

# Mathematics and Cryptography

Jean-Sébastien Coron

Université du Luxembourg

October 12, 2009

- C programming
  - Structures
- Algorithmic number theory.
  - Computing with large integers.

- Structures in C enable to group data of different type.
- Example 1: informations about someone
  - First name, last name, age.
- Example 2: a point in a plane
  - x and y coordinates.
- Exemple 3: a circle
  - Center and radius

- The `struct` keyword :

- ```
struct point {  
    float x;  
    float y;  
};
```

- This defines a new type: `struct point`.

- Each variable of this type has two fields :

- `x` of type `float`
  - `y` of type `float`

# Using structures

- To define a variable  $p$  with this new type :
  - `struct point p;`
- We access the  $x$  and  $y$  fields with `p.x` and `p.y`

```
struct point {  
    float x;  
    float y;  
};  
struct point p;  
p.x=2;  
p.y=3;  
printf( "%f\n" ,p.x );
```

- Replacing `struct point` by something shorter :
  - `typedef struct point Point2d;`
  - `Point2d p;` instead of `struct point p;`
- Or directly :

```
typedef struct {  
    float x;  
    float y;  
} Point2d;  
Point2d p;  
p.x=2;
```

- The new type can be used as any other type :

```
Point2d milieu(Point2d p1,Point2d p2)
{
    Point2d m;
    m.x=(p1.x+p2.x)/2;
    m.y=(p1.y+p2.y)/2;
    return m;
}
```

- The function takes as input two parameters of type `Point2d` and returns a `Point2d`.

# Assignment and comparison

- Assignment :
  - One can copy a `struct` variable into another, as with any other type :
  - ```
Point2d p1,p2;  
p1.x=3;p1.y=4;  
p2=p1; // copy p1 into p2.
```
- Comparison:
  - One can not compare two `struct` variables with `if (p1==p2)`
  - One must compare each field separately.



## Other example

- Function taking as input two points and outputting the distance between them.
  - For two points  $(x_1, y_1)$ ,  $(x_2, y_2)$ , their distance is :

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- ```
float distance(Point2d p1, Point2d p2)
{
    float dx, dy;
    dx=p2.x-p1.x;
    dy=p2.y-p1.y;
    return sqrt(dx*dx+dy*dy);
}
```

- To print a `struct` variable, one must print each of its field.

```
void affiche(Point p)
{
    printf("Coordonnée x:%f\n",p.x);
    printf("Coordonnée y:%f\n",p.y);
}
```

# Another example

- New type with first name, last name and age :

```
typedef struct {  
    char *nom;  
    char *prenom;  
    int age;  
} Personne;
```

- The new type `Personne` contains three fields :
  - Two strings `nom` and `prenom`
  - One `int` named `age`

- Printing a `Personne` variable :

```
void affiche(Personne p)
{
    printf("nom: %s, ", p.nom);
    printf("prenom: %s, ", p.prenom);
    printf("age: %d\n", p.age);
}
```

# Main program

```
int main()
{
    Personne a;
    a.nom=(char *) strdup("Dupond");
    a.prenom=(char *) strdup("Jean");
    a.age=25;
    affiche(a);
}
```

- strdup

- Allocates memory and copy the string given as input.

# Computing with large integers

- Limited precision in C :
  - `int`: 32 bits. Computing with values  $< 2^{32}$ .
- Computing with large integers :
  - One represents the big integers in base  $B$  in an array.
  - One implements addition, multiplication, division on big integers.
  - Existing libraries :
    - GMP: [www.swox.com/gmp](http://www.swox.com/gmp)
    - NTL: [www.shoup.net](http://www.shoup.net)
    - Some parts written in assembly for better efficiency.

- Representing large integers :
  - An integer is represented as an array of digits in base  $B$ , with a sign bit.

$$a = \pm \sum_{i=0}^{k-1} a_i B^i = \pm (a_{k-1} \dots a_0)_B$$

with  $0 \leq a_i < B$ . If  $a \neq 0$ , we must have  $a_{k-1} \neq 0$ .

- Basis :
  - One generally takes  $B = 2^v$  for some  $v$ .
  - One can also take  $B = 10$ .

- Computing  $c = a + b$  with  $a, b > 0$ 
  - Let  $a = (a_{k-1} \dots a_0)$  and  $b = (b_{\ell-1} \dots b_0)$  with  $k \geq \ell \geq 1$ .  
Ket  $c = (c_k c_{k-1} \dots c_0)$

$carry \leftarrow 0$

for  $i = 0$  to  $\ell - 1$  do

$tmp \leftarrow a_i + b_i + carry$

$carry \leftarrow tmp / B; c_i \leftarrow tmp \bmod B$

for  $i = \ell$  to  $k - 1$  do

$tmp \leftarrow a_i + carry$

$carry \leftarrow tmp / B; c_i \leftarrow tmp \bmod B$

$c_k \leftarrow carry$



- Computing  $c = a - b$  with  $a, b > 0$ .
  - $a_i + b_j$  is replaced by  $a_i - b_j$ .

- Computing  $c = a \cdot b$  with  $a, b > 0$ 
  - Let  $a = (a_{k-1} \dots a_0)$  and  $b = (b_{\ell-1} \dots b_0)$  avec  $k, \ell \geq 1$ . Let  $c = (c_{k+\ell-1} \dots c_0)$

*carry*  $\leftarrow 0$

for  $i = 0$  to  $k + \ell - 1$  do

*c*<sub>*i*</sub>  $\leftarrow 0$

for  $i = 0$  to  $k - 1$  do

*carry*  $\leftarrow 0$

    for  $j = 0$  to  $\ell - 1$  do

*tmp*  $\leftarrow a_i \cdot b_j + c_{i+j} + \textit{carry}$

*carry*  $\leftarrow \textit{tmp} / B$ ; *c*<sub>*i+j*</sub>  $\leftarrow \textit{tmp} \bmod B$

*c*<sub>*i+\ell*</sub>  $\leftarrow \textit{carry}$