까지 반복 (배열 인덱스는 1부터 max_deg+1)
    for (int i =1; i <=max_deg +1; i++) {
        int exp =max_deg - (i -1);
        int coef =poly[i];
        if (coef !=0) {
            if (!firstTerm) { // 첫 항이 아니라면 앞에 " + " 추가
                fprintf(fp, " + ");
            }
            // 항을 "계수 x^지수" 형식으로 출력
            // 예를 들어 coef=2, exp=4인 경우 "2x^4" 출력
            fprintf(fp, "%dx^%d", coef, exp);
            firstTerm =0;
        }
    }
    fprintf(fp, "\n");
}


//배열_개선된 방법으로 배열을 할당
int**make_array2(FILE*fp, int n1, int n2, int**out_data) {
    Term*termsA, *termsB;//계수와 지수를 함께 저장할 구조체

    //배열에 동적할당
    MALLOC(termsA, n1 *sizeof(Term));
    for (int i =0; i <n1; i++) {//첫 다항식,구조체 형식으로 배열에 지수, 계수
저장
        fscanf(fp, "%d\t%d", &termsA[i].coef, &termsA[i].exp);
    }
    MALLOC(termsB, n2 *sizeof(Term));
    for (int i =0; i <n2; i++) {//두번째 다항식,구조체 형식으로 배열에 지수,
계수 저장
        fscanf(fp, "%d\t%d", &termsB[i].coef, &termsB[i].exp);
    }
```

```c
    //개선된 배열 만들기
    // 1: 전체 데이터 메모리 한 덩어리 할당
    int rows =n1 +n2 +1;
    int*data = (int*)malloc(rows *2 *sizeof(int));
    // 2: 행 포인터 배열 할당
    int**arr = (int**)malloc(rows *sizeof(int*));
    // 3: 각 행 포인터에 적절한 위치 연결
    for (int i =0; i <rows; i++) {
        arr[i] =data + (i *2);
        //arr[i]는 이제 data에서 행을 기준으로 시작위치 저장
        //data+0->row 0시작, data+2->row1시작(cols=2)
    }
    // 4: 값 할당
    for (int i =0; i <n1; i++) {
        arr[i][0] =termsA[i].coef;
        arr[i][1] =termsA[i].exp;
    }
    for (int i =0; i <n2; i++) {
        arr[n1 +i][0] =termsB[i].coef;
        arr[n1 +i][1] =termsB[i].exp;
    }

    // 5: 메모리 해제
    free(termsA);
    free(termsB);

    //arr을 밖에서도 생존시키고,
    // free(data)하기 위해 data도 밖으로 보낸다.
    *out_data =data;
    return arr;
}

// 두 다항식의 덧셈 결과를 2차원 배열로 반환하는 함수
// out_rows는 결과 다항식의 항 개수를 저장하는 변수의 주소
int**add_array2(int**arr, int n1, int n2, int*out_rows) {
    int total_terms =n1 +n2;
    int max_exp =0;

    // 1. 두 다항식의 최대 지수 찾기
    for (int i =0; i <total_terms; i++) {
        if (arr[i][1] >max_exp)
            max_exp =arr[i][1];
    }

    // 2. 임시 합산 배열 할당 (인덱스: 지수 0 ~ max_exp)
    int*sum_temp = (int*)calloc(max_exp +1, sizeof(int));
    if (sum_temp ==NULL) {
        fprintf(stderr, "메모리 할당 실패\n");
        exit(EXIT_FAILURE);
    }

    // 3. 두 다항식의 모든 항에 대해 계수를 누적
```

```c
    // 각 항: arr[i][0] = 계수, arr[i][1] = 지수
    for (int i =0; i <total_terms; i++) {
        int coef =arr[i][0];
        int exp =arr[i][1];
        sum_temp[exp] +=coef;
    }

    // 4. 누적 결과 중 계수가 0이 아닌 항의 개수 계산
    int count =0;
    for (int exp =0; exp <=max_exp; exp++) {
        if (sum_temp[exp] !=0)
            count++;
    }

    // 5. 결과 2차원 배열 할당
    // 결과 배열은 각 행이 [계수, 지수] 형태이며, 총 count 행
    int**result = (int**)malloc(count *sizeof(int*));
    int*result_data = (int*)malloc(count *2 *sizeof(int));
    if (result ==NULL ||result_data ==NULL) {
        fprintf(stderr, "메모리 할당 실패\n");
        exit(EXIT_FAILURE);
    }
    for (int i =0; i <count; i++) {
        result[i] =result_data +i *2;
    }

