Universidad de Costa Rica
Escuela de Ingeniería Eléctrica
Programación Bajo Plataformas Abiertas
MSc. Andrés Mora Zúñiga
I Ciclo 2020
IE-0117

Práctica #4: Bucles, arreglos y parametros de linea de comandos

1. Segundo Numero Mayor

Para esta sección se realizó un único archivos. Con el nombre secondgreat.c el cual corresponde a:

```
#include <stdio.h>
#include<stdlib.h>

int main (int argc, char const *argv[])

{
    int arr [argc];
    int numerospositivos=0;
    int numerosnegativos=0;
    for( int i = 1;i < argc; i++)
    {

      arr[i]= atoi(argv[i]);
      if (arr[i]>=0)
      {
        numerospositivos++;
      }

      if (arr[i]<0)
      {
        numerosnegativos++;
      }


    }
    if (numerospositivos>1 )
    {
      int a=1;
      int b=1;
      int numeromayor=0;
      int segundomayor =0;
      while (b<argc){
        int numero1 = arr[b];
        if (numero1>numeromayor)
        {
          numeromayor= numero1;
        };
```

```
38          b++;
39
40        }
41
42        while (a<argc)
43        {
44         int numero = arr[a];
45          if (numero>segundomayor && numero<numeromayor){
46          segundomayor=numero;
47          };
48        a++;
49
50        }
51
52        int c=1;
53        int datocero=0;
54        while (c<argc){
55          int num2= arr[c];
56          c++;
57          if (num2==0)
58          {
59            datocero=1;
60          }
61
62        }
63
64        if (segundomayor ==0 && datocero==0)
65        {
66          segundomayor= numeromayor;
67        }
68
69
70
71        if   (argc<3)
72        {
73          printf("error\n");
74        }else
75        {
76
77          printf("Segundo numero mayor: %d\n",segundomayor);
78        }
79
80      }
81      if (numerospositivos==1 && numerosnegativos>0)
82      {
83        int a=1;
84        int b=1;
85        int numeromayor=0;
86        int segundomayor =0;
87        while (b<argc){
88          int numero1 = arr[b];
89          if (numero1>numeromayor)
90          {
91            numeromayor= numero1;
92          };
```

```
93          b++;
94        }
95
96
97        while (a<argc)
98        {
99         int numero = arr[a];
100        if (numero<0){
101          if (segundomayor==0)
102          {
103            segundomayor=numero;
104          }
105          if (numero>segundomayor)
106          {
107            segundomayor==numero;
108          }
109
110
111        };
112         a++;
113       }
114
115       printf("Segundo numero mayor: %d\n",segundomayor);
116
117
118
119
120     }
121     if (numerospositivos==0 && numerosnegativos>1)
122     {
123       int a=1;
124       int b=1;
125       int numeromayor=0;
126       int segundomayor =0;
127       while (b<argc){
128         int numero1 = arr[b];
129         if (numeromayor== 0)
130         {
131           numeromayor= numero1;
132         };
133         if (numero1>numeromayor)
134         {
135           numeromayor= numero1;
136         };
137        b++;
138       }
139       while (a<argc)
140       {
141        int numero = arr[a];
142        if (numero<numeromayor && segundomayor==0){
143          segundomayor= numero;
144        };
145        if (numero<numeromayor && numero >segundomayor)
146        {
147          segundomayor= numero;
```

```
148          };
149           a++;
150         }
151         if(segundomayor==0){
152           segundomayor= numeromayor;
153         }
154
155         printf("Segundo numero mayor: %d\n",segundomayor);
156
157
158
159
160     }
161
162     if (numerospositivos==0 && numerosnegativos==1){
163       printf("error\n");
164     }
165     if (numerospositivos==1 && numerosnegativos==0){
166       printf("error\n");
167     }
168     if (numerospositivos==0 && numerosnegativos==0){
169       printf("error\n");
170     }
171
172     return 0;
173 }
```

