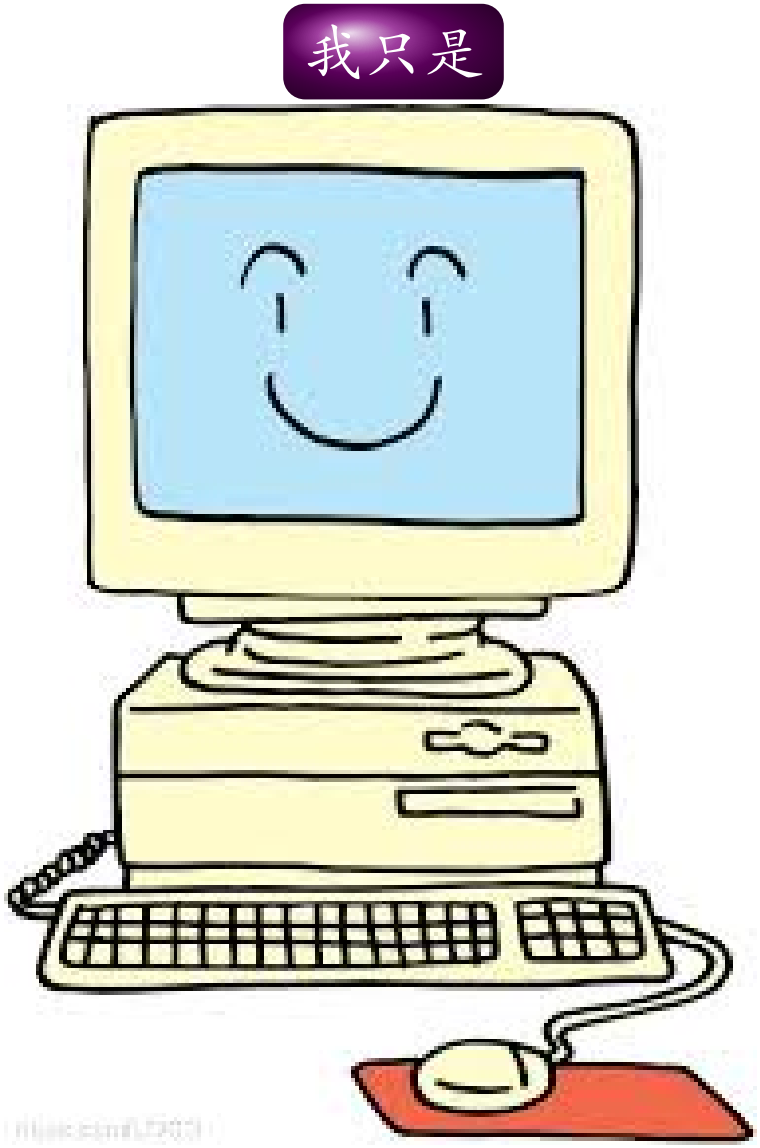
1 感性认识计算机程序

郑海永
zhenghaiyong@ouc.edu.cn

中国海洋大学 电子工程学院


我只是

我不是

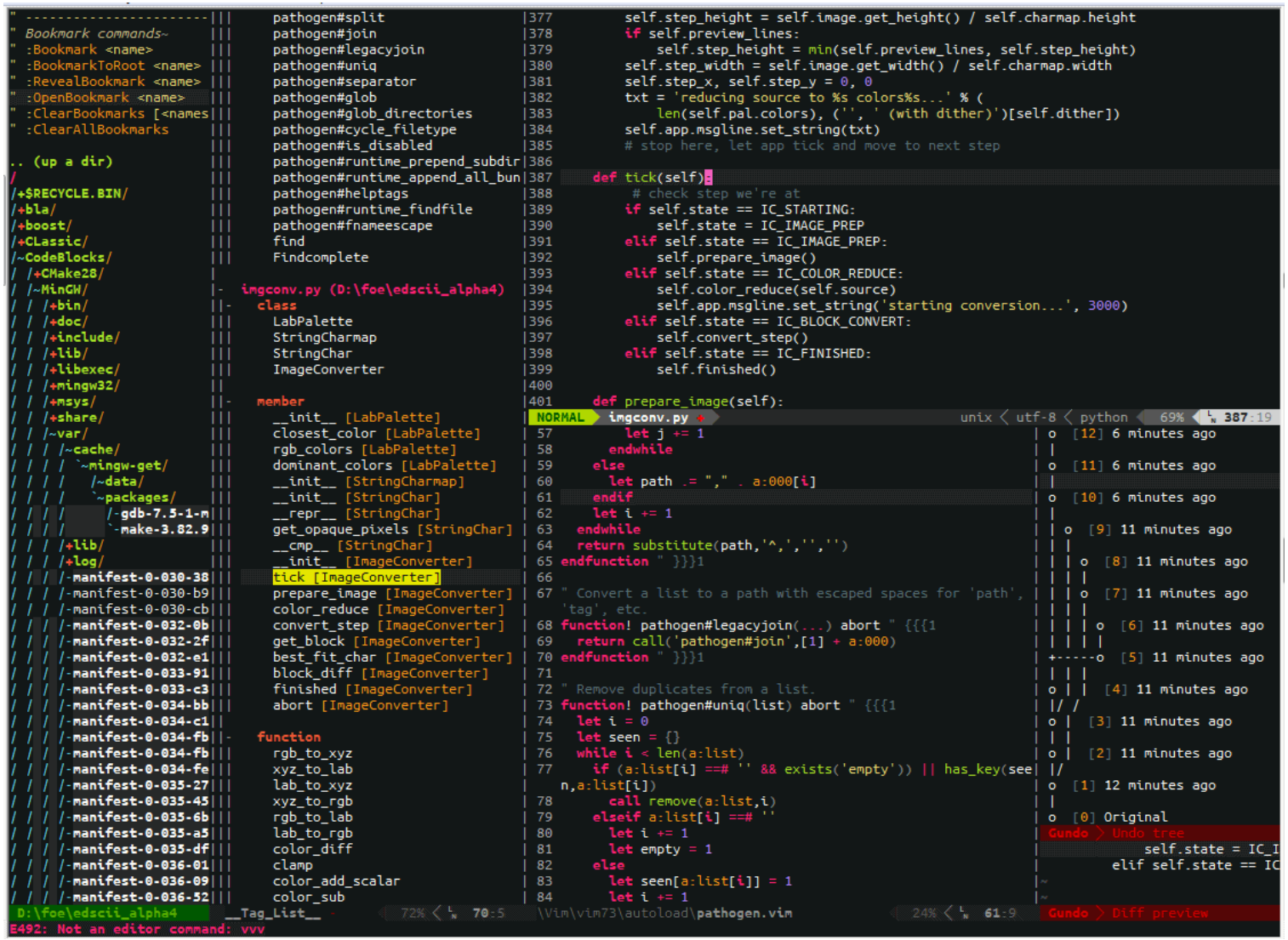
请不要

强脑所难！

编辑、编译、运行

```
vim main.cpp
g++ -o program main.cpp
./program
```

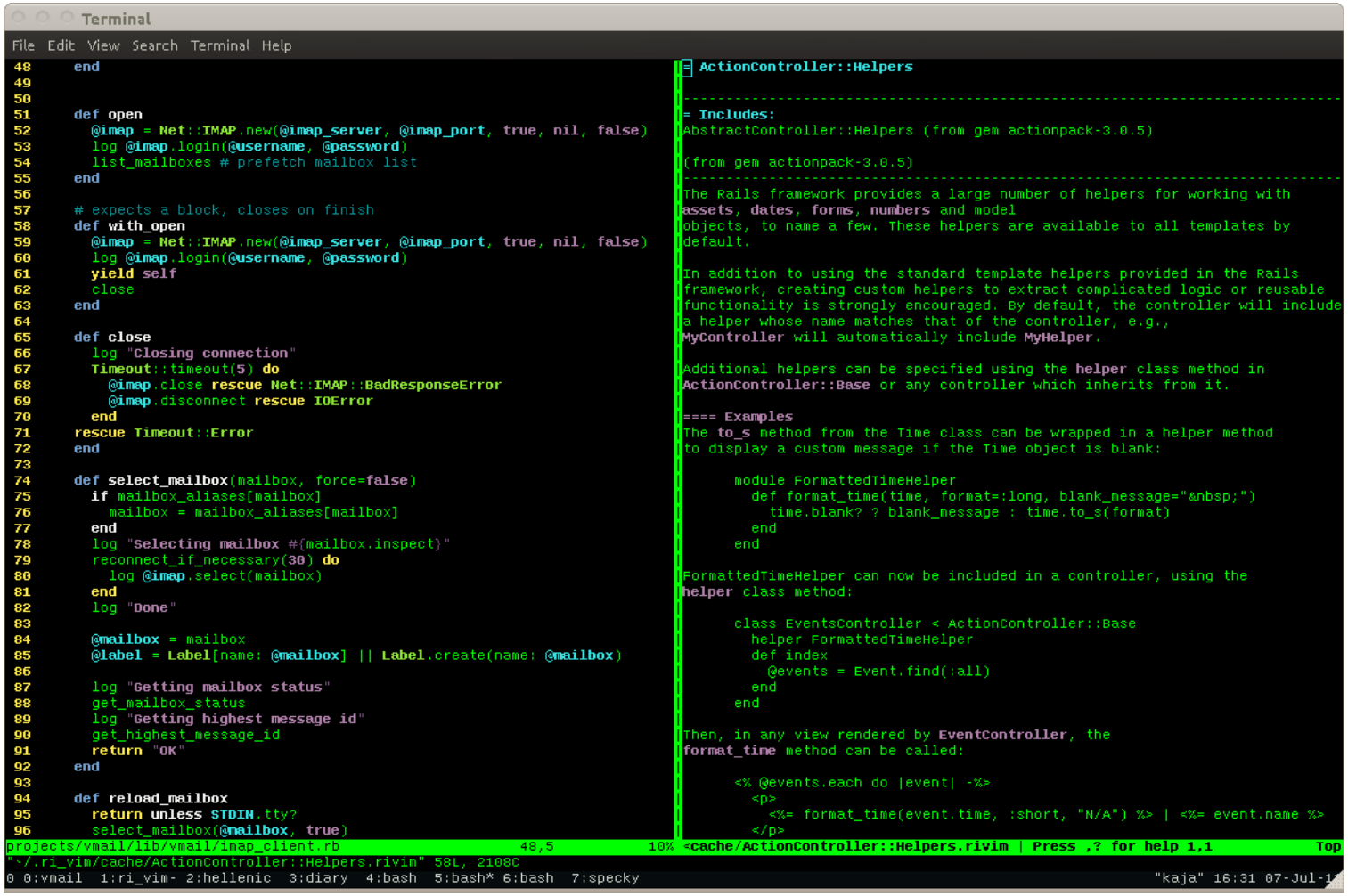
vim


Bookmark commands~
:Bookmark <name>
:BookmarkToRoot <name>
:RevealBookmark <name>
:OpenBookmark <name>
:ClearBookmarks [<names
:ClearAllBookmarks
.. (up a dir)
/+$RECYCLE.BIN/
/+bla/
/+boost/
/+CLassic/
/~CodeBlocks/
/+CMake28/
/~MinGW/
/+bin/
/+doc/
/+include/
/+lib/
/+libexec/
/+mingw32/
/+msys/
/+share/
/~var/
/~cache/
`~mingw-get/
/~data/
`~packages/
/-gdb-7.5.1-m
`-make-3.82.9
/+lib/
/+log/
/-manifest-0-030-38
/-manifest-0-030-b9
/-manifest-0-030-cb
/-manifest-0-032-0b
/-manifest-0-032-2f
/-manifest-0-032-e1
/-manifest-0-033-91
/-manifest-0-033-c3
/-manifest-0-034-bb
/-manifest-0-034-c1
/-manifest-0-034-fb
/-manifest-0-034-fb
/-manifest-0-034-fe
/-manifest-0-035-27
/-manifest-0-035-45
/-manifest-0-035-6b
/-manifest-0-035-a5
/-manifest-0-035-df
/-manifest-0-036-01
/-manifest-0-036-09
/-manifest-0-036-52
pathogen#split
pathogen#join
pathogen#legacyjoin
pathogen#uniq
pathogen#separator
pathogen#glob
pathogen#glob_directories
pathogen#cycle_filetype
pathogen#is_disabled
pathogen#runtime_prepend_subdir
pathogen#runtime_append_all_bun
pathogen#helptags
pathogen#runtime_findfile
pathogen#fnameescape
find
Findcomplete
imgconv.py (D:\foe\edscii_alpha4)
class
LabPalette
StringCharmap
StringChar
ImageConverter
member
__init__ [LabPalette]
closest_color [LabPalette]
rgb_colors [LabPalette]
dominant_colors [LabPalette]
__init__ [StringCharmap]
__init__ [StringChar]
__repr__ [StringChar]
get_opaque_pixels [StringChar]
__cmp__ [StringChar]
__init__ [ImageConverter]
tick [ImageConverter]
prepare_image [ImageConverter]
color_reduce [ImageConverter]
convert_step [ImageConverter]
get_block [ImageConverter]
best_fit_char [ImageConverter]
block_diff [ImageConverter]
finished [ImageConverter]
abort [ImageConverter]
function
rgb_to_xyz
xyz_to_lab
lab_to_xyz
