Práctica #2
Recursividad

25/05/2020

Juan Blanco Martín

Miguel Marazuela Bella

Metodología de la Programación

Laboratorio 3L – David Atauri Grupo 49

Objetivos

Debemos implementar una aplicación que muestre en pantalla un menú el cual se ejecutara de forma repetitiva con cinco opciones las cuales serán subprogramas recursivos y cuyas funciones serán:

1-Solicitar al usuario que introduzca tres cadenas de caracteres y diez números enteros.

2-Determinar cuántas veces aparece un carácter en una de las cadenas.

3-Sustituir los caracteres a por O y p por Z en una de las cadenas.

4-Eliminar la primera aparición de los caracteres que aparecen al menos dos veces seguidos en una de las cadenas.

5-De los dos arrays de enteros se selecciona uno y muestra en pantalla la posición que ocupa el elemento que coincida con el número total de elementos impares que existen en el array.

Algoritmos recursivos

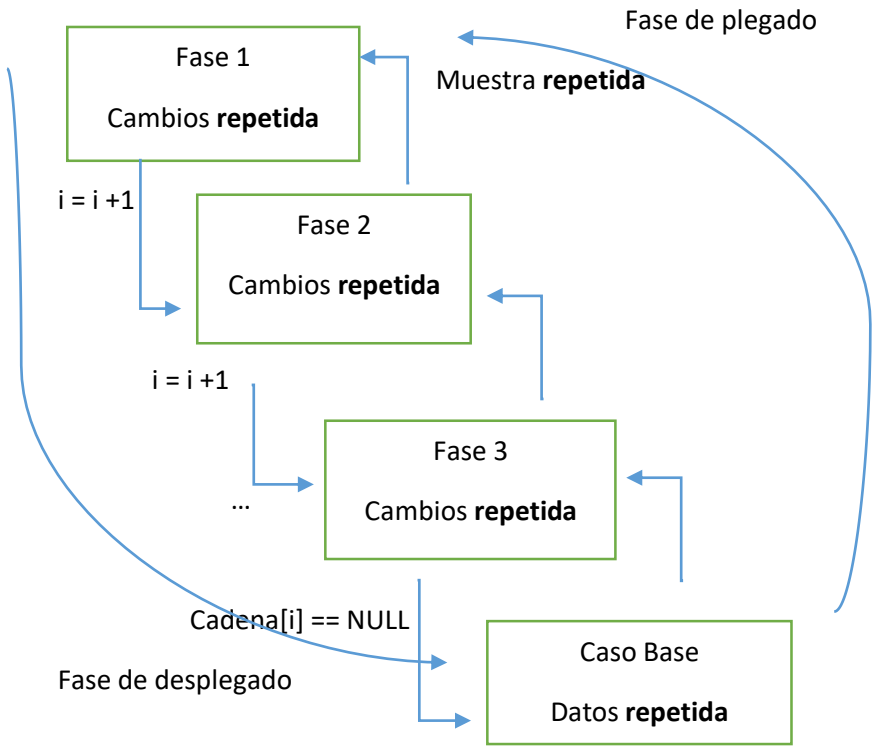
ContarApariciones

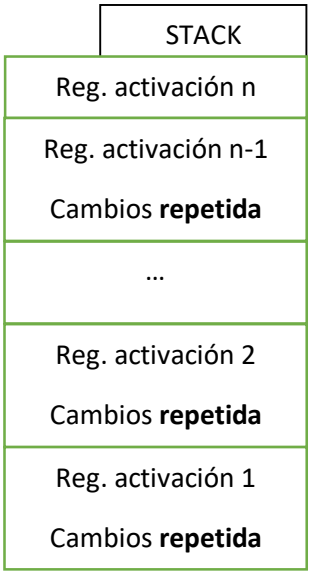
Identificación de casos:

- Caso general (recursivo):
  - El carácter de la cadena de caracteres no es un valor vacío (NULL).
  - Hay caracteres repetidos en la cadena de caracteres.
- Caso base:
  - El carácter de la cadena de caracteres es un valor vacío (NULL).

Tipo de recursividad:

Recursividad **Lineal final.** El resultado se obtiene una vez alcanzado el caso base y las llamadas recursivas que se utilizan son individuales.