El código empieza incluyendo la biblioteca stdio.h como es frecuente. Además se importa stdib.h para importar los datos de la terminal.Primeramente se crean dos variables enteras que corresponden a numerosnegativos y numerospositivos.Estas dos variables son importantes debido que se analiza por separado cuando solo se ingresan numeros positivos, negativos o cuando solo se ingresa un numero positivo y el resto de negativos.El for que se aprecia en la linea 10 es el encargado de crear un arreglo conformado por los datos que se introdujeron desde la terminal, además que llena las variables numerospositivos y numerosnegativos por ende se logra saber cuantos hay de cada uno. En el primer if que se ve en la linea 20, analiza la situación cuando se introdujeron 2 o más números positivos(el cero cuenta como positivo en este caso). Cabe mencionar que todo lo que se va a mencionar ahorita corresponde adentro del if hasta que se diga lo contrario.Posteriormente se crean las variables a,b,numeromayor y segundomayor. El primer while que aparece en la linea 32 es el encargado de encontrar el numero mayor y guardarlo en la variable **numeromayor**. Dentro de este while se recorre el arreglo creado anteriormente. Consiste básicamente en que la variable **numero1** va a adquirir el valor de todos los datos del arreglo, además que mediante el if de la linea 34 dentro del while se tiene la condición de que si numero1 es mayor a numeromayor guarde el valor del numero1 en **numeromayor**, cabe mencionar que primeramente el valor de numeromayor es 0 hasta que se reemplace por algún otro valor del arreglo.Por ende así se logra obtener el valor de la variable numeromayor.Ese while tiene como condición de parada cuando b sea igual a argc y cada ciclo se le suma 1 mediante el comando **b++** por ende no aparecen problemas cíclicos. El siguiente while de la linea 42 corresponde al que va a encontrar el valor a la variable segundomayor. Como en el anterior se recorre el arreglo y se coloca un if dentro del while en la linea 45. Este if posee la condición de que si la variable numero es mayor a

la variable **segundomayor**(inicialmente en 0) y menor a la variable **numeromayor**(de la parte anterior ya se tiene el valor) entonces la variable segundomayor adquiere el valor de la variable numero.

Posteriormente se topa con el tercer while de la linea 54 es el encargado de solucionar la siguiente situación. En el caso de ingresar solamente 2 numero iguales el programa tendría que la variable segundomayor es cero. En el caso de ingresar un numero y el numero 0 esto también daría como resultado sucedería entonces se hace el if dentro del while de la linea 57 que nos diga si hay un dato cero o simplemente se ingresaron 2 números positivos con el mismo valor. Por ende en caso de que datocero siga siendo 0 significa que no se introdujo un cero y así el programa sabe que se introdujeron 2 únicos datos iguales. Por ultimo se imprime un error en caso de que solo se inserte un numero sino se imprime el segundo numero mayor. Y acá terminaría el if de la linea 26 que analizaba solo numeros positivos.

En el if de la linea 81 se evalua el caso donde solo se introduzca un numero positivo y el resto sean negativos. Por consecuente se sabe que el positivo va a ser el numero mayor por ende se realiza de la misma manera que la sección anterior. A diferencia de la sección pasada para obtener el segundomayor realiza primero el if de la linea 100 para garantizar que solo se trabajen con numeros negativos.Una vez garantizado que estaos trabajando con números negativos se dice que si el segundomayor es 0, por lo tanto se cambia al valor de numero como se ve en el if de la linea 101. En el if de la linea 105 ya se trata de ir comparando a ver cual es el mayor entre todos los negativos. Finalmente se imprime el segundo numero mayor.

Ahora en el if de la linea 121 se analiza cuando solo hay numeros negativos. El while de la linea 130 se trabaja igual que los anteriores con la diferencia que si se tiene que primeramente la variable numeromayor es 0, se cambia por el valor de la variable numero y el resto procede de la misma forma.

