

SISTEMAS OPERATIVOS

## TAREA 2 - UNIDAD 3

---

PROFESOR: JORGE CORTÉS GALICIA

GRUPO: 2CM9

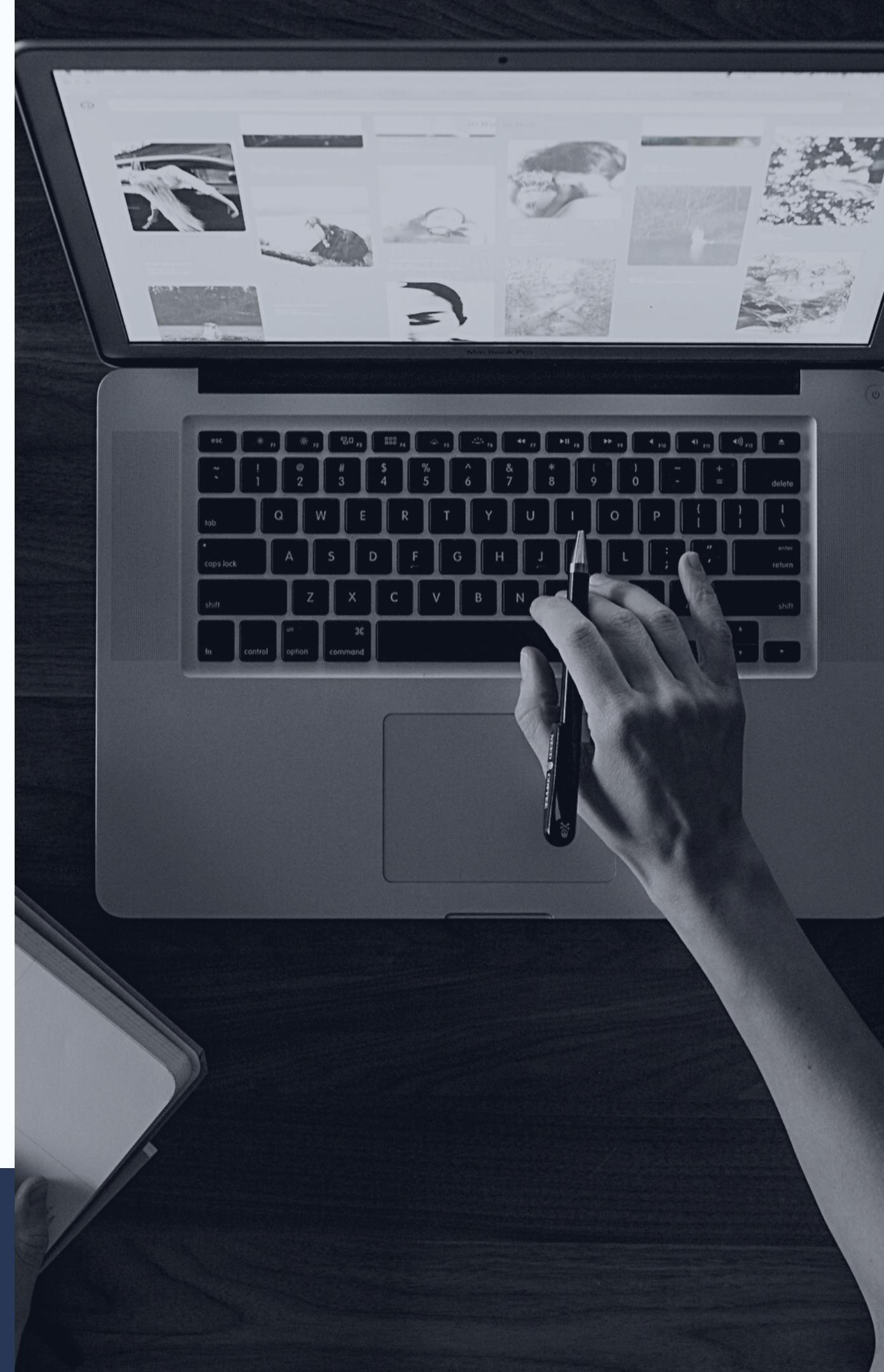
INTEGRANTES DEL EQUIPO:

