Universidad de las Fuerzas Armadas ESPE

Departamento: Eléctrica, Electrónica y Telecomunicaciones.

Carrera: Electrónica y Automatización.

Taller académico Nª: 1 Parcial 2

---

1. Información General

- **Asignatura: Fundamentos de la Programación.**
- **Apellidos y nombres de los estudiantes:**

  Marco Joel Chuquisala Guanoluisa, Leandro Zamir Safla Tenorio, Kerlly Viviana Bonilla Chulca.

- **NRC: 20823**
- **Fecha de realización: 03/06/2025**

---

2. **Objetivo del Taller y Desarrollo:** Realzar los 4 ejercicios propuestas

Problema 2.1.3 Vector con termino general dado

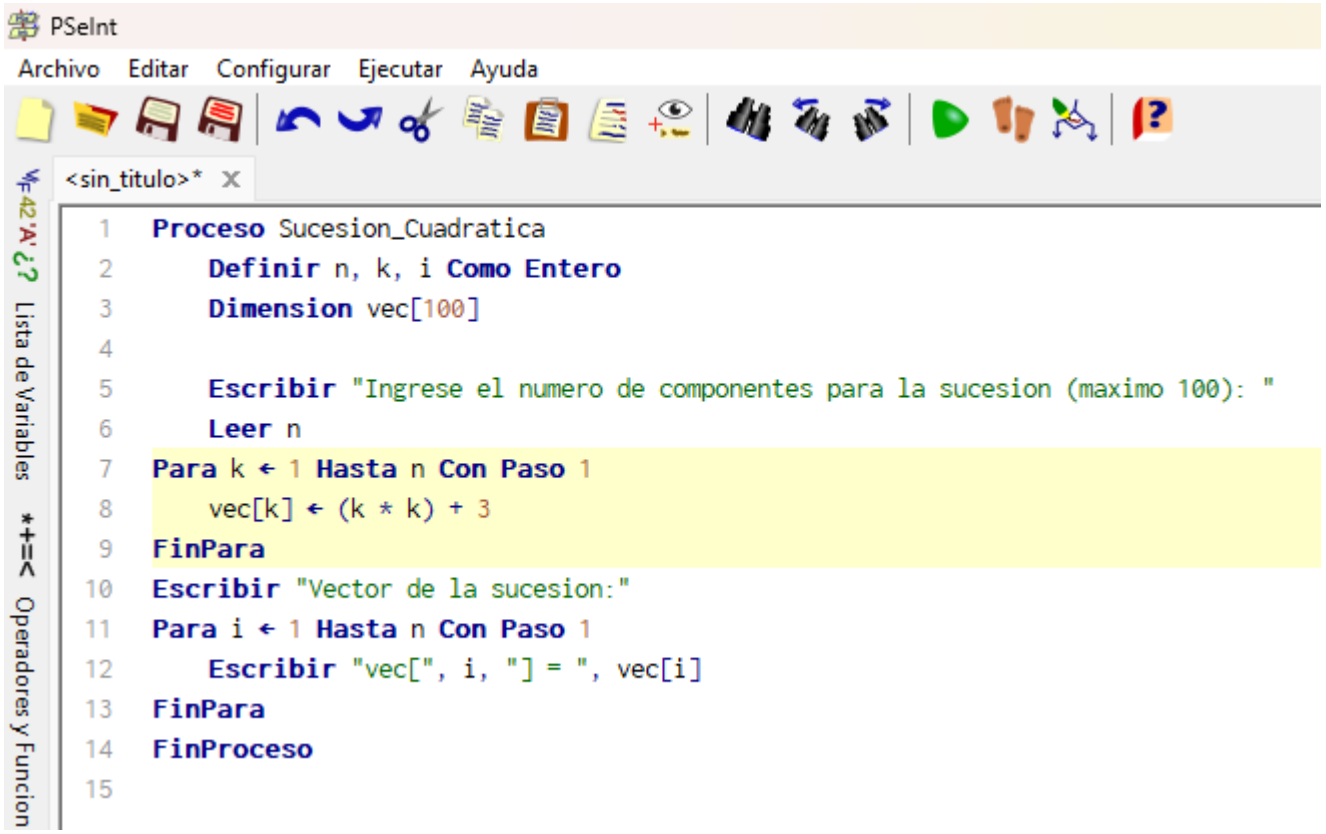
Sea la razón $V_k=k^2+3$

Desarrolle un programa que lea el número n de componentes que se quieren calcular de la sucesión y almacenarlas en un vector vec, tal que vec(i) $=v_i$. Se mostrará el vector en pantalla. Puede asumir que n será siempre menor o igual a 100

Tabla de objetos

Objeto	Nombre	Valor	Tipo
n	cantidad	Variable	Entero
k	componente	Variable	Entero
i	Contador	Variable	entero
vec	Vector	Constante	Entero