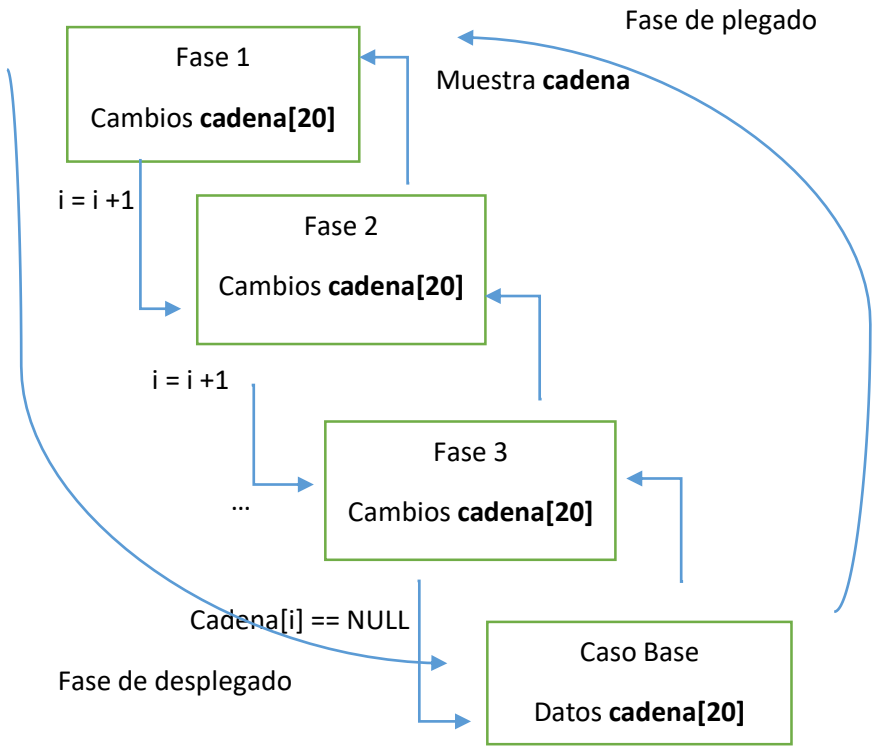
CambiarCaracteres

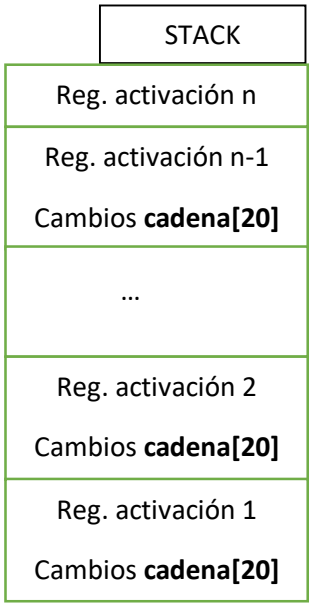
Identificación de casos:

- Caso general (recursivo):
  - El carácter de la cadena de caracteres no es un valor vacío (NULL).
  - Hay caracteres que son “a” o “p”.
- Caso base:
  - El carácter de la cadena de caracteres es un valor vacío (NULL).

Tipo de recursividad:

Recursividad **Lineal final.** El resultado se obtiene una vez alcanzado el caso base y las llamadas recursivas que se utilizan son individuales.





ReducirCadena

Identificación de casos:

- Caso general (recursivo):
    - El carácter de la cadena de caracteres no es un valor vacío (NULL).
- Caso base:
    - El carácter de la cadena de caracteres es un valor vacío (NULL).

Tipo de recursividad:

Recursividad **Lineal final.** El resultado se obtiene una vez alcanzado el caso base y las llamadas recursivas que se utilizan son individuales.

