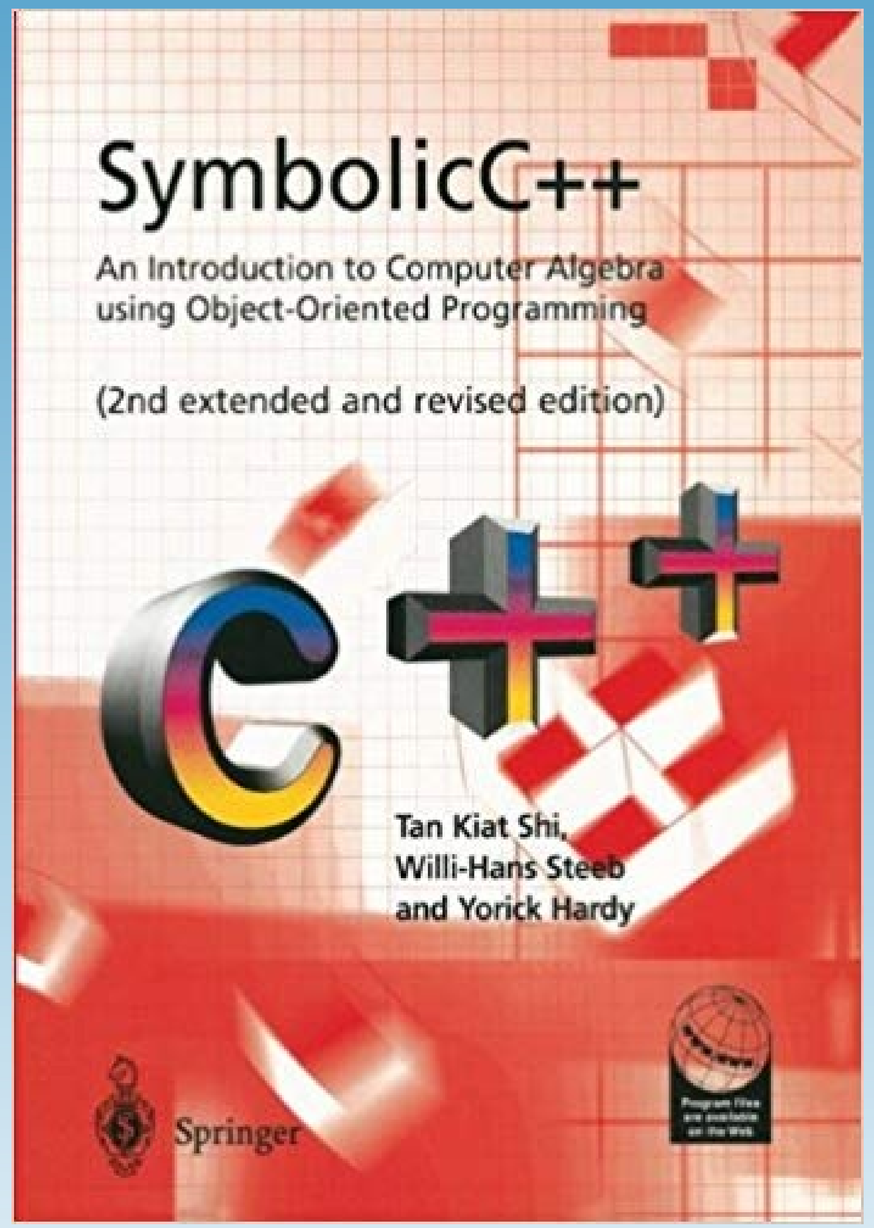
mini-cas

(Mais) Uma biblioteca para matemática simbólica em C++

Arthur Gonçalves do Carmo
Luiz Carlos de Abreu Albuquerque

A proposta



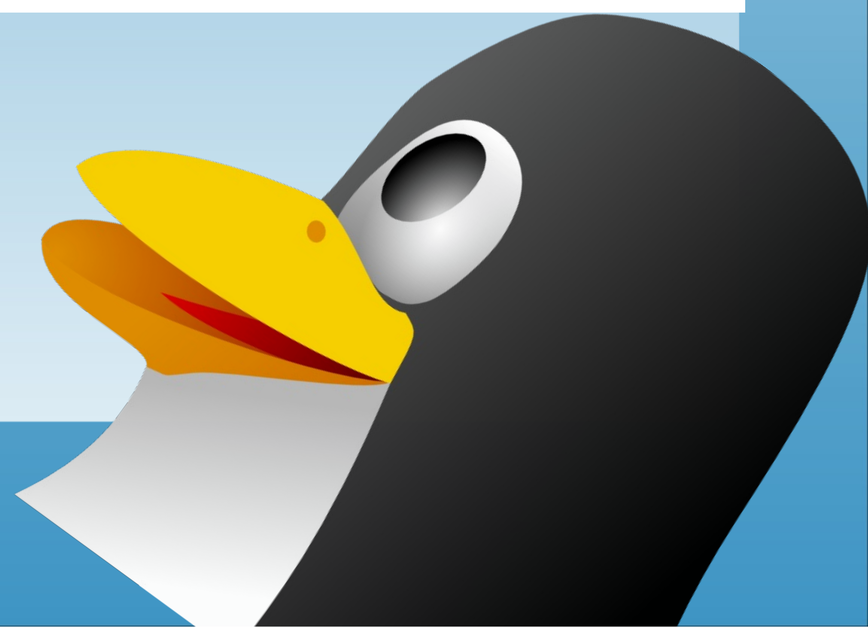
- Uma biblioteca para C++ que permita à linguagem realizar algumas formas de manipulação simbólica

Sumário

- O que já está feito
- Dificuldades até aqui
- O que ainda pode ser feito

```
#include<mini-cas>
```

```
Number **my_vector = new Number * [3];

my_vector[0] = new num_z("184467440737
my_vector[1] = new num_zm<7>(24);
my_vector[2] = new num_q(2, 6);
```



```
class Number{
```

```
virtual Number & value() = 0;
virtual num_z z_value() = 0;
virtual num_q q_value() = 0;
```

```
virtual Number & operator=(const num_q &) = 0;
virtual Number & operator+=(const num_q &) = 0;
```

Precisa de um namespace?

Talvez, mas não vai receber um por enquanto

A classe Number deve mesmo ser abstrata?

A princípio sim, a intenção é que seja uma classe invisível para o usuário

É difícil manter o código "limpo"

```
std::memcpy(this->_num
```

```
memcpy(this->_num,
```

```
operator bool(){return *this != 0;};
```

