

# Programación Orientada al Objeto - 549108

## Tarea 3

Alumnos:

Nicolás Borcoski S

Aldo Mellado O.

Esteban Paz F.

Profesor: Mario Medina C.

# Programación Orientada al Objeto - 549108

## Tarea 3

Alumnos:

Nicolás Borcoski S

Aldo Mellado O.

Esteban Paz F.

Profesor: Mario Medina C.

22 de Junio de 2018

## Índice

1. El método de búsqueda de la proporción áurea	2
2. Buscando anagramas	4

# 1. El método de búsqueda de la proporción áurea

Para comenzar, se diseñó el código pensando en un vector de enteros pequeño y siguiendo el paso a paso entregado en el enunciado del problema.

De esta manera primero se ordena mediante la función `sort()` el vector de entrada. Luego, se calcula el valor de  $\tau = \frac{\sqrt{5}-1}{2}$  y se genera un contador y un control de salida `out`.

```
1  vector<int> v = {9, 1, 5, 4, 3, 6, 8, 7, 2};
2  sort(v.begin(), v.end());
3  auto tau = (sqrt(5)-1)/2;
4  int contador = 0;
5  bool out = 0;
6  int dato = 5;
```

Se divide la operación en dos casos, el primero que se ejecuta una sola vez, donde se crean los iteradores a los espacios de memoria del vector y se verifica que el dato no se encuentre en los extremos del vector.

```
1  vector<int>::iterator x1 = v.begin();
2  vector<int>::iterator x2 = (v.end()-1);
3  vector<int>::iterator x3 = (v.end());
4  contador++;
5  contador++;
6  if(*x1 == dato || *x2 == dato){
7      out = 1;
8  }
```

Luego, un caso iterativo, donde se genera un nuevo  $x_3$  en cada ciclo y, según el caso, un nuevo  $x_1$  y  $x_2$ . En caso de encontrarse el número, o bien cumplirse la condición que indica según enunciado que el número no está en el vector, se activa el control de salida.

```
1  while(out != 1){
2      *x3 = *x2 - (tau * ((*x2)-(*x1)));
3      contador++;
4      if(*x3 == dato){
5          out = 1;
6      }
7      contador++;
8      if(*x3 > dato){
9          *x2 = *x3;
10     }
11     contador++;
12     if(*x3 < dato){
13         *x1 = *x3;
14     }
15     contador++;
16     contador++;
17     contador++;
18     if(*x1 == *x2 || *x1 == dato || *x2 == dato){
19         contador = contador*(-1);
20         out = 1;
21     }
22 }
```

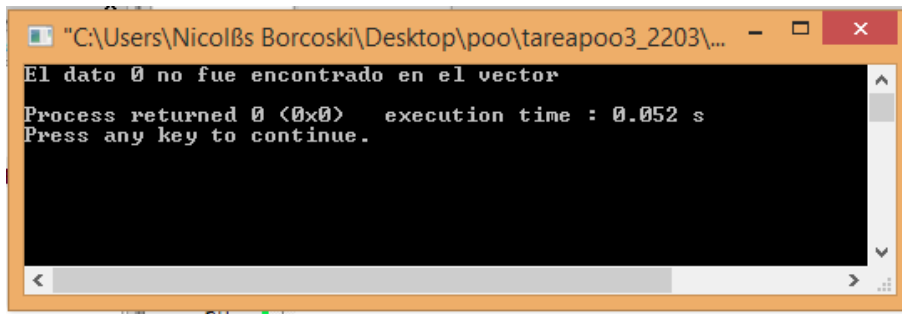


Figura 1: Ejecución del código para  $dato = 0$

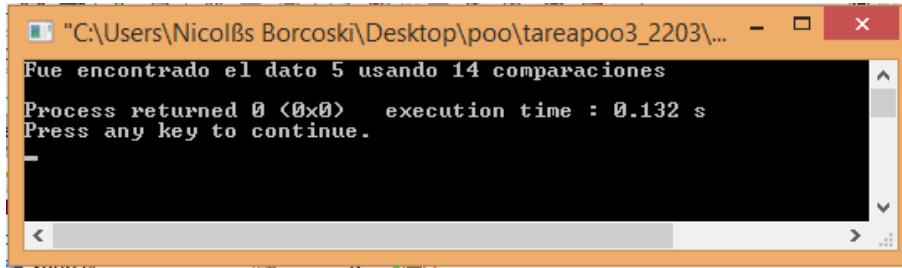


Figura 2: Ejecución del código para  $dato = 5$

Ahora, generalizamos la función para todo tipo de dato, así:

```

1  template <class T> bool GoldenSectionSearch(vector<T>& v, const T& dato){
2      ...
3      typename vector<T>::iterator x1 = v.begin();
4      typename vector<T>::iterator x2 = (v.end() - 1);
5      typename vector<T>::iterator x3 = (v.end());
6      ...
7  }