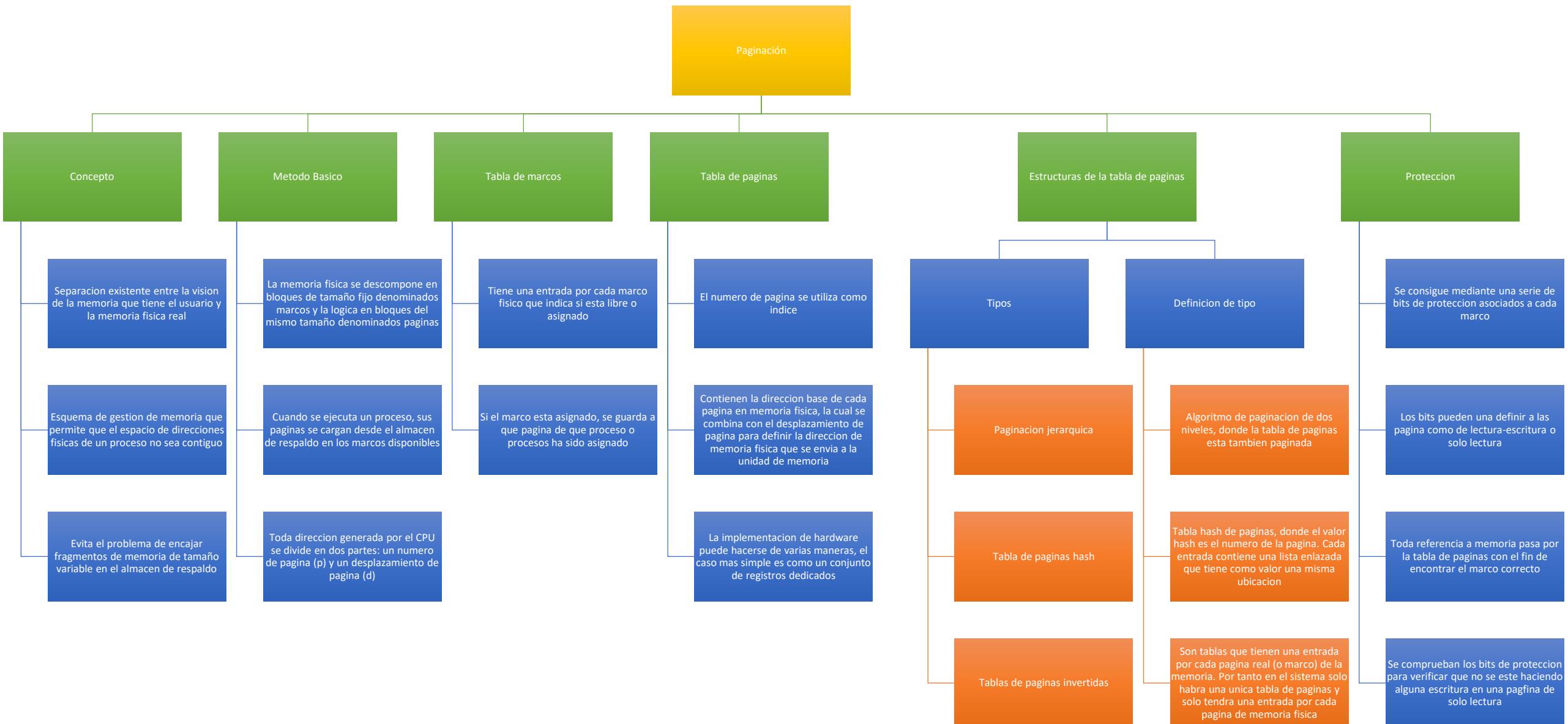
- BELTRAN GARCIA JUAN
- HERNÁNDEZ MÉNDEZ OLIVER MANUEL
- LÓPEZ LÓPEZ RODRIGO
- RANGEL LOZADA KEVIN SEBASTIÁN

