Hoja de ejercicios 6

Tipos Definidos

4 de mayo de 2015

Informática
Año 2014/2015
Facultad de CC.
Matemáticas

▷ 1. Fracciones

Diseña un tipo de datos que permitan representar los números racionales. Implementa las operaciones básicas de suma, resta, multiplicación y división, considera también los operadores de comparación.

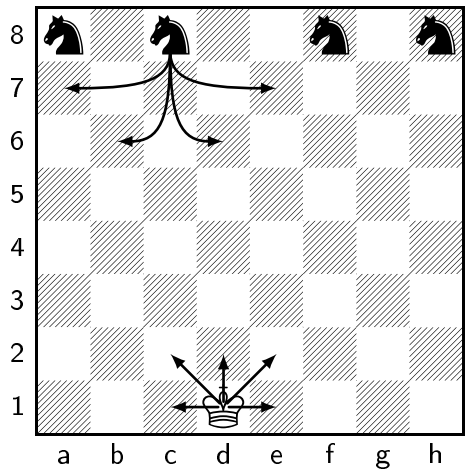
▷ 2. Calendario perpetuo: fechas absolutas

Las implementaciones de las fecha basadas en ternas *dia-mes-anyo* presentan bastantes problemas. En este ejercicio se dará una implementación alternativa. Para ello elegirá un día que sirva como referencia, el día que vamos a fijar es el 1 de enero de 1601. Cualquier otro día lo representaremos como la diferencia de días con respecto a ese día origen (nos vamos a limitar a representar fechas posteriores). Recordemos que un año es bisiesto si es múltiplo de 4, salvo si es múltiplo de 100 en cuyo caso debe ser también múltiplo de 400.

- Haz un programa en Python que dados tres enteros correspondientes a una terna día-mes-año, compruebe si son correctos, y en caso afirmativo calcule el entero que representa la fecha.
- Implementa una clase fecha que internamente guarde el número de días transcurrido desde la fecha origen. La clase debe tener las siguientes características:
  - El constructor debe admitir como parámetros una terna día, mes, año.
  - Debe tener una función que devuelva una cadena de caracteres con el formato `<dia>/<mes>/<año>`.
  - Debe tener un método que devuelva el día de la semana del día en cuestión. Para ello es suficiente saber que el 7 de octubre de 1968 es un lunes.
  - Debe tener un método que calcule el número de días hasta otra fecha que se indique como parámetro.
  - Devuelva otra fecha incrementándola un cierto número de días.
  - Devuelva otra fecha que se corresponda con del día siguiente.
  - Devuelva otra fecha que se corresponda con del día anterior.
- Realiza un programa que muestre todas las fechas entre dos fechas dadas.

▷ 3. Captura al rey!

Te proponemos un juego basado en el ajedrez, en el que el usuario mueve una pieza del ajedrez (el rey) y el ordenador mueve a sus adversarios (cuatro caballos). El rey, al comienzo de la partida, está en un lado del tablero y los caballos en el lado opuesto.



Las reglas del juego son sencillas: para ganar la partida, el usuario debe dirigir al rey hasta alcanzar el lado opuesto del tablero. Los caballos, manejados por el ordenador, deben intentar comerse al rey.

En cada turno, el rey se puede mover una casilla en cualquiera de las direcciones (adelante, atrás, a los lados, y hacia las diagonales).

- El ordenador mueve un sólo caballo por turno. Cada caballo se mueve en L, para cualquier dirección.
- Siempre empieza moviendo el rey.
- Si un caballo se come al rey, la partida termina.
- Si el rey se come a un caballo, la partida continua y, por tanto, el ordenador juega con una pieza menos.
- El rey gana cuando llega al otro lado del tablero (la fila 8).

Movimiento aleatorio Realizar el simulador del juego, de tal manera que el ordenador mueva las piezas aleatoriamente en cualquiera de las ocho direcciones posibles; y el usuario mueva la suya utilizando las teclas `q`, `w`, `e`, `a` , `d`, `z`, `x` y `c` del teclado.

En este apartado el ordenador no es inteligente: sencillamente elegirá al azar tanto el caballo a mover como el movimiento a realizar. La única situación en la que el ordenador debe actuar inteligentemente es cuando el rey está amenazado, entonces el caballo que lo amenaza debe comerlo en su turno.

Por supuesto, el programa debe controlar el final de la partida y mostrar un mensaje diciendo quien ha ganado. También debe dar al usuario la posibilidad de volver a jugar.

Debido a que no contamos en este momento con librerías gráficas para mostrar el tablero como en la figura, deberemos utilizar la salida estándar. Así pues, podemos representar los caballos negros con el carácter `n` y el rey blanco con el carácter `K`, las casillas blancas con el carácter espacio y las negras con el carácter `#`. Así pues la configuración inicial que se muestra es:

```
n # n #   n   n
#   #   #   #
  #   #   #   #
#   #   #   #
  #   #   #   #
#   #   #   #
  #   #   #   #
#   # K #   #
```

Un poco de inteligencia Podemos conseguir que el ordenador sea un poco más listo. Para ello deberá elegir uno de los siguientes movimientos, por orden de prioridad:

1. Comer al rey.
2. Amenazar una de las casillas en las que rey puede continuar.
3. Situarse lo más cerca posible del rey, sin estar bajo su amenaza.

Mejoras Pensar posibles mejoras en la estrategia de los caballos.