Código en Pseint



Código en C

```c
#include <stdio.h>
int main() {
    int n, k, i;
    int vec[100];
    printf("Ingrese el numero de componentes para la sucesion(maximo 100): ");
    scanf("%i", &n);
    if (n<1 || n>100){
        printf ("Numero de componentes invalidos.");
        return 0;
    }
    for (k = 0; k < n; k++) {
        vec[k] = (k * k) + 3;
    }
    printf("Vector de la sucesion:\n");
    for (i = 0; i < n; i++) {
        printf("vec[%i] = %i\n", i, vec[i]);
    }
    return 0;
}
```

Prueba de escritorio

```
Ingrese el numero de componentes para la sucesion(maximo 100): 15
Vector de la sucesion:
vec[0] = 3
vec[1] = 4
vec[2] = 7
vec[3] = 12
vec[4] = 19
vec[5] = 28
vec[6] = 39
vec[7] = 52
vec[8] = 67
vec[9] = 84
vec[10] = 103
vec[11] = 124
vec[12] = 147
vec[13] = 172
vec[14] = 199

--------------------------------
Process exited after 5.326 seconds with return value 0
Presione una tecla para continuar . . .
```

Problema 2.1.4

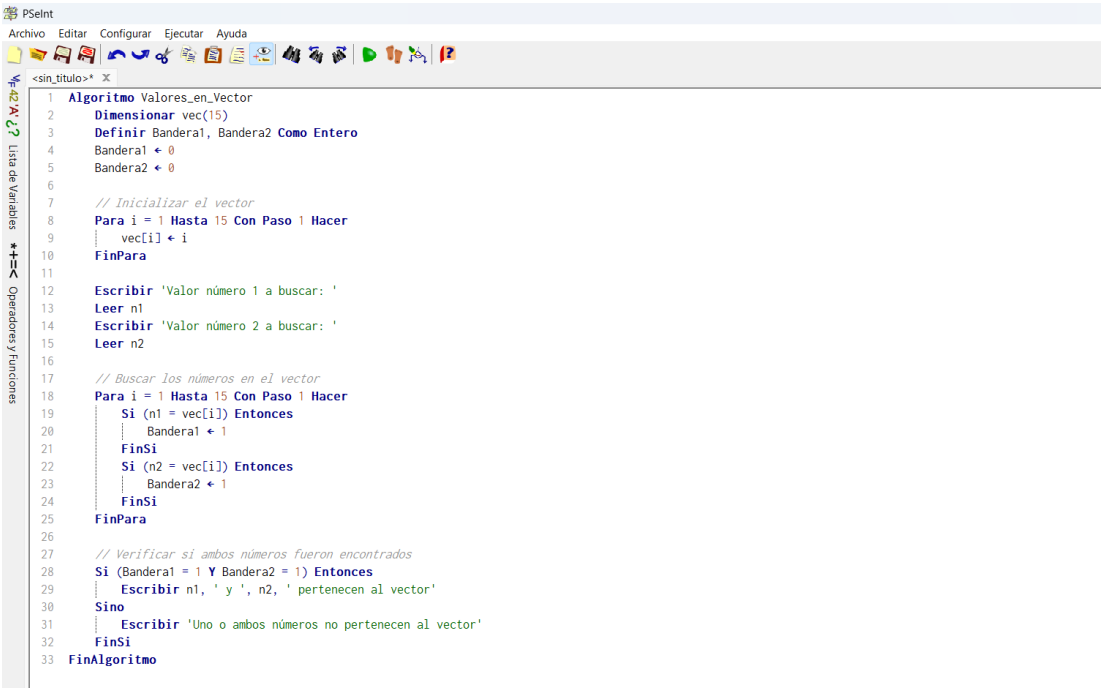
Comprobar si dos valores pertenecen a un vector.

