05/08/20 11:23:28 /Users/mmar/Documents/docencia/PSC/19-20/PrimerControlC/Tercera Parte/Lopez PerezMarta_3325004_assignsubmission_file_/Polinomio.c

ejecuta bien, salvo que no imprime los ceros

```c
/*
 * Polinomio.c
 *
 *  Created on: 29 abr. 2020
 *      Author: marta
 */

#include "Polinomio.h"
#include <stdlib.h>
#include <stdio.h>


//Parte 1. PARA APROBAR

/*Crea el polinomio 0 (es decir, un polinomio vacío).*/
void polinomioCero(TPolinomio *p)
{
    *p = NULL;
}


/*Devuelve el grado del polinomio, es decir, el mayor exponente de los
monomios que no son nulos. En el ejemplo, el grado es 7.
El grado del polinomio cero es 0.*/
unsigned int grado(TPolinomio p)
{
    int maximo_grado = 0;

    while (p!=NULL)
    {
        if (p->exp > maximo_grado)
        {
            maximo_grado = p->exp;
        }
        p = p->sig;
    }
    return maximo_grado;
}

```

```
40
41  /* Devuelve el coeficiente del exponente exp del polinomio p.*/
42  unsigned int coeficiente(TPolinomio p, unsigned int exp)
43  {
44      int coef = 0;
45
46      while (p!=NULL)
47      {
48          if (p->exp == exp)
49          {
50              coef = p->coef;
51          }
52          p = p->sig;
53      }
54      return coef;
55  }
56
57
58  /* Insertar el monomio con coeficiente coef, y exponente exp en el
    polinomio,
59   * de manera que el polinomio quede ordenado. Asegurarse que no se
    insertan
60   * monomios cuyo coeficiente sea 0 y tampoco dos monomios con el
    mismo exponente.
61   * Si al insertar un monomio ya hay otro con el mismo exponente los
    coeficientes
62   * se sumarán. Se puede asumir que el valor del coeficiente siempre
    será un numero
63   * natural (entero no negativo).*/
64  TPolinomio crearNodoPolinomio(unsigned int coef, unsigned int exp)
65  {
66      TPolinomio aux = malloc(sizeof(struct TMonomio));
67      aux->coef = coef;
68      aux->exp = exp;
69      aux->sig = NULL;
70      return aux;
71  }
72
73  void insertar(TPolinomio *p, unsigned int coef, unsigned int exp)
74  {
75      TPolinomio ant = NULL;
76      TPolinomio act = *p;
77      TPolinomio aux = NULL;
78      while (act!=NULL && act->exp > exp)
```

```
79      {
80          ant = act;
81          act = act->sig;
82      }
83
84      if(coef != 0)
85      {
86          if(ant==NULL)
87          {
88              if(act==NULL)   //Polinomio vacio
89              {
90                  *p = crearNodoPolinomio(coef, exp);
91              }
92              else
93              {
94                  if(act->exp==exp)
95                  {
96                      act->coef = act->coef + coef;
97                  }
98                  else    //lo inserto como primer nodo
99                  {
100                     aux = crearNodoPolinomio(coef,exp);
101                     aux->sig = *p;
102                     *p = aux;
103                 }
104             }
105         }
106         else if (act==NULL)     //el nuevo nodo va al final
107         {
108             ant->sig =crearNodoPolinomio(coef,exp);
109
110         }
111         else
112         {
113             if(act->exp == exp)
114             {
115                 act->coef = act->coef + coef;
116             }
117             else
118             {
119                 aux = crearNodoPolinomio(coef,exp);
120                 aux->sig = act;
121                 ant->sig = aux;
122             }
```

```
123         }
124     }
125 }
126 
127 
128 /*Escribe por la pantalla el polinomio con un formato similar al
    siguiente:
129  * [(3,7)(0,6)(2,5)(0,4)(3,3)(0,2)(5,1)(9,0)] para el polinomio
    ejemplo.
130  * Ten en cuenta que los monomios de exponente menor al grado del
    polinomio
131  * con coeficiente 0 también aparecen en la salida, aunque no estén
    almacenados
132  * en el polinomio. */
133 void imprimir(TPolinomio p)
134 {
135     printf("[");
136     while(p!=NULL)
137     {
138         printf("(%d,%d)", p->coef,p->exp);
139         p=p->sig;
140 
141         /*
142          * Aquí habría que incluir un if que ponga como restriccion
143          * que se incluyan los monomios con coeficiente cero aunq no
144          * hayan sido insertados. Un if que reste los exponentes y
    en caso de que se salte alguno
145          * incluya por defecto el del coeficiente 0,
146          * acaba de resolverme la duda y hay que entregar, no me da
    tiempo a implementarlo
147          * pero se haría asi, un saludo.
148          */
149     }
150 
151     printf("] \n");
152 }
153 
154 
155 
156 /* Elimina todos los monomios del polinomio haciendo que el
    polinomio resultante
157  * sea el polinomio 0.*/
158 void destruir(TPolinomio *p)
159 {
```

```
160     TPolinomio aux ;
161
162     while(*p!=NULL)
163     {
164         aux = *p;
165         *p = aux->sig;
166         free(aux);
167     }
168
169     *p=0;
170 }
171
172 //Parte 2. Notable
173 /* Lee los datos de un polinomio de un fichero de texto, y
174  * crea la lista de monomios p. El formato del polinomio en el
    fichero contiene
175  * una secuencia de pares de dígitos correspondientes al coeficiente
    y exponente
176  * de cada monomio del polinomio, incluyendo los que tienen
    coeficiente nulo.
177  * En ambos casos, suponemos que los coeficientes y exponentes son
    dígitos del 0 al 9
178  * (no hay números superiores).
179  * Por ejemplo, para el polinomio de ejemplo el fichero de texto
    estaría compuesto
180  * por la secuencia de caracteres “0690332551370402”.
181  * Observa que los monomios pueden venir desordenados en el fichero
    de entrada.
182  *
183  * La conversión de un valor de tipo ‘char’ que contenga
184  * un valor númerico (ej. char c = ‘2’)
185  * a su correspondiente valor entero (int valor),
186  * se puede hacer de la siguiente forma: valor = c – ‘0’
187 */
188 //void crearDeFichero(TPolinomio *p, char *nombre);
189
190 //Parte 3. Sobresaliente
191 /* Evalúa el polinomio para el valor x y devuelve el resultado.
192  * Para la evaluación del polinomio debes utilizar el método de
    Horner,
193  * de manera que ax^4 + bx^3+ cx^2+dx+e puede evaluarse
194  * en un valor cualquiera x teniendo en cuenta que es equivalente
195  * a: (((ax+b)x+c)x+d)x+e.
196 */
```

```
197  //int evaluar(TPolinomio p,int x);
198
```