Computing

A difficult introduction

Dr D P Jones

The Cedars School

October 2013

https://github.com/TheCedarsSchool/computing

Language of the machine

Dennis Ritchie (1941 - 2011); creator of C

Overview

Counting

Definition

A natural number is any integer (whole number) greater than or equal to zero

Counting

Definition

A natural number is any integer (whole number) greater than or equal to zero

$$c \geq 0$$

Counting

Definition

A natural number is any integer (whole number) greater than or equal to zero

$$c \geq 0$$

Example

- 0
- 145
- 68370912

But knowing when to stop

Definition

A finite natural number is any natural number less than or equal to some maximum value

But knowing when to stop

Definition

A finite natural number is any natural number less than or equal to some maximum value

$$0 \leq c \leq max$$

But knowing when to stop

Definition

A finite natural number is any natural number less than or equal to some maximum value

$$0 \leq c \leq max$$

Example

- 00000000
- 00000145
- 68370912

where 99999999 is the maximum

Computers can add too

```c
#include "stdio.h"

int main()
{
  unsigned short a =  5000;
  unsigned short b = 61000;
  unsigned short c;

  c = a + b;

  printf("%d", c);
}
```

When it all goes wrong

Ariane 5 rocket, Flight 501, 4 June 1996 exploded

because a line of its code tried to store a number which was too big.

So maybe "max" was important after all...

One way find "max" is to keep adding 1 until something strange happens

```
#include "stdio.h"

int main()
{
  unsigned short c = 0;
  unsigned short c_plus_1 = 1;

  while (c_plus_1 == c + 1)
  {
    c = c + 1;
    c_plus_1 = c_plus_1 + 1;
  }

  printf("%d", c);
}
```

A 12 hour clock for a 24 hour day

What time is 14:00 on a clock?


```
print(14 % 12)
```

Back to the mysterious addition

```
a =  5000
b = 61000

c = a + b

print(c % 65535)
```

So can we not have bigger numbers?

1988: Sega Genesis (16-bit console)

So can we not have bigger numbers?

1994: Sony PlayStation (32-bit console)

So can we not have bigger numbers?

1996: Nintendo 64 (64-bit console)

Everything is made of bits

- binary digits (bits) are a like ordinary digits (0 to 9), but there are less of them.
  In fact there are only two: 0 and 1.

Everything is made of bits

- binary digits (bits) are a like ordinary digits (0 to 9), but there are less of them.
  In fact there are only two: 0 and 1.

- 105 is "1 times 100, plus 0 times 10, plus 5 times 1".

Everything is made of bits

- binary digits (bits) are a like ordinary digits (0 to 9), but there are less of them.
  In fact there are only two: 0 and 1.

- 105 is "1 times 100, plus 0 times 10, plus 5 times 1".

- More importantly, it is also 01101001, expressed in (8-bit) binary.

