Algorithms for Numbers and Public-Key Cryptography

Jean-Sébastien Coron

Université du Luxembourg

February 25, 2011

Course Summary

- Algorithms for numbers
  - Describe basic algorithms for dealing with numbers
  - Implement them on a computer
- Public-key cryptography
  - Describe the basic public-key algorithms
  - Implement them on a computer

Course 1

- Basics of C programming
  - This is to ensure that everybody has the same minimal background in programming.
  - However you can choose any other language.
    - Python, with the Sage Library.
- Number theory.
  - GCD
  - Euclid's algorithm

C programming

- Bases of C programming.
  - Structure of a C program.
  - Variables and types.
  - Printing.
  - Control structures: if and while.
  - For loop.
  - Arrays
  - `argc` and `argv`

Structure of a C program

```
#include <stdio.h>
#define A 10
int main()
{
  printf("Hello world \n");
  printf("A=%d\n",A);
}
```

- #include: include libraries
- #define: definition of constants.
- int main(): definition of main function.

Variables

- A program can store variables in memory.
- One must declare a variable before using it.
  - `int a;` declaration of variable a as integer.
- Integer variables
  - `short`: 16 bits ± 32767.
  - `int`: 16 or 32 bits ± 32767 or $\pm 2.10^9$.
  - `long`: 32 bits $\pm 2.10^9$.
- `unsigned short`, `unsigned int`, `unsigned long` $\rightarrow$ non-negative integers.

Floats

- Encoding
  - Mantissa: m.
  - Exponent: e.
  - $m * 2^e$.
- `float`: 24+8 bits.
  - $< 10^{38}$.
- `double`: 53+11 bits.
  - $< 10^{308}$.
- `long double`: 64+16 bits.
  - $< 10^{4932}$.

Operations on variables

- Assignation:
  - `a=b;`
  - the content of variable b is copied in variable a.
- Arithmetic operations:
  - $a + b$: addition.
  - $a - b$: subtraction.
  - $a * b$: multiplication.
  - a/b: division.
    - Euclidean division for integers :
  - $a\%b$: remainder.

Example

- Incrementation of a variable :
  - `i=i+1;`
- Circumference of a circle with radius in variable `float r` :
  - ```
    float c;
    c=2*3.14*r;
    ```
- Average of variables `x` and `y`:
  - ```
    float x,y,m;
    m=(x+y)/2;
    ```

Initialization of variables

- When a variable has been declared, its value is arbitrary.
- One can initialize it simultaneously :

```
#include <stdio.h>
int u=3;
int main()
{
  int a=2;
  printf("a=%d,u=%d\n",a,u);
}
```

Printing variables

- `printf` can print text and variable value on the standard output.
  - %d for an `int` or `long`.
  - %f for an `float` or `double`.

```
float a=2.3;
int b=4;
printf("a=%f,b=%d\n",a,b);
```

Input

- `scanf` enables to read a variable value from keyboard.

```
float a;
int b;
printf("Give a float:");
scanf("%f",&a);
printf("Give an integer:");
scanf("%d",&b);
```

Example

- Computing the circumference and area of a disk :

```
#include <stdio.h>
int main()
{
  float x;
  scanf("%f",&x);
  float pi=3.1415926;
  float c=2*pi*x;
  float a=pi*x*x;
  printf("circonference=%f\n",c);
  printf("aire=%f\n",a);
}
```

Conditions

- `if then else`

```
if (test)
{
   instructions if true
}
else
{
   instructions if false
}
```

- `else {...}` is optional.

Tests

- Possibles tests :
  - Equality: `a==b`
  - Non-equality: `a!=b`
  - Comparison: `a<b`
  - Comparison: `a<=b`
- Operations on tests :
  - Negation: `!(test)`.
  - And: `((test1) && (test2))`
  - Or: `((test1) || (test2))`

Example

- Ask for two integers and print them in increasing order :

```
#include <stdio.h>
int main()
{
  int a,b;
  printf("entrez deux entiers:\n");
  scanf("%d%d",&a,&b);
  if(a<b)
  {
    printf("%d %d\n",a,b);
  }
  else
  {
    printf("%d %d\n",b,a);
  }
}
```

While

- Repeat `instruction` while `test` is true.

```
while (test)
{
   instruction
}
```

- Example: determine the bit-size of a :

```
unsigned int a; int t=0;
while(a>0)
{
  a=a/2;
  t=t+1;
}
```

For loop

- Repeat the same instruction many times with a counter.
- Syntax: `for(`$<$`init`$>$`;`$<$`test`$>$`;`$<$`counter`$>$`)`
- Example: print integers from 1 to 10.
  - `for (i=1;i<=10;i++) printf("%d\n",i);`
- $<$`init`$>$: initialize counter.
- $<$`test`$>$: test counter.
- $<$`counter`$>$: increment counter.

Example

- Compute 2^n given n :

```
int c=1;
int i;
for(i=0;i<n;i++)
{
  c=c*2;
}
// c contains 2^n.
```

Arrays in C

- Arrays can store a group of variables of the same type.
  - For example:

```
int notes[5]; // array of 5 integers
notes[0]=15; // first entry
notes[1]=8;
notes[2]=16;
notes[3]=17;
notes[4]=9;// 5th entry
```

Arrays type

- Arrays type:
  - `float tabf[5]`: array of 5 `float`.
  - `double tabd[10]`: array of 10 `double`.
  - `int tabi[7]`: array of 7 `int`.
- Index:
  - An array of n elements is indexed from 0 to $n - 1$:
  - `int tabi[7]`.
    - From `tab[0]` to `tab[6]`.