# MAPA CONCEPTUAL

Lectura 4 , 5 y video (tema 3.3)

## MEMORIA VIRTUAL



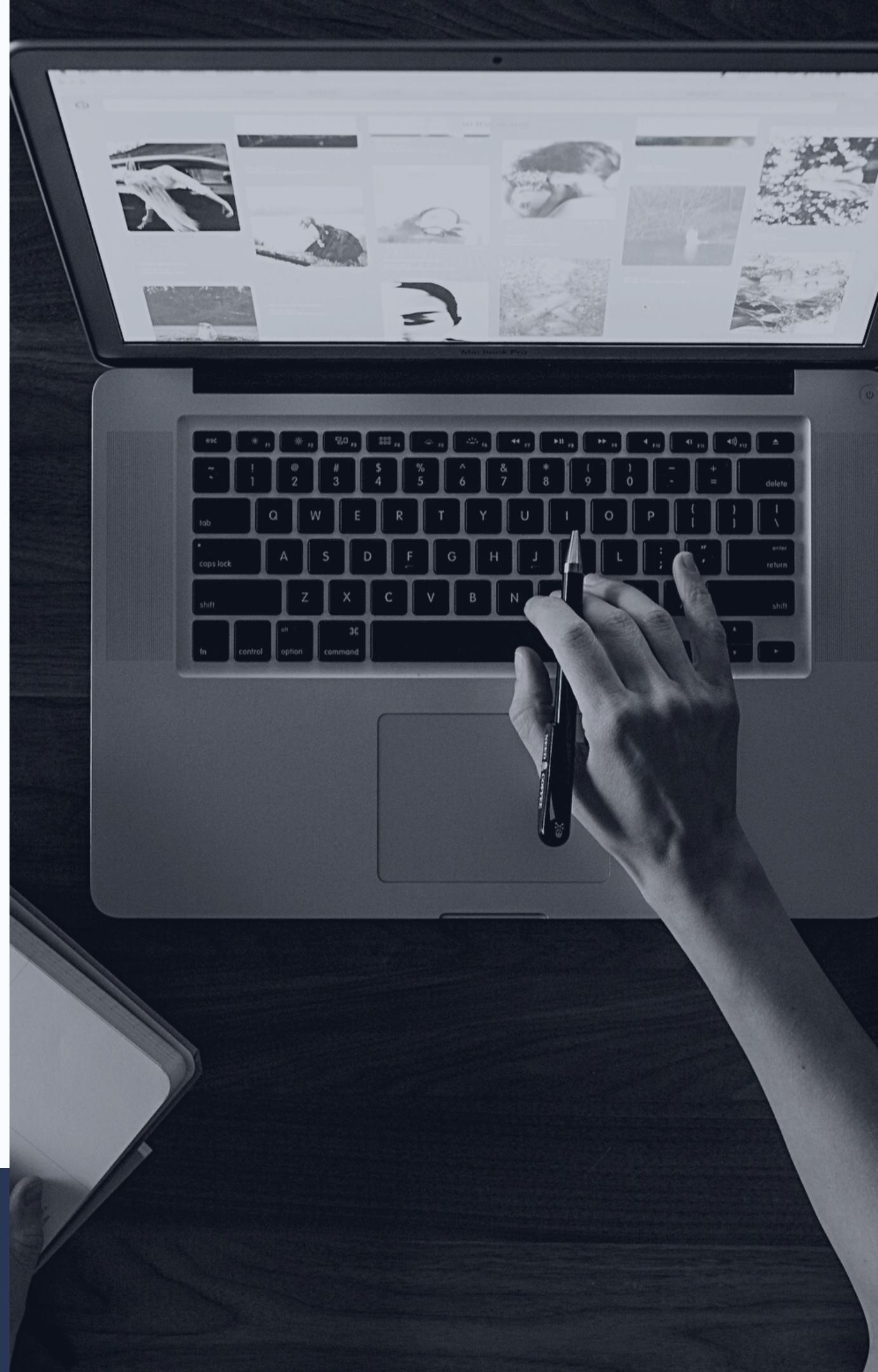




# PSEUDOCÓDIGOS

Lectura 6 y video (tema 3.4)

