

# Memoria PI1

José Gallardo Harillo

## **Ej1:**

En este ejercicio metiendo como parámetro una lista de listas de enteros debemos imprimir una lista de enteros que contenga todos los números primos que se encuentren en la lista de listas del parámetro.

### Java

Método principal:

```
public static List<Integer> ej1(List<List<Integer>> ll){
    List<Integer> l = new ArrayList<Integer>();
    int i=0;
    while(i<ll.size()) {
        List<Integer> sl=ll.get(i);
        for(int j=0;j<sl.size();j++) {
            if(Math2.esPrimo(sl.get(j))==true) {
                l.add(sl.get(j));
            }
        }
        i++;
    }
    return l;
}
```

- 1-Crea una lista vacía l de tipo entero y la cuál será la que devolvamos al final.
- 2-Recorre por medio de un while la lista de listas y cada sublista i se anidará temporalmente en la variable sl.
- 3-Crea un bucle que recorre cada sublista para analizar cada entero contenido en ellas.
- 4-Por medio del método esPrimo de Math2 comprueba si es primo, si lo es se añadirá a la lista creada al principio.
- 5-Terminados los bucles devolverá la lista l de enteros.

Lectura del fichero:

```
public static List<List<Integer>> leer_f(String fichero){
    List<List<Integer>> datos = Files2.streamFromFile(fichero).map(linea->leerLineas(linea))
        .collect(Collectors.toList());
    return datos;
}

public static List<Integer> leerLineas(String linea){
    List<Integer> r = new ArrayList<>();
    if(linea.length()>0) {
        for(String n: linea.split(", ")) {
            r.add(Integer.parseInt(n));
        }
    }
    return r;
}
```

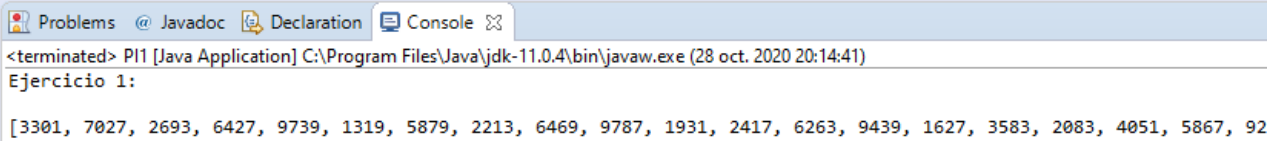
1-leer\_f coge el fichero y lo mapea usando el método auxiliar leer linear, que coge cada línea del fichero y lo vuelve lista de enteros.

2-Cada línea mapeada se contiene en la lista vacía datos creada al principio, siendo devuelta y devolviendo así el método una lista de listas de enteros.

Testeo:

```
List<List<Integer>> s = Ejercicio1.leer_f("ficheros/PI1Ej1DatosEntrada.txt");
List<Integer> res1 = Ejercicio1.ej1(s);
System.out.println("Ejercicio 1:");
System.out.println("");
System.out.println(res1);
```

Impresión:



The screenshot shows a Java IDE window with tabs for Problems, Javadoc, Declaration, and Console. The Console tab is active, displaying the output of the program. The output starts with a system message: "<terminated> PI1 [Java Application] C:\Program Files\Java\jdk-11.0.4\bin\javaw.exe (28 oct. 2020 20:14:41)". Below this, it prints "Ejercicio 1:" followed by a line break. The final output is a list of integers: "[3301, 7027, 2693, 6427, 9739, 1319, 5879, 2213, 6469, 9787, 1931, 2417, 6263, 9439, 1627, 3583, 2083, 4051, 5867, 92".

## C

Método principal:

```
list ej1(list l1){
    list lres= list_empty(int_type);
    iterator it = list_iterable(&l1);
    while(iterable_has_next(&it)){
        list sl = *(list*) iterable_next(&it);
        for(int j=0;j<sl.size;j++){
            bool esPrimo= es_primo( *(int*) list_get(&sl,j));
            if(esPrimo==true){
                list_add(&lres,list_get(&sl,j));
            }
        }
    }
    return lres;
}
```

Los pasos son los mismos que en java pero traducidos a c y utilizando para el recorrido de la lista de listas un iterador en vez de la i tradicional, y utilizando el método es\_Prime del Math2 de la librería de Miguel Toro.

Lectura del fichero:

```
list leer_f(char* file){
    list r = list_empty(list_type);
    iterator it = file_iterable_pchar(file);
    while(iterable_has_next(&it)){
        list li = list_empty(int_type);
        char * c = (char*) iterable_next(&it);
        prelectura(&li, c);
        list_add(&r,&li);
    }
    return r;
}

void prelectura(list *l, char* caracter){
    iterator it = split_iterable_pchar(caracter, " ");
    while(iterable_has_next(&it)){
        char* chara = (char*) iterable_next(&it);
        int n = int_parse_s(chara);
        list_add(l, &n);
    }
}
```

1-leer\_f en el caso de C mediante un iterador itera al fichero entrante y aparte crea una lista vacía l de tipo lista, mientras que haya una línea para iterar en el fichero cada línea se irá anidando temporalmente en el puntero c, y creando para cada una una lista de enteros vacía li para añadir ahí los enteros contenidos en la línea iterada.