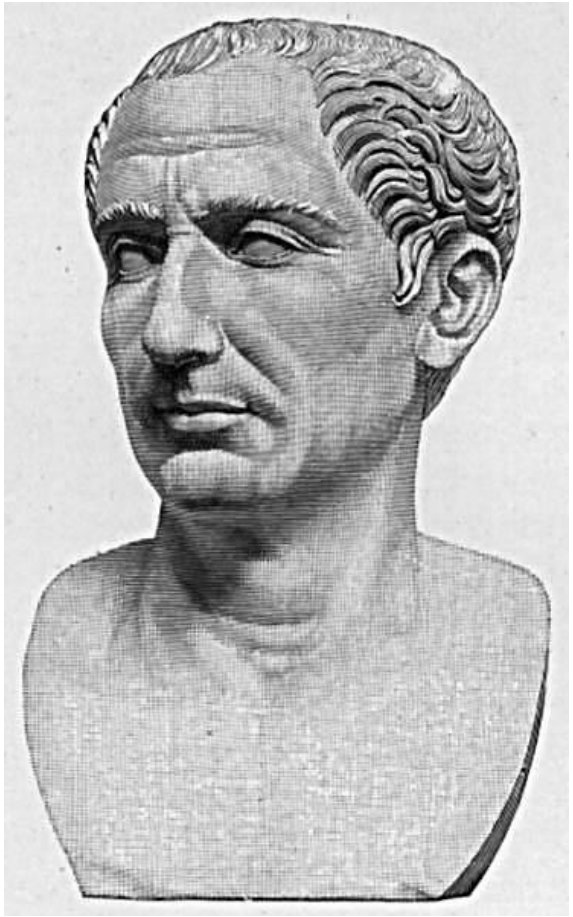
Everything is made of bits

- binary digits (bits) are a like ordinary digits (0 to 9), but there are less of them.
  In fact there are only two: 0 and 1.
- 105 is "1 times 100, plus 0 times 10, plus 5 times 1".
- More importantly, it is also 01101001, expressed in (8-bit) binary.
- And the maximum an 8-bit number can represent is 11111111, which is 255.

How many bits does my number need?

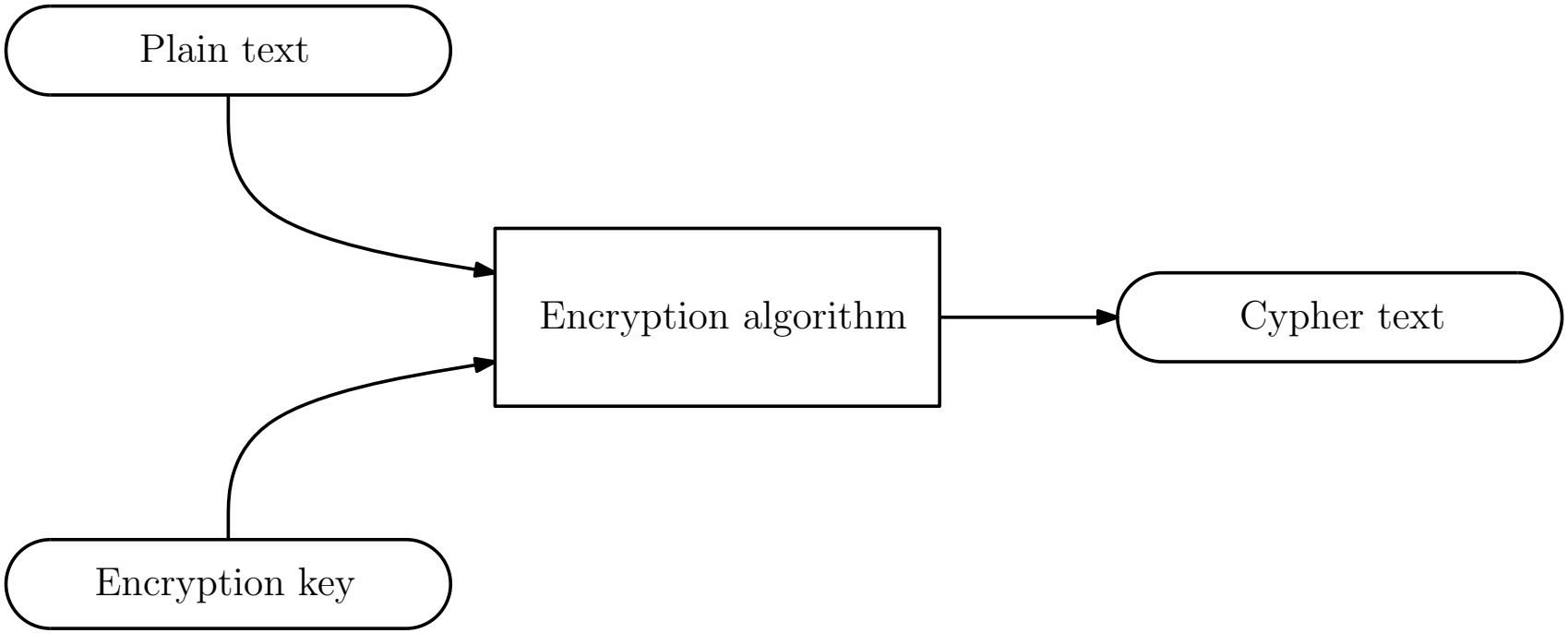
```c
#include "stdio.h"

int main()
{
  unsigned long c = 0;
  unsigned long c_max = 65535;

  unsigned long order  = 1;
  unsigned long n_bits = 0;

  while (c < c_max)
  {
    c = c + order;
    order = 2 * order;
    n_bits = n_bits + 1;
  }

  printf("%d", n_bits);
}
```