xyz_to_rgb
rgb_to_lab
lab_to_rgb
color_diff
clamp
color_add_scalar
color_sub
self.step_height = self.image.get_height() / self.charmap.height
if self.preview_lines:
self.step_height = min(self.preview_lines, self.step_height)
self.step_width = self.image.get_width() / self.charmap.width
self.step_x, self.step_y = 0, 0
txt = 'reducing source to %s colors%s...' % (
len(self.pal.colors), ('', ' (with dither)')[self.dither])
self.app.msgline.set_string(txt)
# stop here, let app tick and move to next step
def tick(self):
# check step we're at
if self.state == IC_STARTING:
self.state = IC_IMAGE_PREP
elif self.state == IC_IMAGE_PREP:
self.prepare_image()
elif self.state == IC_COLOR_REDUCE:
self.color_reduce(self.source)
self.app.msgline.set_string('starting conversion...', 3000)
elif self.state == IC_BLOCK_CONVERT:
self.convert_step()
elif self.state == IC_FINISHED:
self.finished()
def prepare_image(self):
NORMAL imgconv.py + unix utf-8 python 69% 387:19
let j += 1
endwhile
else
let path .= "," . a:000[i]
endif
let i += 1
endwhile
return substitute(path,'^,','','')
endfunction " }}}1
" Convert a list to a path with escaped spaces for 'path', 'tag', etc.
function! pathogen#legacyjoin(...) abort " {{{1
return call('pathogen#join',[1] + a:000)
endfunction " }}}1
" Remove duplicates from a list.
function! pathogen#uniq(list) abort " {{{1
let i = 0
let seen = {}
while i < len(a:list)
if (a:list[i] ==# '' && exists('empty')) || has_key(see
n,a:list[i])
call remove(a:list,i)
elseif a:list[i] ==# ''
let i += 1
let empty = 1
else
let seen[a:list[i]] = 1
let i += 1
o [12] 6 minutes ago
o [11] 6 minutes ago
o [10] 6 minutes ago
o [9] 11 minutes ago
o [8] 11 minutes ago
o [7] 11 minutes ago
o [6] 11 minutes ago
o [5] 11 minutes ago
o [4] 11 minutes ago
o [3] 11 minutes ago
o [2] 11 minutes ago
o [1] 12 minutes ago
o [0] Original
Gundo Undo tree
self.state = IC_I
elif self.state == IC
D:\foe\edscii_alpha4 __Tag_List__ 72% 70:5
\Vim\vim73\autoload\pathogen.vim 24% 61:9 Gundo Diff preview
E492: Not an editor command: vvv