2-Mediante prelectura iteramos la línea actual cortándola por las comas y mediante while anidamos cada trozo en chara para luego parsearlo a entero y añadirlo en la lista metida como parámetro.

3-Tras prelectura cada lista li rellena será añadida a la lista r creada al principio para ser devuelta por el método.

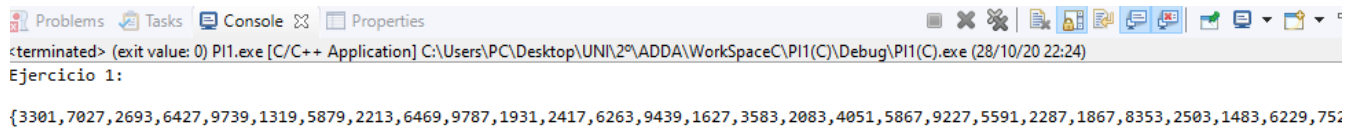
Testeo:

```
char mem[256];

printf("Ejercicio 1:");
printf(" \n\n");

char* fichero1 = "ficheros/PI1Ej1DatosEntrada.txt";
list lineas = leer_f(fichero1);
list res1 = ej1(lineas);
list_tostring(&res1,mem);
printf("%s\n",mem);
printf(" \n\n");
```

Impresión:



The screenshot shows a Visual Studio Code console window with the following content:

```
<terminated> (exit value: 0) PI1.exe [C/C++ Application] C:\Users\PC\Desktop\UNI\2º\ADDA\WorkSpaceC\PI1(C)\Debug\PI1(C).exe (28/10/20 22:24)
Ejercicio 1:
{3301,7027,2693,6427,9739,1319,5879,2213,6469,9787,1931,2417,6263,9439,1627,3583,2083,4051,5867,9227,5591,2287,1867,8353,2503,1483,6229,751}
```

## Ej2:

En este ejercicio metiendo como parámetro un entero debemos imprimir todos los números primos al cuadrado que haya desde el 2 hasta el número metido como parámetro (límite) mediante una cadena.

### Java

Método principal:

```

public static String ej2(Integer limite) {
    int i = 2;
    String mensaje= "";
    while(i<=limite) {
        if(Math2.esPrimo(i)==true) {
            Integer primoCuadrado = i*i;
            String primoStr = primoCuadrado.toString().concat("\n");
            mensaje = mensaje.concat(primoStr);
        }
        i++;
    }
    return mensaje;
}

```

1-Iniciamos la i del bucle while con 2 para así descartar los número 0 y 1.

2-Creamos una cadena vacia ("" ) a la que iremos concatenando los números primos y los saltos de renglón.

3-Mediante el bucle while, que terminará cuando i sea igual al límite, filtrará mediante if todos los números primos que se encuentre para luego multiplicarlos por ellos mismos y luego mediante concat ir concatenando a la cadena vacía los números parseados a string y los saltos de renglón (\n).

4-Devolverá la cadena con todos los números primos cada uno en un renglón distinto.

