JuanVLL
www.wuolah.com/student/JuanVLL

Funciones-basicas.pdf

Funciones_Básicas_MP

1° Metodología de la Programación

Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

CLASE VECTOR DINÁMICO BÁSICO

```
class Vector {
    private:
        int n;
        double * datos;
        void reservar ( int n)
        void copiar ( double * ptr, int n);
        void liberar ( );
        void copiarV ( const Vector & otro);
    public:
        Vector ( );
        Vector ( int n);
        ~Vector ( );
        Vector (const Vector & otro);
        Vector & operator= ( const Vector & otro);
};
```

¡ MUY IMPORTANTE !

FUNCIÓN PARA RESERVAR MEMORIA

```
void Vector :: reservar ( int n) {
    this -> n = n;
    if (n > 0) {
        datos = new double [n];
    }
    else {
        datos = NULL;
    }
}
```

FUNCIÓN PARA COPIAR

```
void Vector :: copiar (double * ptr, int n) {
    for (int i=0; i < n; i++) {
        datos [i] = ptr [i];
    }
}
```

```
void Vector::liberar(){
    delete[] datos;
    datos = NULL;
    n = 0;
}
```

FUNCIÓN PARA LIBERAR MEMORIA

```
void Vector::copiarV(const Vector & otro){
    reservar(otro.n);
    copiar(otro.datos, otro.n);
}
```

FUNCIÓN PARA COPIAR UN OBJETO DE LA CLASE VECTOR

```
Vector::Vector(){
    n = 0;
    datos = NULL;
}
```

CONSTRUCTOR POR DEFECTO

```
Vector::Vector(int n){
    this->n = n;
    reservar(n);
}
```

CONSTRUCTOR POR PARÁMETROS

```
Vector::~Vector(){
    liberar();
}
```

DESTRUCTOR

```
Vector::Vector(const Vector & otro){
    copiarV();
}
```

CONSTRUCTOR DE COPIA

```
Vector& Vector::operator=(const Vector & otro){
    if(this != &otro){
        liberar();
        reservar(otro.n);
        copiar(otro.datos, otro.n);
    }
    return *this;
}
```

SOBRECARGA DEL OPERADOR DE ASIGNACIÓN

CLASE MATRIZ DINÁMICA BÁSICA

```
class Matriz {
    private:
        int fil, col;
        double ** datos;

        void reservar (int fil, int col);
        void copiar (double ** ptr, int fil, int col);
        void liberar ();
        void copiarM (const Matriz & M);
    public:
        Matriz ();
        Matriz (int fil, int col);
        ~Matriz ();
        Matriz (const Matriz &otra);
        Matriz & operador = (const Matriz & M);
};
```

¡MUY IMPORTANTES!

```
void Matriz:: liberar () {
    for (int i=0; i < fils; i++) {
        delete [] datos [i];
    }
    delete [] datos;
    datos = NULL;
    fil = col = 0;
}
```

FUNCIÓN PARA LIBERAR MEMORIA

```
void Matriz::reservar (int fil, int col) {
    this -> fil = fil;
    this -> col = col;
    if (fil * col == 0) {
        datos = NULL;
    else {
        datos = new double *[fil];
        for (int i=0; i< col; i++) {
            datos [i] = new double [col];
        }
    }
}
```

FUNCIÓN PARA RESERVAR MEMORIA

```
void Matriz::copiar (double ** ptr, int fils, int cols) {
    for (int i=0; i< fils; i++ ) {
        for (int j=0; j< cols; j++ ) {
            datos [i][j] = ptr [i][j];
        }
    }
}
```

FUNCIÓN PARA COPIAR

```
void Matriz::copiarM( const Matriz &M) {
    reservar (M.fil, M.col);
    copiar ( M.datos, M.fil, M.col);
}
```

FUNCIÓN PARA COPIAR UN OBJETO DE LA CLASE MATRIZ

```
Matriz::Matriz() {
    fil = col = 0;
    datos = NULL;
}
```

CONSTRUCTOR POR DEFECTO

```
Matriz::Matriz ( int fil, int col) {
    this -> fil = fil;
    this -> col = col;
    reservar (fil, col);
}
```

CONSTRUCTOR POR PARÁMETROS

```
Matriz::~Matriz() {
    liberar();
}
```

DESTRUCTOR

```
Matriz::Matriz(const Matriz &otra) {
    copiarM(otra);
}
```

CONSTRUCTOR DE COPIA

```
Matriz& Matriz::operator=(const Matriz &M) {
    if (this != &M) {
        liberar();
        reservar(M.fil, M.col);
        copiar(M.datos, M.fil, M.col);
        liberar();
        copiarM(M);
    }
    return *this;
}
```

SOLUCIÓN ALTERNATIVA

SOBRECARGA DEL OPERADOR DE ASIGNACIÓN