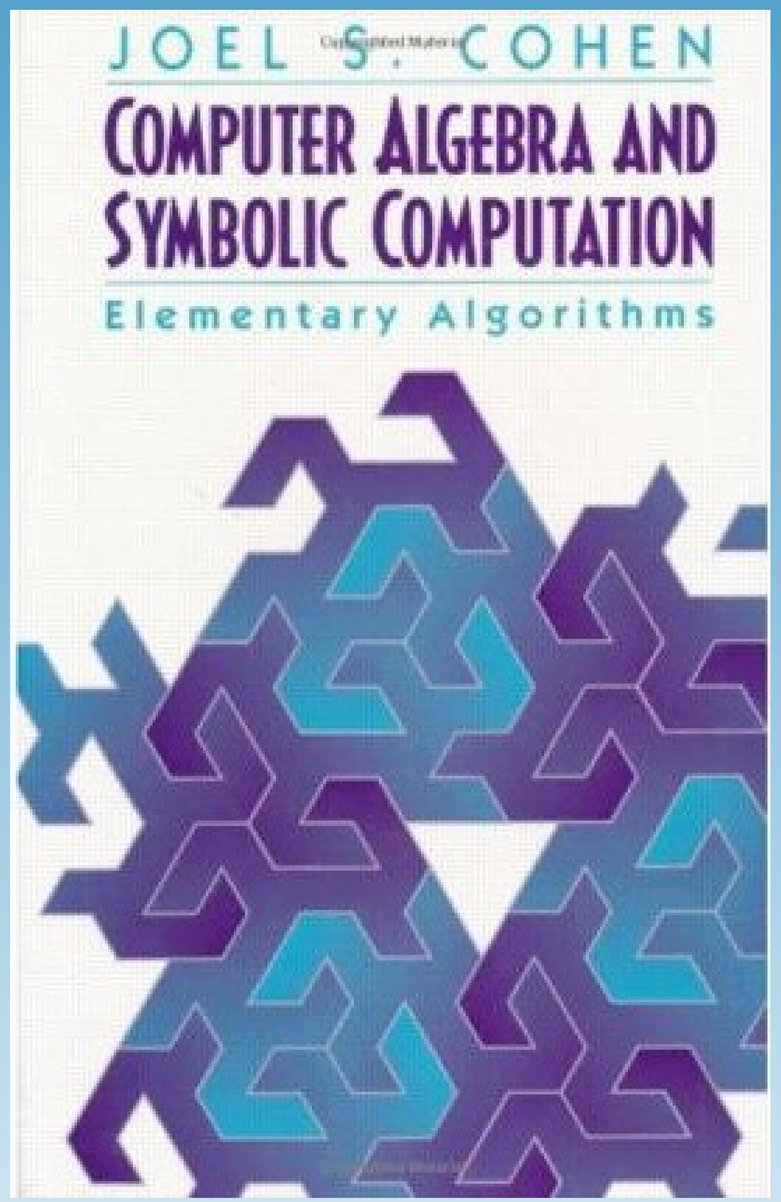
```
#define _CAS_TYPE_Z_  0
#define _CAS_TYPE_Q_  1
#define _CAS_TYPE_ZM_ 2
```