Lectura del fichero:

```

public static List<Integer> leer_f2(String fichero){
    List<Integer> datos = Files2.streamFromFile(fichero)
        .filter(x->x.startsWith("L")).map(linea->leerLineas2(linea))
        .collect(Collectors.toList());
    return datos;
}

public static Integer leerLineas2(String linea){
    String[] subLinea = linea.split(": ");
    Integer n = Integer.parseInt(subLinea[1]);
    return n;
}

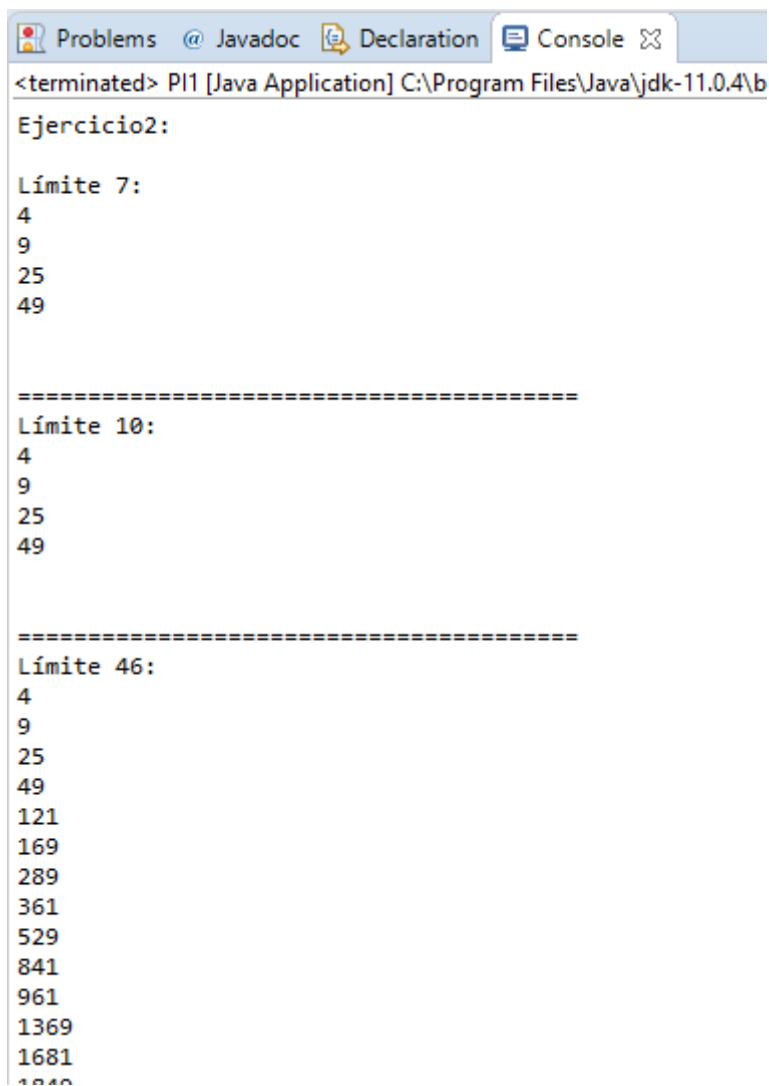
```

La estructura es básicamente la misma que la del primer ejercicio solo que en esta Cacheamos que las líneas comienzan por L (Cada línea con contenido empieza con "Limite"), y leerLineas2 crea subStrings troceando por ":" para así ignorar la parte la línea que contiene "limite" y así la otra parsearla a entero y devolver el entero para al final del método leer\_f2 devolver una lista con todos los enteros que harán de límites.

Testeo:

```
System.out.println("");
System.out.println("Ejercicio2:");
System.out.println("");
List<Integer> s2= Ejercicio2.leer_f2("ficheros/PI1Ej2DatosEntrada.txt");
for(int i=0;i<s2.size();i++) {
    String res2 = Ejercicio2.ej2(s2.get(i));
    System.out.println("Límite "+s2.get(i)+":");
    System.out.println(res2);
    System.out.println("");
    System.out.println("=====");
}
```

Impresión:



```
<terminated> PI1 [Java Application] C:\Program Files\Java\jdk-11.0.4\bin
Ejercicio2:

Límite 7:
4
9
25
49

=====

Límite 10:
4
9
25
49

=====

Límite 46:
4
9
25
49
121
169
289
361
529
841
961
1369
1681
1849
```

## C

Método principal:

```
string ej2(int limite){
    int i = 2;
    string mensaje = string_of_pchar("");
    char mem[10];
    while(i<=limite) {
        bool esPrimo= es_primo(i);
        if(esPrimo==true) {
            int primoCuadrado = i*i;
            string memstr = string_of_pchar(int_tostring(&primoCuadrado,mem));
            string aparte = string_of_pchar("\n");
            string_add_string(&mensaje,&memstr);
            string_add_string(&mensaje,&aparte);
        }
        i++;
    }
    return mensaje;
}
```

---

La estructura es básicamente la misma que hemos usado en java solo que aquí usamos los string y sus métodos de la librería string.h.

Lectura del fichero:

```
list leer_f2(char* file){
    list r = list_empty(int_type);
    iterator it = file_iterable_pchar(file);
    while(iterable_has_next(&it)){
        char * c = (char*) iterable_next(&it);
        iterator it2 = split_iterable_pchar(c,": ");
        char* s = iterable_next(&it2);
        char* s2 = iterable_next(&it2);
        int n = int_parse_s(s2);
        list_add(&r,&n);
    }
    return r;
}
```

La estructura es básicamente la misma que la de la lectura anterior solo que aquí no necesitamos un método auxiliar, sino que directamente en el bucle while trozamos la línea iterada por ":" y nos quedamos con el segundo trozo que es el que contiene el número para así luego parsearlo y devolverlo (por eso ignoramos el puntero char s).

