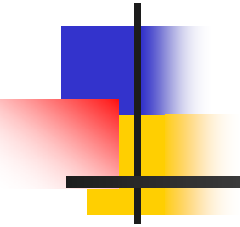


# Ampliación de la Programación



M Procesadores  
Programación Dinámica



# M Procesadores: Prog. Dinámica

## Resolución

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

```
#define DIMMAX 20
```

```
#define INFINITO 10000
```

```
#define TRUE 1
```

```
#define FALSE 0
```

```
typedef struct {
```

```
    int coste;
```

```
    int trabajo [DIMMAX]; // trabajo[i] es del procesador que hace el trabajo i
```

```
} TipoSolucion;
```

```
void MProcesadores (int k, int tproc[DIMMAX], TipoSolucion *solOptima,  
                    int M, int N, int tiempo[DIMMAX][DIMMAX]);
```

```
void Salida (TipoSolucion sol, int M, int N, int tiempo[DIMMAX][DIMMAX]);
```



# M Procesadores: Prog. Dinámica

---

## Resolución

```
int maximo (int tproc[DIMMAX], int M) {  
    int i, max;
```

```
    max = tproc[0];  
    for (i=1; i<M; i++) if (max < tproc[i]) max = tproc[i];  
    return max;  
}
```

```
void Inicializa (int tproc[DIMMAX], int M) {  
    int i;
```

```
    for (i=0; i<M; i++) tproc[i] = 0;  
}
```

```
void Entrada (int *M, int *N, int tiempo[DIMMAX][DIMMAX]);
```



# M Procesadores: Prog. Dinámica

---

## Resolución

```
int main () {  
  
    int tiempo [DIMMAX][DIMMAX]; // matriz de costes  
    int M, N;  
    TipoSolucion solOptima;  
    int tproc [DIMMAX];  
  
    Entrada(&M,&N,tiempo);  
  
    Inicializa (tproc,M);  
  
    MProcesadores(0,tproc,&solOptima,M,N,tiempo);  
  
    Salida(solOptima,M,N,tiempo);  
  
    return 0;  
  
}
```



# M Procesadores: Prog. Dinámica

## Resolución

```
void MProcesadores (int k, int tproc[DIMMAX], TipoSolucion *solOptima,  
                    int M, int N, int tiempo[DIMMAX][DIMMAX]) {
```

```
    int p; TipoSolucion solParc, solMejor;
```

```
    if (k == N)
```

```
        solOptima->coste = maximo (tproc,M);
```

```
    else {
```

```
        solMejor.coste = INFINITO;
```

```
        for (p=0; p<M; p++) {
```

```
            // lo hace el procesador p
```

```
            tproc[p] += tiempo[p][k];
```

```
            MProcesadores (k+1, tproc, &solParc, M, N, tiempo);
```

```
            solParc.trabajo[k] = p;
```

```
            // vemos si es mejor que solMejor
```

```
            if (solParc.coste < solMejor.coste) solMejor = solParc;
```

```
            // deshacemos para la siguiente iteración
```

```
            tproc[p] -= tiempo[p][k];
```

```
        }
```

```
        // nos quedamos con la mejor de las dos
```

```
        *solOptima = solMejor;
```

```
    }
```

```
}
```

marzo de 2011



# M Procesadores: Prog. Dinámica

## Resolución

```
void Entrada (int *M, int *N, int tiempo[DIMMAX][DIMMAX]) {
    int i, j;

    printf("Introduzca el número de trabajos "); scanf("%d",N);
    printf("Introduzca el número de procesadores "); scanf("%d",M);
    for (i=0; i<*M; i++) {
        printf("\nDuración de los trabajos en el procesador %d (separados por un espacio)\n",i);
        for (j=0; j<*N; j++)
            scanf(" %d", &tiempo[i][j]);
    }

    printf("\n\n");
    for (i= 0; i<*M; i++) {
        for (j=0; j<*N; j++)
            printf(" %4d",tiempo[i][j]);
        printf("\n");
    }
}
```



# M Procesadores: Prog. Dinámica

---

## Resolución

```
void Salida (TipoSolucion sol, int M, int N, int tiempo[DIMMAX][DIMMAX]) {  
    int i, p;  
  
    for (p= 0; p<M; p++) {  
        printf("\nTrabajos realizados por el procesador %d: ", p);  
        for (i=0; i<N; i++)  
            if (sol.trabajo[i] == p) printf("%d (+ %d ) ", i, tiempo[p][i]);  
        printf("\n");  
    }  
    printf("\n\nEl coste de la solucion es: %d",sol.coste);  
}
```