Finalmente mediante los if de la linea 162 se pone que cuando se reciba solo un parámetro ya sea 0, positivo o negativo se imprima error. Las pruebas se pueden ver en la figura 1.

```
alberto@debian:~/laboratorio4/segundonumeromayor$ gcc -o salida.x secondgreat.c
alberto@debian:~/laboratorio4/segundonumeromayor$ ./salida.x 31 14 9 7 20
Segundo numero mayor:20
alberto@debian:~/laboratorio4/segundonumeromayor$ 100
bash: 100: orden no encontrada
alberto@debian:~/laboratorio4/segundonumeromayor$ ./salida.x 100
error
alberto@debian:~/laboratorio4/segundonumeromayor$ ./salida.x 99 99
Segundo numero mayor:99
alberto@debian:~/laboratorio4/segundonumeromayor$ 11 4 9 5 8 9 11
bash: 11: orden no encontrada
alberto@debian:~/laboratorio4/segundonumeromayor$ ./salida.x 11 4 9 5 9 11
Segundo numero mayor:9
alberto@debian:~/laboratorio4/segundonumeromayor$ ./salida.x 0 -1 -100 100
Segundo numero mayor:0
alberto@debian:~/laboratorio4/segundonumeromayor$ ./salida.x  -1 -100 100
Segundo numero mayor:-1
alberto@debian:~/laboratorio4/segundonumeromayor$ ./salida.x  -1 -100
Segundo numero mayor:-100
alberto@debian:~/laboratorio4/segundonumeromayor$
```

Figura 1: Segundo numero mayor

2. Imprimir elementos únicos

Para esta sección se realizó un único archivos. Con el nombre uniqueelement.c el cual corresponde a:

```
#include <stdio.h>
#include<stdlib.h>

int main (int argc, char const *argv[])

{
    int arr [argc];


    for( int i = 1;i < argc; i++)
    {
      arr[i]= atoi(argv[i]);

    }

    if(argc >21||argc==1){
        printf("error\n");
    }else
    {
    int a=1;
    int aparecio=0;

    while (a<argc)
    {
        int numero = arr[a];

        for (int b = 1; b < argc; b++)
        {
            int numero2 = arr[b];

            if (numero == numero2)
            {
                aparecio = aparecio + 1;

            }
            if(aparecio >1){

                int cantidad = sizeof(arr)/4;
                // se divide entre 4 porque el entero ocupa 4 bites
                int d =b;

                while (d< (cantidad-1))
                {
                    arr[d]=arr[d+1];
                    d++;
                    //funcion para eliminar un numero repetido
                }


            }
```

```
51          }
52          aparecio=0;
53          a++;
54      }
55
56      int cantidad2 =(sizeof(arr)/4);
57      printf("%d\n",arr[1]);
58      for (int i = 2; i < cantidad2; i++)
59      {
60          if(arr[i]!=arr[i+1]){
61          printf("%d\n",arr[i]);
62          }
63      }
64
65
66      }
67
68
69
70      return 0;
71  }
```