Constant size

- An array must be of constant size.
  - This size must be written in the program, for example `int tab[10]`
  - `#define`:

```
#include <stdio.h>
#define N 10   // one defines N=10
int main()
{
  int tab[N];
  int autretab[5];
}
```

Characters

- Stored in a byte (8 bits).
  - ASCII encoding:
  - `'A'` $\longrightarrow$ 65, `'B'` $\longrightarrow$ 66,...
  - `'0'` $\longrightarrow$ 48,...
- Printing a character:

```
char x;
x='A';
printf("%c",x);
```

Strings

- A string is an array of characters.
  - `char ch[10]="hello";` creates an array of characters such that :
  - `ch[0]='h'`, `ch[1]='e'`, `ch[2]='l'`, `ch[3]='l'`, `ch[4]='o'`
  - `ch[5]='\0'` is the last character.
  - The others elements are not initialized.
- Printing a string :
  - `printf("%s",ch);`

Initialization of an array

- Using `for`:

```
#define N 10
int main()
{
  int tab[N];
  int i;
  for(i=0;i<N;i++)
  {
    tab[i]=0;
  }
}
```

Example

- Factorial using array :
  - $n! = n \cdot (n-1) \cdots 2 \cdot 1$

```
#define N 10
int main()
{
  int fac[N];
  int i;
  fac[0]=1;
  for(i=1;i<N;i++)
  {
    fac[i]=fac[i-1]*i;
  }
}
```

2-dimensional arrays

- One can declare arrays with two dimensions or more :
  - `int tab[4][3];` declares an array of size 4*3.
- Initialization :

```
#define M 10
#define N 5
int main()
{
  int tab[M][N];
  int i,j;
  for(i=0;i<M;i++)
    for(j=0;j<N;j++)
      tab[i][j]=0;
}
```

Command-line arguments

- Obtaining command-line arguments :
  - One would like to be able to write :
    ```
    $ fact 5
    120
    ```
- Advantage :
  - No need to write `int n=5` in the code (then code needs to be recompiled each time `n` is changed).
  - Avoid a `scanf`.

argc and argv

- Command-line arguments are stored in array `argv`.
- `argc` contains the number of arguments (size of `argv`).

```
#include <stdio.h>
int main(int argc,char *argv[])
{
  int i;
  for(i=0;i<argc;i++)
  {
    printf("%s\n",argv[i]);
      // print each argv[i] word
  }
}
```

Using `argc` and `argv`

- If the previous program is named `affiche`, then :
  - `$ affiche hello world 2`
    ```
    affiche
    hello
    world
    2
    ```
- Here `argc=4`.

Converting a string to an integer

- `int atoi()` enables to convert a string to an integer.
  - Example : print the square of an integer.

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc,char *argv[])
{
  int a=atoi(argv[1]); // conversion
  printf("%d\n",a*a);
}
```

  - `$ carre 3`
    `9`

GCD

- Common divisor :
  - Let a, b be two integers. A common divisor of a and b is an integer m such that $m|a$ and $m|b$.
- GCD.
  - GCD of two integers a and b is the greatest common divisor of a and b.
  - If $d = \mathrm{GCD}(a, b)$, then for all m such that $m|a$ and $m|b$, we have $m|d$.
- Example
  - $\mathrm{GCD}(9, 6) = 3$
  - $\mathrm{GCD}(7, 5) = 1$.

Euclid's algorithm

- Euclid's algorithm :
  - Input: a, b.
  - Let $r_0 = a$ and $r_1 = b$.
  - For $i \geq 0$, one defines the sequence (r_i) and (q_i) such that :

$$r_i = q_i \cdot r_{i+1} + r_{i+2}$$

    where q_i and r_{i+2} are the quotient and remainder of the division of r_i by r_{i+1}
  - The exists $k > 0$ such that $r_k = 0$.
  - Then $\mathrm{GCD}(a, b) = r_{k-1}$.

Proof

- Let $a > 0$ and $b \geq 0$.
  - If $b = 0$, then $\mathrm{GCD}(a, b) = \mathrm{GCD}(a, 0) = a$
  - Otherwise, let $a = b \cdot q + r$ with $0 \leq r < b$.
  - Then $\mathrm{GCD}(a, b) = \mathrm{GCD}(b, r)$.
  - (b, r) is less than (a, b).
- $\mathrm{GCD}(a, b) = \mathrm{GCD}(b, r)$
  - If $d|a$ and $d|b$, then $d|r$, and then $d|\mathrm{GCD}(b, r)$. Then $\mathrm{GCD}(a, b)|\mathrm{GCD}(b, r)$.
  - If $d'|b$ and $d'|r$, then $d'|a$, and then $d'|\mathrm{GCD}(a, b)$. Then $\mathrm{GCD}(b, r)|\mathrm{GCD}(a, b)$.
  - Then $\mathrm{GCD}(a, b) = \mathrm{GCD}(b, r)$.