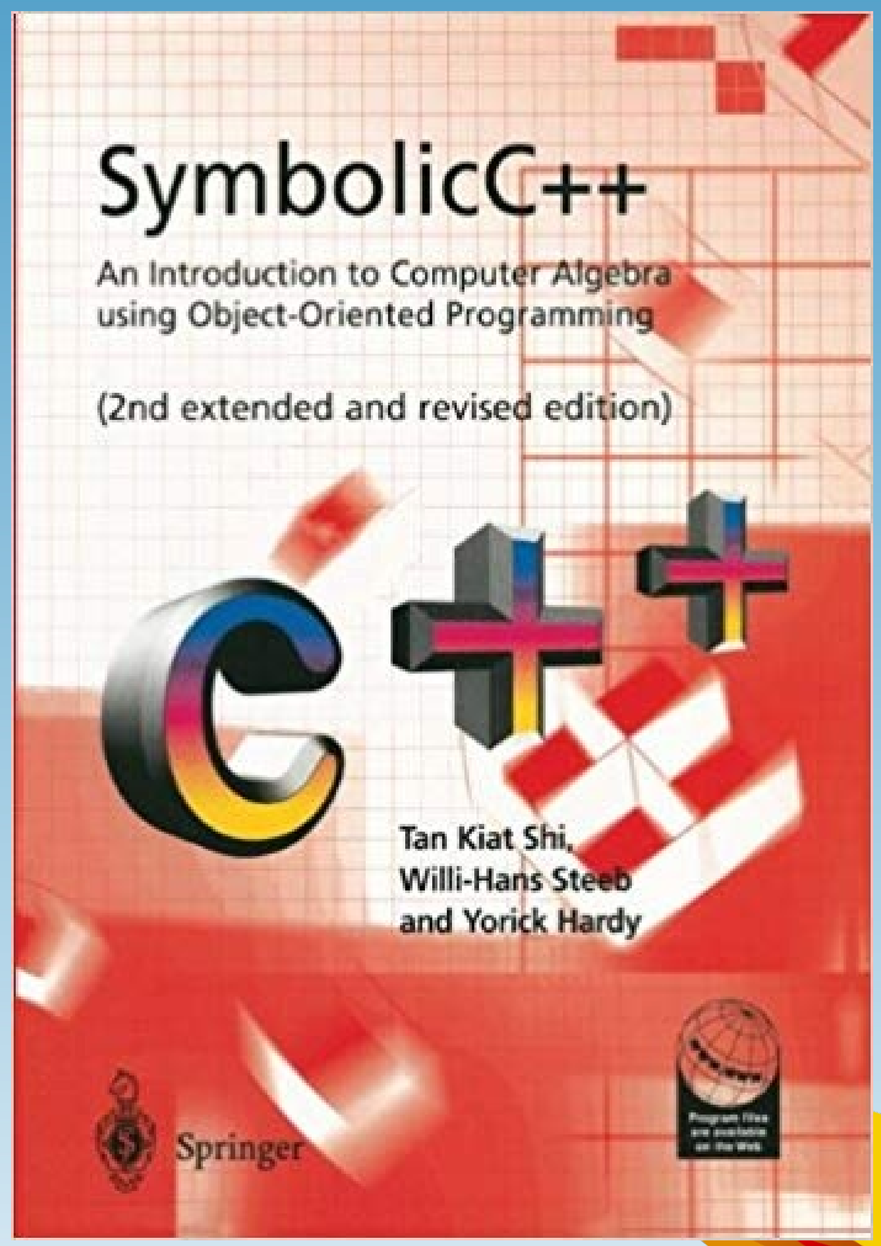
```
protected:
        int _type;
```

```
55          n_size = n._blocks - 1;
56          for(j = m._blocks - 1; j > n_size; --j){
57                      q_guess = (m._num[j] == n._num[n._blocks-1])?
  ((uint64_t)_MAX_DIGIT_BASE_):(((uint64_t)m._num[j]*(uint64_t)_BASE_ + m.
  n._num[n._blocks - 1]);
58
59                  while(((uint64_t)n._num[n._blocks - 2]*(uint64_t)q_guess
  ((uint64_t)m._num[j] * (uint64_t)_BASE_ - (uint64_t)n._num[n._blocks - 1
  (uint64_t)q_guess + m._num[j-1])* (uint64_t)_BASE_ + m._num[j-2])) --q_g
60
61                  //Multiplícar e subtrair
62                  parc_n = n * q_guess;
63                  parc_m._blocks = 1 + n._blocks;
64
65                  for(i = 0; i <= n._blocks; i++){
66                          parc_m._num[n._blocks - i] = m._num[j - i];
67                  }
68
69                  if(parc_m._num[parc_m._blocks - 1] == 0) parc_m._num[--p
  = 0;
70
71                  while(parc_n > parc_m){
```