El código empieza incluyendo la biblioteca stdio.h como es frecuente. Además se importa stdib.h para importar los datos de la terminal. Primeramente en el código main se crea un array que contiene todos lo numeros ingresados en la linea de comandos como se ve en el for de la linea 10. Posteriormente el if de la linea 16 corresponde a que si se introducen 0 o más de 20 números el programa indique error. Todo lo demás corresponde dentro del else. En la linea 20 y 21 se declaran los enteros a y apareció. El while de la función 23 incluyendo todo lo que tiene en su interior, corresponde a la idea de agarrar numero por numero del array e irlo comparando con cada numero de este mismo array, por ende si sale más de una vez, la otra vez que aparece este opta por eliminarlo y así queda el array sin números repetidos. Primeramente en la linea 25 se define la variable entera llamada numero y en el for de la linea 27 se define otra variable entera llamada numero2. Así es que se va comparando un numero(el de variable numero) con todos los números del array(con variable numero2). Así que como se ve en el if de la linea 31, si numero y numero2 son iguales se le suma a la cantidad de veces que aparecio, si aparece más de 1 vez esto significa que es un numero repetido y entra en el if de la linea 36. Para saber el tamaño del arreglo se crea el entero cantidad y se usa el sizeof que nos da la cantidad en bites. Como un entero ocupa 4 bites, se divide entre bites para saber la cantidad de enteros en el array.La variable d de la linea 40 corresponde a la posición donde se repite el arreglo. Para eliminar este elemento repetido en el arreglo se procede al while de la linea 42 que va a tomar todos los elementos posteriores al elemento repetido y los va a colocar una posición detrás, de esta manera se puede decir que se elimina un elemento. En la linea 52 se resetea el valor de apareció para volver a empezar el ciclo con el siguiente numero. Finalmente en la linea 56 se calcula el tamaño del array final de la misma forma que se calculó antes.El for de la linea 58 se introduce para imprimir el arreglo final, cabe mencionar que el último dígito se imprimia varias veces entonces para corregir este error se introdujo el if de la linea 60.Las pruebas se pueden ver en la figura 2.

```
alberto@debian:~/laboratorio4/elementounico$ ./salida.x 31 14 9 7 20
31
14
9
7
20
alberto@debian:~/laboratorio4/elementounico$ ./salida.x 100
100
alberto@debian:~/laboratorio4/elementounico$ ./salida.x 99 70 99 99 70 11
99
70
11
alberto@debian:~/laboratorio4/elementounico$ ./salida.x 11 4 9 5 8 11 9 11 4 9 5 8 11 9
 11 4 3 66 7 99 2 3 4
error
alberto@debian:~/laboratorio4/elementounico$
```

Figura 2: Imprimir elementos únicos

3. Busqueda de un elemento

Para esta sección se realizó un único archivos. Con el nombre search.c el cual corresponde a:

```c
#include <stdio.h>
#include<stdlib.h>

void search(int arr[],int tamanio, int buscar){
    int aparecio=0;
    for( int i = 0;i < tamanio; i++)
    {
        if (buscar==arr[i])
        {
            printf("%d ",i);
            aparecio++;
        }

    }

}

int main (int argc, char const *argv[])

{
    int tamanio= 6; //se debe de cambiar si se cambia el tamanio
    // del arreglo
    int arr[]={31,14,9,7,20,14};//uno lo introduce
    int buscar=14;// numero que se quiere buscar

    search(arr,tamanio,buscar);




    return 0;
}
```

Primeramente en el main se crea el arreglo, el tamaño de arreglo y el numero que se quiere buscar. Luego estos parámetros los colocamos en la función search. La función search se puede ver en la linea 4 y es de tipo void. Inicia definiendo la variable apareció con valor nulo. En el for de la linea 6 se utiliza para recorrer el arreglo e irlo comparando con el numero que se quiere buscar. Por ende si los dos son iguales mediante el if de la linea 8 se imprime la posición cada vez que entra al if por ende nos da lo que buscamos.Las pruebas se pueden ver en la figura 3. Cabe mencionar que el primer caso es el del que se ve en el codigo, en el seundo caso es buscando el numero 21(el cual no imprime nada ya que no esta en el arreglo)

```
alberto@debian:~/laboratorio4/searchelement$ gcc -o salida.x search.c
alberto@debian:~/laboratorio4/searchelement$ ./salida.x
1 5
alberto@debian:~/laboratorio4/searchelement$ gcc -o salida.x search.c
alberto@debian:~/laboratorio4/searchelement$ ./salida.x
```