Realice un algoritmo que lea dos números enteros por teclado y determine si ambos valores forman parte de un vector de enteros previamente definido de dimensión 15.

Tabla de objetos

Objeto	Nombre	Valor	Tipo
Vector	Vec	Variable	Entero
Bandera	Bandera1	Constante	Entero
Bandera	Bandera2	Constante	Entero
Numero 1	n1	Variable	Entero
Numero 2	n2	Variable	Entero

Código en Pseint




```
Algoritmo Valores_en_Vector
    Dimensionar vec(15)
    Definir Bandera1, Bandera2 Como Entero
    Bandera1 ← 0
    Bandera2 ← 0

    // Inicializar el vector
    Para i = 1 Hasta 15 Con Paso 1 Hacer
        vec[i] ← i
    FinPara

    Escribir 'Valor número 1 a buscar: '
    Leer n1
    Escribir 'Valor número 2 a buscar: '
    Leer n2

    // Buscar los números en el vector
    Para i = 1 Hasta 15 Con Paso 1 Hacer
        Si (n1 = vec[i]) Entonces
            Bandera1 ← 1
        FinSi
        Si (n2 = vec[i]) Entonces
            Bandera2 ← 1
        FinSi
    FinPara

    // Verificar si ambos números fueron encontrados
    Si (Bandera1 = 1 Y Bandera2 = 1) Entonces
        Escribir n1, ' y ', n2, ' pertenecen al vector'
    Sino
        Escribir 'Uno o ambos números no pertenecen al vector'
    FinSi
FinAlgoritmo
```

Código en C

[*] Ejercicio2.cpp

```c
#include <stdio.h>
int main() {
    int vec[15];
    int Bandera1 = 0, Bandera2 = 0;
    int n1, n2;
    for (int i = 0; i < 15; i++) {
        vec[i] = i + 1;
    }
    printf("Valor numero 1 a buscar: ");
    scanf("%d", &n1);
    printf("Valor numero 2 a buscar: ");
    scanf("%d", &n2);

    for (int i = 0; i < 15; i++) {
        if (n1 == vec[i]) {
            Bandera1 = 1;
        }
        if (n2 == vec[i]) {
            Bandera2 = 1;
        }
    }
    if (Bandera1 == 1 && Bandera2 == 1) {
        printf("%d y %d pertenecen al vector\n", n1, n2);
    } else {
        printf("Uno o ambos numeros no pertenecen al vector\n");
    }
    return 0;
}
```

Prueba de escritorio

```
Valor numero 1 a buscar: 12
Valor numero 2 a buscar: 4
12 y 4 pertenecen al vector

--------------------------------
Process exited after 4.679 seconds with return value 0
Presione una tecla para continuar . . .
```

Problema 2.1.5 Vector de factoriales.

Dado un vector, Vec, que contiene los primeros 15 números naturales, calcule un vector fact con sus factoriales y mostrarlo por pantalla.

Nota: ¿Por qué a partir del número 12 no funciona correctamente el cálculo del factorial? ¿Qué se podría hacer para evitarlo?

El algoritmo consiste en dos bucles anidados, uno externo, que da valores a la variable i entre 1 y 15, y otro interno, que calcula el factorial de la componente i-ésima. El algoritmo deja de funcionar para el valor 12 porque en la codificación en C se han empleado variables de tipo entero. El valor de 13! excede la capacidad de almacenamiento de una variable entera, y el cálculo se corrompe. Una posible solución es emplear variables de tipo unsigned long int (entero largo si signo), que incrementan la capacidad de almacenamiento.