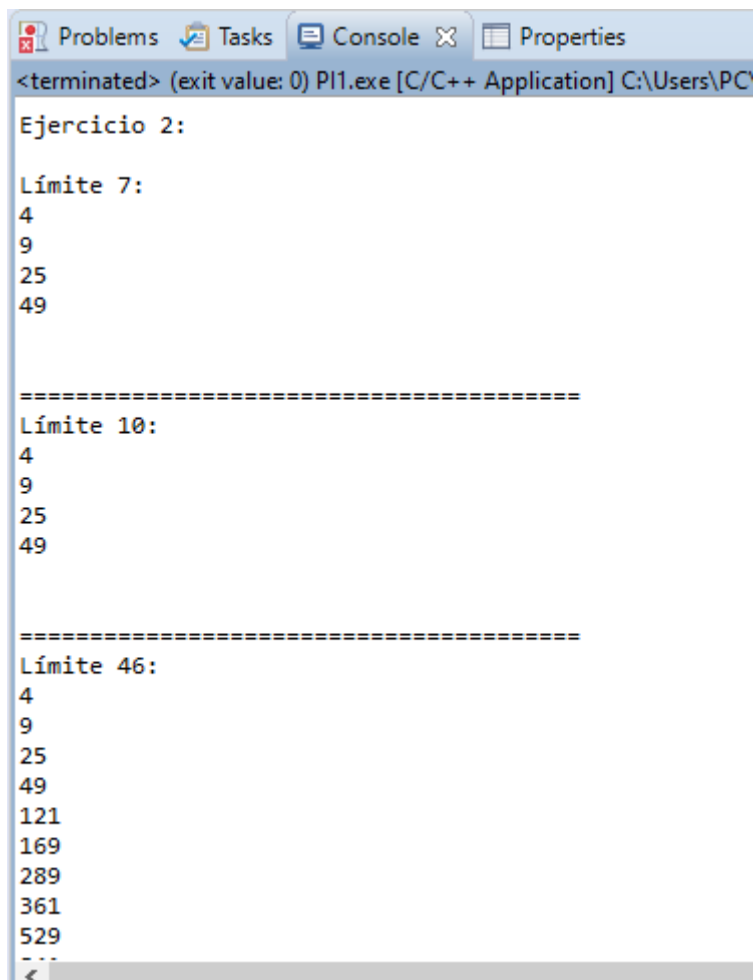
Testeo:

```
char mem2[256];

printf("Ejercicio 2:");
printf(" \n\n");

char* fichero2 = "ficheros/PI1Ej2DatosEntrada.txt";
list lineas2 = leer_f2(fichero2);
for(int i=0;i<lineas2.size;i++){
    int n = *(int*) list_get(&lineas2,i);
    string res2 = ej2(n);
    string_tostring(&res2,mem2);
    printf("Límite %d:\n",n);
    printf("%s",mem2);
    printf(" \n\n");
    printf("=====\n");
}
```

