

Matemáticas.

Material de referencia.

Índice

1. Big Numbers.	2
1.1. Implementación	2
2. Sistema de Ecuaciones Lineales.	5
2.1. Eliminación Gaussiana	5

1. Big Numbers.

1.1. Implementación

```
1  #include <iostream>
2  #include <algorithm>
3  #include <utility>
4  using namespace std;
5
6  typedef string BigInteger;
7
8  //Regresa el i-esimo dígito de derecha a izquierda de un número.
9  int digit(const BigInteger &num, int i) {
10     if (i < num.size())
11         return num[num.size() - 1 - i] - '0';
12     return 0;
13 }
14
15 //Compara dos números y regresa: 1 si el primero es mayor; 0 si son iguales;
16 // -1 si el segundo es mayor.
17 int compareTo(const BigInteger &a, const BigInteger &b) {
18     for (int i = max(a.size(), b.size()) - 1; i >= 0; --i) {
19         if (digit(a, i) > digit(b, i))
20             return 1;
21         if (digit(b, i) > digit(a, i))
22             return -1;
23     }
24     return 0;
25 }
26
27 //Regresa la suma de dos números.
28 BigInteger sum(const BigInteger &a, const BigInteger &b) {
29     BigInteger ans;
30     int carry = 0;
31     for (int i = 0; i < max(a.size(), b.size()); ++i) {
32         carry += digit(a, i) + digit(b, i);
33         ans.push_back((carry % 10) + '0');
34         carry /= 10;
35     }
36     if (carry)
37         ans.push_back(carry + '0');
38     reverse(ans.begin(), ans.end());
39     return ans;
40 }
41
42 //Regresa la diferencia de dos números. El primer número debe ser mayor o
43 // igual que el segundo.
44 BigInteger subtract(const BigInteger &a, const BigInteger &b) {
45     BigInteger ans;
46     int carry = 0;
47     for (int i = 0; i < a.size(); ++i) {
48         carry += digit(a, i) - digit(b, i);
49         if (carry >= 0) {
50             ans.push_back(carry + '0');
51             carry = 0;
52         }
53     }
```

```

51         else {
52             ans.push_back(carry + 10 + '0');
53             carry = -1;
54         }
55     }
56     while (ans.size() > 1 && ans.back() == '0')
57         ans.pop_back();
58     reverse(ans.begin(), ans.end());
59     return ans;
60 }
61
62 //Regresa el producto de dos numeros (BigInteger x int).
63 BigInteger multiply(const BigInteger &a, int b) {
64     if (b == 0)
65         return "0";
66     BigInteger ans;
67     int carry = 0;
68     for (int i = 0; i < a.size(); ++i) {
69         carry += digit(a, i) * b;
70         ans.push_back((carry % 10) + '0');
71         carry /= 10;
72     }
73     while (carry) {
74         ans.push_back((carry % 10) + '0');
75         carry /= 10;
76     }
77     reverse(ans.begin(), ans.end());
78     return ans;
79 }
80
81 //Regresa el producto de dos numeros (BigInteger x BigInteger).
82 BigInteger multiply(const BigInteger &a, const BigInteger &b) {
83     BigInteger ans;
84     for (int i = 0; i < b.size(); ++i)
85         ans = sum(ans, multiply(a, digit(b, i)).append(i, '0'));
86     return ans;
87 }
88
89 //Regresa el cociente y el residuo de la division (BigInteger / int).
90 pair<BigInteger, int> divide(const BigInteger &a, int b) {
91     pair<BigInteger, int> ans;
92     for (int i = a.size() - 1; i >= 0; --i) {
93         ans.second = 10*ans.second + digit(a, i);
94         if (!ans.first.empty() || ans.second >= b || i == 0)
95             ans.first.push_back((ans.second / b) + '0');
96         ans.second %= b;
97     }
98     return ans;
99 }
100
101 //Regresa el cociente y el residuo de la division (BigInteger / BigInteger).
102 pair<BigInteger, BigInteger> divide(const BigInteger &a, const BigInteger &b)
103 {
104     pair<BigInteger, BigInteger> ans;
105     BigInteger table[10];
106     for (int i = 0; i < 10; ++i)

