プログラミング演習２

演習１

```
/*
 ２分探索木に対するMEMBER, INSERT, MIN, DELETEのプログラム例
   テキスト プログラム例をswk用に修正
   コンパイル方法： cl bst2.c
   実行方法(例)：   bst2 Samples¥n0-6.txt

   (注)実行時の入力データは、Samplesの中にいくつか用意あります。
*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <ctype.h>
#define FALSE 0
#define TRUE 1
#ifdef min
#undef min
#endif
#ifdef max
#undef max
#endif

enum yn {yes, no};          /* 列挙型データynの定義 */
struct node                 /* 構造体nodeの宣言 */
{
        int element;
        struct node *left;
        struct node *right;
};

/* 関数の宣言 */
enum yn member(int x, struct node *init);
void insert(int x, struct node **p);
int min(struct node *p);
```

```
void delete(int x, struct node **p);
struct node *off(struct node *p);
void printpre(struct node *p);
void inorder(struct node *p);
//void print_tree(struct node *init);
int height_of_tree(int reset, struct node *init);
void array_tree(struct node *init, int *array);
void show_tree(struct node *init);

int main(int argc, char *argv[])
/* ２分探索木の処理のテストプログラム */
{
        struct node *init;
        int i,j,k,x;
        enum yn a;
        FILE *file; /* 2分木入力データファイル */
        int tmp0,
            counter=0;
        char sel,     /* 処理の選択を指定する文字変数 */
             tmp[20];/* 一時的な変数(scanf用) */

        /* 2分木入力データファイルを読み込む準備 */
        if(argc !=2){
                printf("error! --> 使用法：%s 入力データファイル名¥n",argv[0]);
                exit(1);
        }
        if((file=fopen(argv[1],"rt"))==NULL){
                printf("error! --> %s が見つかりません¥n",argv[1]);
                exit(1);
        }

        init=NULL;            /* データの生成 */
        printf("入力データ： ");
        while(1){
                if(fscanf(file," %d",&tmp0)==1){
                        counter++;
```

```
                printf(" %d",tmp0);
                insert(tmp0,&init);
            }
            else{
                printf(" (%d個)¥n",counter);
                break;
            }
        }
        fclose(file);

        /* 入力データ読み込み後 */
        printf("入力データ読込完了 : init = %p¥n", init);

        while(1){

            /*========== 処理方法の選択 ==========*/
            if(init==NULL){
                printf("挿入(i),終了(q) ?  : ");
                scanf("%1s%*c" ,&sel);
                if(sel=='d'||sel=='D') continue;
                if(sel=='p'||sel=='P') continue;
                if(sel=='s'||sel=='S') continue;
                if(sel=='m'||sel=='M') continue;
            }
            else{
                printf("挿入(i),削除(d),表示(p),検索(s),最小値(m),終了(q) ?  :
");
                scanf("%1s%*c" ,&sel);
            }

            /* selの値に応じて処理を選択 */
            switch(sel){

                /*++++++++++ 挿入処理 ++++++++++*/
                case 'i':
                case 'I':
```

```
                printf("=== 挿入 ===¥n");
                /* 2分木へ挿入するデータの入力 */
                while(1){
                        printf(" 挿入データ ? : ");
                        scanf("%s",tmp);
                        if(isdigit(tmp[0])==0 && tmp[0]!='-')
                                printf("error¥n");
                        else{
                                x=atoi(tmp);
                                break;
                        }
                }
                /* insert関数の呼び出し */
                insert(x,&init);
                printf("init = %p¥n", init);

                break;

        /*++++++++++ 削除処理 ++++++++++*/
        case 'd':
        case 'D':
                printf("=== 削除 ===¥n");
                /* 2分木から削除するデータの入力 */
                while(1){
                        printf(" 削除するデータ ? : ");
                        scanf("%s",tmp);
                        if(isdigit(tmp[0])==0 && tmp[0]!='-')
                                printf("error¥n");
                        else{
                                x=atoi(tmp);
                                break;
                        }
                }
                /* delete関数の呼び出し */
                delete(x, &init);
                printf("init = %p¥n", init);
```

```
            break;

        /*++++++++++ 2分木を図示 ++++++++++*/
        case 'p':
        case 'P':
            printf("=== 表示 ===¥n");
            /* print_tree関数の呼び出し */
            //print_tree(init);
