

```
/*
 * GF.h
 *
 * Created on: Dec 28, 2009
 * Author: koscum
 */

#ifndef GF_H_
#define GF_H_

#include "Error.h"
#include "Polynomial.h"
#include <iostream>
#include <vector>
#include <stdlib.h>

namespace GF
{
    class Error;
    class Polynomial;

    struct Inverse
    {
        Inverse();

        int add;
        int mul;
    };

    class GF
    {
    public:
        GF(int);
        GF(int, int);
        GF(int, const Polynomial&);
        GF(int, int, const Polynomial&);
        GF(const GF&);
        GF(const GF&, int);
        GF(const GF&, const Polynomial&);
        GF(const GF&, int, const Polynomial&);
        virtual ~GF();

        Polynomial& Add(const Polynomial& a, const Polynomial& b) const;
        Polynomial& Sub(const Polynomial& a, const Polynomial& b) const;
        Polynomial& Mul(const Polynomial& a, const Polynomial& b) const;
        Polynomial& Div(const Polynomial& a, const Polynomial& b) const;
        Polynomial& Mod(const Polynomial& a, const Polynomial& b) const;
        Polynomial& Pow(const Polynomial& a, const Polynomial& b) const;

        int GetCharasteristic() const;
        int GetBase() const;
        int GetPower() const;

        GF& GetBaseField() const;

        const Polynomial& GetAlpha() const;
    };
}
```

```

const Polynomial& GetIrreduciblePolynomial() const;
const Polynomial& GetElement(int) const;
const Polynomial& operator [](int) const;

bool IsBase() const;
bool IsExtended() const;
bool IsPrimitive() const;

static bool IsIrreducible(const Polynomial&, int base, int power);
static bool IsPrime(int);
static bool IsPrimePower(int);
static int BaseOf(int);
static int PowerOf(int);
static const Polynomial& FindIrreduciblePolynomial(int base, int power);

protected:
    void Initialize(int base, int power, const Polynomial& irreduciblePolynomial);
    void Free();
    void Copy(const GF&);
    void Out(std::ostream&);

private:
    int GetIndexOf(const Polynomial&) const;
    const Polynomial& GetAddInverse(int) const;
    const Polynomial& GetMulInverse(int) const;

    void GenerateInversionsTable();
    void FindAlpha();
    void GenerateAlphaTable();

    static std::vector<Polynomial>& GenerateAllPolynomials(int base, int powerLimit);

    friend std::ostream& operator <<(std::ostream&, const GF&);

private:
    int base;
    int power;
    int alpha;
    Polynomial irreduciblePolynomial;
    std::vector<int> alphaTable;
    std::vector<Inverse> inversionsTable;
    std::vector<Polynomial> elements;
};
}

#endif /* GF_H_ */

```