```
1   import os
2   from time import sleep #utilizado como efecto estetico al principio de main()
3   import re
4
5   EXTENSIONES_BUSCADAS = (".txt",".md",".py",".c")
6   RUTA_CODIGO = os.path.join(".",os.path.basename(__file__))
7   MODOS = ("and:","or:","not:")
8
9   AYUDA = """
10  Se disponen de tres modos de busqueda: AND,OR o NOT. Las coincidencias son aquellos
    archivos que en su ruta o nombre respeten las condiciones de busqueda, ignorando
    mayusculas y minusculas.
11  En caso de ser archivos '.txt', '.md', '.py' o '.c' se buscara terminos dentro de
    los mismos, analizando su contenido.
12  Introduciendo <comando>: <terminos> se realiza una consulta. Un termino debe
    coincidir completamente al encontrado para ser contado como valido. DEBE DEJARSE UN
    ESPACIO LUEGO DE LOS DOS PUNTOS(':')
13
14  Busqueda OR:
15      Ejecutada con 'OR: <terminos>' o '<terminos>'. Devuelve las rutas de los
        archivos que coincidan con AL MENOS un termino.
16
17  Busqueda AND:
18      Ejecutada con 'AND: <terminos>'. Devuelve las rutas de los archivos que
        coincidan con TODOS los terminos.
19
20  Busqueda NOT:
21      Ejecutada con 'NOT: <termino>'. Devuelve las rutas de los archivos que NO
        coinciden con el unico termino dado.
22
23  Otros comandos especiales: "/*" para salir, "/c" para creditos.
24  """
25
26  CREDITOS = """TP2 - Catedra Wachenchauzer - Algoritmos y Programacion I - FIUBA 2017
27  Franco Giordano - 100608
28  """
29
30
31
32
33  def indexar_archivos():
34      """No recibe nada. Recorre los directorios y todos los subdirectorios desde
        donde se ejecuta el programa. Si un archivo coincide con EXTENSIONES_BUSCADAS se
        llamara a terminos_en_archivo.
35      Devuelve un diccionario con terminos (palabras) como claves, y rutas de archivos
        que contengan dicho termino como valores."""
36      INDICE_POR_RUTA = {}
37      for tupla in os.walk("."):
38          ruta_actual = tupla[0]
39          archivos_actuales = tupla[2]
40          terminos_ruta_actual = re.split("\W+", ruta_actual.lower())
41          for nombre_archivo in archivos_actuales:
42              ruta_archivo = os.path.join(ruta_actual,nombre_archivo)
43              if ruta_archivo == RUTA_CODIGO:
44                  continue
45
46              terminos_actuales = terminos_ruta_actual + re.split("\W+",
                nombre_archivo.lower())
47
48              if nombre_archivo.endswith(EXTENSIONES_BUSCADAS):
49                  terminos_actuales += terminos_en_archivo(ruta_archivo)
50
51              terminos_actuales_sin_vacios = [term for term in terminos_actuales if
                term != ""]
52              terminos_actuales_sin_repetidos =
                remover_repetidos(terminos_actuales_sin_vacios)
53              INDICE_POR_RUTA[ruta_archivo] = terminos_actuales_sin_repetidos
54
55      INDICE_POR_TERMINO = invertir_diccionario(INDICE_POR_RUTA)
56      cantidad_indexados = len(INDICE_POR_RUTA)
57      return INDICE_POR_TERMINO,cantidad_indexados
58
```

```
59
60
61  def invertir_diccionario(diccionario):
62      '''Dado un diccionario, devuelve un nuevo diccionario donde cada clave sera un
        valor del anterior, y sus valores seran claves del anterior.
63      Ej: {"animal": ["perro"], "persona": ["alumno","bombero"]}  ----->  {"perro":
        ["animal"], "alumno": ["persona"], "bombero": ["persona"]}'''
64      diccionario_invertido = {}
65      for clave, lista in diccionario.items():
66          for elemento in lista:
67              diccionario_invertido[elemento] = diccionario_invertido.get(elemento,
                []) + [clave]
68      return diccionario_invertido
69
70
71
72  def terminos_en_archivo(ruta):
73      '''Dada una ruta de un archivo, devuelve los terminos (palabras) que encuentra
        en el mismo como una lista, ignorando MAYUS/minus y repetidos.'''
