ESQUEMAS DE ESTRUCTURAS PARA TABLAS C++

RELLENAR TABLA UNA DIMENSION

```
#define M 10
class clase {
        int tabla[M];
public:
        void rellenar();
};

void clase::rellenar(){
    for(int i = 0; i < M; i++){
        cout << "Introduzca un numero entero (" << (i+1) << "): ";
        cin >> tabla[i];
    }
    cout << "\n";
}
```

RELLENAR TABLA DOS DIMENSIONES

```
#define M 10
#define N 15
class clase {
        int tabla[M][N];
public:
        void cargar();
};

void clase::rellenar(){
   int a;
   for(int i = 0; i < M; i++){
        for(int j=0; j < N; j++){
             cout << “Introduce un numero, que se almacenara en  <<i+1<<” , “<<j+1<<endl;
              cin >> a;
        }
    tabla[i][j]=a;
    }
}
```

RELLENAR TABLA DOS DIMENSIONES CON CONDICIONAL

```
#define N 3
#define M 4
class datos {
   int valores[N][M];
public:
   void Cargar();
};

void datos::Cargar()
{
   int a;
   for (int i=0; i<N; i++)
   {
      for (int j=0; j<M; j++)
      {
         cout << "Introduzca un valor entre 10 y 20 para la posicion " << i+1 << ", " << j+1 << endl;
         cin >> a;
         while (a < 10 || a > 20)
         {
            cout << "El numero introducido no es correcto, introduzcalo de nuevo" << endl;
            cin >> a;
         }
         valores[i][j] = a;
      }
   }
}
```

RELLENAR TABLA UNA DIMENSION CON ESTRUCTURA

```
#define M 5
typedef char cadena[30];

struct producto {
    cadena nombre;
    float precio;
};

class matrices {
   producto tabla[M];
public:
   void cargar();
};

void matrices::cargar(){
for(int i = 0; i  <M; i++){
```

```
    cout << "\nIndica el nombre del producto en la psicon: " << i+1 <<"\n";
    cin>>tabla[i].nombre;
    cout << "\nIndica el precio por kilo: ";
    cin >> tabla[i].precio;
    }
}
```

RELLENAR TABLA DOS DIMENSIONES CON ESTRUCTURA

```
#define M 2
#define N 4
typedef char cadena[30];

struct persona {
   int dni;
   cadena nombre;
};

class matrices {
      persona tabla[M][N];
public:
      void cargar();
};

void matrices::cargar(){
for(int i = 0; i  <M; i++)
  for(int j = 0; j < N; j++){
    cout << "\nIndica el nombre de la persona que estara en la posicion: " << i << "," <<j<< "\n";
    cin>>tabla[i][j].nombre;
    cout << "\nIndica su DNI: ";
    cin >> tabla[i][j].dni;
    }
}
```

MOSTRAR TABLA (RECORRIDO) UNA DIMENSION

```
#define M 10
class clase {
      int tabla[M];
public:
      void rellenar();
      void mostrar();
};
```

```
void clase::Mostrar(){
   for (int i=0; i<M; i++){
       cout << tabla[i]<< " ";
       cout << endl;
    }
}
```

MOSTRAR TABLA (RECORRIDO) DOS DIMENSIONES

```
#define N 3
#define M 4
class datos {
    int valores[N][M];
public:
    void Cargar();
    void mostrar();
};

void datos::Mostrar(){
   for (int i=0; i<N; i++){
       cout << endl;
       for (int j=0; j<M; j++){
           cout << valores[i][j] << " ";
       }
   }
   cout << endl;
}
```

BUSCAR VALOR MAYOR Y MINIMO VALOR (RECORRIDO) EN UNA TABLA

UNA DIMENSION

```
#define M 10
class clase {
        int tabla[M];
public:
        void rellenar();
        int maximmo();
        int minimo();
};

int clase::maximo(){
    int max = tabla[0];
```

```
    for(int i = 0; i < M; i++)
        if(tabla[i] > max)
            max = tabla[i];
    return max;
}
int clase::minimo(){
    int min = tabla[0];
    for(int i = 1; i < M; i++)
        if(tabla[i] < min)
            min = tabla[i];
    return min;
}
```