Figura 3: Busqueda dde un elemento

4. Criba de Eratóstenes

Para esta sección se realizó un único archivos. Con el nombre Eratostenes.c el cual corresponde a:

```
1  #include <stdio.h>
2  #include<stdlib.h>
3
4  int tacharmultiplos(int arr[],int limitenuevo){
5
6      int d=0;
7
8
9      for ( int i =1;i <= limitenuevo; i++)
10     {
11         int modulo = arr[i] % arr[0];
12
13         if (modulo ==0 )
14         {
15             d=i;
16
17             while(d< limitenuevo)
18             {
19                 arr[d]=arr[d+1];
20
21                  d++;
22
23
24             }
25             i--;
26             limitenuevo--;
27
28
29         }
```

```
30          d=0;
31
32      }
33
34
35
36      return limitenuevo;
37
38  }
39
40  void Quitarprimerelemento(int arr[], int limitenuevo){
41      int d=0;
42
43      while(d< limitenuevo)
44              {
45                  arr[d]=arr[d+1];
46
47                   d++;
48              }
49
50  }
51
52
53
54
55  int main (int argc, char const *argv[])
56
57  {
58      int limite=atoi(argv[1]);
59      int lim= limite-1;
60      int y=-1;
61      int ymil=-1;
62      int arr[lim];
63      int arrmil[999];
64      int hastanumero[lim];
65      int hastamil[999];
66
67      for( int i =2;i <= limite; i++)
68      {
69         y++;
70         arr[y]= i;
71
72      }
73      for( int i =2;i <= 1000; i++)
74      {
75         ymil++;
76         arrmil[ymil]= i;
77
78      }
79
80      int limitenuevo=y;
81      int limitenuevomil=ymil;
82      int primerelemento=(2*2);
83      int primerelementomil=(2*2);
84      int k=0;
```

```
85      int p=0;
86
87
88      while (primerelemento<limite )
89      {
90
91          hastanumero[k]=arr[0];
92          limitenuevo = tacharmultiplos(arr,limitenuevo);
93          Quitarprimerelemento(arr, limitenuevo);
94          primerelemento=(arr[0]*arr[0]);
95          k++;
96
97      }
98      while (primerelementomil<1000 )
99      {
100
101
102         hastamil[p]=arrmil[0];
103         limitenuevomil = tacharmultiplos(arrmil,limitenuevomil);
104         Quitarprimerelemento(arrmil, limitenuevomil);
105         primerelementomil=(arrmil[0]*arrmil[0]);
106         p++;
107
108     }
109
110
111     for( int i =0;i < limitenuevo; i++)
112     {
113         if (arr[i]==arr[i-1])
114        {
115        }else
116        {
117            //printf("%d\n",arr[i]);
118            hastanumero[k]=arr[i];
119            k++;
120        }
121
122     }
123
124     for( int i =0;i < limitenuevomil; i++)
125     {
126         if (arrmil[i]==arrmil[i-1])
127        {
128        }else
129        {
130
131            hastamil[p]=arrmil[i];
132            p++;
133        }
134
135     }
136     for( int i =0;i < p; i++)
137     {
138         if (hastamil[i]==limite &&hastamil[i]==hastanumero[i])
139            {
```

```
140                 printf("%d \n",hastamil[i]);
141             }
142         if(hastamil[i]!=hastanumero[i]){
143             printf("%d \n",hastamil[i]);
144         }
145 
146     }
147 
148 
149     return 0;
150 }
```