vim


3+ ../s/manual.html test
if same
$log.info("#{@splayd}: Session OK")
else
@so.write "INFOS"
@so.write @ip
if @so.read != "OK" then raise ProtocolError, "INFOS not OK" end
infos = @so.read # no addslashes (json)
@splayd.insert_splayd_infos(infos)
bl = Splayd.blacklist
@so.write "BLACKLIST"
@so.write bl.to_json
if @so.read != "OK" then raise ProtocolError, "BLACKLIST not OK" end
logv = {}
logv['ip'] = @@logd_ip
logv['port'] = @@logd_port + rand(@@num_logd)
logv['max_size'] = @@log_max_size
@so.write "LOG"
@so.write logv.to_json
if @so.read != "OK" then raise ProtocolError, "LOG not OK" end
$log.info("#{@splayd}: Log port: #{logv['port']}")
end
$log.info("#{@splayd}: Auth OK")
@splayd.available
rescue => e
# restore previous status (REGISTER, UNAVAILABLE or RESET)
@splayd.update('status', @splayd.row['status'])
raise e
end
if @ip != old_ip and @@localize
$log.info("#{@splayd}: Localization")
@splayd.localize
end
TODO Invariant check @splayd.row must be == to a new fetch of infos
end
def main
begin
last_contact = @splayd.last_contact
running = true
while running
action = @splayd.next_action
if not action
if Time.now.to_i - last_contact > @@ping_interval
"Inlining PING" Avoid 2 DB operations
@so.write "PING"
if @so.read != "OK" then raise ProtocolError, "PING not OK" end
last_contact = @splayd.last_contact
end
sleep(rand(@@sleep_time * 2 * 100).to_f / 100)
else
$log.debug("#{@splayd}: Action #{action['command']}")
start_time = Time.now.to_f
@so.write action['command']
if action['data']
[utf-8,unix,RUBY,760 lines] [char:000/0x000] [tab:2] 202,1 28%
local ref = assert(so:receive_one())
if splayd.jobs[ref] then
free(ref)
assert(so:send("OK"))
else
assert(so:send("UNKNOWN_REF"))
end
end
function n_log(so)
splayd.settings.log = json.decode(assert(so:receive_one()))
if not splayd.settings.log.ip then
splayd.settings.log.ip = splayd.settings.controller.ip
end
assert(so:send("OK"))
end
function list(so)
local list = json.decode(assert(so:receive_one()))
local ref = list.ref
if not splayd.jobs[ref] then
assert(so:send("UNKNOWN_REF"))
return
end
job = splayd.jobs[ref]
-- We append the list to the job configuration.
list.ref = nil
job.network.list = list
assert(so:send("OK"))
end
function loadavg(so)
assert(so:send("OK"))
local f = string.gmatch(io.open("/proc/loadavg"):read(), "%d+.%d+")
assert(so:send(f()..." "..f()..." "..f()))
end
function n_start(so)
local ref = assert(so:receive_one())
if splayd.jobs[ref] then
if splayd.jobs[ref].status == "running" then
assert(so:send("RUNNING"))
return
end
start(ref)
assert(so:send("OK"))
else
assert(so:send("UNKNOWN_REF"))
end
end
function n_stop(so)
local ref = assert(so:receive_one())
if splayd.jobs[ref] then
if splayd.jobs[ref].status ~= "running" then
assert(so:send("NOT_RUNNING"))
return
end
stop(ref)
assert(so:send("OK"))
else
assert(so:send("UNKNOWN_REF"))
end
end
[utf-8,unix,LUA,1089 lines] [char:000/0x000] [tab:2] 451,1 42%
font-size: 85%;
margin: 2cm 15%;
text-align: center;
}
.content {
text-align: justify;
line-height: 140%;
margin: 0 auto;
}
h1, h2, h3, h4, h5, h6 {
font-family: "Lucida Grande", "Lucida Sans", "Lucida Sans Unicode", sans-serif;
margin-top: 1.5em;
}
pre {
background-color: #ddd;
font-family: "monospace", sans-serif;
font-size: 85%;
padding: 10px;
margin: 10px;
}
li {
margin-bottom: 0.5em;
}
table {
text-align: left;
margin: 0px;
}
table td {
padding: 5px;
border-bottom: 1px solid black;
font-size: 85%;
}
table th {
font-size: 80%;
width: 250px;
padding: 5px;
vertical-align: top;
background-color: #CCCCFF;
border-bottom: 1px solid black;
}
</style>
<link type="text/css" rel="stylesheet" href="syntax/syntax.css"></link>
<script language="javascript" src="syntax/shCore.js"></script>
<script language="javascript" src="syntax/shBrushLua.js"></script>
del
dfn
dir
div
dl
address
bdo
code
fieldset
isindex />
kbd
<div
[utf-8,unix,HTML,1267 lines] [char:000/0x000] [tab:2] 69,5 0%
-- Complètement selon le type de fichier (Omni) (^O^N^P) Correspondance 4 sur 11

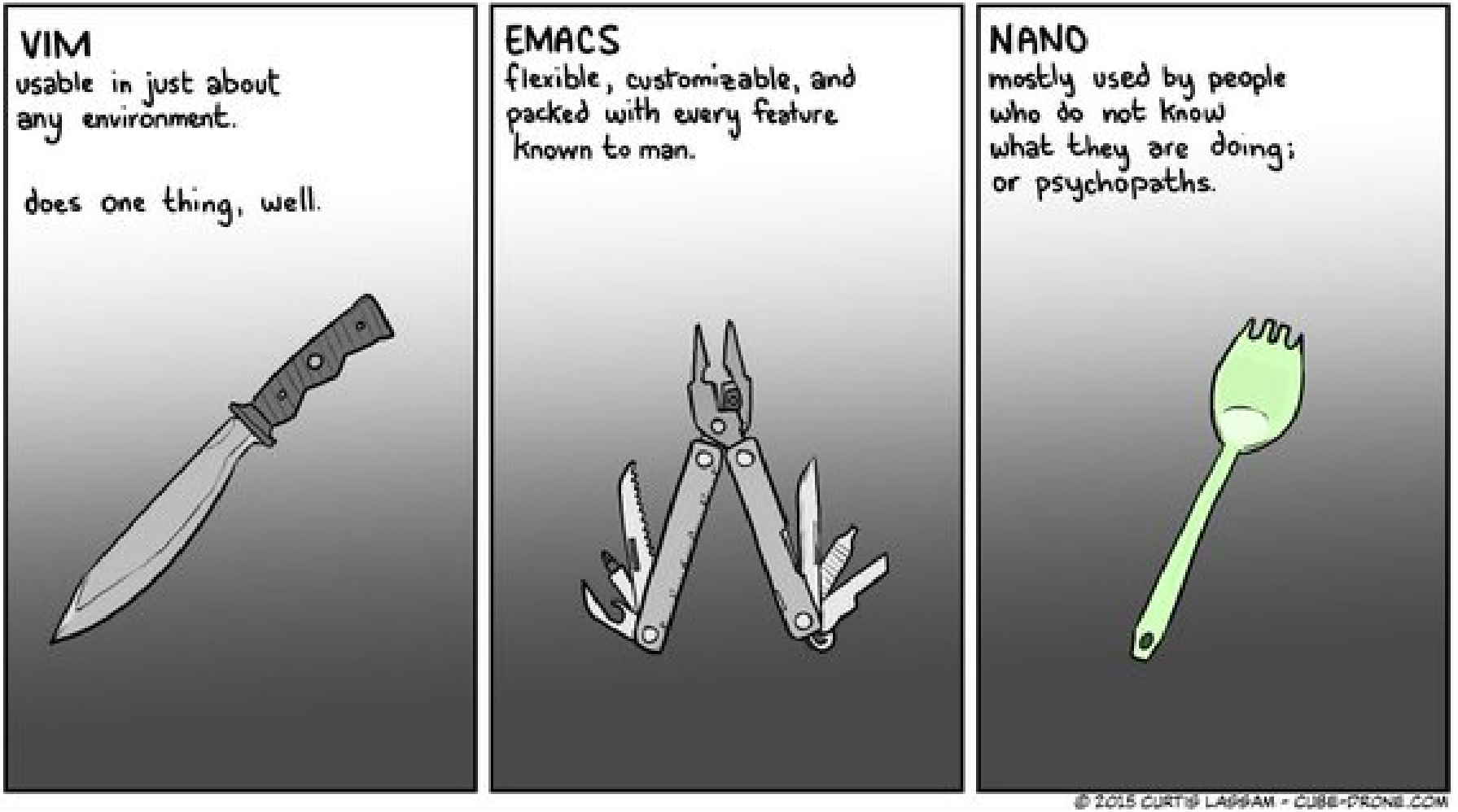
vim


Terminal
File Edit View Search Terminal Help
48 end
49
50
51 def open
52 @imap = Net::IMAP.new(@imap_server, @imap_port, true, nil, false)
53 log @imap.login(@username, @password)
54 list_mailboxes # prefetch mailbox list
55 end
56
57 # expects a block, closes on finish
58 def with_open
59 @imap = Net::IMAP.new(@imap_server, @imap_port, true, nil, false)
60 log @imap.login(@username, @password)
61 yield self
62 close
63 end
64
65 def close
66 log "Closing connection"
67 Timeout::timeout(5) do
68 @imap.close rescue Net::IMAP::BadResponseError
69 @imap.disconnect rescue IOError
70 end
71 rescue Timeout::Error
72 end
73
74 def select_mailbox(mailbox, force=false)
75 if mailbox_aliases[mailbox]
76 mailbox = mailbox_aliases[mailbox]
77 end
78 log "Selecting mailbox #{mailbox.inspect}"
79 reconnect_if_necessary(30) do
80 log @imap.select(mailbox)
81 end
82 log "Done"
83
84 @mailbox = mailbox
85 @label = Label[name: @mailbox] || Label.create(name: @mailbox)
86
87 log "Getting mailbox status"
88 get_mailbox_status
89 log "Getting highest message id"
90 get_highest_message_id
91 return "OK"
92 end
93
94 def reload_mailbox
95 return unless STDIN.tty?
96 select_mailbox(@mailbox, true)
projects/vmail/lib/vmail/imap_client.rb 48,5 10%
= ActionController::Helpers
= Includes:
AbstractController::Helpers (from gem actionpack-3.0.5)
(from gem actionpack-3.0.5)
The Rails framework provides a large number of helpers for working with assets, dates, forms, numbers and model objects, to name a few. These helpers are available to all templates by default.
In addition to using the standard template helpers provided in the Rails framework, creating custom helpers to extract complicated logic or reusable functionality is strongly encouraged. By default, the controller will include a helper whose name matches that of the controller, e.g., MyController will automatically include MyHelper.
Additional helpers can be specified using the helper class method in ActionController::Base or any controller which inherits from it.
==== Examples
The to_s method from the Time class can be wrapped in a helper method to display a custom message if the Time object is blank:
module FormattedTimeHelper
def format_time(time, format=:long, blank_message=" ")
time.blank? ? blank_message : time.to_s(format)
end
end
FormattedTimeHelper can now be included in a controller, using the helper class method:
class EventsController < ActionController::Base
helper FormattedTimeHelper
def index
@events = Event.find(:all)
end
end
Then, in any view rendered by EventController, the format_time method can be called:
<% @events.each do |event| -%>

