Master 1 Master de Mathématiques et Applications Cours en commun : Université Paris VII et Université VIII
(Parcours ACC et CSSD)
Programmation en cryptographie S. Mesnager

Corps des entiers de Gauss

1. PRÉLIMINAIRES

Nous allons écrire les primitives permettant de faire des calculs algébriques sur le corps des entiers de Gauss :

$$\mathbb{Q}(\sqrt{5}) := \left\{ \frac{a+b\cdot\sqrt{5}}{d},\ (a,b,d)\in\mathbb{Q},\ d\neq 0 \right\}$$

Un entier de Gauss sera representé en `C` par la structure `GaussType` définie par :

```
typedef struct
          {
              long int a,b,d;
          } GaussType;
```

Pour définir un entier de Gauss x, il suffit d'écrire

```
GaussType x;
```

Pour affecter une valeur à x, par exemple $x = \frac{-1+8\sqrt{5}}{2}$, il existe trois fas de procéder soit par les affectations

```
x.a=-1;
x.b= 8;
x.d= 2;
```

soit lors de la déclaration de x

```
GaussType x={-1,8,2};
```

soit en affectant à x le contenu d'une variable y de type `GaussType` déjà affectée

```
GaussType y={-1,8,2}

x=y;
```

Pour afficher les entiers de Gauss, on utilisera la procédure `GaussAffiche` suivante :

```
// affichage
// si cr!=0, passe \`a la ligne, sinon, affiche un blanc

void GaussAffiche(GaussType x,int cr)
{
   printf("(%d %s %d V5) / %d%s",
     x.a,
     x.b<0?"-":"+", // on affiche a - b V5 si b est n\'egatif
     x.b<0?-x.b:x.b,
     x.d,
     cr?"\n":" ");
}
```

On appellera forme réduite d'un entier de Gauss x son écriture sous la forme $x = \frac{a+b\sqrt{5}}{d}$ avec $d > 0$ et $\text{pgcd}\,(\text{pgcd}(a,b),d) = 1$ (c'est-à-dire a, b et d n'ont pas de diviseur commun). Pour déterminer le plus grand commun diviseur de deux entiers, on utilisera l'algorithme d'Euclide :

```
// Algorithme d'Euclide
long int pgcd(long int a, long int b)
{
   // valeurs absolues de x et y
   if (a<0) a=-a;
   if (b<0) b=-b;
   if (b==0) return a; // cette fonction rend donc 0 si a et b sont
   // tous les deux nuls. Il suffit de le savoir.
   while (a)
   {
      b%=a;
      if (b==0) return a;
      a%=b;
   }
   return b;
}
```

Exercice (sur machine) 1.
Ecrire la procédure de réduction d'un entier de Gauss. La signature de la fonction est

```
void Reduire(GaussType *x)
```

Une fois écrite la proécdure, éxecuter le programme suivant :

```
#include <stdio.h>
#include <stdlib.h>

typedef struct
           {
               long int a,b,d;
           } GaussType;

// affichage
// si cr!=0, passe \'a la ligne, sinon, affiche un blanc

void GaussAffiche(GaussType x,int cr)
{
  ...
}

// Algorithme d'Euclide
long int pgcd(long int a, long int b)
{
  ...
}

void Reduire(GaussType *x)
{
  ...
```

```
}

main()
{
  GaussType x={9,-72,-45};

  GaussAffiche(x,1);
  Reduire(&x);
  GaussAffiche(x,1);

}
```

2. Opérations sur les entiers de Gauss

Exercice (sur machine) 2.
On souhaite écrire des procédures permettant d'additionner et de multiplier deux entiers de Gauss.

(1) Soient $x = \frac{a+b\sqrt{5}}{d}$ et $y = \frac{e+f\sqrt{5}}{g}$ deux entiers de Gauss. On pose $s = x + y$ et $p = xy$. Ecrire les expressions algébriques de s et p sous la forme $\frac{u+v\sqrt{5}}{w}$.

(2) Ecrire deux procédures `C` réalisant l'addition et la multiplication de deux entiers de Gauss. Les signatures de ces procédures sont

```
void GaussAdd(GaussType *s,GaussType x,GaussType y)
{
...
}

void GaussMul(GaussType *p,GaussType x,GaussType y)
{
...
}
```

(3) Tester les procédures `GaussAdd` et `GaussMul` en exécutant le programme principal suivant :

```
main()
{
  GaussType x={-1,4,45};
  GaussType y={-6,7,100};

  GaussType s,p;

  GaussAdd(&s,x,y);
  GaussMul(&p,x,y);

  printf("Somme = ");
  GaussAffiche(s,1);
  printf("Produit = ");
  GaussAffiche(p,1);
}
```

Exercice (sur machine) 3.
On souhaite maintenant écrire deux procédures qui déterminent, respectivement, l'opposé et l'inverse d'un entier de Gauss.

(1) Soit $x = \frac{a+b\sqrt{5}}{d}$. On pose $o = -x$ et $i = \frac{1}{x}$. Ecrire les expressions algébriques de o et i sous la forme $\frac{u+v\sqrt{5}}{w}$.

(2) Ecrire deux procédures `C` déterminant l'opposé et l'inverse d'un entier de Gauss. Les signatures de ces procédures sont

```
void GaussOpp(GaussType *o,GaussType x)
{
...
}

void GaussInv(GaussType *i,GaussType x)
{
...
}
```

(3) Tester les procédures `GaussOpp` et `GaussInv` en exécutant le programme principal suivant :

```
main()
{
  GaussType x={-1,4,45};

  GaussType o,i,r;

  GaussOpp(&o,x);
  printf("Oppose = ");
  GaussAffiche(o,1);
  printf("Verification : ");
  GaussAffiche(x,0);
  printf(" + ");
  GaussAffiche(o,0);
  GaussAdd(&r,x,o);
  printf(" = ");
  GaussAffiche(r,1);

  GaussInv(&i,x);
  printf("Inverse = ");
  GaussAffiche(i,1);
  printf("Verification : ");
  GaussAffiche(x,0);
  printf(" x  ");
  GaussAffiche(i,0);
  GaussMul(&r,x,i);
  printf(" = ");
  GaussAffiche(r,1);
}
```

Exercice (sur machine) 4.

Ecrire deux procédures `GaussSoustrait` et `GaussDivise` qui calcule, respectivement, la différence et le quotient de deux entiers de Gauss.

Exercice (sur machine) 5.
Ecrire une procedure `GaussPuiss` calculant la puissance d'un entier de Gauss. La signature de la procédure est

```
void GaussPuiss(GaussType *r,GaussType x,unsigned int n)
```

Pour calculer la puissance, on utilisera l'algorithme suivant (écrit en meta-langage)

```
Y = X ;
Produit = 1 ;
Expo = n ;

Faire Tant que Expo not= 0
 If Expo mod 2 = 1 Then Produit = Produit*Y;
Y = Y^2
Expo = Expo / 2 ;
Fin de la boucle  "tant que"
```