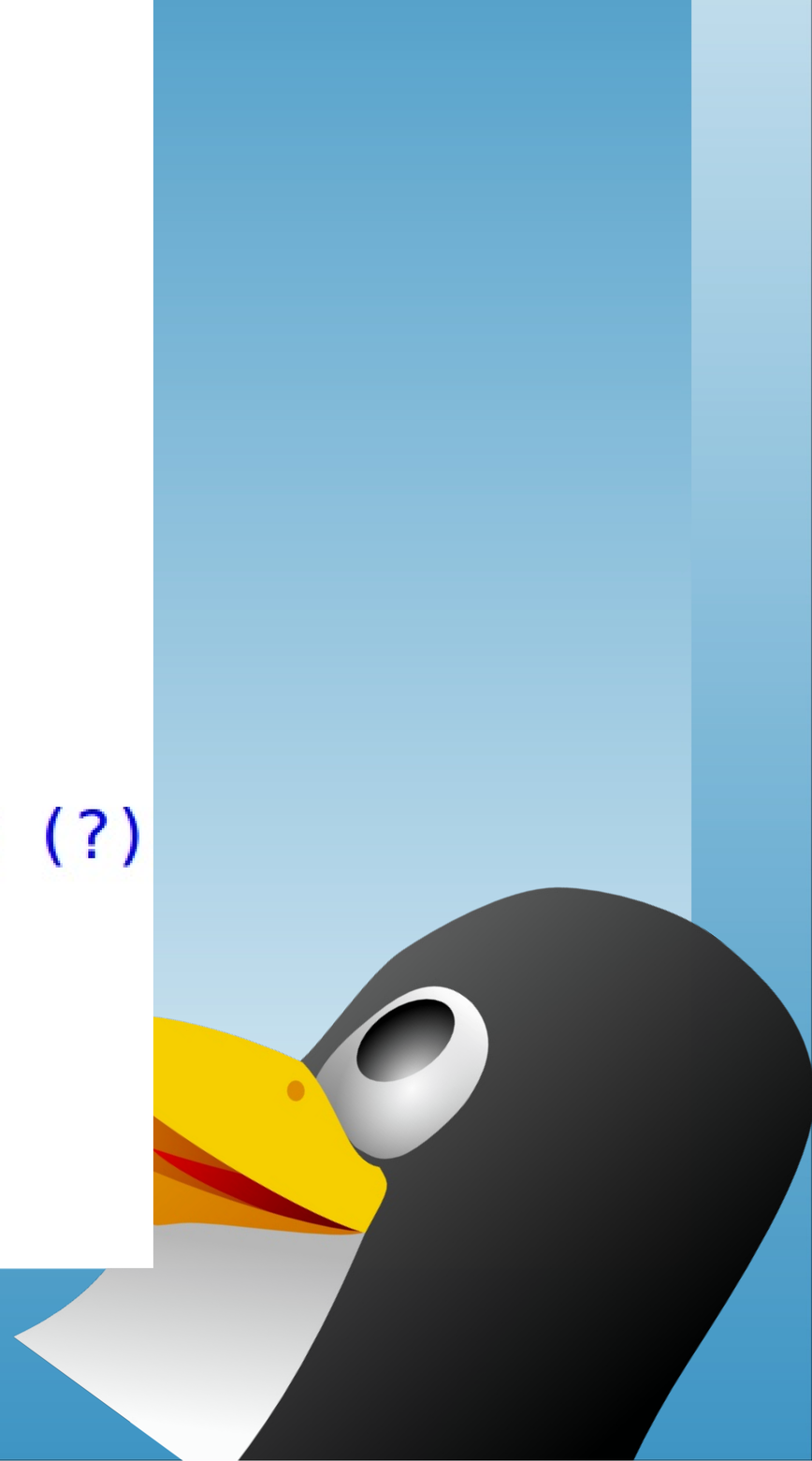
```
if(this->_n_blocks < maior)
        this->__resize(maior);
else
        memset(this->_num + this->_blocks, 0, (maior - this
        for(i = this->_blocks; i < maior; i++)
                this->_num[i] = 0;

this->_blocks = maior - 1;

for(i = 0; i < this->_blocks; i++){
        w = (this->_num[i] + n._num[i] + vai_um);
        this->_num[i] = w % _BASE_;
        vai_um = (w >= _BASE_);
}
```

Novas referências





```
 1 #ifndef __SIGNATURES_
 2 #define __SIGNATURES_
 3
 4 class Number;
 5 class num_z;
 6 class num_q;
 7 class q_vector;     //
 8 class q_matrix;     //    Talvez (?)
 9 class polynomial;   //
10 class symbol;
11
12 struct div_tuple;
13 struct mod_tuple;
14 struct monomial;    //    struct term (?)
15
16 template<int64_t N>
17 class num_zm;
18
19 #endif
```



O que já está pronto é utilizável!

Fiquem à vontade para baixar, testar e reportar bugs / comportamentos inesperados

Tarball mais recente no link

https://github.com/ArthurCarmo/mini-cas/blob/master/minicas-beta.tar.gz

Obrigado!

Contato: arthur.carmo@ufv.br

https://github.com/ArthurCarmo/mini-cas

Os livros que foram mostrados

(apesar de que dá pra ler bem nas imagens)

SymbolicC++ (2nd extended and revised edition)

Shi, T. K. Steeb, W. Hardy, Y.

Computer Algebra and Symbolic Computation: Elementary Algorithms

Cohen, J. S.

-----> *https://github.com/ArthurCarmo/mini-cas/blob/master/minicas-beta.tar.gz* <-----