    // 6. 결과 배열에 내림차순(높은 지수부터)으로 값 저장
    // exp: max_exp → 0
    int row =0;
    for (int exp =max_exp; exp >=0; exp--) {
        if (sum_temp[exp] !=0) {
            result[row][0] =sum_temp[exp]; // 계수
            result[row][1] =exp;           // 지수
            row++;
        }
    }
    free(sum_temp);
    *out_rows =count;  // 결과 다항식의 항 개수를 반환
    return result;
}

//다항식 출력함수
// poly: 2차원 배열, 각 행은 [계수, 지수] 형태로 저장됨
// row_count: poly에 저장된 항의 개수
// 출력 형식: 각 항을 "계수x^지수"로 출력하며, 항들 사이에는 " + " 또는 " - "를
공백 포함하여 삽입
void write_poly_file(FILE*fp, int**poly, int row_count) {
    int firstTerm =1;  // 첫 항이면 " + "를 출력하지 않기 위해 사용
    for (int i =0; i <row_count; i++) {
        int coef =poly[i][0];
        int exp =poly[i][1];
        if (coef ==0) continue;  // 계수가 0이면 출력하지 않음
```

```
        if (firstTerm) {
            // 첫 항일 경우, 음수이면 "-" 표시, 양수이면 그냥 출력
            if (coef <0)
                fprintf(fp, "-%dx^%d", -coef, exp);
            else
                fprintf(fp, "%dx^%d", coef, exp);
            firstTerm =0;
        }
        else {
            // 두 번째 항부터는 " + " 혹은 " - " 추가
            if (coef <0)
                fprintf(fp, " - %dx^%d", -coef, exp);
            else
                fprintf(fp, " + %dx^%d", coef, exp);
        }
    }
    fprintf(fp, "\n");  // 한 다항식 출력 후 줄바꿈
}

//노드 생성 함수(지수, 계수)
polyNode*create_node(int coef, int exp) {//노드 생성 함수(계수,지수 입력받
음)
    polyNode*node = (polyNode*)malloc(sizeof(polyNode));//구조체 생성
    //변환된 주소를 polyNode*형으로 캐스팅하여 node에 저장.
    if (node ==NULL) {
        printf("메모리 할당 실패\n");
        exit(1);
    }
    node->coef =coef;
    node->exp =exp;
    node->link =NULL;
    return node;
}
//노드 삽입 함수
// (지수 내림차순, 동일한 지수의 항이 이미 존재하면 계수끼리 더해준다.)
void insert_term(polyNode**head, int coef, int exp) {
    if (coef ==0) return;  // 계수 0은 무시

    polyNode*new_node =create_node(coef, exp);//새 노드 동적 생성

    // 1. head가 비었거나, 새 항이 head보다 큰 지수면 맨 앞 삽입(내림차순 정렬)
    if (*head ==NULL ||exp > (*head)->exp) {
        new_node->link =*head;
        *head =new_node;
        return;
    }

    //순회 준비
    polyNode*cur =*head;//현재 노드 포인터 지시
    polyNode*prev =NULL;//이전 노드 포인터 지시
    //현재 항보다 지수가 작거나 같은 곳을 찾을 때까지 이동(내림차순 유지)
    while (cur !=NULL &&cur->exp >exp) {
```

```
        prev =cur;
        cur =cur->link;
    }
    if (cur !=NULL &&cur->exp ==exp) {
        // 지수가 같으면 계수만 더하기
        cur->coef +=coef;
        free(new_node);  // 새 노드는 필요 없음
        if (cur->coef ==0) {
            // 계수가 0이 되면 노드 제거
            if (prev) prev->link =cur->link;
            else *head =cur->link;
            free(cur);
        }
    }
    else {//지수가 더 작거나, 끝까지 갔을 경우-> 중간 혹은 끝에 삽입
        new_node->link =cur;
        if (prev) prev->link =new_node;
        else *head =new_node;
    }
}

//폴리 다항식 모양으로 프린트 로직 참고
// polyNode 링크드 리스트에 저장된 다항식을 파일에 출력하는 함수
void write_poly_linked(FILE*fp, polyNode*head) {
    int first =1;  // 첫 항이면 구분자 없이 출력하도록 함

    while (head !=NULL) {
        // 계수가 0이면 건너뛰기 (필요하면 0항도 출력할 수 있음)
        if (head->coef !=0) {
            if (first) {
                // 첫 항: 양수이면 그냥 출력, 음수이면 '-' 처리
                if (head->coef <0)
                    fprintf(fp, "-%dx^%d", -head->coef, head->exp);
                else
                    fprintf(fp, "%dx^%d", head->coef, head->exp);
                first =0;
            }
            else {
                // 이후 항들: 부호에 따라 " + " 또는 " - "로 출력
                if (head->coef <0)
                    fprintf(fp, " - %dx^%d", -head->coef, head->exp);
                else
                    fprintf(fp, " + %dx^%d", head->coef, head->exp);
            }
        }
        head =head->link;
    }
    fprintf(fp, "\n");
}