            printf("init = %p¥n", init);
            printf("  ---tree---¥n");
            show_tree(init);
            printf("  ---inorder---¥n");
            inorder(init);
            printf("  ---preorder---¥n");
            printpre(init);

            break;

        /*++++++++++ 検索 ++++++++++*/
        case 's':
        case 'S':
            printf("=== 検索 ===¥n");
            /* 2分木で検索するデータの入力 */
            while(1){
                printf(" 検索するデータ ? : ");
                scanf("%s",tmp);
                if(isdigit(tmp[0])==0 && tmp[0]!='-')
                    printf("error¥n");
                else{
                    x=atoi(tmp);
                    break;
                }
            }
            /* member関数の呼び出し */
            a=member(x, init);
```

```
                        if(a==yes) printf("Yes: x = %d¥n", x);
                        else printf("No: x = %d¥n", x);

                        break;

                /*++++++++++ 最小値 ++++++++++*/
                case 'm':
                case 'M':
                        printf("=== 最小値 ===¥n");
                        /* min関数の呼び出し */
                        x=min(init);
                        printf("min = %d¥n", x);

                        break;

                /*++++++++++ プログラム終了 ++++++++++*/
                case 'q':
                case 'Q':
                        exit(0);

                /* 指定の処理以外はエラーとして扱う */
                default:
                        printf("入力エラー: %c¥n",sel);
                        printf("¥n¥n");
                        break;

                }
        }

        return(0);
}

enum yn member(int x, struct node *p)
/* pが指す２分探索木にxの存在を判定 */
{
        struct node *q;
```

```
        q=p;                                    /* 根initから探索開始 */
        while(q!=NULL)
        {
                if(q->element == x) return(yes);  /* xを発見 */
                if(q->element < x)  q = q->right;
                else q = q->left;
        }
        return(no);                             /* x存在せず */
}

void insert(int x, struct node **p)
/* ポインタ*pが指す２分探索木にxを挿入し、*pを更新 */
{
        struct node *q, **r;            /* qは探索中のポインタ、rはその親 */

        r=p; q=*r;                              /* rとqの初期化 */
        while(q != NULL)
        {
                if(q->element == x) return;       /* xはすでに存在 */
                if(q->element < x)                /* 右の子へ */
                        {r=&(q->right); q=q->right;}
                else                              /* 左の子へ */
                        {r=&(q->left); q=q->left;}
        }
    /* *pの指す木が空の場合、*rと*pは同じところを指している */
        *r=(struct node *)malloc(sizeof(struct node));  /* xを入れる場所 */
        (*r)->element=x; (*r)->left=NULL; (*r)->right=NULL;   /* xの挿入 */
        return;
}

int min(struct node *p)
/* pが指す２分探索木の最小要素を出力 */
{
        struct node *q, *r;             /* qは探索中のポインタ、rはその親 */
```

```
        q=p;                            /* 根から左端の路を下がる */
        if(q==NULL) {printf("Error: Tree is empty.¥n"); exit(1);} /* 木は空 */
        while(q!=NULL) {
                r = q; q = q->left;
        }
        return(r->element);             /* 結果を返す */
}

void delete(int x, struct node **p)
/* ポインタ*pが指す２分探索木から要素xを除去し、*pの更新 */
{
        struct node *q, **r;            /* qは探索中のポインタ、rはその親 */

        r=p; q=*r;                      /* 根から探索の開始 */
        while(q!=NULL){
                if(q->element == x){        /* xを発見 */
                        *r=off(q);              /* xのセルqを除去し、qの部分木を更新 */
                        return;                 /* 終了 */
                }
                if(q->element < x) {r = &(q->right);}    /* 探索を続ける */
                else {r = &(q->left);}
                q=*r;
   }
        return;                         /* xは存在せず */
}

struct node *off(struct node *p)
/* ポインタpが指す節点を除去、pの部分木を更新、根へのポインタを返す */
{
        struct node *q, **r;                /* qは探索中のポインタ、rはその親 */

        if(p->left==NULL && p->right==NULL) /* pの子は共に空 */
                {free(p); return(NULL);}
        if(p->left==NULL)                   /* pの左の子が空 */
                {q=p->right; free(p); return(q);}
        if(p->right==NULL)                  /* pの右の子が空 */
```

```
                {q=p->left; free(p); return(q);}