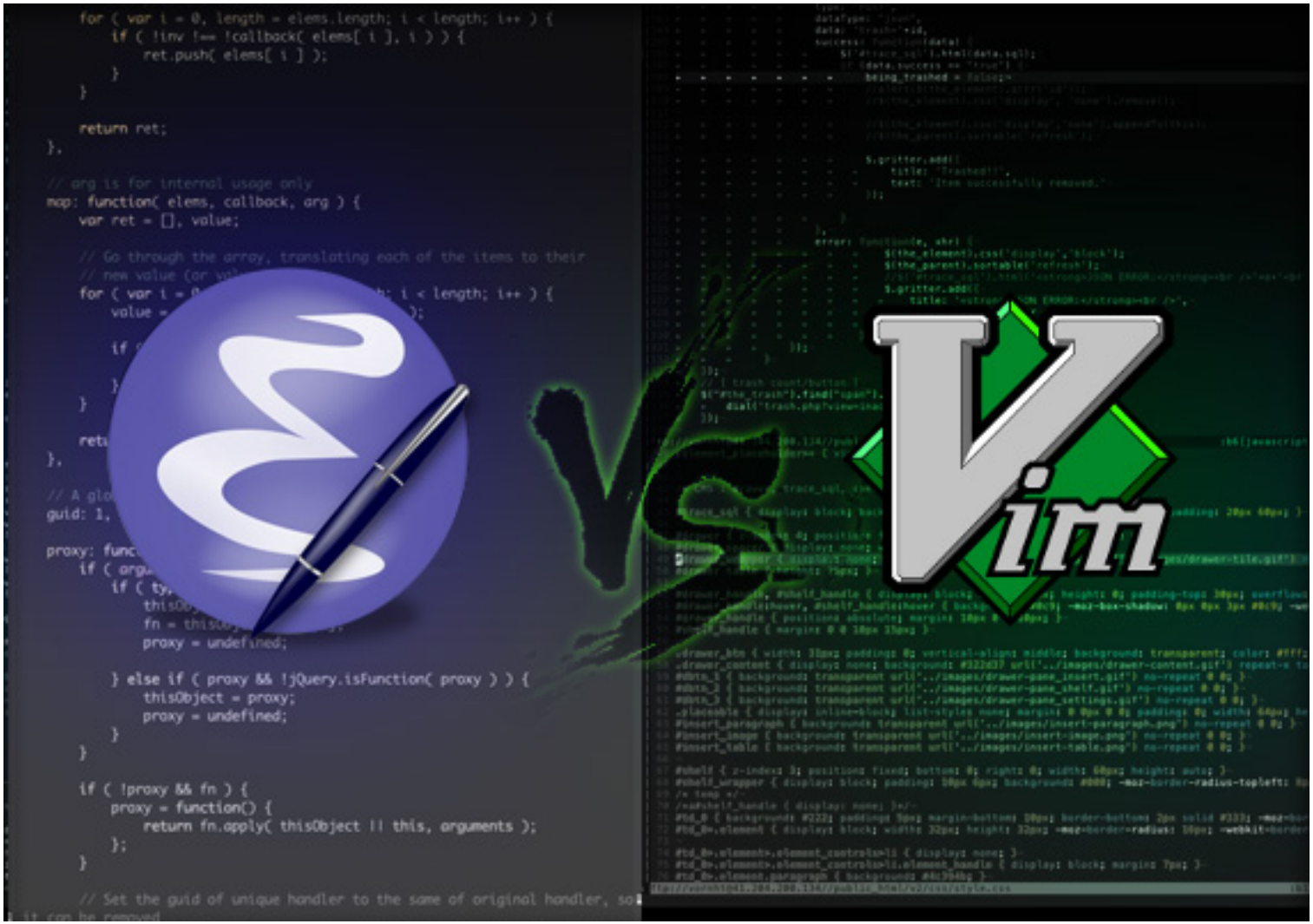
<%= format_time(event.time, :short, "N/A") %> | <%= event.name %>

<cache/ActionController::Helpers.rivim | Press ,? for help 1,1 Top
"~/.ri_vim/cache/ActionController::Helpers.rivim" 58L, 2108C
0 0:vmail 1:ri_vim- 2:hellenic 3:diary 4:bash 5:bash* 6:bash 7:specky
"kaja" 16:31 07-Jul-1

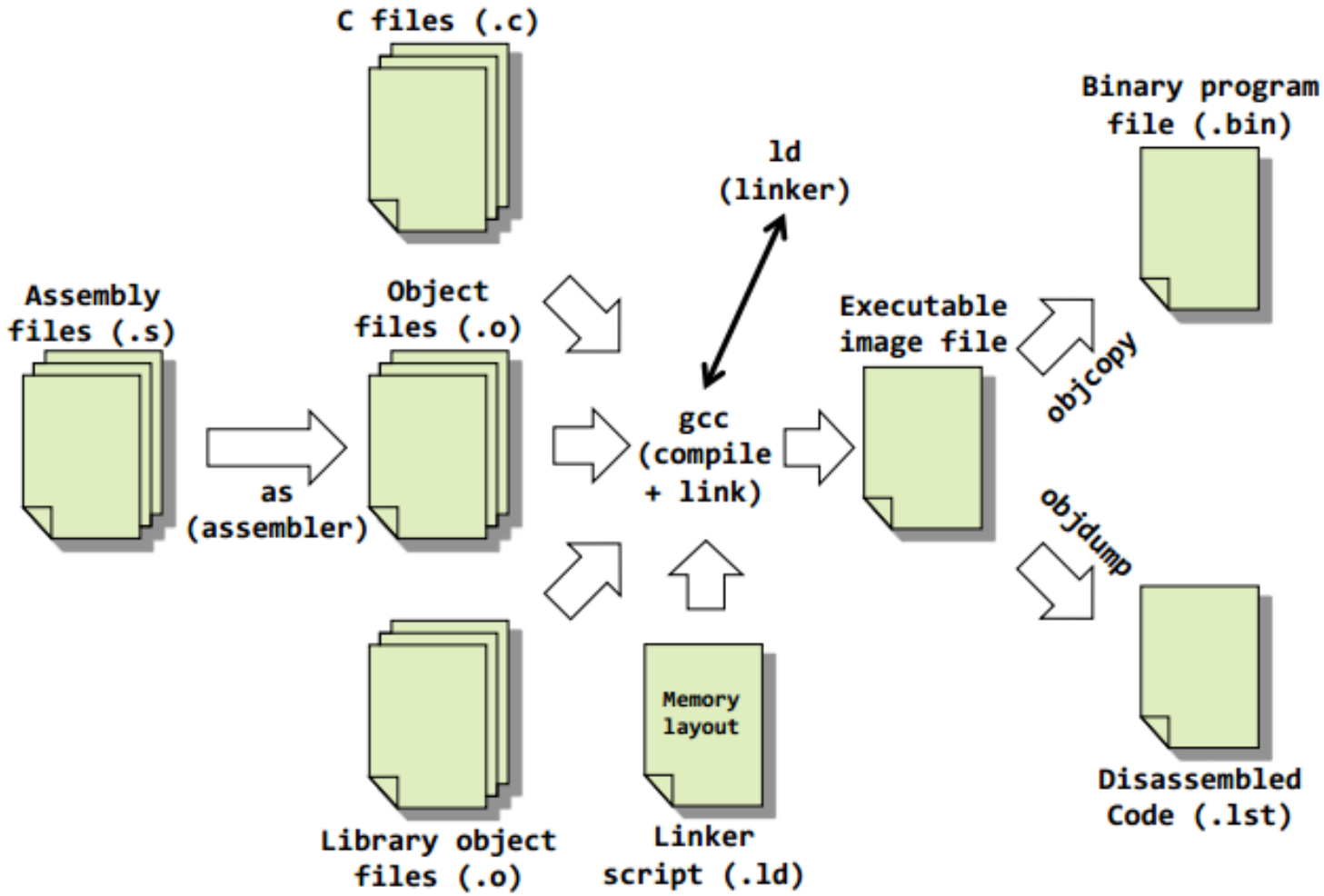
vim


VIM
usable in just about any environment.
does one thing, well.
EMACS
flexible, customizeable, and packed with every feature known to man.
NANO
mostly used by people who do not know what they are doing; or psychopaths.
© 2015 CURTIS LAGHAM - CUBE-DRONE.COM