## ALGORITMOS DE SUSTITUCIÓN DE PÁGINAS



## Algoritmos de sustitución de páginas

### Sustitución de páginas no usada recientemente variables

```
Estructura pagina paginaEntrante
    instrucciones : tipo_de_elemento;
    rotulo : numero_de_instrucciones;
    bitR : entero;
    bitM : entero;
    bitV-I : caracter;
    array de tipo pagina tablapaginacion;
    entero i,j;

Funcion ComprobarTabla(tablapaginacion)
booleano bandera <- falso;
Desde i=0 hasta numero de filas del arreglo
    Si tablapaginacion[i][1] = V entonces
        bandera <- verdadero;
    finSi;
    siguiente;
finLoop;
retornar bandera;

Inicio
    Si      (ComprobarTabla(tablapaginacion)      ==      verdadero)
entonces
    Desde i=0 hasta numero de filas del arreglo
        Si tablapaginacion[i]->bitV-I = v entonces ( v
            significa valido, se puede asignar)
            tablapaginacion[i]<- paginaEntrante;
            ColaP.push(pagina);
            finSi;
            siguiente;
    finLoop;
sino
    Desde i=0 hasta numero de filas del arreglo
        Si      (tablapaginacion[i]->bitM      ==      0      Y
            tablapaginacion[i]->bitR==0)entonces
            tablapaginacion[i] <- paginaEntrante;
            finSi;
            ActualizarBitRM(tablapaginacion[i]);
    finSi;
Fin
```

## **sustitución de Paginas FIFO**

```
FIFO
variables
    Estructura pagina paginaEntrante
        instrucciones : tipo_de_elemento;
        bitV-I : caracter;
    array de tipo pagina tablapaginacion;
    entero i,j;
    Estructura Cola de paginas ColaP;
    booleano bandera <- falso;

    Funcion ComprobarTabla(tablapaginacion)
        booleano bandera <- falso;
        Desde i=0 hasta numero de filas del arreglo
            Si tablapaginacion[i][1] = V entonces
                bandera <- verdadero;
            finSi;
            siguiente;
        finLoop;
        retornar bandera;

    Inicio
        Si      (ComprobarTabla(tablapaginacion)      ==      verdadero)
        entonces
            Desde i=0 hasta numero de filas del arreglo
                Si tablapaginacion[i]->bitV-I = v entonces ( v
                    significa valido, se puede asignar)
                    tablapaginacion[i]<- paginaEntrante;
                    ColaP.push(pagina);
                    finSi;
                    siguiente;
                finLoop;
            sino
                pagina paginaAUX <- ColaP.pop();
                Desde i=0 hasta numero de filas del arreglo
                    Si tablapaginacion[i] == paginaAUX entonces
                        tablapaginacion[i]<- paginaEntrante\
                    finSi;
                finSi;
            fin
```