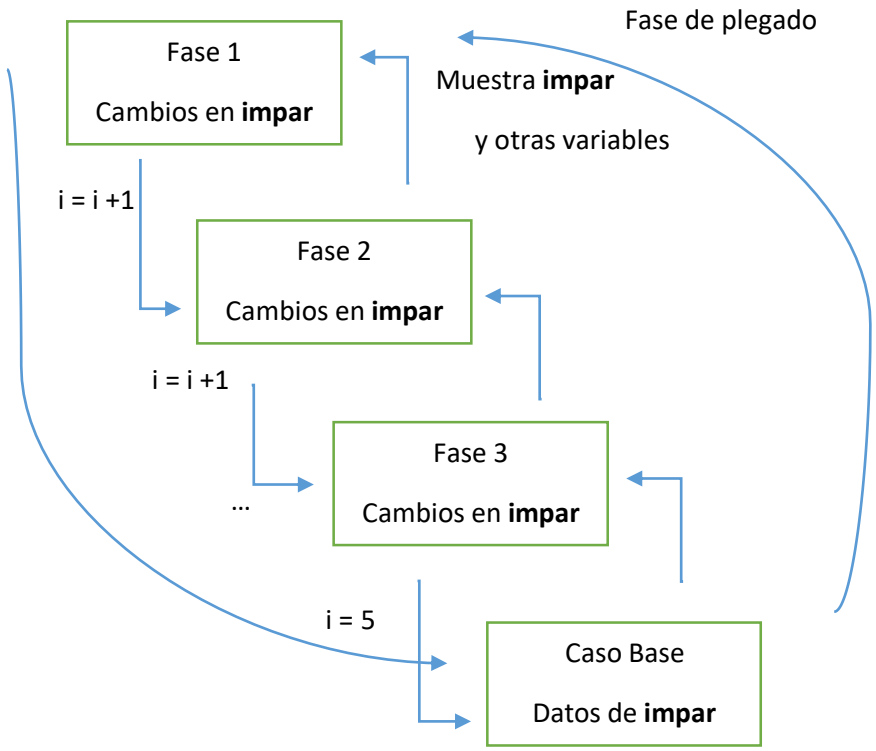
MostrarPosicionTotalImpares

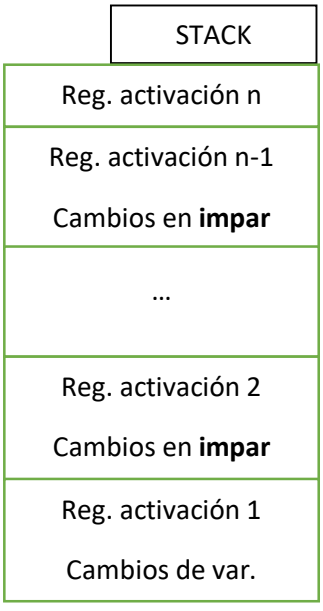
Identificación de casos:

- Caso general (recursivo):
  - El número en la posición del array sea par.
  - El número en la posición del array sea impar.
- Caso base:
  - El iterador con el que recorro el array sea igual a 4.

Tipo de recursividad:

Recursividad **Lineal final.** El resultado se obtiene una vez alcanzado el caso base y las llamadas recursivas que se utilizan son individuales.





Pruebas de ejecución

```
MENU RECURSIVIDAD
--------------

1- Peticion de datos
2- Contar apariciones
3- Reemplazar caracteres
4- Eliminar un caracter repetido
5- Mostrar posicion total de impares

0-Salir

Introduzca una opcion:
```

```
Numero entero: 2

Numero entero: 2

ERROR: Numeros repetidos...
Vuelva a introducir los datos

Pulse <Intro> para continuar...
```

```
Cadena de caracteres 1: holla proobando

Cadena de caracteres 2: ccadena 222

Cadena de caracteres 3: proobando

Numero entero: 1

Numero entero: 23

Numero entero: 56

Numero entero: 4

Numero entero: 8

Numero entero: 98

Numero entero: 8877

Numero entero: 42

Numero entero: 420

Numero entero: 3
```

```
Que cadena quieres analizar?:
```

```
Que cadena quieres analizar?: 1

Que caracter quieres buscar?: l
```

```
Que cadena quieres analizar?: 1

Que caracter quieres buscar?: l

El caracter l aparece 2 veces en la cadena holla proobando

Pulse <Intro> para continuar...
```

```
En que cadena quieres quitar caracteres repetidos?: 1

Cadena de caracteres ORIGINAL: hollO ZroobOndo
Cadena de caracteres REVISADA: holO ZrobOndo

Pulse <Intro> para continuar...
```

```
Que cadena quieres analizar?: 1

Cadena de caracteres ORIGINAL: holla proobando
Cadena de caracteres MODIFICADA: hollO ZroobOndo

Pulse <Intro> para continuar...
```

```
Que array de numeros quieres usar?:
```

```
Que array de numeros quieres usar?: 1

Array considerado: (1, 23, 56, 4, 8)
Total de elementos impares: 2
Elemento no encontrado

Pulse <Intro> para continuar...
```

```
Que array de numeros quieres usar?: 1

Array considerado: (0, 1, 2, 3, 4)
Total de elementos impares: 2
Elemento 2 encontrado en la posicion 2 del array

Pulse <Intro> para continuar...
```