Tabla de objetos:

Objeto	Nombre	Valor	Tipo
Dato 1	TAM	Constante	Entero
Dato 2	Vec	Variable	Entero
Dato 3	Fac	Variable	Entero largo sin signo
Dato 4	i	Variable	Entero
Dato 5	j	Variable	Entero
Proceso 1	Inicializarvector	Subproceso	Procedimiento
Proceso 2	Calclularfactoriales	Subproceso	Procedimiento
Proceso 3	Mostrarfactoriales	Subproceso	Procedimiento

Código Pseint:

```
Algoritmo factorial_vector
   Definir TAM <- 15
   Definir vec[15], fac[15] Como Entero Largo Sin Signo
   Definir i, j Como Entero
   SubProceso inicializarVector(vec Por Referencia, TAM)
      Para i <- 0 Hasta TAM - 1 Con Paso 1
         vec[i] <- i + 1
      FinPara
   FinSubProceso
   SubProceso calcularFactoriales(vec, fac Por Referencia, TAM)
      Para i <- 0 Hasta TAM - 1 Con Paso 1
         fac[i] <- 1
         Para j <- 1 Hasta vec[i] Con Paso 1
```

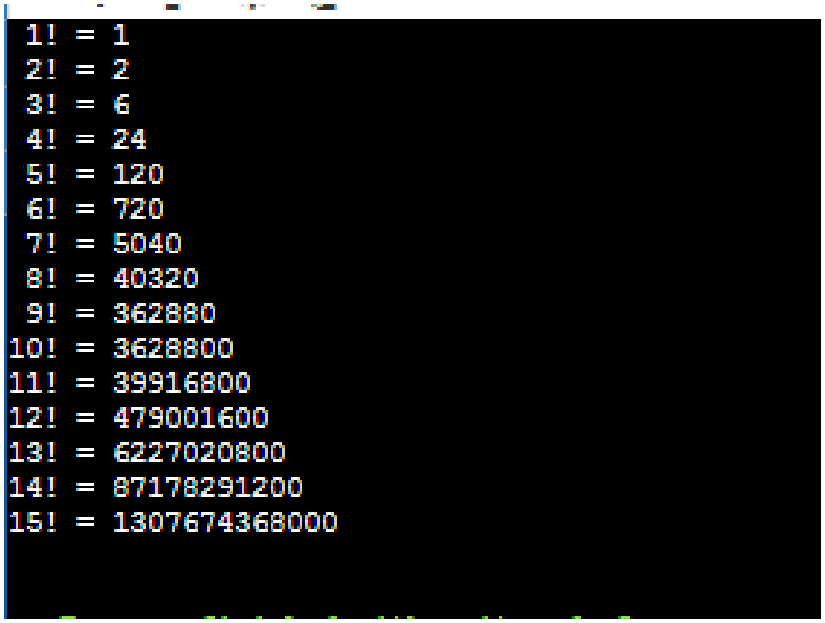
```
            fac[i] <- fac[i] * j
        FinPara
    FinPara
FinSubProceso
SubProceso mostrarFactoriales(vec, fac, TAM)
    Para i <- 0 Hasta TAM - 1 Con Paso 1
        Escribir vec[i], "! = ", fac[i]
    FinPara
FinSubProceso
inicializarVector(vec, TAM)
calcularFactoriales(vec, fac, TAM)
mostrarFactoriales(vec, fac, TAM)
FinAlgoritmo
```

