Junco de los Heros 2-3-2019

TAD Two Stack

```
#define DIM 256
struct _TwoStack{
    EleStack *datos[DIM];
    int top1; // Podía haber usado un array
    int top2;
};
/* Este TAD utiliza un array donde una pila usará los primeros posiciones y
   la segunda pila los últimos */

TwoStack *twostack_ini(){
    TwoStack *t=NULL;
    int i;
    t=(TwoStack *) malloc(sizeof(TwoStack));
    if(!t){
      fprintf(stderr,"%s",strerror(errno));
      return NULL;
    }
    for(i=0;i<DIM;i++){
       t->datos[i]=NULL;
    }
    t->top1=-1;
    t->top2=DIM;
    return t;
}

Bool twostack_isFull(const TwoStack* t){
    if(!t) return TRUE;
    if(t->top1+1==t->top2) return TRUE;
    return FALSE;
}

Bool twostack_isEmpty(const TwoStack *t, TypeStack stipo){
    if(!t) return TRUE;
    if(stipo==STACK1) return (t->top1==-1);
    else return (t->top2==DIM);
}
```

5.)
```
STATUS twostack_push (const EleStack *e, TwoStack *t, TypeStack stipo){
    if(!e || !t) return ERROR;
    if(twostack_isFull(t)) return ERROR;
    EleStack *aux = NULL;
    aux = elestack_copy(e);
    if(!aux) return ERROR;
    if(stipo == STACK1){
        t->top1++;
        t->data[t->top1] = aux;
    }
    else{
        t->top2--;
        t->data[t->top2] = aux;
    }
    return OK;
}
```

6.)
```
EleStack *twostack_pop(TwoStack *t, TypeStack stipo){
    EleStack *aux = NULL;
    if(!t || twostack_isEmpty(t)) return NULL;
    if(stipo == STACK1){
        aux = t->data[t->top1];
        t->data[t->top1] = NULL;
        t->top1--;
    }else{
        aux = t->data[t->top2];
        t->data[t->top2] = NULL;
        t->top2++;
    }
    return aux;
}
```

7.)
```
int twostack_print(FILE *f, const TwoStack *t, TypeStack stipo){
    int cont = 0;
    if(!f || !t) return 0;
    if(stipo == STACK1){
        for(i = t->top1; i>=0; i--){
            cont += EleStack_print(f, t->data[i]);
            cont += fprintf(f, "\n");
        }
    }else{
        for(i = t->top2; i<DIM; i++){
            cont += EleStack_print(f, t->data[i]);
            cont += fprintf(f, "\n");
        }
    }
    return cont;
}
```

8.)
```
void twostack_destroy(TwoStack *t){
    int i;
    if(!t) return;
    for(i=t->top1; i>=0; i--){
        elestack_destroy(t->datos[i]);
    }
    for(i=t->top2; i<DIM; i++){
        elestack_destroy(t->datos[i]);
    }
    free(t);
}
```

9.)
```
int main(void){
    int i;
    TwoStack *t = NULL;
    EleStack *ele = NULL;
    t = twostack_ini();
    if(!t) return EXIT_FAILURE;
    ele = elestack_ini();
    if(!ele){
        twostack_destroy(t);
        return EXIT_FAILURE;
    }
    i=4;
    elestack_setInfo(ele, &i);
    twostack_push(ele, t, STACK1);
    i=2;
    elestack_setInfo(ele, &i);
    twostack_push(ele, t, STACK2);
    twostack_print(stdout, t, STACK1);
    twostack_print(stdout, t, STACK2);
    twostack_destroy(t);
    elestack_destroy(ele);
    return EXIT_SUCCESS;
}
```