

Evaluar un polinomio

Un polinomio $P(x) = a_{n-1}x^{n-1} + \dots + a_1x + a_0$ con coeficientes enteros puede ser representado de distintas maneras. Supongamos que tenemos una clase `Polinomio` con la siguiente interfaz:

```
class Polinomio {
public:
    Polinomio();
    void anyadir_monomio(int coef, int exp);
    long evaluar(int valor) const;
private:
    // ...
};
```

Una instancia recién construida de esta clase denota el polinomio $P(x) = 0$. El método `anyadir_monomio` añade al polinomio `this` un monomio con el coeficiente y exponente pasados como parámetro. El método `evaluar` devuelve el resultado de sustituir la variable x del polinomio `this` por el valor indicado como parámetro. La especificación del constructor y de ambos métodos es la siguiente:

```
{ true }
Polinomio() // constructor
{ this = 0 }

{ this = P(x) }
void anyadir_monomio(int coef, int exp)
{ this = P(x) + coef · xexp }

{ this = P(x) }
long evaluar(int valor)
{ result = P(valor) }
```

En este ejercicio trataremos dos posibles representaciones de los polinomios, que se describen en las siguientes secciones. En ambos casos, el programa a escribir debe leer un polinomio P por la entrada y un valor v , y a partir de ambos imprimir el resultado de evaluar el polinomio para el valor dado, esto es, el resultado de sustituir la variable x de P por el valor v .

Primera representación

En esta primera sección representaremos el polinomio $P(x) = a_{n-1}x^{n-1} + \dots + a_1x + a_0$ mediante un array de números enteros, en el que la posición i -ésima del array contiene el valor del coeficiente a_i .

```
const int GRADO_MAX = 10000;

class Polinomio {
public:
    // ...
private:
    int coeficientes[GRADO_MAX + 1];
};
```

Suponiendo esta representación, se pide:

1. Implementar el constructor de la clase `Polinomio`, y los métodos `anyadir_monomio` y `evaluar`. Sobre este último método, pon atención al coste. Si n es el grado del polinomio, no se van a admitir soluciones $O(n^2)$ ni $O(n \log n)$. Tampoco está permitido el uso de la función `std::pow()`.
2. Indicar el coste en tiempo, en el caso peor, del constructor de la clase y de los métodos mencionados en el apartado anterior.
3. Escribe un programa que lea de la entrada estándar varios polinomios, construya los objetos `Polinomio` correspondientes y que los evalúe para un valor determinado, también leído de la entrada. El formato de la entrada se describe al final de este enunciado.

Segunda representación

Otra forma de representar un polinomio consiste en tener un array de registros de tipo `Monomio`, donde cada uno de ellos guarda un coeficiente y un exponente:

```
const int MAX_MONOMIOS = 10000;

class Polinomio {
public:
    Polinomio();
    // ...
private:
    struct Monomio {
        int coeficiente;
        int exponente;
    };
    Monomio monomios[MAX_MONOMIOS];
    int num_monomios;
};
```

Suponemos que se cumple el siguiente invariante de representación para toda instancia p de `Polinomio`:

$$I(p) = \begin{aligned} &0 \leq p.\text{num_monomios} \leq \text{MAX_MONOMIOS} \\ &\wedge \forall i: 0 \leq i < p.\text{num_monomios} \Rightarrow p.\text{monomios}[i].\text{coeficiente} \neq 0 \\ &\wedge \forall i, j: 0 \leq i < j < p.\text{num_monomios} \Rightarrow p.\text{monomios}[i].\text{exponente} < p.\text{monomios}[j].\text{exponente} \end{aligned}$$

Es decir, el array de monomios debe estar ordenado de manera que la secuencia de exponentes sea estrictamente ascendente desde la posición 0 hasta `num_monomios - 1`. El array no puede contener exponentes duplicados, y no puede contener coeficientes nulos.

Utilizando esta representación, se pide:

4. Implementar el constructor de la clase `Polinomio`, y los métodos `anyadir_monomio` y `evaluar`. Sobre este último método, pon atención al coste. Si n es el grado del polinomio, no se van a admitir soluciones $O(n^2)$ ni $O(n \log n)$. Tampoco está permitido el uso de la función `std::pow()`.
5. Indicar el coste en tiempo, en el caso peor, del constructor de la clase y de los métodos mencionados en el apartado anterior.
6. Reutiliza el programa del apartado 3, pero utilizando la clase `Polinomio` del apartado 4.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso ocupa dos líneas. La primera línea contiene dos números N y V ($0 \leq N \leq 2000$, $-10 \leq V \leq 10$), donde N indica el número de pares (*coeficiente, exponente*) que vendrán en la línea siguiente, y V es el valor para el cual quiere evaluarse el polinomio. La segunda línea contiene una secuencia de $2 * N$ números que indican los monomios del polinomio que quiere evaluarse. Cada pareja de números indica el coeficiente y el exponente del monomio correspondiente. Ten en cuenta que en la entrada puede haber varios monomios con el mismo exponente, y estos no están necesariamente ordenados.

La entrada finaliza con una línea que contiene dos ceros, caso que no deberá procesarse.

Salida

Para cada caso se escribirá una línea con el número resultante de evaluar el polinomio para el valor dado. Se garantiza que este número siempre estará contenido en el rango permitido por una variable de tipo long de 64 bits.

Entrada de ejemplo

```
2 4
3 2 5 0
4 2
3 2 1 0 7 1 -2 2
0 0
```

Salida de ejemplo

```
53
19
```

Créditos

Manuel Montenegro. Adaptación del problema *Evaluar un polinomio* de Alberto Verdejo e Isabel Pita.