//메모리 반환
void free_poly(polyNode*head) {
```

```
    polyNode*cur;
    while (head !=NULL) {
        cur =head;
        head =head->link;
        free(cur);
    }
}
//배열_초기 방식으로 덧셈계산된 배열 반환
//시간 계산 구현 필요
int*calculate_Array1() {
    FILE*input =fopen("input.txt", "r");//파일 읽기 모드로 열기
    if (input ==NULL) {//파일 열기 오류처리
        printf("파일열기 실패\n");
        return NULL;
    }
    int n1, n2;//각 다항식의 지수의 수를의미
    //([number_of_exponents]\t[number_of_exponents]\n)
    fscanf(input, "%d\t%d", &n1, &n2);

    int*A =make_array1(input, n1);
    int*B =make_array1(input, n2);
    fclose(input);//파일 닫기

    int degA =A[0];  // A의 최고 차항
    int degB =B[0];  // B의 최고 차항
    int max_deg =MAX(degA, degB);

    int size =max_deg +2; // [최고차수, 계수 ...] 이므로 +2

    int*answer;//덧셈 완료된 배열 동적할당
    MALLOC(answer, size *sizeof(int));
    answer[0] =max_deg;

    // 계수 0으로 초기화
    for (int i =1; i <=max_deg +1; i++) {
        answer[i] =0;
    }
    // A의 항들 복사
    for (int i =1; i <=degA +1; i++) {
        int exp =degA - (i -1);  // 지수 계산
        int pos =max_deg -exp +1;
        answer[pos] +=A[i];
    }

    // B의 항들 더하기
    for (int i =1; i <=degB +1; i++) {
        int exp =degB - (i -1);  // 지수 계산
        int pos =max_deg -exp +1;
        answer[pos] +=B[i];
    }
    //텍스트 파일로 출력
    FILE*out_fp =fopen("output.txt", "w");//텍스트 파일 열기
```

```
    if (out_fp ==NULL) {
        perror("output.txt 파일 열기 실패");
        exit(EXIT_FAILURE);
    }
    write_poly(out_fp, A);
    write_poly(out_fp, B);
    write_poly(out_fp, answer);

    fclose(out_fp);
    //메모리 누수 방지
    free(A);
    free(B);

    return answer;
}

//배열_개선된 방법으로 배열을 계산하는 함수
int calculate_Array2() {
    FILE*input =fopen("input.txt", "r");//파일 읽기 모드로 열기
    if (input ==NULL) {//파일 열기 오류처리
        printf("파일열기 실패\n");
        return 1;
    }
    int n1, n2;//각 다항식의 지수의 수를의미
    //([number_of_exponents]\t[number_of_exponents]\n)
    fscanf(input, "%d\t%d", &n1, &n2);

    //data 포인터 받기위한 인자 선언
    int*data =NULL;
    int**arr =make_array2(input, n1, n2, &data);
    fclose(input);//파일 닫기



    //다항식 덧셈 구현
    int out_rows;
    int**sum =add_array2(arr, n1, n2, &out_rows);

    //덧셈 결과 파일 출력
    FILE*fp =fopen("output.txt", "a");//내용 추가하는 것이기 때문에 append로
열기
    if (fp ==NULL) {
        perror("파일 열기 실패");
        exit(EXIT_FAILURE);
    }
    write_poly_file(fp, arr, n1);  // arr의 첫 n1개의 행이 f1의 항들

    write_poly_file(fp, arr +n1, n2);  // arr+n1은 두번째 다항식 시작, n2개
의 행 사용

    write_poly_file(fp, sum, out_rows); // add_array2()를 통해 계산된 덧셈
결과
```

```
    fclose(fp);