Impresión:



```
Problems Tasks Console Properties
<terminated> (exit value: 0) PI1.exe [C/C++ Application] C:\Users\PC

Ejercicio 2:

Límite 7:
4
9
25
49

=====
Límite 10:
4
9
25
49

=====
Límite 46:
4
9
25
49
121
169
289
361
529
729
961
1296
1681
2116
2601
3136
3721
4356
5041
5776
6561
7400
8281
9216
10201
11236
12321
13456
14641
15876
17161
18500
19881
21316
22801
24336
25921
27556
29241
30976
32761
34600
36481
38401
40360
42361
44401
46481
48600
50761
52961
55200
57481
59800
62161
64561
67000
69481
71996
74541
77120
79736
82381
85060
87776
90521
93300
96116
98961
101836
104741
107676
110641
113636
116660
119716
122800
125916
129056
132221
135416
138640
141896
145176
148481
151816
155180
158576
161996
165440
168916
172416
175940
179481
183040
186616
190216
193840
197481
201140
204816
208516
212240
215981
219740
223516
227316
231140
234981
238840
242716
246616
250540
254481
258440
262416
266416
270440
274481
278540
282616
286716
290840
294981
299140
303316
307516
311740
315981
320240
324516
328816
333140
337481
341840
346216
350616
355040
359481
363940
368416
372916
377440
381981
386540
391116
395716
400340
404981
409640
414316
419016
423740
428481
433240
438016
442816
447640
452481
457340
462216
467116
472040
476981
481940
486916
491916
496940
501981
507040
512116
517216
522340
527481
532640
537816
543016
548240
553481
558740
564016
569316
574640
579981
585340
590716
596116
601540
606981
612440
617916
623416
628940
634481
639981
645540
651116
656716
662340
667981
673640
679316
685016
690740
696481
702240
708016
713816
719640
725481
731340
737216
743116
749040
754981
760940
766916
772916
778940
784981
791040
797116
803216
809340
815481
821640
827816
834016
840240
846481
852740
859016
865316
871640
877981
884340
890716
897116
903540
909981
916440
922916
929416
935940
942481
949040
955616
962216
968840
975481
982140
988816
995516
1002240
1008981
1015740
1022516
1029316
1036140
1042981
1049840
1056716
1063616
1070540
1077481
1084440
1091416
1098416
1105440
1112481
1119540
1126616
1133716
1140840
1147981
1155140
1162316
1169516
1176740
1183981
1191240
1198516
1205816
1213140
1220481
1227840
1235216
1242616
1250040
1257481
1264940
1272416
1279916
1287440
1294981
1302540
1310116
1317716
1325340
1332981
1340640
1348316
1356016
1363740
1371481
1379240
1387016
1394816
1402640
1410481
1418340
1426216
1434116
1442040
1449981
1457940
1465916
1473916
1481940
1489981
1498040
1506116
1514216
1522340
1530481
1538640
1546816
1555016
1563240
1571481
1579740
1588016
1596316
1604640
1612981
1621340
1629716
1638116
1646540
1654981
1663440
1671916
1680416
1688940
1697481
1706040
1714616
1723216
1731840
1740481
1749140
1757816
1766516
1775240
1783981
1792740
1801516
1810316
1819140
1827981
1836840
1845716
1854616
1863540
1872481
1881440
1890416
1899416
1908440
1917481
1926540
1935616
1944716
1953840
1962981
1972140
1981316
1990516
1999740
2008981
2018240
2027516
2036816
2046140
2055481
2064840
2074216
2083616
2093040
2102481
2111940
2121416
2130916
2140440
2149981
2159540
2169116
2178716
2188340
2197981
2207640
2217316
2227016
2236740
2246481
2256240
2266016
2275816
2285640
2295481
2305340
2315216
2325116
2335040
2344981
2354940
2364916
2374916
2384940
2394981
2405040
2415116
2425216
2435340
2445481
2455640
2465816
2476016
2486240
2496481
2506740
2517016
2527316
2537640
2547981
2558340
2568716
2579116
2589540
2599981
2610440
2620916
2631416
2641940
2652481
2663040
2673616
2684216
2694840
2705481
2716140
2726816
2737516
2748240
2758981
2769740
2780516
2791316
2802140
2812981
2823840
2834716
2845616
2856540
2867481
2878440
2889416
2900416
2911440
2922481
2933540
2944616
2955716
2966840
2977981
2989140
3000316
3011516
3022740
3033981
3045240
3056516
3067816
3079140
3090481
3101840
3113216
3124616
3136040
3147481
3158940
3170416
3181916
3193440
3204981
3216540
3228116
3239716
3251340
3262981
3274640
3286316
3298016
3309740
3321481
3333240
3345016
3356816
3368640
3380481
3392340
3404216
3416116
3428040
3439981
3451940
3463916
3475916
3487940
3499981
3512040
3524116
3536216
3548340
3560481
3572640
3584816
3597016
3609240
3621481
3633740
3646016
3658316
3670640
3682981
3695340
3707716
3720116
3732540
3744981
3757440
3769916
3782416
3794940
3807481
3820040
3832616
3845216
3857840
3870481
3883140
3895816
3908516
3921240
3933981
3946740
3959516
3972316
3985140
3997981
4010840
4023716
4036616
4049540
4062481
4075440
4088416
4101416
4114440
4127481
4140540
4153616
4166716
4179840
4192981
4206140
4219316
4232516
4245740
4258981
4272240
4285516
4298840
4312140
4325481
4338840
4352216
4365616
4379040
4392481
4405940
4419416
4432916
4446440
4459981
4473540
4487116
4500716
4514340
4527981
4541640
4555316
4569016
4582740
4596481
4610240
4624016
4637816
4651640
4665481
4679340
4693216
4707116
4721040
4734981
4748940
4762916
4776916
4790940
4804981
4819040
4833116
4847216
4861340
4875481
4889640
4903816
4917981
4932140
4946316
4960516
4974740
4988981
5003240
5017516
5031816
5046140
5060481
5074840
5089216
5103616
5118040
5132481
5146940
5161416
5175916
5190440
5204981
5219540
5234116
5248716
5263340
5277981
5292640
5307316
5322016
5336740
5351481
5366240
5381016
5395816
5410640
5425481
5440340
5455216
5470116
5485040
5499981
5514940
5529916
5544916
5559940
5574981
5589981
5605040
5620116
5635216
5650340
5665481
5680640
5695816
5711016
5726240
5741481
5756740
5772016
5787316
5802640
5817981
5833340
5848716
5864116
5879540
5894981
5910440
5925916
5941416
5956940
5972481
5988040
6003616
6019216
6034840
6050481
6066140
6081816
6097516
6113240
6128981
6144740
6160516
6176316
6192140
6207981
6223840
6239716
6255616
6271540
6287481
6303440
6319416
6335416
6351440
6367481
6383540
6399616
6415716
6431840
6447981
6464140
6480316
6496516
6512740
6528981
6545240
6561516
6577816
6594140
6610481
6626840
6643216
6659616
6676040
6692481
6708940
6725416
6741916
6758440
6774981
6791540
6808116
6824716
6841340
6857981
6874640
6891316
6908016
6924740
6941481
6958240
6975016
6991816
7008640
7025481
7042340
7059216
7076116
7093040
7109981
7126940
7143916
7160916