El código empieza incluyendo la biblioteca stdio.h como es frecuente. Además se importastdib.h para importar los datos de la terminal. El main empieza definiendo el limite que corresponde al numero ingresado por el usuario. Se define lim que es 1 menos que el limite debido a que en el array se toma en cuanto el 0. Se crean 4 arrays, 2 para poner la cadena de todos los numeros(arr seria hasta el valor limite y arrmil hasta 1000) y 2 para colocar los números primos hasta su limite. El for de la linea 67 y el de la 73 corresponde al que va a llenar los arrays arr y arrmil con todos los numeros hasta su limite. Ahora se define limitenuevo y limitenuevomil con el tamaño del array y las variables primerelemento y primerelementomil que equivalen a 2 al cuadrado. Y se definen k y p que ocupan la misma funcion ero una es para el limite y otra es para mil.En los while de la 88 y la 89 se aplica el algoritmo de la Criba de Eratóstenes. Con el limite siendo el numero hasta donde se quiere llegar. Dentro del while primeramente se guarda en el array hastanumero(hastanumeromil) el primer elemento del array que corresponde a arr[0]. Posteriormente entra a la funcion tacharmultiplos y esta elimina todos los multiplos de arr[0]. Posteriormente se elimina el primer elemento mediante la función Quitarelemento y por ende ahora nuestro arr[0] corresponderá al segundo elemento(osea al 3 no al 2).Ahora en la linea 94 el elemento primerelemento corresponde a 3 al cuadrado y se compara con el limite como indica en el while y este proceso sigue hasta que no se cumpla la condición.

En la linea 4 esta la función tacharmultiplos. Para realizarla primeramente se define una d en valor nulo. Luego el for de la linea 9 va a ir recorriedo el array y si el modulo da 0 significa que es multiplo del arr[0] por ende con el if de la linea 13 se elimina este vaor. Se logra eliminar ya que dentro del if el d adquiere el valor de la posición y se seleccionan en la casilla anterior a todos los numeros del array posteriores al del modulo 0 y de esta manera se elimina. Se elimina un numero del limitenuevo después de que entre al módulo ya que este va a ser el nuevo tamaño del array y se elimina un i ya que si entra al if debido al while el valor siguiente ocupa la posición del que se eliminó actual. En la linea 40 se ve la función Quitarprimerelemento que consiste en que se elimine el primer digito y esto se realiza moviendo todos los datos posteriores una celda atrás.

Una vez explicadas las funciones se sigue con el resto del programa. Del while de la linea 88 y 98 se tiene que los únicos números que quedan en el arr y arrmil son los numeros primos. Por ende mediante el for de la linea 111 y 124 se encargan de recorrer este arr y arrmil y además de que se agregan a los arrays hastanumero y hastamil. Cabe mencionar que para prevenir cualquier error se pone el if de la linea 113 que se encarga que ningun numero que entre al array hastanumero o hasta mil sea repetido. Por último ya que se cuenta con los 2 array hastanumero y hastamil, uno cuenta con numeros primos desde 2 hasta los menores al que el usuario ingresó y el otro con los numeros primos de 2 hasta mil. Por ende se recorren

estos 2 arrays y solo se imprimen los que tengan condiciones de que salgan en hastamil y no salgan en hastanumero. Así se obtienen los números mayores e iguales primos hasta el mil desde el numero que el usuario ingrese. Cabe destacar que se realizaron varias pruebas y en algunas incluía el numero si era primo sin embargo en otras no lo incluía. Para corregir este error se utilizó el if de la linea 138 que establece que si el numero que aparece en hastamil es igual a limite(numero que el usuario ingresó) y se cumple la condición de que el numero hastamil y el numero hastanumero son iguales(condición inversa al pasado), esta condición inversa evita que en algún caso en especifico el numero se imprima más de una vez.Las pruebas se pueden ver en las figuras 4,4 y 4 se ven ejemplos.

Figura 4: Números primos con Criba de Eratóstenes

```
alberto@debian:~/laboratorio4/algoritmoCriba$ gcc -o salida.x Eratostenes.c
alberto@debian:~/laboratorio4/algoritmoCriba$ ./salida.x 888
907
911
919
929
937
941
947
953
967
971
977
983
991
997
```

Figura 5: Números primos con Criba de Eratóstenes

```
alberto@debian:~/laboratorio4/algoritmoCriba$ ./salida.x 0
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
```

Figura 6: Números primos con Criba de Eratóstenes