Old-style encryption

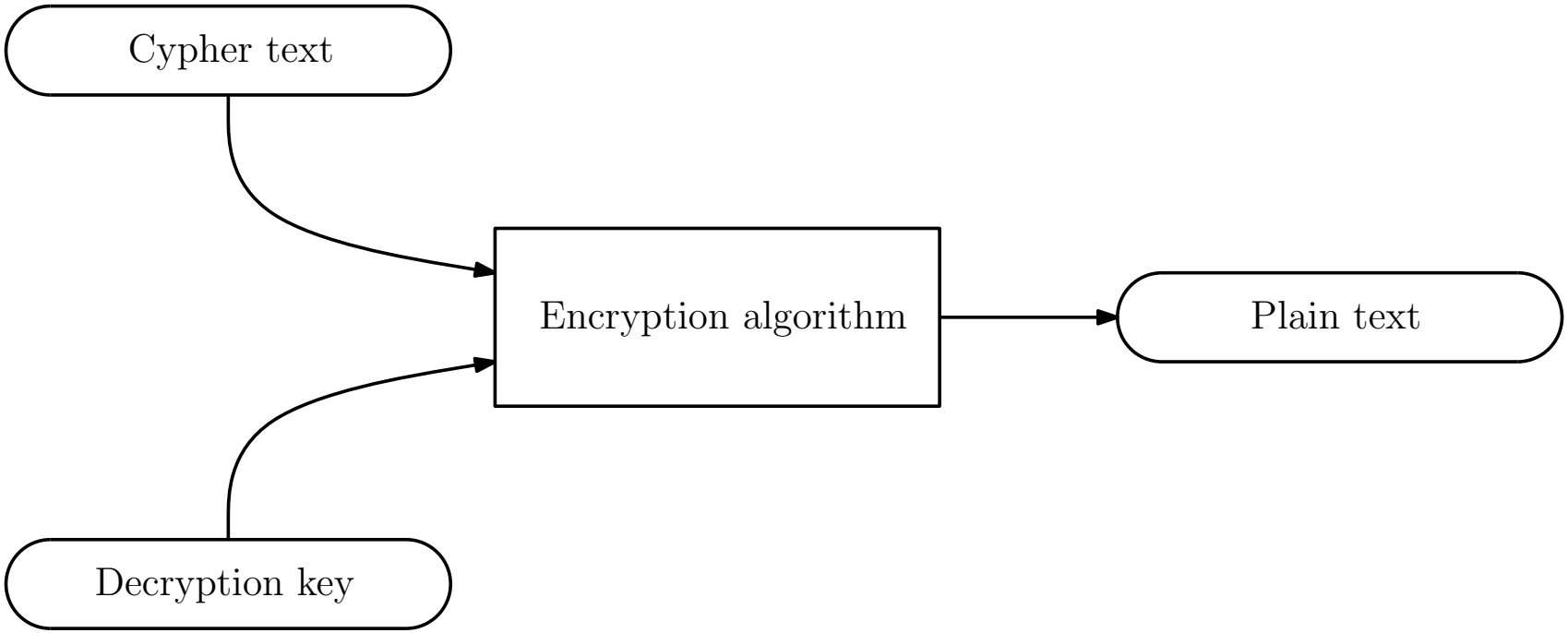


Julius Caesar (100 - 44BC)

How it works (encryption)



How it works (decryption)



Julius didn't trust everyone...

Key = 25: A becomes Z, B becomes A,...

...only his closest pals

```
def cypher(message, key = 13):
  result = ""
  a_value = ord('a')

  for letter in message:

    value = ord(letter)
    value_0 = value - a_value

    if letter.islower():
      value_0 = (value_0 + key) % 26

    value = value_0 + a_value
    result = result + chr(value)

  return result

print(cypher("nyrn vnpgn rfg"))
```

ASCII Tables