7177940
7194981
7212040
7229116
7246216
7263340
7280481
7297640
7314816
7332016
7349240
7366481
7383740
7401016
7418316
7435640
7452981
7470340
7487716
7505116
7522540
7539981
7557440
7574916
7592416
7609940
7627481
7645040
7662616
7680216
7697840
7715481
7733140
7750816
7768516
7786240
7803981
7821740
7839516
7857316
7875140
7892981
7910840
7928716
7946616
7964540
7982481
8000440
8018416
8036416
8054440
8072481
8090540
8108616
8126716
8144840
8162981
8181140
8199316
8217516
8235740
8253981
8272240
8290516
8308840
8327140
8345481
8363840
8382216
8400616
8419040
8437481
8455940
8474416
8492916
8511440
8529981
8548540
8567116
8585716
8604340
8622981
8641640
8660316
8679016
8697740
8716481
8735240
8754016
8772816
8791640
8810481
8829340
8848216
8867116
8886040
8904981
8923940
8942916
8961916
8980940
9000016
9019140
9038216
9057340
9076481
9095640
9114816
9134016
9153240
9172481
9191740
9211016
9230316
9249640
9268981
9288340
9307716
9327116
9346540
9365981
9385440
9404916
9424416
9443940
9463481
9483040
9502616
9522216
9541840
9561481
9581140
9600816
9620516
9640240
9659981
9679740
9699516
9719316
9739140
9758981
9778840
9798716
9818616
9838540
9858481
9878440
9898416
9918416
9938440
9958481
9978540
9998616
10018716
10038840
10058981
10079140
10099316
10119516
10139740
10159981
10180240
10200516
10220816
10241140
10261481
10281840
10302216
10322616
10343040
10363481
10383940
10404416
10424916
10445440
10465981
10486540
10507116
10527716
10548340
10568981
10589640
10610316
10631016
10651740
10672481
10693240
10714016
10734816
10755640
10776481
10797340
10818216
10839116
10860040
10880981
10901940
10922916
10943916
10964940
10985981
11007040
11028116
11049216
11070340
11091481
11112640
11133816
11155016
11176240
11197481
11218740
11239981
11261240
11282516
11303816
11325140
11346481
11367840
11389216
11410616
11432040
11453481
11474940
11496416
11517916
11539440
11560981
11582540
11604116
11625716
11647340
11668981
11690640
11712316
11734016
11755740
11777481
11799240
11821016
11842816
11864640
11886481
11908340
11930216
11952116
11974040
11995981
12017940
12039916
12061916
12083940
12105981
12128040
12150116
12172216
12194340
12216481
12238640
12260816
12283016
12305240
12327481
12349740
12372016
12394316
12416640
12438981
12461340
12483716
12506116
12528540
12550981
12573440
12595916
12618416
12640940
12663481
12686040
12708616
12731216
12753840
12776481
12799140
12821816
12844516
12867240
12889981
12912740
12935516
12958316
12981140
13003981
13026840
13049716
13072616
13095540
13118481
13141440
13164416
13187416
13210440
13233481
13256540
13279616
13302716
13325840
13348981
13372140
13395316
13418516
13441740
13464981
13488240
13511516
13534816
13558140
13581481
13604840
13628216
13651616
13675040
13698481
13721940
13745416
13768916
13792440
13815981
13839540
13863116
13886716
13910340
13933981
13957640
13981316
14005016
14028740
14052481
14076240
14099981
14123740
14147516
14171316
14195140
14218981
14242840
14266716
14290616
14314540
14338481
14362440
14386416
14410416
14434440
14458481
14482540
14506616
14530716
14554840
14578981
14603140
14627316
14651516
14675740
14699981
14724240
14748516
14772816
14797140
14821481
14845840
14870216
14894616
14919040
14943481
14967940
14992416
15016916
15041440
15065981
15090540
15115116
15139716
15164340
15188981
15213640
15238316
15263016
15287740
15312481
15337240
15362016
15386816
15411640
15436481
15461340
15486216
15511140
15536081
15561040
15586016
15611016
15636040
15661081
15686140
15711216
15736316
15761440
15786581
15811740
15836916
15862116
15887340
15912581
15937840
15963116
15988416
16013740
16039081
16064440
16089816
16115240
16140681
16166140
16191616
16217140
16242681
16268240
16293816
16319416
16345040
16370681
16396340
16422016
16447716
16473440
16499181
16524940
16550716
16576540
16602381
16628240
16654116
16680016
16705940
16731881
16757840
16783816
16809816
16835840
16861881
16887940
16914016
16940116
16966240
16992381
17018540
17044716
17070916
17097140
17123381
17149640
17175916
17202216
17228540
17254881
17281240
17307616
17334016
17360440
17386881
17413340
17439816
17466316
17492840
17519381
17545940
17572516
17599116
17625740
17652381
17679040
17705716
17732416
17759140
17785881
17812640
17839416
17866216
17893040
17919881
17946740
17973616
18000516
18027416
18054340
18081281
18108240
18135216
18162216
18189240
18216281
18243340
18270416
18297516
18324640
18351781
18378940
18406116
18433316
18460540
18487781
18515040
18542316
18569616
18596940
18624281
18651640
18679016
18706416
18733840
18761281
18788740
18816216
18843716
18871240
18898781
18926340
18953916
18981516
19009140
19036781
19064440
19092116
19119816
19147540
19175281
19203040
19230816
19258616
19286440
19314281
19342140
19370016
19397916
19425840
19453781
19481740
19509716
19537716
19565740
19593781
19621840
19649916
19678016
19706140
19734281
19762440
19790616
19818816
19847040
19875281
19903540
19931816
19960140
19988481
20016840
20045216
20073616
20102040
20130481
20158940
20187416
20215916
20244440
20272981
20301540
20330116
20358716
20387340
20415981
20444640
20473316
20502016
20530740
20559481
20588240
20617016
20645816
20674640
20703481
20732340
20761216
20790116
20819040
20847981
20876940
20905916
20934916
20963940
20992981
21022040
21051116
21080216
21109340
21138481
21167640
21196816
21226016
21255240
21284481
21313740
21343016
21372316
21401640
21430981
21460340
21489716
21519116
21548540
21577981
21607440
21636916
21666416
21695940
21725481
21755040
21784616
21814216
21843840
21873481
21903140
21932816
21962516
21992240
22021981
22051740
22081516
22111316
22141140
22170981
22200840
22230716
22260616
22290540
22320481
22350440
22380416
22410416
22440440
22470481
22500540
22530616
22560716
22590840
22620981
22651140
22681316
22711516
22741740
22771981
22802240
22832516
22862816
22893140
22923481
22953840
22984216
23014616
23045040
23075481
23105940
23136416
23166916
23197440
23227981
23258540
23289116
23319716
23350340
23380981
23411640
23442316
23473016
23503740
23
```