```

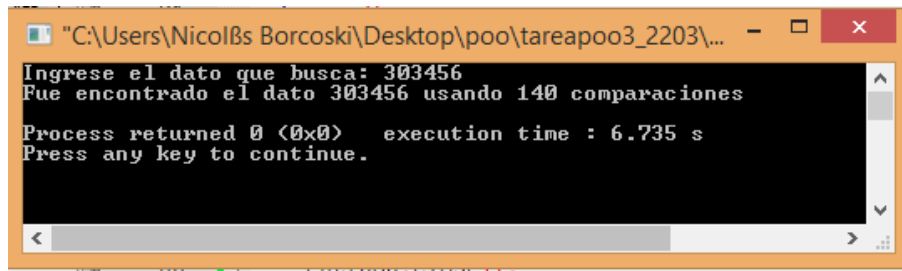
Y en el main del código se cargan los archivos entregados vía *INFODA*, además de interacción vía consola:

```

1  int main()
2  {
3      ifstream entrada("MillonDeInts.txt");
4      int temp;
5      vector<int> v;
6      while(entrada >> temp)
7      {
8          v.push_back(temp);
9      }
10     entrada.close();
11
12     int dato;
13     cout << "Ingresa el dato que busca: ";
14     cin >> dato;
15
16     GoldenSectionSearch(v, dato);
17
18     return 0;
19 }

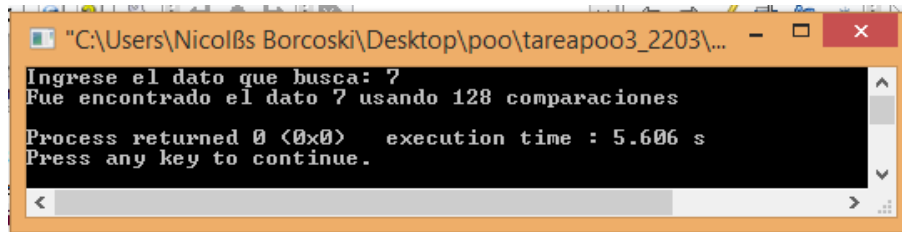
```

Esto se hace tanto con el .txt de enteros como con el de double, obteniendo, por ejemplo, los siguientes resultados:



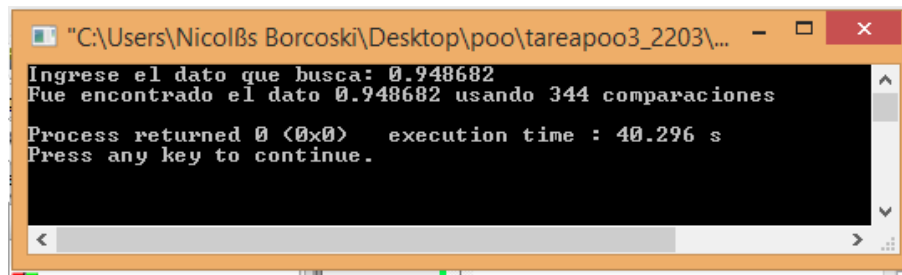
```
"C:\Users\NicolBs Borcoski\Desktop\poo\tareapoo3_2203\... - [X]
Ingrese el dato que busca: 303456
Fue encontrado el dato 303456 usando 140 comparaciones
Process returned 0 (0x0)   execution time : 6.735 s
Press any key to continue.
```

Figura 3: Ejecución del código para  $dato = 303456$  en *MillonDeInts.txt*



```
"C:\Users\NicolBs Borcoski\Desktop\poo\tareapoo3_2203\... - [X]
Ingrese el dato que busca: 7
Fue encontrado el dato 7 usando 128 comparaciones
Process returned 0 (0x0)   execution time : 5.606 s
Press any key to continue.
```

Figura 4: Ejecución del código para  $dato = 7$  en *MillonDeInts.txt*



```
"C:\Users\NicolBs Borcoski\Desktop\poo\tareapoo3_2203\... - [X]
Ingrese el dato que busca: 0.948682
Fue encontrado el dato 0.948682 usando 344 comparaciones
Process returned 0 (0x0)   execution time : 40.296 s
Press any key to continue.
```

Figura 5: Ejecución del código para  $dato = 0,948682$  en *MillonDeReals.txt*

## 2. Buscando anagramas

Para este problema lo que se hizo fue ingresar todos los nombres, de hombre y mujeres, además de los apellidos, en minúsculas; solamente las del principio, pues se nota que hay nombres con mayúsculas entremedio, pero aquellas letras fueron consideradas como caracteres diferentes a su minúscula, no así la primera letra, que se escribe en mayúsculas por regla ortográfica.

Luego se recibieron los datos ingresados por usuario y también fueron llevados a minúsculas, y se concatenó el nombre y el apellido, en la línea 66. Además se ordenan las letras en orden alfabético para no tener problemas con la función *is\_permutation()*, la cual eventualmente podría saltarse opciones, si es que está configurada de forma alfabética.

