```cpp
#include <iostream>
using namespace std;
#define QTDNC 2
typedef struct {
 float real, imaginaria;
} numcomplex;
void leNumComplex(numcomplex *nc) {
 cout << endl << "Parte Real: ";
 cin >> nc->real;
 cout << endl << "Parte Imaginaria: ";
 cin >> nc->imaginaria;
}

void escreveNumComplex(numcomplex nc) {
 cout << endl << "Numero Complexo: ";
 if ((nc.real != 0) || (nc.imaginaria == 0)) {
 cout << nc.real;
 }
 if (nc.imaginaria != 0) {
 if ((nc.imaginaria > 0) && (nc.real != 0)) {
 cout << "+" << nc.imaginaria;
 } else {
 cout << nc.imaginaria;
 }
 cout << "i";
 }
 cout << endl;
}
//somando os numeros complexos
numcomplex somaNumComplex(numcomplex a, numcomplex b) {
    numcomplex aux_soma;

    aux_soma.real = a.real + b.real;
    aux_soma.imaginaria = a.imaginaria+ b.imaginaria;

    return aux_soma;
}
//produto dos numeros
numcomplex produtoNumComplex(numcomplex a, numcomplex b) {
    numcomplex aux_mult;

    aux_mult.real = a.real * b.real - a.imaginaria * b.
imaginaria;
    aux_mult.imaginaria = a.real * b.imaginaria + a.
imaginaria * b.real;

    return aux_mult;
}
//verificando a igualdade dos dois numeros
bool igualdadecomplex (numcomplex a, numcomplex b){
    if((a.real == b.real)&& (a.imaginaria == b.imaginaria
)){
        return true;
    }else{
        return false;
    }

}
//Oposto de apenas um número
numcomplex opostoNumComplex(numcomplex a){
    numcomplex aux_op;
    aux_op. real = a.real * -1;
    aux_op.imaginaria = a.imaginaria * -1;
    return aux_op;
}
//Conjugado
```

```
64  numcomplex conjugadoNumComplex(numcomplex a){
65      numcomplex aux_conj;
66      aux_conj.real = a.real;
67      aux_conj.imaginaria = a.imaginaria * -1;
68      return aux_conj;
69  }
70  int main()
71  {
72   int i;
73   numcomplex vetorNC[QTDNC];
74   numcomplex sNC, pNC, oPC, cNC;
75   bool resig;
76   for (i=0; i<QTDNC; i++) {
77   cout << endl << "====================";
78   cout << endl << i+1 << "o Numero Complexo";
79   cout << endl << "====================" << endl;
80   leNumComplex(&vetorNC[i]);
81   escreveNumComplex(vetorNC[i]);
82   }
83   cout << endl << "====================" << endl;
84   cout << endl << "====================";
85   cout << endl << "Soma";
86   cout << endl << "====================" << endl;
87   sNC = somaNumComplex(vetorNC[0], vetorNC[1]);
88   escreveNumComplex(sNC);
89   cout << endl << "====================" << endl;
90   cout << endl << "====================";
91   cout << endl << "Produto";
92   cout << endl << "====================" << endl;
93   pNC = produtoNumComplex(vetorNC[0], vetorNC[1]);
94   escreveNumComplex(pNC);
95
96   resig = igualdadecomplex (vetorNC[0], vetorNC[1]);
97   cout << endl << "====================" << endl;
98   cout << endl << "====================" << endl;
99   cout << endl << "Igualdade";
100  cout << endl << "====================" << endl;
101  if(resig == 1){
102     cout << "OS numeros complexos são iguais!";
103  }else{
104     cout <<"Sem igualdade";
105  }
106  cout << endl << "====================" << endl;
107  cout << endl << "====================";
108  cout << endl << "Oposto";
109  cout << endl << "====================" << endl;
110  oPC = opostoNumComplex (vetorNC[0]);
111  escreveNumComplex(oPC);
112
113  cout << endl << "====================" << endl;
114  cout << endl << "====================";
115  cout << endl << "Conjugado";
116  cout << endl << "====================" << endl;
117  cNC = conjugadoNumComplex (vetorNC[0]);
118  escreveNumComplex(cNC);
119
120  return 0;
121 }
```