## sustitución optima de paginas

```
variables
    Estructura pagina paginaEntrante
        instrucciones : tipo_de_elemento;
        rotulo : numero_de_instrucciones;
        bitV-I : caracter;
        array de tipo pagina tablapaginacion;

        entero i,j,mayor,posicion;
    Estructura Cola de paginas ColaP;

    Funcion ComprobarTabla(tablapaginacion)
        booleano bandera <- falso;
        Desde i=0 hasta numero de filas del arreglo
            Si tablapaginacion[i][1] = V entonces
                bandera <- verdadero;
            finSi;
            siguiente;
        finLoop;
        retornar bandera;

    Inicio
        Si      (ComprobarTabla(tablapaginacion)      ==      verdadero)
    entonces
        Desde i=0 hasta numero de filas del arreglo
            Si tablapaginacion[i]->bitV-I = v entonces ( v
            significa valido, se puede asignar)
                tablapaginacion[i]<- paginaEntrante;
                ColaP.push(pagina);
                finSi;
                siguiente;
            finLoop;
        sino
            mayor <- tablapaginacion[0]->;
            Desde i=0 hasta número de filas del arreglo
            Si(tablapaginacion[i]->rotulo > mayor) entonces
                mayor <- tablapaginacion[i]->rotulo;
                posicion <- i;
                finSi;
                siguiente;
            fin;
            tablapaginacion[posicion] <- paginaEntrante;
        finSi;
    Fin
```

## sustitución segunda oportunidad de paginas

variables

```
Estructura página paginaEntrante;
    instrucciones: tipo_de_elemento;
    bitReferencia : entero;
array de tipo página tablapaginacion;
entero i,j;
colaCircular de tipo paginas ColaMarcosC;
```

```
Funcion ComprobarTabla(tablapaginacion)
booleana bandera <- falso;
Desde i=0 hasta numero de filas del arreglo
    Si tablapaginacion[i][1] = 0 entonces
        bandera <- verdadero;
    finSi;
    siguiente;
finLoop;
retornar bandera;
```

Inicio

```
    Si      (ComprobarTabla(tablapaginacion)      ==      verdadero)
entonces
```

```
        Desde i=0 hasta número de filas del arreglo
            Si tablapaginacion[i]->bitV-I = 0 entonces
                tablapaginacion[i]<- paginaEntrante;
                tablapaginacion[i]->bitReferencia =1;
                ColaMarcosC.push(paginaEntrante);
            finSi;
            siguiente;
        finLoop;
```

sino

```
    Mientras (ColaMarcosC->bitReferencia != 0 )
        pagina paginaAUX <- ColaMarcosC.pop();
        Desde i=0 hasta número de filas del arreglo
            Si      tablapaginacion[i]      ==      paginaAUX
            entonces
                tablapaginacion[i]<- paginaEntrante\
            finSi;
        finSi;
    Sino
        pagina paginaAUX <- ColaMarcosC.consulta();
        Desde i=0 hasta número de filas del arreglo
            Si      tablapaginacion[i]      ==      paginaAUX
            entonces
                tablapaginacion[i]->bitReferencia =0;
```

```

                finSi;
            finSi;
        ColaMarcosC<- ColaMarcosC->nextPagina;
        finSi;
        finLoop;
    finSi;
Fin

```

### **sustitución segunda oportunidad de páginas mejorado variables**

```

Estructura página paginaEntrante;
    instrucciones: tipo_de_elemento;
    bitReferencia : entero;
    bitMoficacion : entero;
    array de página tablapaginacion;
    entero i,j;
    colaCircular de tipo paginas ColaMarcosC;

Funcion ComprobarTabla(tablapaginacion)
booleana bandera <- falso;
Desde i=0 hasta numero de filas del arreglo
    Si tablapaginacion[i][1] = 0 entonces
        bandera <- verdadero;
    finSi;
    siguiente;
finLoop;
retornar bandera;

Inicio
    Si      (ComprobarTabla(tablapaginacion)      ==      verdadero)
entonces
    Desde i=0 hasta número de filas del arreglo
        Si tablapaginacion[i]->bitV-I = 0 entonces
            tablapaginacion[i]<- paginaEntrante;
            tablapaginacion[i]->bitReferencia =1;
            ColaMarcosC.push(paginaEntrante);
        finSi;
        siguiente;
    finLoop;
sino

Mientras      (ColaMarcosC->bitReferencia!=      0      Y
bitMoficacion != 0 )
    pagina paginaAUX <- ColaMarcosC.pop();
    Desde i=0 hasta número de filas del arreglo
        Si      tablapaginacion[i]      ==      paginaAUX