74      with open(ruta) as archivo:
75          terminos_del_archivo = []
76          for linea in archivo:
77              linea_sin_salto = linea.rstrip("\n").lower()
78              terminos_linea = re.split("\W+",linea_sin_salto)
79              for palabra in terminos_linea:
80                  terminos_del_archivo += [palabra]
81      return remover_repetidos(terminos_del_archivo)
82
83  def recibir_comandos():
84      """Utilizada en main(). Imprime un prompt y recibe un input. Devuelve tres
        elementos: un elemento de MODOS (correspondiente al elegido), terminos elegidos
        (palabras), e input original (sin MAYUS/minus)."""
85      input_raw = input("> ").lower()
86      palabras = input_raw.split()
87      modo_elegido = "or:"
88
89      if input_raw.startswith(MODOS):
90          modo_elegido = palabras[0]
91          palabras.pop(0)
92
93      return modo_elegido,palabras,input_raw
94
95  def remover_repetidos(lista):
96      '''Dada una lista, devuelve una nueva desordenada sin elementos repetidos.'''
97      return list(set(lista))
98
99  def obtener_elem_compartidos_entre(lista1, lista2):
100     """Dadas dos listas, encuentra los elementos compartidos en ambas y los devuelve
        como una nueva lista."""
101     return [elemento1 for elemento1 in lista1 if elemento1 in lista2]
102
103 def decidir_comando_especial(cadena):
104     '''Utilizada en main(). Dada una cadena, decide qué "comando especial" ejecutar
        (imprimir ayuda, creditos o salir).'''
105     if cadena == "/*":
106         print("Adios!\n")
107         exit()
108     elif cadena == "/h":
109         print(AYUDA)
110     elif cadena == "/c":
111         print(CREDITOS)
112     else:
113         print("Comando no reconocido. ¿Has probado con '/h'?\n")
114
115
116 def main():
117     '''Funcion principal del programa.'''
118     print("Indexando archivos",end="",flush=True)
119     sleep(0.5)
120     for punto in "...":
121         print(punto,flush=True,end="")
122         sleep(0.5)
```

```
123
124         indice_invertido,cantidad = indexar_archivos()
125         print("\nListo! Se indexaron {} archivos".format(cantidad))
126         print("Puedes utlizar '/*' para salir en cualquier momento, '/h' para ayuda o
            '/c' para creditos\n")
127
128         while True:
129             modo_busqueda,terminos_a_buscar,input_raw = recibir_comandos()
130
131             if input_raw.startswith("/"):
132                 decidir_comando_especial(input_raw)
133                 continue
134
135             rutas_coincidentes = []
136             un_solo_term_a_buscar = len(terminos_a_buscar) == 1
137
138             if modo_busqueda == "or:":
139                 for termino in terminos_a_buscar:
140                     rutas_coincidentes += indice_invertido.get(termino,[])
141
142
143             elif modo_busqueda == "and:":
144                 for i,termino in enumerate(terminos_a_buscar):
145                     if termino not in indice_invertido:
146                         rutas_coincidentes = []
147                         break
148                     if i == 0:
149                         rutas_coincidentes = indice_invertido[termino]
150                     rutas_coincidentes =
                    obtener_elem_compartidos_entre(rutas_coincidentes,indice_invertido[ter
                    mino])
151
152             elif modo_busqueda == "not:":
153                 if not un_solo_term_a_buscar:
154                     print("La busqueda 'not' recibe un solo termino! Intente
                    nuevamente\n")
155                     continue
156                 rutas_no_validas = indice_invertido.get(terminos_a_buscar[0], [])
157                 for termino,rutas in indice_invertido.items():
158                     rutas_coincidentes += rutas
159                 rutas_coincidentes = [ruta for ruta in rutas_coincidentes if ruta not in
                    rutas_no_validas]
160
161             if rutas_coincidentes == []:
162                 print("No hay coincidencias\n")
163                 continue
164
165             rutas_coincidentes = remover_repetidos(rutas_coincidentes)
166             for r_coinc in rutas_coincidentes:
167                 print(r_coinc)
168             print()
169
170     main()
```