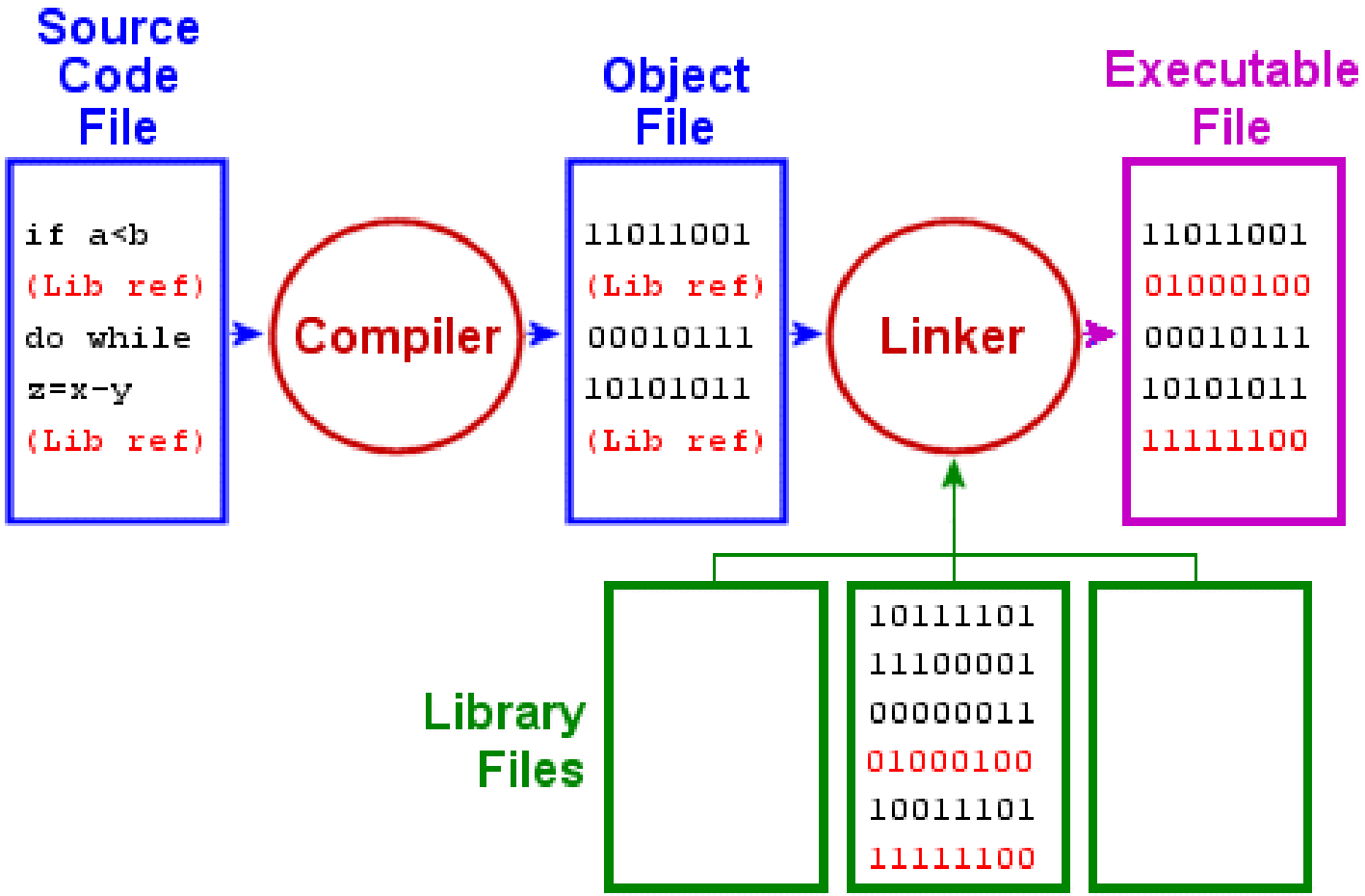
vim



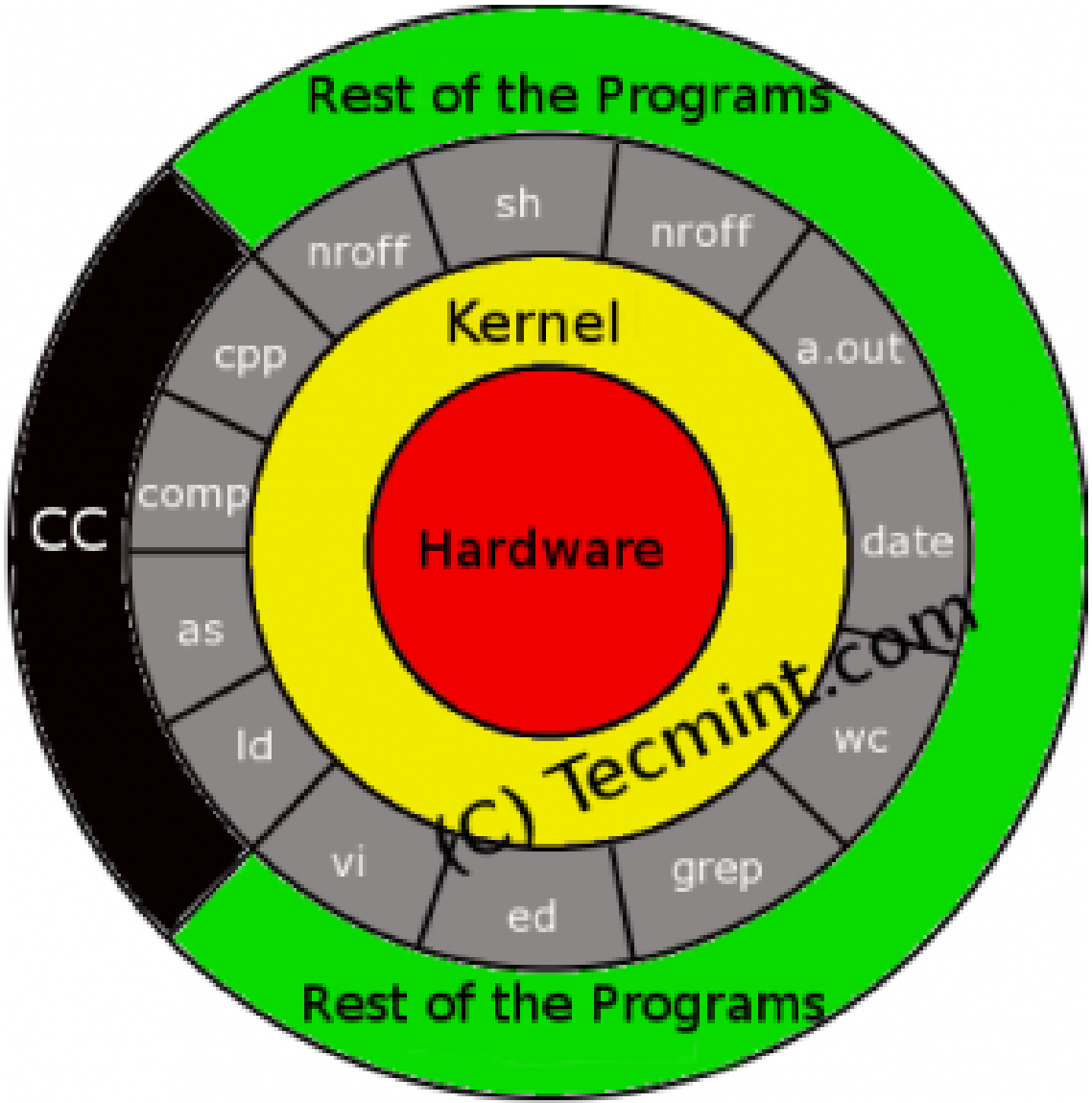
gcc/g++


C files (.c)
ld
(linker)
Binary program
file (.bin)
Assembly
files (.s)
Object
files (.o)
Executable
image file
objcopy
gcc
(compile
+ link)
as
(assembler)
objdump
Memory
layout
Library object
files (.o)
Linker
script (.ld)
Disassembled
Code (.lst)