Implementación

```
/*****************************************************
                Alumno 1: Juan Blanco Martín
                Alumno 2: Migel Marazuela Bella
                Grupo: 49 - Turno:  3L
                Fecha: 25/05/2020
*****************************************************/
/* Compilar y ejecutar desde terminal ******************
/* Windows*********************************************
    gcc src/main.c -o rec && rec.exe

* Linux***********************************************
    gcc src/main.c -o rec
    ./rec

*****************************************************/

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

typedef struct
{
    char char1[20];
    char char2[20];
    char char3[20];
    int num1[5];
    int num2[5];
} datos;


/*****************************************PROTOTIPOS*****************************
**********************/

//UTILIDADES
void Pausa();

//SUBPROGRAMAS PRINCPALES
void PeticionDatos(datos tablaDatos[1]);
void CambiarCaracteres(char cadena[20], int i);
int ContarApariciones(char cadena[20], char letra[2], int i, int repetida);
void ReducirCadena(char cadena[20], int i);
void MostrarPosicionTotalImpares(int num[5], int i, int impar);


/*****************************************************************************
**********************/

int main()
```

```
{
    datos tablaDatos[1];

    int opcion, num;
    char letra[2];

    do{
        system("cls");
        printf("\nMENU RECURSIVIDAD\n");
        printf("--------------\n\n");
        printf("1- Peticion de datos\n");
        printf("2- Contar apariciones\n");
        printf("3- Reemplazar caracteres\n");
        printf("4- Eliminar un caracter repetido\n");
        printf("5- Mostrar posicion total de impares\n\n");
        printf("0-Salir\n");
        printf("\n");
        printf("Introduzca una opcion: ");
        fflush(stdin);
        scanf("%d",&opcion);

        switch(opcion){

            case 1:
                PeticionDatos(tablaDatos);
                break;

            case 2:
                system("cls");
                printf("Que cadena quieres analizar?: ");
                fflush(stdin);
                scanf("%d", &num);
                printf("\nQue caracter quieres buscar?: ");
                fflush(stdin);
                scanf("%c", &letra[0]);

                switch(num) {
                    case 1:
                        printf("\nEl caracter %s aparece %d veces en la cadena
%s\n",letra, ContarApariciones(tablaDatos[0].char1, letra, 0, 0),
tablaDatos[0].char1);
                        Pausa();
                        break;

                    case 2:
                        printf("\nEl caracter %s aparece %d veces en la cadena
%s\n",letra, ContarApariciones(tablaDatos[0].char2, letra, 0, 0),
tablaDatos[0].char1);
                        Pausa();
                        break;

                    case 3:
                        printf("\nEl caracter %s aparece %d veces en la cadena
%s\n",letra, ContarApariciones(tablaDatos[0].char3, letra, 0, 0),
```

```
tablaDatos[0].char1);
                        Pausa();
                        break;

                    default:
                        break;
                }
                break;

            case 3:
                system("cls");
                printf("Que cadena quieres analizar?: ");
                fflush(stdin);
                scanf("%d", &num);

                switch(num) {
                    case 1:
                        printf("\nCadena de caracteres ORIGINAL: %s",
tablaDatos[0].char1);
                        CambiarCaracteres(tablaDatos[0].char1, 0);
                        printf("\nCadena de caracteres MODIFICADA: %s \n",
tablaDatos[0].char1);
                        Pausa();
                        break;

                    case 2:
                        printf("\nCadena de caracteres ORIGINAL: %s",
tablaDatos[0].char2);
                        CambiarCaracteres(tablaDatos[0].char2, 0);
                        printf("\nCadena de caracteres MODIFICADA: %s \n",
tablaDatos[0].char2);
                        Pausa();
                        break;

                    case 3:
                        printf("\nCadena de caracteres ORIGINAL: %s",
tablaDatos[0].char3);
                        CambiarCaracteres(tablaDatos[0].char3, 0);
                        printf("\nCadena de caracteres MODIFICADA: %s \n",
tablaDatos[0].char3);