```

```
106         table[i] = multiply(b, i);
107     for (int i = a.size() - 1; i >= 0; --i) {
108         int q = 0;
109         ans.second.push_back(digit(a, i) + '0');
110         while (q < 9 && compareTo(ans.second, table[q + 1]) >= 0)
111             ++q;
112         if (!ans.first.empty() || q > 0 || i == 0)
113             ans.first.push_back(q + '0');
114         ans.second = subtract(ans.second, table[q]);
115     }
116     return ans;
117 }
118
119 int main() {
120     BigInteger a, b;
121     cin >> a >> b;
122
123     cout << a << " + " << b << " = " << sum(a, b) << "\n";
124     if (compareTo(a, b) >= 0)
125         cout << a << " - " << b << " = " << subtract(a, b) << "\n";
126     else
127         cout << b << " - " << a << " = " << subtract(b, a) << "\n";
128     cout << a << " * " << b << " = " << multiply(a, b) << "\n";
129     cout << a << " = " << b << " * " << divide(a, b).first << " + " << divide(
        a, b).second << "\n";
130
131     return 0;
132 }
```

Entrada	Salida
1894821 589613	1894821 + 589613 = 2484434 1894821 - 589613 = 1305208 1894821 * 589613 = 1117211094273 1894821 = 589613 * 3 + 125982

2. Sistema de Ecuaciones Lineales.

Consideremos un sistema de ecuaciones lineales dado por la matriz \mathbf{A} de $n \times n$ y el vector \mathbf{b} de dimensión n . Decimos que \mathbf{x} es solución si $\mathbf{Ax} = \mathbf{b}$.

2.1. Eliminación Gaussiana

Complejidad $O(n^3)$.

```

1  #include <iostream>
2  #include <algorithm>
3  #include <cmath>
4  using namespace std;
5
6  #define maxn 100 //Maximo numero de ecuaciones-incognitas.
7
8  int n, m; //Dimensiones.
9  double AugMatrix[maxn][maxn]; //Matriz aumentada.
10
11 //Encuentra la forma escalonada reducida de la matriz aumentada [A | B]
12 //con A de n x n y B de n x m. Regresa el determinante de A.
13 double GaussianElimination() {
14     double det = 1;
15     for (int k = 0; k < n; ++k) {
16         int r = k;
17         for (int i = k + 1; i < n; ++i)
18             if (fabs(AugMatrix[i][k]) > fabs(AugMatrix[r][k]))
19                 r = i;
20
21         if (fabs(AugMatrix[r][k]) < 1e-9)
22             return 0;
23         if (r != k) {
24             for (int j = k; j < n + m; ++j)
25                 swap(AugMatrix[k][j], AugMatrix[r][j]);
26             det *= -1;
27         }
28         det *= AugMatrix[k][k];
29
30         for (int j = n + m - 1; j >= k; --j) {
31             AugMatrix[k][j] /= AugMatrix[k][k];
32             for (int i = 0; i < n; ++i)
33                 if (i != k)
34                     AugMatrix[i][j] -= AugMatrix[i][k] * AugMatrix[k][j];
35         }
36     }
37     return det;
38 }
39
40 int main() {
41     ios_base::sync_with_stdio(0); cin.tie();
42     cin >> n >> m;
43
44     for (int i = 0; i < n; ++i)
45         for (int j = 0; j < n + m; ++j)
46             cin >> AugMatrix[i][j];
47

```

```
48     cout << "Determinante: " << GaussianElimination() << "\nSolucion:\n";
49     for (int i = 0; i < n; ++i) {
50         for (int j = 0; j < m; ++j)
51             cout << AugMatrix[i][j + n] << ' ';
52         cout << '\n';
53     }
54     return 0;
55 }
```

Entrada	Salida
4 1 1 -2 2 -3 15 3 4 -1 1 -6 2 -3 2 -1 17 1 1 -3 -2 -7	Determinante: 142 Solucion: 2 -2 3 -1