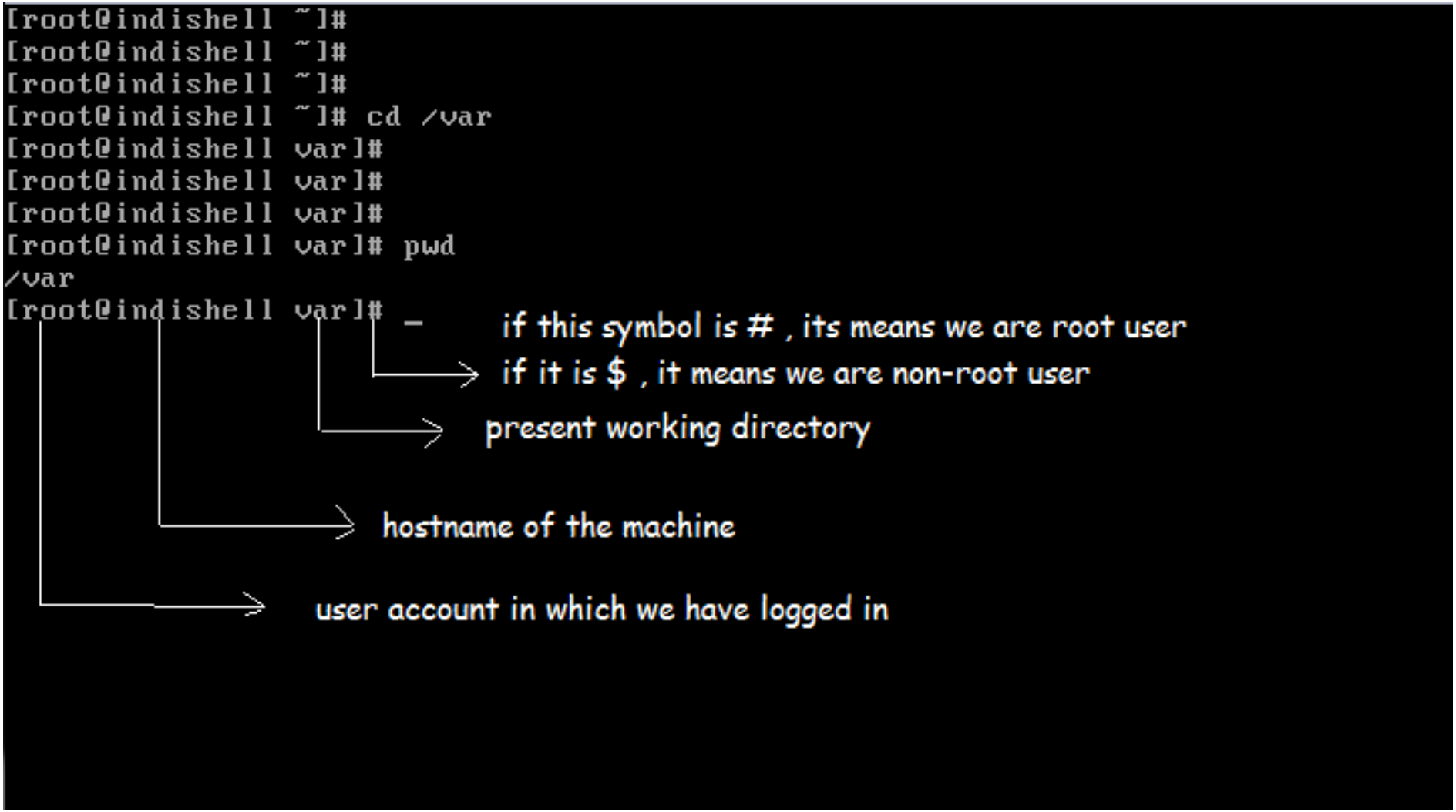
gcc/g++


Source Code File
if a<b
(Lib ref)
do while
z=x-y
(Lib ref)
Compiler
Object File
11011001
(Lib ref)
00010111
10101011
(Lib ref)
Linker
Executable File
11011001
01000100
00010111
10101011
11111100
Library Files
10111101
11100001
00000011
01000100
10011101
11111100

shell


Rest of the Programs
sh
nroff
nroff
Kernel
cpp
a.out
comp
CC
Hardware
date
as
ld
wc
(C) Tecmint.com
vi
grep
ed
Rest of the Programs

shell


[root@indishell ~]#
[root@indishell ~]#
[root@indishell ~]#
[root@indishell ~]# cd /var
[root@indishell var]#
[root@indishell var]#
[root@indishell var]#
[root@indishell var]# pwd
/var
[root@indishell var]# _
if this symbol is # , its means we are root user
if it is $, it means we are non-root user
present working directory
hostname of the machine
user account in which we have logged in

关于空格

```
Thisisanexample(forexample).
This is an example(for example).
This is an example (for example).
This  is  an example (for  example).
```

咿呀学语

目录I

1. 感性认识计算机程序
   - 说在前面的话
   - 程序是你告诉计算机的话
   - 如果你的大脑是台计算机
   - 如果你来设计一门编程语言
2. 快步走进C程序
   - 最简单的程序—“模仿”
   - 很简单的程序
   - 简单的程序
3. 从现实问题到计算机程序
   - 没有解决方案就没有程序
   - 先有构想再写程序
   - 体验结构化的程序

内容提要 I

1 感性认识计算机程序
- 说在前面的话
- 程序是你告诉计算机的话
- 如果你的大脑是台计算机
- 如果你来设计一门编程语言

2 快步走进C程序
- 最简单的程序—“模仿”
- 很简单的程序
- 简单的程序

3 从现实问题到计算机程序
- 没有解决方案就没有程序
- 先有构想再写程序
- 体验结构化的程序

内容提要 I

1 感性认识计算机程序
- 说在前面的话
- 程序是你告诉计算机的话
- 如果你的大脑是台计算机
- 如果你来设计一门编程语言

2 快步走进C程序
- 最简单的程序—“模仿”
- 很简单的程序
- 简单的程序

3 从现实问题到计算机程序
- 没有解决方案就没有程序
- 先有构想再写程序
- 体验结构化的程序

学习编程语言开始阶段要注意

训练得技能

- 知识（Knowledge）⇐ 课堂/视频
- 使用（Skill）⇐**练习，练习，再练习！**

学习编程语言开始阶段要注意

须抓大放小

- 抓大：激发兴趣！抓主要问题
- 放小：放过细节！放纠缠细节

学习编程语言开始阶段要注意

多练简单题

- **模仿**⇐婴儿学语言！
- **抄**程序！
- 沉住气！“磨刀不误砍柴工”“Slow down to speed up”

学习编程语言开始阶段要注意

选一本薄书

- 简单！有效！

学习编程语言开始阶段要注意

- 训练得技能
- 须抓大放小
- 多练简单题
- 选一本薄书

内容提要 I

1 感性认识计算机程序
- 说在前面的话
- 程序是你告诉计算机的话
- 如果你的大脑是台计算机
- 如果你来设计一门编程语言

2 快步走进C程序
- 最简单的程序—“模仿”
- 很简单的程序
- 简单的程序

3 从现实问题到计算机程序
- 没有解决方案就没有程序
- 先有构想再写程序
- 体验结构化的程序

什么是程序？

计算机是机器！

- 必须“设置”好才能运行！
  - ENIAC 采用“手工插线”的方式“编程”
  - 存储程序式（冯诺伊曼式）计算机
- “编程序” $\Rightarrow$ 给计算机设置好运行的步骤！

什么是程序？

计算机是机器！

- 必须“设置”好才能运行！
  - ENIAC 采用“手工插线”的方式“编程”
  - 存储程序式（冯诺伊曼式）计算机
- “编程序”$\Rightarrow$ 给计算机设置好运行的步骤！

程序

- 人们用来告诉计算机应该做什么的东西！

什么是程序？

程序

- 人们用来告诉计算机应该做什么的东西！

问题：怎么告诉它呢？

1. **告诉计算机一些什么东西，它才能运行？**
2. **以什么形式告诉它，它才能明白？**

什么是程序？

程序

- 人们用来告诉计算机应该做什么的东西！

问题：怎么告诉它呢？

1. **告诉计算机一些什么东西，它才能运行？内容！**
2. **以什么形式告诉它，它才能明白？形式！**

内容提要 I

1 感性认识计算机程序
- 说在前面的话
- 程序是你告诉计算机的话
- 如果你的大脑是台计算机
- 如果你来设计一门编程语言

2 快步走进C程序
- 最简单的程序—“模仿”
- 很简单的程序
- 简单的程序

3 从现实问题到计算机程序
- 没有解决方案就没有程序
- 先有构想再写程序
- 体验结构化的程序

告诉计算机一些什么东西，它才能运行？

给你一个数列，问哪个数字最大？ 假如你的大脑就是一台计算机

78，56，69，31，36，67，31，47，69，34，45，74，61，82，43，
41，76，79，81，66，54，50，76，51，53，28，74，39，45，61，
52，41，43，75，78，84，72，51，43，64，75，81，69，55，74

你是怎么做的？

给你一个数列，问哪个数字最大？ 假如你的大脑就是一台计算机

78，56，69，31，36，67，31，47，69，34，45，74，61，82，43，
41，76，79，81，66，54，50，76，51，53，28，74，39，45，61，
52，41，43，75，78，84，72，51，43，64，75，81，69，55，74

