Faculdade de Ciências da Universidade do Porto
Departamento de Matemática
Christian Lomp

Projeto **1**
entrega: 26.10.2011

Projeto de Álgebra Computacional (M342)

1. A classe inteiro

Objetivo: Definir uma classe em C++ que permita a aritmética de inteiros em precisão arbitrária ("arbitrary-precision arithmetic").

Especificação da classe: A classe <inteiro > deve representar inteiros na base $\mathbf{q} = 10^9$ e deve utilizar a classe <vector> para guardar os coeficientes da representação $\mathbf{q}$-ária.

A classe <inteiro> deve ter a seguinte estrutura:

```
class inteiro {

  bool negativo;
  vector <unsigned int> coeficientes;

  public:
    inteiro();
    inteiro(int n);
    inteiro(bool, vector<unsigned int>);
    bool operator < (inteiro);
    bool operator > (inteiro);
    bool operator == (inteiro);
    bool operator != (inteiro);
    inteiro operator + (inteiro);
    inteiro operator - (inteiro);
    inteiro operator * (inteiro);
    inteiro operator / (inteiro);
    inteiro operator % (inteiro);

    string ConvertToString();
    inteiro ConvertToInteiro(string);

}
```

Especificações das funções:

1. **O construtor:**

```
inteiro::inteiro() { ... }
```

Criação dum objecto *inteiro* que representa o número 0.

```
inteiro::inteiro(int n) { ... }
```

Criação dum objecto *inteiro* que representa o número n.

```
inteiro::inteiro(bool sinal, vector<unsigned int> coef) { ... }
```

Criação dum objecto *inteiro* que representa o número dado pela representação $(sinal, coef)$.

2. **Comparações:**

```
bool inteiro::operator == (inteiro b) { ... }
```

O valor de retorno é *true* se os inteiros representados pelo objeto atual e pelo objeto b forem iguais.

```
bool inteiro::operator != (inteiro b) { ... }
```

O valor de retorno é *true* se os inteiros representados pelo objeto atual e pelo objeto b não forem iguais.

```
bool inteiro::operator < (inteiro b) { ... }
```

O valor de retorno é *true* se o inteiros representado pelo objeto atual for menor do que o inteiro representado pelo objeto `b`.

```
bool inteiro::operator > (inteiro b) { ... }
```

O valor de retorno é *true* se o inteiros representado pelo objeto atual for maior do que o inteiro representado pelo objeto `b`.

3. **Aritmética:**

```
inteiro inteiro::operator + (inteiro b) { ... }
```

O valor de retorno é um objeto *inteiro* que representa a soma dos inteiros representados pelo objeto atual e pelo objeto `b`.

```
inteiro inteiro::operator - (inteiro b) { ... }
```

O valor de retorno é um objeto *inteiro* que representa a diferenca dos inteiros representados pelo objeto atual e pelo objeto `b`.

```
inteiro inteiro::operator * (inteiro b) { ... }
```

O valor de retorno é um objeto *inteiro* que representa a multiplicação dos inteiros representados pelo objeto atual e pelo objeto `b`.

```
inteiro inteiro::operator / (inteiro b) { ... }
```

O valor de retorno é um objeto *inteiro* que representa o quociente da divisão dos inteiros representados pelo objeto atual e pelo objeto `b`.

```
inteiro inteiro::operator % (inteiro b) { ... }
```

O valor de retorno é um objeto *inteiro* que representa o resto da divisão dos inteiros representados pelo objeto atual e pelo objeto `b`.

1. Problemas adicionais

Escreva uma função **inteiro factorial(inteiro)** que permite calcular o factorial de um inteiro. Calcue `100!` e `1000!`.

Escreva uma função **inteiro divisor(inteiro)** que encontre o menor divisor maior que `1` de um inteiro. Factorize o número `10000000001` em números primos.

1. Recurso Auxiliar

Podem utilisar as seguintes funções que permitem a conversação dos coeficientes entre o formato *vectores<unsigned int>* e o formato de texto *string*.

Para utilizar as funções precisa de incluir as bibliotecas <vector>, <string>, <cstdlib> e <sstream>.

```
#include <vector>
#include <string>
#include <cstdlib>
#include <sstream>
```

BASE e BASE_LEN são constantes que são preciso para a conversação.

```
#define BASE 1000000000
#define BASE_LEN 9
```

```
string ConvertVectorToString(vector<unsigned int> v)
{
        int n=v.size();
        ostringstream vector_string;
        for(int i=n-1;i>=0;i--)
        {
          unsigned int z=v[i];
          if(i<n-1)
          {
            int s=BASE/10;
            while(z%s == z)
            {
                vector_string << "0";
                s=s/10;
            };
          };
          vector_string << z;
        };
        return vector_string.str();
};

vector<unsigned int> ConvertStringToVector(string s)
{
        vector<unsigned int> v;
        int n=s.length();
        int q=n/BASE_LEN;
        int r=n%BASE_LEN;
        string sub;
        for(int i=q-1;i>=0;i--)
        {
                sub=s.substr(r+i*BASE_LEN,BASE_LEN);
                v.push_back(atoi(sub.c_str()));
        };
        if(r>0)
        {
                sub=s.substr(0,r);
                v.push_back(atoi(sub.c_str()));
        };
        return v;
};
```