    //메모리 해제
    free(sum);
    free(data);
    free(arr);

    return 1;
}
// 두 정렬된 링크드 리스트(내림차순 상태)를 병합하여 덧셈 결과 링크드 리스트를
생성하는 함수
polyNode*merge_poly(polyNode*A, polyNode*B) {
    // 임시 더미 노드를 생성하여 결과 리스트의 시작을 쉽게 처리
    polyNode dummy;
    dummy.link =NULL;
    polyNode*tail =&dummy;

    while (A !=NULL &&B !=NULL) {
        if (A->exp >B->exp) {
            // A의 항이 더 큰 경우: 결과에 A의 노드를 연결하고 A를 한 칸 이동
            tail->link =A;
            A =A->link;
            tail =tail->link;
        }
        else if (A->exp <B->exp) {
            // B의 항이 더 큰 경우: 결과에 B의 노드를 연결하고 B를 한 칸 이동
            tail->link =B;
            B =B->link;
            tail =tail->link;
        }
        else {
            // 두 항의 지수가 같으면 계수를 합산
            int newCoef =A->coef +B->coef;
            if (newCoef !=0) {
                // 합산한 계수가 0이 아니면, 새 노드를 생성하여 결과에 연결
                tail->link =create_node(newCoef, A->exp);
                tail =tail->link;
            }
            // 두 노드를 모두 소비
            A =A->link;
            B =B->link;
        }
    }
    // 남은 리스트 중 남은 항을 결과에 연결
    if (A !=NULL) {
        tail->link =A;
    }
    else if (B !=NULL) {
        tail->link =B;
    }
```

```
    return dummy.link;
}

//Linkedlist_방법으로 배열을 계산하는 함수
int calculate_Linkedlist() {
    FILE*input =fopen("input.txt", "r"); // 파일 읽기 모드
    if (input ==NULL) {
        printf("파일열기 실패\n");
        return 1;
    }

    int n1, n2; // f₁(x)와 f₂(x)의 항 개수
    fscanf(input, "%d\t%d", &n1, &n2);

    int co, ex;
    polyNode*poly1 =NULL;
    polyNode*poly2 =NULL;

    // 첫 번째 다항식의 n1개의 항 읽기
    for (int i =0; i <n1; i++) {
        fscanf(input, "%d\t%d", &co, &ex);
        insert_term(&poly1, co, ex);
    }
    // 두 번째 다항식의 n2개의 항 읽기 (루프 조건 수정)
    for (int i =0; i <n2; i++) {
        fscanf(input, "%d\t%d", &co, &ex);
        insert_term(&poly2, co, ex);
    }
    fclose(input);


    polyNode*sum =merge_poly(poly1, poly2);

    // output.txt에 결과를 append 모드로 기록
    FILE*out_fp =fopen("output.txt", "a");
    if (out_fp ==NULL) {
        perror("output.txt 파일 열기 실패");
        exit(EXIT_FAILURE);
    }

    // 각각의 다항식을 출력 (라벨 없이 다항식만 출력)
    write_poly_linked(out_fp, poly1);
    write_poly_linked(out_fp, poly2);
    write_poly_linked(out_fp, sum);

    fclose(out_fp);

    // merge_poly()는 새 노드를 생성할 때만 new 메모리를 할당
    // 남은 노드들은 poly1이나 poly2의 노드를 재사용
    // 따라서 두 리스트가 병합된 후 poly1과 poly2는 더 이상 유효하지 않으므로,
    // merge_poly() 결과 sum 리스트만 free_poly()를 호출해서 해제
    free_poly(sum);
```

```
    return 0;
}


int main() {
    clock_t start, end;
    double time_array1, time_array2, time_linkedlist;// 시간 체크를 위한 타
임 변수 설정

    //solution#1
    start =clock();
    int*result1 =calculate_Array1();
    end =clock();
    time_array1 = ((double)(end -start)) /CLOCKS_PER_SEC;
    printf("calculate_Array1 실행 시간: %f 초\n", time_array1);
    free(result1);

    //solution#2
    start =clock();
    calculate_Array2(); // 반환형이 int 형(예: 성공 여부)이라 가정
    end =clock();
    time_array2 = ((double)(end -start)) /CLOCKS_PER_SEC;
    printf("calculate_Array2 실행 시간: %f 초\n", time_array2);

    //solution#3
    start =clock();
    calculate_Linkedlist(); // 반환형이 int 형(예: 성공 여부)이라 가정
    end =clock();
    time_linkedlist = ((double)(end -start)) /CLOCKS_PER_SEC;
    printf("calculate_Linkedlist 실행 시간: %f 초\n", time_linkedlist);


    FILE*fp =fopen("output.txt", "a");
    fprintf(fp, "%f\t%f\t%f", time_array1, time_array2, time_linkedlist);
    fclose(fp);

    return 0;
}
```

2. TA의 컴퓨터에서 실행하기 위한 방법

비주얼스튜디오_
Microsoft Visual Studio Community 2022 (64-bit) - Current
버전 17.13.5