1. 把某一个数字取出来，当作一个临时的“特别数字”记住，并假设这个数字最大；
2. 拿这个临时的“特别数字”与其他数字相比较；
3. 如果有其他数字比临时的“特别数字”更大，就把“特别数字”换成这个更大的数字；
4. 重复上述过程直到把所有的数字都比较完毕；
5. 最后大脑中这个“特别数字”就记录了最大的数字。

我的大脑这样运行……

把自己的大脑当作计算机：

1. 在大脑中开辟一片“存储空间”存放输入的数字；
2. 使用了一个另一片“存储空间”存放“特别数字”；
3. 按照某种规律“反复”选定“输入存储空间中的数字”与“特别数字”比较；
4. 每次比较时，判断“选定的数字”是否大于“特别数字”；
5. 如果大于，重新“刷新”“特别数字”；
6. 如果“特别数字”与其他数字都进行了比较，说出“特别数字”。

把这个过程稍作整理……

更清楚的描述方式：

1. 在你的大脑里开辟一片存储空间存放输入的数字；
2. 开辟另一片存储空间存放“特别数字”；
3. 从存储空间中的第一个数字开始，直到最后一个数，重复以下操作：
   1. 比较“存储空间中的数字”与“特别数字”；
   2. 如果“存储空间中的数字”大于“特别数字”；
   3. 那么将“特别数字”换成“存储空间中的数字”。
4. 说出“特别数字”。

让计算机用程序来解决的话

计算机程序也是一样的**逻辑结构**只不过换了一种语言来表达！

```cpp
#include<iostream>
using namespace std;
int main( )
{
  int number[45] = {78, 56, 69, 31, 36, 67, 31, 47, 69,
  ↪  34, 45, 74, 61, 82, 43, 41, 76, 79, 81, 66, 54, 50,
  ↪  76, 51, 53, 28, 74, 39, 45, 61, 52, 41, 43, 75, 78,
  ↪  84, 72, 51, 43, 64, 75, 81, 69, 55, 74};
  int max = 0;
  int i = 0;
  for(i = 0; i < 45; i++)
  {
    if(number[i] > max) max = number[i];
  }
  cout<<"The Maximal Number is: "<<max;
  return 0;
}
```

内容提要 I

1 感性认识计算机程序
- 说在前面的话
- 程序是你告诉计算机的话
- 如果你的大脑是台计算机
- 如果你来设计一门编程语言

2 快步走进C程序
- 最简单的程序—“模仿”
- 很简单的程序
- 简单的程序

3 从现实问题到计算机程序
- 没有解决方案就没有程序
- 先有构想再写程序
- 体验结构化的程序

假如你要设计一门编程语言

如果你要创造一门“程序设计语言”

假如你要设计一门编程语言

如果你要创造一门“程序设计语言”

问题①：

是不是无论我们在程序里写什么“单词”，计算机都能明白？

假如你要设计一门编程语言

如果你要创造一门“程序设计语言”

问题①：

是不是无论我们在程序里写什么“单词”，计算机都能明白？

问题②：

是不是无论我们在程序里写什么“数”和“计算符号”，计算机都能明白？

假如你要设计一门编程语言

如果你要创造一门“程序设计语言”

问题①：

是不是无论我们在程序里写什么“单词”，计算机都能明白？

问题②：

是不是无论我们在程序里写什么“数”和“计算符号”，计算机都能明白？

问题③：

世界上可能要用“程序来表达的逻辑”纷繁复杂，程序设计语言里面得有多少种“句式”才能够用？

问题①：

是不是无论我们在程序里写什么“单词”，计算机都能明白？

问题①：

是不是无论我们在程序里写什么“单词”，计算机都能明白？

NO!

编程语言定义了一些有特定含义的“**关键字**”，计算机“只能明白”这些“词”的含义。

问题①：

是不是无论我们在程序里写什么“单词”，计算机都能明白？

NO!

编程语言定义了一些有特定含义的“**关键字**”，计算机“只能明白”这些“词”的含义。

计算机能够认识的单词（35）

auto	break	case	char	const	continue	default
do	double	else	enum	extern	float	for
goto	if	int	long	register	return	short
signed	sizeof	static	struct	switch	typedef	unsigned
union	void	volatile	while	bool	catch	class

```cpp
#include<iostream>
using namespace std;
int main()
{
  enum day{Mon, Tue, Wed, Thu, Fri, Sat, Sun};
  double times, wages, hourlyRate, hours;
  cout<<"Enter the hourly wages rate."<<endl;
  cin>>hourlyRate;
  cout<<"Enter hours worked daily\n";
  for (int workDay=Mon; workDay<=Sun; workDay++)
    { cin>>hours; //输入周一到周日的工作时间;
      switch(workDay)
        { case Sat: times=1.5*hours; break;
          case Sun: times=2.0*hours; break;
          default: times=hours; }
      wages = wages + times*hourlyRate;
    }
  cout<<"The wages for the week are "<<wages;
  return 0;
}
```

问题②：

是不是无论我们在程序里写什么“数”和“计算符号”，计算机都能明白？

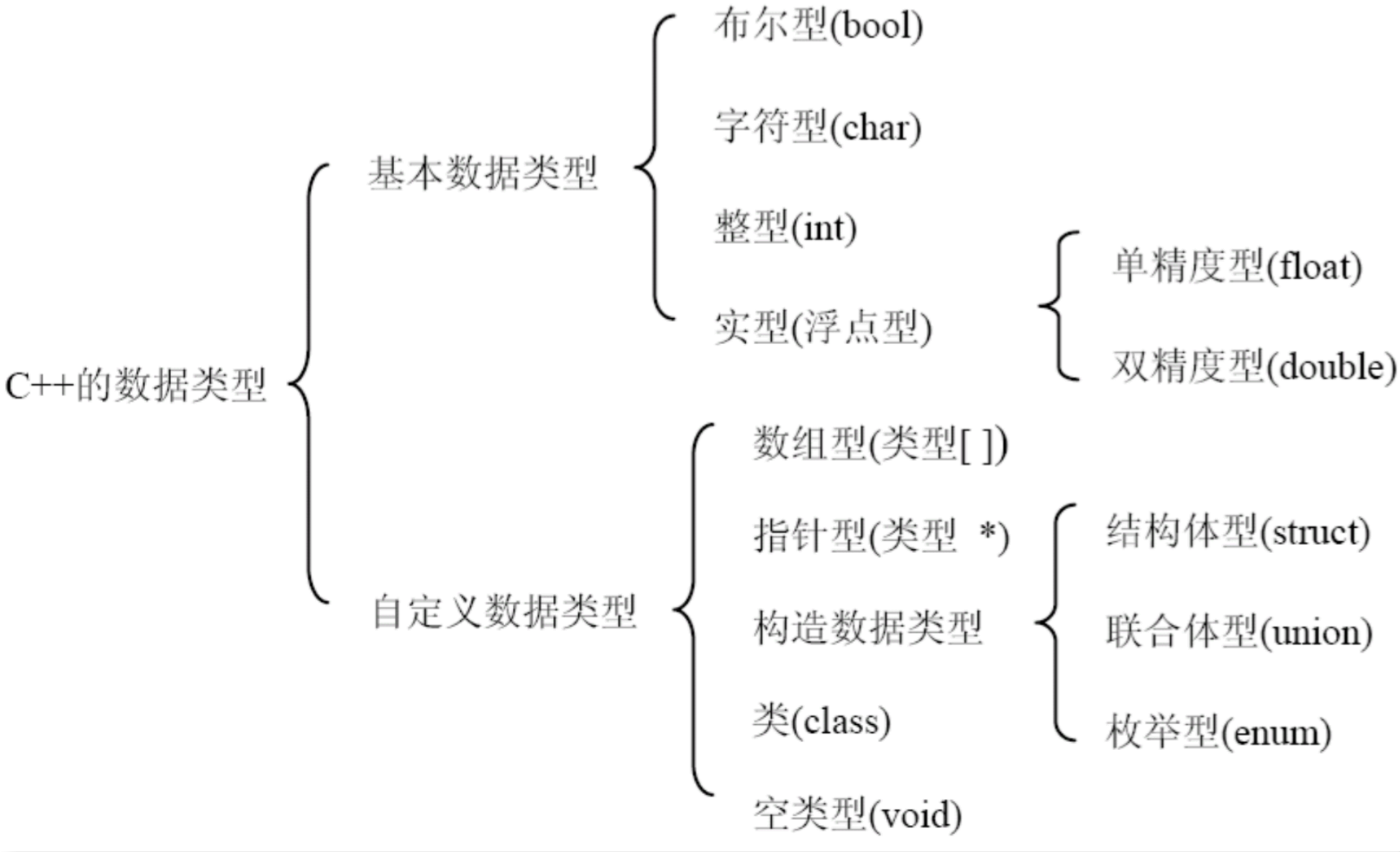
问题②：

是不是无论我们在程序里写什么“数”和“计算符号”，计算机都能明白？

NO!

