Theoretical foundations

Introduction to Algorithmic Number Theory

Jean-Sébastien Coron

Université du Luxembourg

September 26, 2009

Course summary

- Introduction to algorithmic number theory.
  - Part of the Theoretical Foundations course.
  - Professor: Jean-Sebastien Coron
- Goal:
  - Recall and supply the required mathematical background.
  - Necessary for the other courses of the curriculum.
    - Cryptography
    - Information Theory and Coding

C programming

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

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.
```