▷ 4. Implementación de polinomios

Un polinomio se puede ver como una lista de monomios de la siguiente forma:

$$x^{100} - 3x^2 + 1 \quad \rightsquigarrow \quad \boxed{(1,0)} \boxed{(-3,2)} \boxed{(1,100)}$$

Tipos de datos Define las clases necesarias para para representar monomios y polinomios.

Suma y resta de polinomios Diseña métodos para sumar y restar polinomios.

Multiplicación de polinomios Diseña métodos para multiplicar dos polinomios.

Polinomio de interpolación de Lagrange Haz un programa que calcule el polinomio de interpolación de Lagrange. Este polinomio se calcula como se indica a continuación. Tenemos los puntos $(x_0, y_0), \ldots, (x_k, y_k)$, queremos calcular un polinomio cuyo valor en x_i sea exactamente y_i $(0 \leq i \leq k)$. Dicho polinomio $L(X)$ se calcula de la siguiente manera:

$$L(x) = \sum_{j=0}^{k} y_j \ell_j(x)$$

donde cada $\ell_j(x)$ es un polinomio que verifica lo siguiente

$$\ell_j(x_j) = 1 \ , \quad \ell_j(x_i) = 0 \ \forall \ i \neq j$$

Los polinomios $\ell_j(x)$ se pueden calcular de la siguiente forma:

$$\ell_j(x) = \prod_{i=0, i\neq j}^{k} \frac{x - x_i}{x_j - x_i} = \frac{x - x_0}{x_j - x_0} \cdots \frac{x - x_{j-1}}{x_j - x_{j-1}} \frac{x - x_{j+1}}{x_j - x_{j+1}} \cdots \frac{x - x_k}{x_j - x_k}$$

Nótese que los denominadores son escalares, no polinomios.

El programa debe leer los puntos de interpolación de un fichero. Por ejemplo, si queremos calcular el polinomio de interpolación para los puntos (0.5, 2.3), (3.1, 6.8) y (5.6, 5.5) el fichero contendrá las siguientes líneas.

```
0.5 2.3
3.1 6.8
5.6 5.5
```

Un poco de historia En análisis numérico, el polinomio de Lagrange, llamado así en honor a Joseph-Louis de Lagrange, es el polinomio que interpola un conjunto de puntos dado en la forma de Lagrange. Fue descubierto por Edward Waring en 1779 y redescubierto más tarde por Leonhard Euler en 1783. Dado que existe un único polinomio interpolador para un determinado conjunto de puntos, resulta algo confuso llamar a este polinomio el polinomio interpolador de Lagrange. Un nombre más conciso es interpolación polinómica en la forma de Lagrange. (fuente Wikipedia, `http://es.wikipedia.org/wiki/Interpolaci%C3%B3n_polin%C3%B3mica_de_Lagrange`)

▷ 5. Monstruos S.A.

Monstruos S.A. es la mayor compañía eléctrica de Monstruópolis. Su método de producción consiste en emparejar cada niño humano con el monstruo asustador más adecuado. De este modo, se obtienen gritos de la mejor calidad, que una vez refinados se convierten en energía limpia y fiable. En la base de datos de la empresa se almacena la información de todos sus monstruos asustadores, que consiste en los datos siguientes:

- Código de identificación, compuesto por 8 dígitos y un carácter alfabético.
- Nombre y primer apellido.
- Antigüedad en la empresa especificada en meses.
- Lista de los niños humanos a los que es capaz de asustar.

De cada niño humano se guarda el nombre y primer apellido, la edad, y el código que identifica la puerta del armario de su habitación dentro del almacén de puertas de la empresa.

- Define los tipos de datos necesarios para representar la información descrita.
- Diseña y escribe un subprograma que encuentre los monstruos que asusten a los niños humanos de mayor edad.
- Cada susto genera una energía que consiste en en el producto de la edad del niño asustado por la antigüedad del monstruo asustante. Cada monstruo solo puede asustar a un niño en una noche y cada niño solo puede ser asustado por un monstruo. Diseña una estrategia para emparejar, al principio de una noche, a cada monstruo con un niño y escribe el programa que calcule la energía que la empresa obtendrá en esa noche siguiendo tu estrategia.

▷ 6. Coleccionando Vectores

En este ejercicio trabajaremos con colecciones de vectores de dimensión arbitraria. Cada colección tiene un número variable de vectores pero todos los vectores de una colección tienen la misma dimensión.

Tipos de datos Define los tipos de datos estructurados adecuados para representar los vectores y las colecciones de ellos.

Cubrevector Diremos que un vector v cubre a otro w si ambos tienen la misma dimensión y para todo índice i se tiene $v(i) >= w(i)$. Escribe un subprograma que implemente la función cubrevector.

Matriz de cubrición Dada una colección podemos construir la matriz de cubrición que nos indica si el elemento i-ésimo de la colección cubre al j-ésimo. Define el tipo adecuado para la matriz de cubrición e implementa un subprograma que dada una colección genere la matriz de cubrición correspondiente.

Cubrecolección Una colección C_1 cubre a otra C_2 si para cada vector de C_2 existe alguno de C_1 que lo cubre. Implementa una función que decida si dadas dos colecciones de vectores C_1 y C_2, la primera cubre a la segunda.

Reducir colecciones Podemos reducir el tamaño de una colección eliminando los elementos que son cubiertos por otros elementos de la propia colección. Implementa, utilizando recursión, un subprograma para recudir una colección lo máximo posible.