```

```

        entonces
            tablapaginacion[i]<- paginaEntrante\
        finSi;
    finSi;
Sino
    pagina paginaAUX <- ColaMarcosC.consulta();
    Desde i=0 hasta número de filas del arreglo
        Si tablapaginacion[i] == paginaAUX
        entonces
            tablapaginacion[i]->bitReferencia =0;
        finSi;
    finSi;
    ColaMarcosC<- ColaMarcosC->nextPagina;
    finSi;
    finLoop;
finSi;
Fin

```

### **sustitución de paginas LRU**

Algortimo LRU

variables

```

Estructura pagina paginaEntrante
    instrucciones : tipo_de_elemento;
    tiempo: entero;
    bitV-I : caracter;
array de tipo pagina tablapaginacion;
entero i,j,mayor, posicion;
Estructura Cola de paginas ColaP;

```

```

Funcion ComprobarTabla(tablapaginacion)
booleano bandera <- falso;
Desde i=0 hasta numero de filas del arreglo
    Si tablapaginacion[i][1] = V entonces
        bandera <- verdadero;
    finSi;
    siguiente;
finLoop;
retornar bandera;

```

Inicio

```

    Si (ComprobarTabla(tablapaginacion) == verdadero)
entonces
    Desde i=0 hasta numero de filas del arreglo
        Si tablapaginacion[i]->bitV-I = v entonces (v
significa valido, se puede asignar)
            tablapaginacion[i]<- paginaEntrante;

```

```

        ColaP.push(pagina);
        finSi;
        siguiente;
    finLoop;
sino
    mayor <- tablapaginacion[0];
    Desde i=0 hasta numero de filas del arreglo
    mayor <- tablapaginacion[0];
    Desde i=0 hasta numero de filas del arreglo
    Si(tablapaginacion[i]->tiempo > mayor) entonces
        mayor <- tablapaginacion[i]->tiempo;
        posicion <- i;
        finSi;
        siguiente;
    fin;
    tablapaginacion[posicion] <- paginaEntrante;
finSi;
Fin

sustitución de páginas basada en contador
variables
    Estructura pagina paginaEntrante
        instrucciones : tipo_de_elemento;
        Ccorrimietos: entero;
        bitV-I : caracter;
    array de tipo pagina tablapaginacion;
    entero i,j,menor,posicion;
    Estructura Cola de paginas ColaP;

Funcion ComprobarTabla(tablapaginacion)
booleano bandera <- falso;
Desde i=0 hasta numero de filas del arreglo
    Si tablapaginacion[i][1] = V entonces
        bandera <- verdadero;
    finSi;
    siguiente;
finLoop;
retornar bandera;

Inicio
    Si (ComprobarTabla(tablapaginacion)== verdadero) entonces
        Desde i=0 hasta numero de filas del arreglo
            Si tablapaginacion[i]->bitV-I = v entonces (v
                significa valido, se puede asignar)
                tablapaginacion[i]->Ccorrimientos<-
                tablapaginacion[i]->Ccorrimientos>> 1;
                tablapaginacion[i]<- paginaEntrante;

```

```

        finSi;
        siguiente;
    finLoop;
sino
    Desde i=0 hasta numero de filas del arreglo
        Si tablapaginacion[i]==paginaEntrante entonces
            tablapaginacion[i]->Ccorrimientos<-
                tablapaginacion[i]->Ccorrimientos>> 1;
        Sino
            menor <- tablapaginacion[0]->Ccorrimiento;
            Desde i=0 hasta numero de filas del arreglo
                Si(tablapaginacion[i]->Ccorrimiento <
                    menor) entonces
                    menor <- tablapaginacion[i]-
                        >Ccorrimiento;
                    posicion <- i;
                    finSi;
                    siguiente;
                fin;
                tablapaginacion[posicion]<-
                    PaginaEntrante;
            finSI;

        Finloop;
    finSi;
Fin

```