```cpp
#include "Matriz.h"
#include <assert.h>

//Matriz cuadrada
Matriz::Matriz(int n)
{
    mat = new int* [n];
    for(int i=0; i<n; i++)
        mat[i] = new int[n];
    dimension=n;
}

Matriz::Matriz(const Matriz& otra)
{
    mat = new int* [otra.getDimension()];
    for(int k=0; k<otra.getDimension(); k++)
        mat[k]= new int [otra.getDimension()];
    dimension=otra.getDimension();
    for (int i=0; i<dimension; i++)
        for (int j=0; j<dimension; j++)
            mat[i][j] = otra.getValue(i, j);
}

void Matriz::obtenerCuadrante(const Matriz& otra, int inicio1, int inicio2)
{
    for(int i=0; i<dimension; i++)
    {
        int aux=inicio2;
        for(int j=0; j<dimension; j++)
        {
            mat[i][j]=otra.getValue(inicio1, aux);
            aux++;
        }
        inicio1++;
    }
}

Matriz& Matriz::operator=(const Matriz& otra)
{
    assert(dimension = otra.getDimension());
    for (int i=0; i<dimension; i++)
        for (int j=0; j<dimension; j++)
            mat[i][j] = otra.getValue(i, j);
    return *this;
}

Matriz Matriz::operator+(const Matriz otra)
{
    assert(dimension = otra.getDimension());
    Matriz new_mat(otra.getDimension());
    for(int i=0;i<dimension; i++)
        for(int j=0; j<dimension; j++){
            new_mat.setValue(i,j, mat[i][j]+ otra.getValue(i, j));
        }
    return new_mat;
}

Matriz Matriz::operator-(const Matriz otra)
{
    assert(dimension = otra.getDimension());
    Matriz new_mat(otra.getDimension());
    for(int i=0;i<dimension; i++)
        for(int j=0; j<dimension; j++)
           new_mat.setValue(i,j, mat[i][j]- otra.getValue(i, j));
    return new_mat;
}
```

```
67
68  int Matriz::getValue(int i, int j) const
69  {
70      assert(i>0 || i<dimension);
71      assert(j>0 || j<dimension);
72      return mat[i][j];
73  }
74
75  void Matriz::setValue(int i, int j, int value)
76  {
77      assert(i>0 || i<dimension);
78      assert(j>0 || j<dimension);
79      mat[i][j]=value;
80  }
81
82  void Matriz::opSuma(const Matriz& A, int i, int j, const Matriz& B, int n, int m)
83  {
84      for(int k=0; k<this->getDimension(); k++)
85      {
86          int aux1=j;
87          int aux2=m;
88          for(int l=0; l<this->getDimension(); l++)
89          {
90              mat[k][l]=A.getValue(i,aux1)+B.getValue(n,aux2);
91              aux1++;
92              aux2++;
93          }
94          i++;
95          n++;
96      }
97  }
98
99  void Matriz::opResta(const Matriz& A, int i, int j, const Matriz& B, int n, int m)
100 {
101     for(int k=0; k<this->getDimension(); k++)
102     {
103         int aux1=j, aux2=m;
104         for(int l=0; l<this->getDimension(); l++)
105         {
106             mat[k][l]=A.getValue(i,aux1)-B.getValue(n,aux2);
107             aux1++;
108             aux2++;
109         }
110         i++;
111         n++;
112     }
113 }
114
115 void Matriz::asignarCuadrante(const Matriz& otra, int inicio1, int inicio2)
116 {
117     for (int i=0; i<otra.getDimension(); i++){
118         int aux=inicio2;
119         for (int j=0; j<otra.getDimension(); j++)
120         {
121             mat[inicio1][aux]=otra.getValue(i,j);
122             aux++;
123         }
124         inicio1++;
125     }
126 }
127
128 Matriz::~Matriz()
129 {
130     for(int i=0; i<dimension;i++)
131         delete mat[i];
132     delete mat;
```

```
}
```