

Iterativas1: Ejemplo de uso de estructuras iterativas. Diseñe y escriba un algoritmo que le permita calcular la suma:

$$\frac{1}{2} + \frac{2}{2^2} + \frac{3}{2^3} + \frac{4}{2^4} + \dots + \frac{n}{2^n}$$

ALGORITMO suma;

ENTRADA: n, entero positivo;

SALIDA : sum, entero positivo;

V. AUXILIARES: nume, deno, frac, ind, ind2: enteros positivos;

S0.	Inicializar;	
S1.	LEER (n);	S2.1 Inicializar_var_auxiliares.;
S2.	HACER n VECES (ind = 1,n)	Nume ← ind;
	Inicializar_var_auxiliares.;	Deno ← 1;
	Determinar_pot_del_denominador;	S2.2 Determinar_pot_del_denominador;
	armar_fracción;	HACER ind VECES (ind2=1, ind)
	sum ← sum + frac;	Deno ← deno *2;
	FIN_HACER	
S3.	ESCRIBIR(sum);	S2.3 Armar_fracción;
S4.	PARAR.	Frac ← nume/deno;
S0.	Inicializar;	
	Sum ← 0;	

Modifique el algoritmo de modo tal que le permita trabajar sólo con datos válidos, caso contrario, el mismo deberá enviar un mensaje de error y terminar. Escriba todos los supuestos que Ud. considere necesarios.

Iterativas2: Ejemplo de uso de estructuras iterativas.

En un observatorio meteorológico se lleva el registro de lluvias diarias. Se desea saber al cabo de un mes (30 días) cual fue el día de mayor cantidad de lluvia caída y cual fue ese registro. (y la misma pregunta para la menor cantidad de lluvia). Además se quiere conocer la cantidad total de lluvias caídas durante el mes.

ALGORITMO observatorio;

ENTRADA: reg: entero positivo;

SALIDA: día, mayor, tot: enteros positivos;

V. AUXILIARES: sum, maxi: enteros positivos;

```

O0. Inicializar;
O1. HACER 30 VECES (día=1,30)
    LEER(reg);
    Seleccionar_mayor_y_dia;
    Seleccionar_menor_y_dia
    Sum ← sum + reg;
    FIN_HACER;
O2. ESCRIBIR(día, sum);
O3. PARAR.
    
```

Modifique el algoritmo de modo tal que: ante la lectura de un registro incorrecto se emita un mensaje de error pero el procesamiento continúe, esto es, que los registros incorrectos sea ignorados. El código siguiente ignora los registros invalidos.

```

#include <stdio.h>
#define REGISTRO_INVALIDO printf("REGISTRO_INVALIDO\n")
main()
{
    
```

```
short dia, reg, sum=0, maxi=0, dia_max, min=1500, dia_min, reg_valido;

for(dia=1; dia <=30; dia++)
{   printf("Ingrese el registro del dia %2d\n",dia);
    scanf("%d",&reg);
    reg_valido = reg >= 0 && reg < 1500;
    if (reg_valido)
    {   if(reg >= maxi)
        {   maxi = reg;
            dia_max = dia;
        }
        if(reg <= min)
        {   min = reg;
            dia_min = dia;
        }
        sum = sum + reg;
    }
    else
        REGISTRO_INVALIDO;
}
printf("El mayor registro de %d mm corresponde al dia %2d\n",maxi,dia_max);
printf("El menor registro de %d mm corresponde al dia %2d\n",min,dia_min);
printf("Lluvia caida durante el mes %d mm",sum);
return 0;
}
```

Iterativas3: Ejemplo de uso de estructuras apropiadas para el tratamiento de secuencias.