### Ej3:

En este ejercicio metiendo como parámetro una lista de puntos debemos de imprimir un diccionario con los 4 cuadrantes como claves y la suma de las x pertenecientes a cada cuadrante como valor.

#### Java

Método principal:

```
public static Map<Punto2D.Cuadrante, Double> ej3(List<Punto2D> l){  
  
    Map<Punto2D.Cuadrante, Double> diccionario = new HashMap<Punto2D.Cuadrante, Double>();  
    int i=0;  
  
    while(i<l.size()) {  
        Punto2D.Cuadrante cuadrante = l.get(i).getCuadrante();  
        Double valor = diccionario.get(cuadrante);  
        Double v = 0.0;  
  
        if (valor==null){  
            v=l.get(i).getX();  
            diccionario.put(cuadrante, v);  
        }  
  
        else if(valor!=null){  
            v = valor+l.get(i).getX();  
            diccionario.put(cuadrante, v);  
        }  
  
        i++;  
    }  
  
    return diccionario;  
}
```

1-Creamos un diccionario vacío de tipo cuadrante como clave y tipo double como valor.

2-Mediante un bucle while recorreremos la lista de puntos metida como parámetro y anidamos en la variable cuadrante el cuadrante al que pertenezca el punto actual.

3-Sacamos el valor contenido en el diccionario con el cuadrante del punto actual como referencia clave e inicializamos a 0 una variable v que sea la que vaya luego a actualizar el valor del cuadrante actual.

4-Mediante un bucle if comprobamos si ese cuadrante lleva algún valor x ya contenido para si no crearlo y ponerle de valor inicial la x del punto actual. Si lleva ya algún valor contenido el cuadrante este será modificado sumándole la x del punto actual.

5-Pasado el bucle while el método devolvería el diccionario que creamos al principio con la suma de las x de cada cuadrante.