DOS DIMENSIONES

```
#define M 4
#define N 3
class datos {
        int tabla[M][N];
public:
        void cargar();
        int maximo();
        int minimo();
};

int datos::Minimo(){
    int vmin;
    vmin = valores[0][0];
    for (int i=0; i<M; i++){
        for (int j=0; j<N; j++){
            if (valores[i][j] < vmin)
                vmin = valores[i][j];
        }
    }
    return vmin;
}

int datos::maximo(){
    int vmin;
    vmin = valores[0][0];
    for (int i=0; i<M; i++){
        for (int j=0; j<N; j++){
            if (valores[i][j] > vmin)
                vmin = valores[i][j];
```

```
        }
    }
    return vmin;
}
```

BUSCAR UN VALOR (BUSQUEDA) EN UNA TABLA UNA DIMENSION

```
#define M 10
class clase {
        int tabla[M];
public:
        void cargar();
        bool buscar();
};
bool clase::buscar(){
    int buscado, i = 0;
    bool encontrado = false;

    cout << "Introduzca el numero que desea buscar: ";
    cin >> buscado;

        while(!encontrado && i < M){
        if(tabla[i] == buscado)
            encontrado = true;
         i++;
        }

    return encontrado;
}
```

BUSCAR UN VALOR (BUSQUEDA) EN UNA TABLA DOS DIMENSIONES

```
#define N 3
#define M 4
class datos {
    int valores[N][M];
public:
    void Cargar();
    bool buscar();
};

bool datos::buscar(){
    bool encontrado = false;
```

```
    int a, i=0, j, k=0;
    cout << "Introduzca el valor que quiere buscar en la tabla" << endl;
    cin >> a;
    while (i<N && !encontrado) {
        j=0;
        while (j<M && !encontrado) {
            if (valores[i][j] == a){
                encontrado=true;
            }
            else
                j++;
        }
        i++;
    }
    return encontrado;
}
```

BUSCAR UNA CADENA UNA DIMENSION (CADENA)(BUSQUEDA) (COMPARACION DE CADENAS)

```
#include <cstring>
#define M 5
typedef char cadena[30];

struct producto {
     cadena nombre;
     float precio;
};

class matrices {
    producto tabla[M];
public:
    void cargar();
    void encontrarpornombre();
};

void matrices::encontrarpornombre(){
    cadena buscado;
    int i = 0;
    bool encontrado = false;
    cout << "Introduzca el producto a buscar: ";
    cin >> buscado;
    while((encontrado==false) && (i < M)){
```

```
        if(strcmp(buscado, tabla[i].nombre)==0)
            encontrado = true;
        else
            i++;
    }

    if(encontrado)
        cout << "\nEl El producto introducido ha sido encontrado. Su precio por kilo es " <<
tabla[i].precio<< ".\n\n";
    else
        cout << "\nEl Producto introducido no esta en la tabla.\n\n";

}
```

BUSCAR UNA VALOR TABLA DOS DIMENSIONES (BUSQUEDA) CON ESTRUCTURA

```
#define M 2
#define N 4

typedef char cadena[30];
struct persona{
    int dni;
    cadena nombre;
};

class matrices{
    private:
        persona tabla[M][N];
    public:
        void cargar();
        void encontrar();
};

void matrices::encontrar(){
    int buscado;
    int j, i = 0;
    bool encontrado = 0;

    cout << "Introduzca el DNI por el que quiere buscar: ";
    cin >> buscado;

    while(!encontrado && i < M){
        j = 0;
        while(!encontrado && j < N){
            if(tabla[i][j].dni == buscado)
```

```
            encontrado = true;
          j++;
        }
        i++;
    }

    if(encontrado)
        cout << "\nEl DNI introducido ha sido encontrado. Su nombre asociado es " << tabla[i-1][j-
1].nombre << ".\n\n";
    else
        cout << "\nEl DNI introducido no esta en la tabla.\n\n";
```