Diseñe un algoritmo para verificar si una frase es un tautograma, es decir, que todas las palabras que componen la frase comienzan con la misma letra. Considere:

- la frase puede estar precedida por blancos
- la frase puede ser nula (secuencia vacía)
- las palabras de la frase pueden estar separadas por mas de un blanco
- una frase de una sola palabra se considerará un tautograma

ALGORITMO tautograma;

ENTRADA: secuencia de caracteres con MF, car : carácter;

SALIDA : mensaje;

V: AUXILIARES: Es_tauto, indicador lógico ó variable booleana; pri: carácter;

NIVEL 1

```
T0.  Es_tauto ← verdadero;
T1.  Saltar_blanco_iniciales;
T2.  SI ( car = MF) ENTONCES
      ESCRIBIR(Secuencia con blancos y MF)
      SINO
      pri ← car;
      MIENTRAS (car <> MF y es_tauto)
          saltar_palabra;
          SI (car <> MF) ENTONCES
              saltar_blanco_intermedios;
              controlar_tauto_con_el_1º_car_de_la_proxima_palabra;
      FIN-MIENTRAS;
T3.  SI (Es_tauto ) ENTONCES
      ESCRIBIR ( Es tautograma)
```

SINO

ESCRIBIR (NO es tautograma)

T4. PARAR.

NIVEL 2

```
Saltar_blanco_iniciales;
LEER(car);
MIENTRAS (car = blanco y car <> MF)
    LEER(car);
FIN_MIENTRAS;
```

```
saltar_palabra;
MIENTRAS (car <> blanco y car <> MF)
    LEER(car);
FIN_MIENTRAS;
```

```
saltar_blanco_intermedios;
MIENTRAS (car = blanco y car <> MF)
    LEER(car);
FIN_MIENTRAS;
```

```
controlar_tauto_con_el_1º_car_de_la_proxima_palabra;
SI (car <> MF ) ENTONCES
    SI (car <> pri) ENTONCES
        Es_tauto ← falso;
    SINO
        Leer(car);
```

Iterativas4: Programa figura.

```
123454321
1234 4321
123 321
12 21
1 1
```

```
#include <stdio.h>
#include <conio.h> /* uso de clrscr(); */

main()
{
    short num, bl, i;
    clrscr();
    for(i=1; i <=5; i++)
    {
        for (num=1; num <= 6-i; num++)
            printf("%1d", num);
        for (bl=1; bl <= 2*i-3; bl++)
            printf(" ");
        for (num=6-i; num >= 1; num--)
            if(num != 5)
                printf("%1d", num);
        printf("\n");
    }
    return 0;
}
```

Iterativas5: Ejemplo de uso de estructuras apropiadas para el tratamiento de secuencias.

Una frase es “*un conjunto de palabras que basta para formar sentido*”. Una palabra es “*un conjunto de caracteres unidos que designan una cosa o expresan una idea*”.

Considere una frase como una secuencia de caracteres con marca final, por ejemplo un punto. Diseñe un algoritmo que lea la frase y cuente la cantidad de diptongos “ai” presentes en la misma. La salida de este algoritmo deben ser el valor del contador.

Por ejemplo: Para la frase “Inauguración del Mundial de Voley”, la cantidad de diptongos “au”.

ALGORITMO diptongo;
 ENTRADA: secuencia de caracteres con MF; car: carácter;
 SALIDA: cont, entero >=0;
 Nivel 1

-
-
- D1. LEER(car);
D2. MIENTRAS (car <> MF)
 Saltar_blanco;
 Procesar_palabra;
 FIN_MIENTRAS;
D3. ESCRIBIR(cont);
D4. PARAR.

```
#include <stdio.h>
#include <ctype.h>
#define MF '.' /* defino la constante Marca Final */
#define A 'a' /* defino la constante 'a' */
#define I 'i' /* defino la constante 'i' */
#define U 'u' /* defino la constante 'u' */

main()
{
    char car; int cont = 0;

    car = getchar();
    while (car != MF)
    {
        do
            car = getchar();
        while ((car == ' ') && (car != MF));

        while ((car != ' ') && (car != MF))
        {
            if (car == A)
            {
                car = getchar();
                if ((car == I) || (car == U))
                    cont = cont + 1;
            }
            if (car != MF)
                car = getchar();
        }
    }
    printf("Hay %d diptongos que comienzan con a\n", cont);
    return 0;
}
```

Sugerencia: modifique el algoritmo de modo tal que le permita contar la cantidad de diptongos del tipo: "au", "ei", "eu", "ou", "oi". Codifique.