Lectura del fichero:

```
public static List<Punto2D> leer_f3 (String f){
    return Files2.streamFromFile(f).filter(x->x.startsWith("(")).map(x->LeerLineas3(x))
        .collect(Collectors.toList());
}

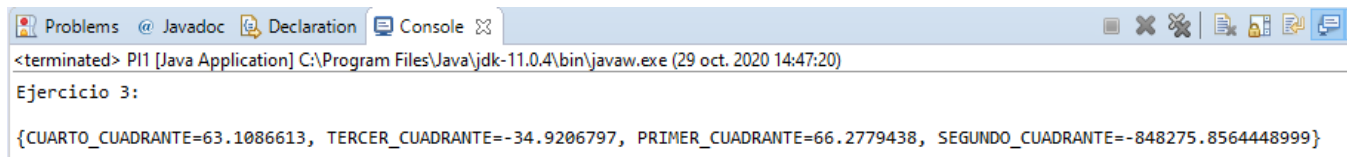
public static Punto2D leerLineas3(String linea) {
    String[] componentes = linea.substring(1,linea.length()-1).split(",");
    Double x = Double.parseDouble(componentes[0]);
    Double y = Double.parseDouble(componentes[1]);
    return Punto2D.create(x,y);
}
```

La estructura es básicamente la de la lectura del ejercicio 2 solo que aquí en el método auxiliar en vez de trocear para sacar la parte entera troceamos para sacar la parte x e y para de esa forma crear un punto con ese x e y.

Testeo:

```
List<Punto2D> lp = Ejercicio3.leer_f3("ficheros/PI1Ej3DatosEntrada.txt");
Map<Punto2D.Cuadrante, Double> res3= Ejercicio3.ej3(lp);
System.out.println("");
System.out.println("Ejercicio 3:");
System.out.println("");
System.out.println(res3);
```

Impresión:



The screenshot shows a Java IDE window with a console tab. The console output is as follows:

```
<terminated> PI1 [Java Application] C:\Program Files\Java\jdk-11.0.4\bin\javaw.exe (29 oct. 2020 14:47:20)
Ejercicio 3:
{CUARTO_CUADRANTE=63.1086613, TERCER_CUADRANTE=-34.9206797, PRIMER_CUADRANTE=66.2779438, SEGUNDO_CUADRANTE=-848275.8564448999}
```

## C

Método principal:

```
hash_table ej3(list l){  
  
    iterator it = list_iterable(&l);  
    hash_table diccionario = hash_table_empty(int_type,double_type);  
    punto *p;  
    double x= 0.0;  
    while(iterable_has_next(&it)) {  
  
        p= (punto*) iterable_next(&it);  
        Cuadrante cuadrante = punto_cuadrante(p);  
        double * valor = (double*) hash_table_get(&diccionario,&cuadrante);  
        x = p->x;  
        double v = 0.0;  
  
        if (valor==NULL){  
            v=x;  
            hash_table_put(&diccionario,&cuadrante,&v);  
        }  
        else if(valor!=NULL){  
            v = *valor+x;  
            hash_table_put(&diccionario,&cuadrante,&v);  
        }  
    }  
    return diccionario;  
}
```

La estructura básicamente es como la de java solo que aquí usamos para recorrer la lista de puntos un iterador y aparte para conseguir el punto actual usamos un puntero tipo punto.

Lectura del fichero:

```
void leer_f3(const char* file, list* l){  
    iterator it = file_iterable_pchar(file);  
    while(iterable_has_next(&it)){  
        char * c = (char*) iterable_file_next(&it);  
        punto p = punto_parse_s(c);  
        list_add(l,&p);  
    }  
}
```

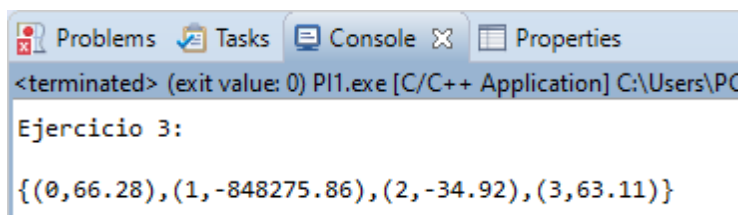
La estructura es parecida a la de las otras dos lecturas solo que en esta tenemos que meterle la lista que queremos llenar de puntos como parámetro.

Testeo:

```
char mem3[256];

printf("Ejercicio 3:");
printf(" \n\n");
const char* fichero3 = "ficheros/PI1Ej3DatosEntrada.txt";
list r = list_empty(punto_type);
leer_f3(fichero3,&r);
hash_table res3 = ej3(r);
hash_table_tostring(&res3,mem3);
printf("%s\n",mem3);
```

Impresión:



The screenshot shows a console window with tabs for Problems, Tasks, Console, and Properties. The Console tab is active, displaying the output of the program. The output consists of the text "Ejercicio 3:" followed by a set of four points in a set notation: {(0,66.28),(1,-848275.86),(2,-34.92),(3,63.11)}.

```
<terminated> (exit value: 0) PI1.exe [C/C++ Application] C:\Users\PC
Ejercicio 3:
{(0,66.28),(1,-848275.86),(2,-34.92),(3,63.11)}
```