Codigo C:

main.c

```c
#include <stdio.h>
#define TAM 15

void inicializarVector(int vec[], int n) {
    for (int i = 0; i < n; i++) {
        vec[i] = i + 1;
    }
}

void calcularFactoriales(int vec[], unsigned long long int fac[], int n) {
    for (int i = 0; i < n; i++) {
        fac[i] = 1;
        for (int j = 1; j <= vec[i]; j++) {
            fac[i] *= j;
        }
    }
}

void mostrarFactoriales(int vec[], unsigned long long int fac[], int n) {
    for (int i = 0; i < n; i++) {
        printf("%2d! = %llu\n", vec[i], fac[i]);
    }
}

int main() {
    int vec[TAM];
    unsigned long long int fac[TAM];

    inicializarVector(vec, TAM);
    calcularFactoriales(vec, fac, TAM);
    mostrarFactoriales(vec, fac, TAM);

    return 0;
}
```

Prueba de escritorio:



Problema 2.1.6 Ordenación de un vector.

Desarrolle un programa que ordene un vector de 10 componentes de mayor a menor valor. Asuma que el vector está ya leído y almacenado en memoria.

Existen varios métodos de ordenación. Se expondrán dos:

Ordenación iterativa. Esta solución emplea dos bucles anidados para comparar cada elemento del vector con los que le siguen, intercambiando las parejas de elementos fuera de orden. Por ejemplo, para ordenar un vector de n elementos numerados de 0 an- n-1 (en DF de 1 an), se comienza comparando el elemento 0 con los que le siguen, es decir, las parejas de elementos (0-1), (0-2), (0-3), ...(0-n-1) (ojo, en DF se numeran los elementos desde 1). Si alguna de estas parejas presenta un orden inverso al deseado, se intercambian las posiciones de sus elementos en el vector. Tras comparar la última pareja, se tiene la garantía de que el elemento de mayor valor está en la posición 0. El procedimiento se repite a continuación con el elemento 1, esto es, se estudian las parejas (1-2), (1-3), ..., (1-n-1), intercambiando aquellas fuera de orden. El procedimiento se repite hasta llegar al elemento n - 2, que debe ser comparado con el último, es decir, (n-2-n-1). La solución aparece detallada en el diagrama de la figura 2.6, junto con su tabla de objetos, así como en la correspondiente codificación en C en el código 2.1.6.

Tabla de objetos:

Objeto	Nombre	Valor	Tipo
vector	vec	Variable	Entero
Dato 1	i	Variable	Entero
Dato 2	j	Variable	Entero
Aux	Auxiliar para cambio de enteros	Variable	Entero

Codigo Pseint:

```
Algoritmo OrdenarVectorMayorAMenor
   Entrada: vector[10] (ya almacenado en memoria)
   Salida: vector[10] ordenado de mayor a menor
   Para i desde 0 hasta 8 hacer:
      Para j desde i+1 hasta 9 hacer:
         Si vector[i] < vector[j] entonces:
                                    // Intercambiar elementos
                                    aux = vector[i]
                                    vector[i] = vector[j]
                                    vector[j] = temp
                              Fin Si
                        Fin Para
                  Fin Para
    Fin Algoritmo
```

Código C:

Código en C

```c
#include <stdio.h>
#define TAM 10

void inicializar(int v[]);
void ordenar(int v[], int tam);
void mostrar(int v[], int tam);

int main() {
    int vec[TAM];

    inicializar(vec);
    ordenar(vec, TAM);
    mostrar(vec, TAM);

    return 0;
}
void inicializar(int v[]) {
    int datos[TAM] = {45, 12, 78, 23, 56, 89, 34, 67, 10, 90};
    for (int i = 0; i < TAM; i++) {
        v[i] = datos[i];
    }
}
void ordenar(int v[], int tam) {
    int aux;
    for (int i = 0; i < tam - 1; i++) {
        for (int j = i + 1; j < tam; j++) {
            if (v[i] < v[j]) {
                aux = v[i];
                v[i] = v[j];
                v[j] = aux;
            }
        }
    }
}
void mostrar(int v[], int tam) {
    printf("Vector ordenado de mayor a menor:\n");
    for (int i = 0; i < tam; i++) {
        printf("Elemento %d: %d\n", i, v[i]);
    }
}

```

Prueba de Escritorio