        r=&(p->right);      /* pの子は共に存在; 右の部分木の最小要素を探索 */
        q=*r;
        while(q->left != NULL)
                {r=&(q->left); q=q->left;}
        p->element = q->element;
        *r=off(q);                          /* 最小要素qの除去 ; 再帰的実行 */
        return(p);
}

void printpre(struct node *p)
/* 節点 *p の全ての子節点を前順に出力する */
{
        printf("p = %p, element = %d, left = %p, right = %p¥n",
                p, p->element, p->left, p->right);
        if(p->left != NULL) printpre(p->left);
        if(p->right != NULL) printpre(p->right);
        return;
}

void print_tree(struct node *p)
/* pが指す２分探索木を表示する */
{
        static int depth = 0;

        if (p->left != NULL) {
                depth++;  print_tree(p->left);  depth--;
        }
        printf("%*c%d¥n", 5 * depth, ' ', p->element);
        if (p->right != NULL) {
                depth++;  print_tree(p->right);  depth--;
        }
}

int height_of_tree(int reset,struct node *p)
{
```

```
        static int height, depth=0;
        if(reset==TRUE) height=0;
        if (p->left != NULL) {
                depth++;
                if(height<depth) height=depth;
                height_of_tree(FALSE,p->left);
                depth--;
        }
        if (p->right != NULL) {
                depth++;
                if(height<depth) height=depth;
                height_of_tree(FALSE,p->right);
                depth--;
        }
        return(height);
}

void array_tree(struct node *p, int *array)
/* pが指す２分探索木を行列表示する */
{
        static int i=1;
        array[i]=p->element;
        if(p->left != NULL){
                i*=2;
                array_tree(p->left,array);
                i/=2;
        }
        if(p->right !=NULL){
                i=i*2+1;
                array_tree(p->right,array);
                i=(i-1)/2;
        }
        return;
}

void show_tree(struct node *p)
```

```
/* pが指す２分探索木を図示する */
{
        int i,j,k,tmp1,tmp2,h,t;
        int *array; /* 木の行列表現(木の表示用) */
        const int upper=4;

        h=height_of_tree(TRUE,p);
        t=(h>upper)?upper:h;

        printf("(height=%d)¥n",h);

        tmp1=(int)pow(2.,(double)(h+1))-1;
        array=(int *)malloc(sizeof(int)*tmp1);  /* 行列割当て */
        for(i=1;i<=tmp1;i++)
                array[i]=-100;
        array_tree(p,array);
        for(i=0;i<=h;i++){
                tmp2=(int)pow(2.,(double)i);
                if(i>upper){
                        printf(" ********* tree表示オーバのため以下略 *********¥n¥n");
                        break;
                }
                for(j=0;j<tmp2;j++){
                        if(array[tmp2+j]!=-100){
                                printf("%3d",array[tmp2+j]);
                                for(k=1;k<(int)pow(2.,(double)(t-i));k++)
printf("%s","    ");
                        }
                        else{
                                printf("  *");
                                for(k=1;k<(int)pow(2.,(double)(t-i));k++)
printf("%s","    ");
                        }
                }
                printf("¥n");
                if(i<t) printf("%s","  |");
```

```
                    for(j=0;j<tmp2;j++){
                            for(k=0;k<(int)pow(2.,(double)(t-i-1));k++)
printf("%s","___");
                            for(k=0;k<(int)pow(2.,(double)(t-i-1))-1;k++) printf("%s","
");
                            printf("%s","   ");
                            if(j<tmp2-1 && i<t ) printf("%s","|");
                    }
                    printf("¥n");
            }
            free(array);
}

void inorder(struct node *p)
/* 節点 *p の全ての子節点を中順に出力する */
{
            if(p->left != NULL) inorder(p->left);
            printf("p = %p, element = %d, left = %p, right = %p¥n",
                    p, p->element, p->left, p->right);
            if(p->right != NULL) inorder(p->right);
            return;
}
```

何回か直してみたりしたのですがうまくできなかったので元のコードをそのまま乗せました。

結果

C:¥Users¥yutow¥OneDrive¥デスクトップ¥演習問題 1、16>mondai#1
error! --> 使用法：mondai#1 入力データファイル名

C:¥Users¥yutow¥OneDrive¥デスクトップ¥演習問題 1、16>mondai#1 input1
.txt
入力データ： 7 3 11 1 5 9 13 0 2 4 6 8 10 12 14 (15 個)
入力データ読込完了：init = 0000015892CE0700
挿入(i),削除(d),表示(p),検索(s),最小値(m),終了(q)？ ：

演習問題 1 の後半以降は手をつけることができなかったです。