                        Pausa();
                        break;

                    default:
                        break;
                }
                break;

            case 4:
                system("cls");
                printf("En que cadena quieres quitar caracteres repetidos?: ");
                fflush(stdin);
                scanf("%d", &num);
```

```
                switch(num) {
                    case 1:
                        printf("\nCadena de caracteres ORIGINAL: %s",
tablaDatos[0].char1);
                        ReducirCadena(tablaDatos[0].char1, 0);
                        printf("\nCadena de caracteres REVISADA: %s \n",
tablaDatos[0].char1);
                        Pausa();
                        break;

                    case 2:
                        printf("\nCadena de caracteres ORIGINAL: %s",
tablaDatos[0].char2);
                        ReducirCadena(tablaDatos[0].char2, 1);
                        printf("\nCadena de caracteres REVISADA: %s \n",
tablaDatos[0].char2);
                        Pausa();
                        break;

                    case 3:
                        printf("\nCadena de caracteres ORIGINAL: %s",
tablaDatos[0].char3);
                        ReducirCadena(tablaDatos[0].char3, 1);
                        printf("\nCadena de caracteres REVISADA: %s \n",
tablaDatos[0].char3);
                        Pausa();
                        break;

                    default:
                        break;
                }

                break;

            case 5:
                system("cls");
                printf("Que array de numeros quieres usar?: ");
                fflush(stdin);
                scanf("%d", &num);
                switch (num) {

                    case 1:
                        printf("\nArray considerado: (%d, %d, %d, %d,
%d)",tablaDatos[0].num1[0], tablaDatos[0].num1[1], tablaDatos[0].num1[2],
tablaDatos[0].num1[3], tablaDatos[0].num1[4]);
                        MostrarPosicionTotalImpares(tablaDatos[0].num1, 0, 0);
                        Pausa();
                        break;
                    case 2:
                        printf("\nArray considerado: (%d, %d, %d, %d,
%d)",tablaDatos[0].num2[0], tablaDatos[0].num2[1], tablaDatos[0].num2[2],
tablaDatos[0].num2[3], tablaDatos[0].num2[4]);
                       MostrarPosicionTotalImpares(tablaDatos[0].num2, 0, 0);
                        Pausa();
```

```
                        break;

                    default:
                        break;
                }
                break;

            case 0:
                break;
        }

    } while(opcion != 0);

    return (0);
}


/*****************************************************************************
*****/
/* Subprograma: Pausa */
/* Tipo: Procedimiento (función tipo void) */
/* Prerrequisitos: No tiene. */
/* Objetivo: Espera a que el usuario pulse <Intro> para continuar el
procedimiento. */
/*****************************************************************************
*****/
void Pausa() {

    printf("\nPulse <Intro> para continuar...");
    fflush(stdin);
    getchar();
}

/*****************************************************************************
**/
/* Subprograma: PeticionDatos */
/* Tipo: Procedimiento (función tipo void) */
/* Prerrequisitos: No tiene. */
/* Objetivo: Pide al usuario 10 enteros y 3 cadenas de caracteres para */
/* ser usados en otros funciones de la aplicación */
/*****************************************************************************
**/
void PeticionDatos(datos tablaDatos[1]) {

    int contNum, i, numAux;

    printf("\nCadena de caracteres 1: ");
    fflush(stdin);
    // %[^\n]s hace que hasta que no sale de linea sigue recogiendo el valor
escrito
    // esto me permite poder escribir cadenas con espacios
    scanf("%[^\n]s",&tablaDatos[0].char1);
    printf("\nCadena de caracteres 2: ");
    fflush(stdin);
```

```
    scanf("%[^\n]s",&tablaDatos[0].char2);
    printf("\nCadena de caracteres 3: ");
    fflush(stdin);
    scanf("%[^\n]s",&tablaDatos[0].char3);

    for(contNum = 0; contNum < 5; contNum++) {
        printf("\nNumero entero: ");
        fflush(stdin);
        scanf("%d",&numAux);

        for(i = 0; i < contNum; i++){
            if(tablaDatos[0].num1[i] == numAux){
                printf("\nERROR: Numeros repetidos...\nVuelva a introducir los
datos\n");
                Pausa();