Luego se ponen condiciones para saber si buscar en la lista de nombres de mujer o de hombre, y se utilizan dos ciclos *while* para recorrerlas completamente, fijando un nombre, y barriendo todos los apellidos. Este proceso haciendo una comparación con las permutaciones de la concatenación del nombre y apellido ingresados por el usuario; si es verdadero entonces se pide que guarde el nombre y apellido en un vector de strings llamado *respuestas*. Donde se utiliza un *push\_back()* y se hace un

pequeño arreglo para que se guarden los nombres y apellidos separados por un espacio y con mayúsculas.

A continuación se muestra una impresión de pantalla de un ejemplo, que tarda 611s en ejecutarse:

Figura 6: Ejecución del código para el nombre Abel López

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 #include <iterator>
5 #include <fstream>
6 #include <string>
7
8
9 using namespace std;
10 string minuscula(string a)
11 {
12     a[0]=tolower(a[0]);
13     return a;
14 }
15 string mayuscula(string a)
16 {
17     a[0]=toupper(a[0]);
18     return a;
19 }
20 int main()
21 {
22
23     ifstream apellidos_US("Apellidos_US.txt");
24     ifstream hombres_US("NombresHombres.txt");
25     ifstream mujeres_US("NombresMujeres.txt");
26     string temp;
27     vector<string> apellidos;
28     vector<string> hombres;
29     vector<string> mujeres;
30     //Se escriben en minusculas para luego poder hacer las permutaciones sin problemas
31     //en el
32     //uso de caracteres
33     while(apellidos_US >> temp)
```

```

33     {
34         temp=minuscula(temp);
35         apellidos.push_back(temp);
36     }
37 while(hombres_US >> temp)
38     {
39         temp=minuscula(temp);
40         hombres.push_back(temp);
41     }
42 while(mujeres_US >> temp)
43     {
44         temp=minuscula(temp);
45         mujeres.push_back(temp);
46     }
47
48 string nombre, apellido, sexo;
49 cout << "Ingrese el nombre a buscar: ";
50 cin >> nombre;
51 nombre=minuscula(nombre);
52 cout << "Ingrese el apellido a buscar: ";
53 cin >> apellido;
54 apellido=minuscula(apellido);
55 cout << "Ingrese el sexo (M/F): ";
56 cin >> sexo;
57 sexo=minuscula(sexo);
58 //Se agregan estas lineas en caso que la persona no siga las instrucciones
59 //y escriba el sexo en una palabra
60 if(sexo=="masculino"){sexo="m";}
61 if(sexo=="femenino"){sexo="f";}
62 // Se crea un string concatenando nombre y apellido para poder econtrar las
63 // permutaciones
64 // ademas se agrega un caracter especial "_", el cual va a servir para poder
65 // separar todas
66 // las permutaciones posibles en dos, una parte siendo el nombre, y la otra el
67 // apellido.
68 string nom_ap;
69 nom_ap=nombre+apellido;
70 //Se ordena este string en orden alfabetico, de esa forma se garantiza que se
71 // encuentren
72 // todas las permutaciones
73 sort(nom_ap.begin(), nom_ap.end());
74
75 //vector<pair<string, string>> respuesta;
76 //vector<pair<string, string>>::iterator it_respuesta=respuesta.begin();
77
78 vector<string> respuesta;
79
80 vector<string>::iterator ap=apellidos.begin();
81 vector<string>::iterator ho=hombres.begin();
82 vector<string>::iterator mu=mujeres.begin();
83 //Se ponen las condiciones para saber si buscar en la lista de mujeres o de hombres
84 //Se utiliza la funsion is_permutation para encontrar las permutaciones posibles, y
85 // ademas
86 //se pone la condicion de que tiene que ser del mismo largo que la suma de los
87 // caracteres de
88 // la palabra original.
89 if(sexo=="m")
90 {
91     while(ho!=hombres.end())

```

```

86 {
87     ap=apellidos.begin();
88     while(ap!=apellidos.end())
89     {
90         auto aux=(*ho)+(*ap);
91         if(is_permutation(nom_ap.begin(),nom_ap.end(),aux.begin())&& aux.size()==
           nom_ap.size())
92         {
93             string resultado=mayuscula(*ho)+" "+mayuscula(*ap);
94             respuesta.push_back(resultado);
95         }
96         ap++;
97     }
98     ho++;
99 }
100 }
101 if(sexo=="f")
102 {
103     while(mu!=mujeres.end())
104     {
105         ap=apellidos.begin();
106         while(ap!=apellidos.end())
107         {
108             auto aux=(*mu)+(*ap);
109             if(is_permutation(nom_ap.begin(),nom_ap.end(),aux.begin())&& aux.size()==
           nom_ap.size())
110             {
111                 string resultado=mayuscula(*mu)+" "+mayuscula(*ap);
112                 respuesta.push_back(resultado);
113             }
114             ap++;
115         }
116         mu++;
117     }
118 }
119
120 cout<<"A continuacion, los posibles alias generados a partir del nombre son:"<<endl
   ;
121 for(auto x:respuesta){cout<<x<<endl;}
122
123 return 0;
124 }

```