                return 0;
            }
        }
        tablaDatos[0].num1[contNum] = numAux;
    }

    for(contNum = 0; contNum < 5; contNum++) {
        printf("\nNumero entero: ");
        fflush(stdin);
        scanf("%d",&numAux);

        for(i = 0; i < contNum; i++){
            if(tablaDatos[0].num2[i] == numAux){
                printf("\nERROR: Numeros repetidos...\nVuelva a introducir los
datos\n");
                Pausa();
                return 0;
            }
        }
         tablaDatos[0].num2[contNum] = numAux;
    }

    for(contNum = 0; contNum < 5; contNum++) { //Compara los 2 arrays entre ellos
        for(i = 0; i < 5; i++) {
            if(tablaDatos[0].num1[contNum] ==  tablaDatos[0].num2[i]){
                printf("\nERROR: Numeros repetidos...\nVuelva a introducir los
datos\n");
                Pausa();
                return 0;
            }
        }
    }
}

/*************************************************************************
**/
/* Subprograma: ContarApariciones */
/* Tipo: Entero (función tipo int) */
/* Prerrequisitos: Ninguno   */
```

```
/* Objetivo: Contar la cantidad de veces que se repite el caracter pasado */
/*  en la cadena pasada */
/*****************************************************************************
**/
int ContarApariciones(char cadena[20], char letra[2], int i, int repetida) {

    while(cadena[i] != NULL) {

        if(cadena[i] == letra[0]) {
            repetida++;
            i++;
            ContarApariciones(cadena, letra, i, repetida);

        } else {
            i++;
            ContarApariciones(cadena, letra, i, repetida);
        }
    }

    return repetida;
}

/*****************************************************************************
**/
/* Subprograma: CambiarCaracteres */
/* Tipo: Procedimiento (función tipo void) */
/* Prerrequisitos: No tiene. */
/* Objetivo: Transforma los caracteres "a" en "O y "p" en "Z" */
/*****************************************************************************
**/
void CambiarCaracteres(char cadena[20], int i) {

    while(cadena[i] != NULL) {

        if(cadena[i] == 'a') {
            cadena[i] = 'O';
            i++;
            CambiarCaracteres(cadena, i);

        } else {
            if(cadena[i] == 'p') {
                cadena[i] = 'Z';
                i++;
                CambiarCaracteres(cadena, i);

            } else {
                i++;
                CambiarCaracteres(cadena, i);
            }
        }
    }
}

/*****************************************************************************
```

```
**/
/* Subprograma: ReducirCadena */
/* Tipo: Procedimiento (función tipo void) */
/* Prerrequisitos: No tiene. */
/* Objetivo: Elimina los caracteres iguales que se encuentran juntos" */
/*****************************************************************************
**/
void ReducirCadena(char cadena[20], int i) {

    int j;

    while(cadena[i] != NULL) {

        if(cadena[i] ==  cadena[i + 1]) {
            // Borro elemento
            for(j = i; j < 20; j++){
                cadena[j] = cadena[j+1];
            }
            i++;
            ReducirCadena(cadena, i);
        }
        else {
            i++;
            ReducirCadena(cadena, i);
        }
    }
}

/*****************************************************************************
**/
/* Subprograma: MostrarTotalImpares */
/* Tipo: Procedimiento (función tipo void) */
/* Prerrequisitos: Que exista el array de enteros que se le pasa. */
/* Objetivo: Recorre el array de enteros y cuenta el numero de numeros impares */
/*****************************************************************************
**/
void MostrarPosicionTotalImpares(int num[5], int i, int impar) {

    int j, find, aux;

    if(i < 5) {

        if (num[i]%2 == 0) {
            i++;
            MostrarPosicionTotalImpares(num, i, impar);
        } else {
            i++;
            impar++;
            MostrarPosicionTotalImpares(num, i, impar);
        }
    } else {
        printf("\nTotal de elementos impares: %d\n", impar);

        for( j = 0; j < 5; j++) {
```

```
            if (num[j] == impar) {
                find = j;
                j = 6; //Para salir del blucle for
            } else find = -1;
        }

        if(find == -1){
            printf("Elemento no encontrado\n");
        } else printf("Elemento %d encontrado en la posicion %d del array\n",
impar, find);

    }
}
```