计算机只能“看懂”某些类型的数据，这些“**数据类型**”和相应的“**操作符号**”也是定义好的。

计算机能够看懂的“**数据的类型**”



计算机能够理解的“运算的种类”

- 求字节数运算符：sizeof
- 下标运算符 []
- 赋值运算符 =
- 算术运算符 + - * / %
- 关系运算符 < > == >= <= !=
- 逻辑运算符 ! && ||
- 条件运算符 ? :
- 逗号运算符 ,
- 位运算符 >> ~ | ^ &
- 指针运算符 *, &
- 强制类型转换运算符：(类型)
- 分量运算符 . →

问题③：

世界上可能要用“程序来表达的逻辑”纷繁复杂，程序设计语言里面得有多少种“句式”才能够用？

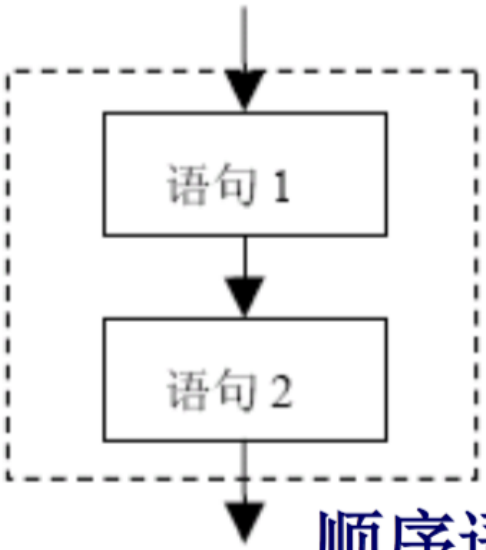
问题③：

世界上可能要用“程序来表达的逻辑”纷繁复杂，程序设计语言里面得有多少种“句式”才能够用？

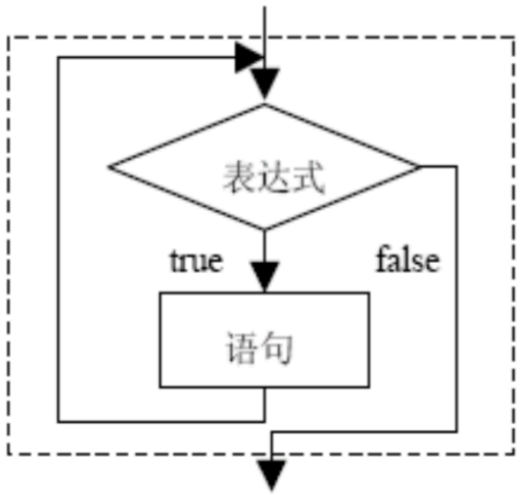
不多～三种而已！

Corrado Böhm, Giuseppe Jacopini.
Flow diagrams, turing machines and languages with only two formation rules
Communications of the ACM, 1966, 9(5):366–371.

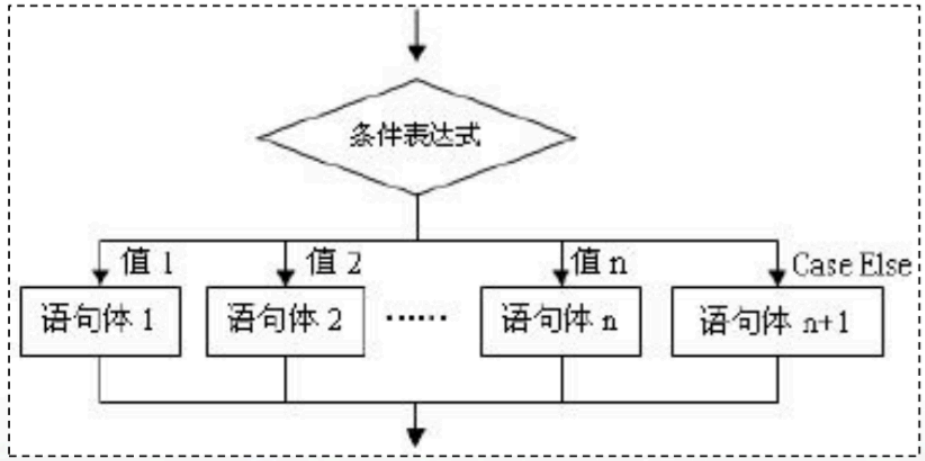
如何表达纷繁复杂的计算逻辑？



顺序语句



循环语句



分支语句

我们要学的编程语言

30 几个关键字

10 几种基本数据类型＋ 30 几个运算符号

3 种基本的逻辑语句

内容提要 I

1. 感性认识计算机程序
   - 说在前面的话
   - 程序是你告诉计算机的话
   - 如果你的大脑是台计算机
   - 如果你来设计一门编程语言
2. 快步走进C程序
   - 最简单的程序—“模仿”
   - 很简单的程序
   - 简单的程序
3. 从现实问题到计算机程序
   - 没有解决方案就没有程序
   - 先有构想再写程序
   - 体验结构化的程序

内容提要 I

1 感性认识计算机程序
- 说在前面的话
- 程序是你告诉计算机的话
- 如果你的大脑是台计算机
- 如果你来设计一门编程语言

2 快步走进C程序
- 最简单的程序—“模仿”
- 很简单的程序
- 简单的程序

3 从现实问题到计算机程序
- 没有解决方案就没有程序
- 先有构想再写程序
- 体验结构化的程序

程序中的“套话”

```
1 #include<iostream>
2 using namespace std;
3 int main()
4 {
5
6   return 0;
7 }
```

内容提要 I

1 感性认识计算机程序
- 说在前面的话
- 程序是你告诉计算机的话
- 如果你的大脑是台计算机
- 如果你来设计一门编程语言

2 快步走进C程序
- 最简单的程序—“模仿”
- 很简单的程序
- 简单的程序

3 从现实问题到计算机程序
- 没有解决方案就没有程序
- 先有构想再写程序
- 体验结构化的程序

很简单的程序①

- 定义变量
- 输出数据

```
1 #include<iostream>
2 using namespace std;
3 int main()
4 {
5   int a=0;
6   cout<<a<<endl;
7   return 0;
8 }
```

很简单的程序②

- 定义变量
- 输出数据

```
1 #include<iostream>
2 using namespace std;
3 int main()
4 {
5   int a=0;
6   cout<<" 我想输出 a 的值："<<a<<endl;
7   return 0;
8 }
```

很简单的程序③

- 定义变量
- 输入数据
- 输出数据

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5    int a=0;
6    cout<<" 请输入一个数："<<endl;
7    cin>>a;
8    cout<<" 我刚刚输入的 a："<<a<<endl;
9    return 0;
   }
```

内容提要 I

1 感性认识计算机程序
- 说在前面的话
- 程序是你告诉计算机的话
- 如果你的大脑是台计算机
- 如果你来设计一门编程语言

2 快步走进C程序
- 最简单的程序—“模仿”
- 很简单的程序
- 简单的程序

3 从现实问题到计算机程序
- 没有解决方案就没有程序
- 先有构想再写程序
- 体验结构化的程序

简单的程序①——顺序结构

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5    int a=0, b=0, result=0;
6    cout<<"Please input two numbers: ";
7    cin>>a>>b;
8    result=3*a-2*b+1;
9    cout<<"The result is: "<<result<<endl;
10   return 0;
11 }
```

简单的程序②——顺序结构

```
#include<iostream>
using namespace std;
int main()
{
  float a=0, b=0, temp=0;
  cout<<"Input a and b: "<<endl;
  cin>>a>>b;
  cout<<"a="<<a<<", b="<<b<<endl;
  temp=a; a=b; b=temp;
  cout<<"a="<<a<<", b="<<b<<endl;
  return 0;
}
```

简单的程序③—分支结构

```
#include<iostream>
using namespace std;
int main()
{
  int x=0, y=0;
  cin>>x>>y;
  if (x>y)
    cout<<"Max number is: "<<x<<endl;
  else
    cout<<"Max number is: "<<y<<endl;
  return 0;
}
```

简单的程序④——分支结构

```
#include<iostream>
using namespace std;
int main()
{
  char a='A';
  cout<<" 猜猜我是哪个字母："<<endl;
  cin>>a;
  if (a != 'G')
    cout<<" 你猜错了！ "<<endl;
  else if (a == 'G')
    cout<<" 被你猜中了！"<<endl;
  return 0;
}
```

简单的程序⑤——循环结构

```
#include<iostream>
using namespace std;
int main()
{
  int i=0;
  cout<<"20 以内的奇数："<<endl;
  for (i=0;i<20;i++)
    {
      if (i%2 != 0)
        cout<<i<<endl;
    }
  return 0;
}
```

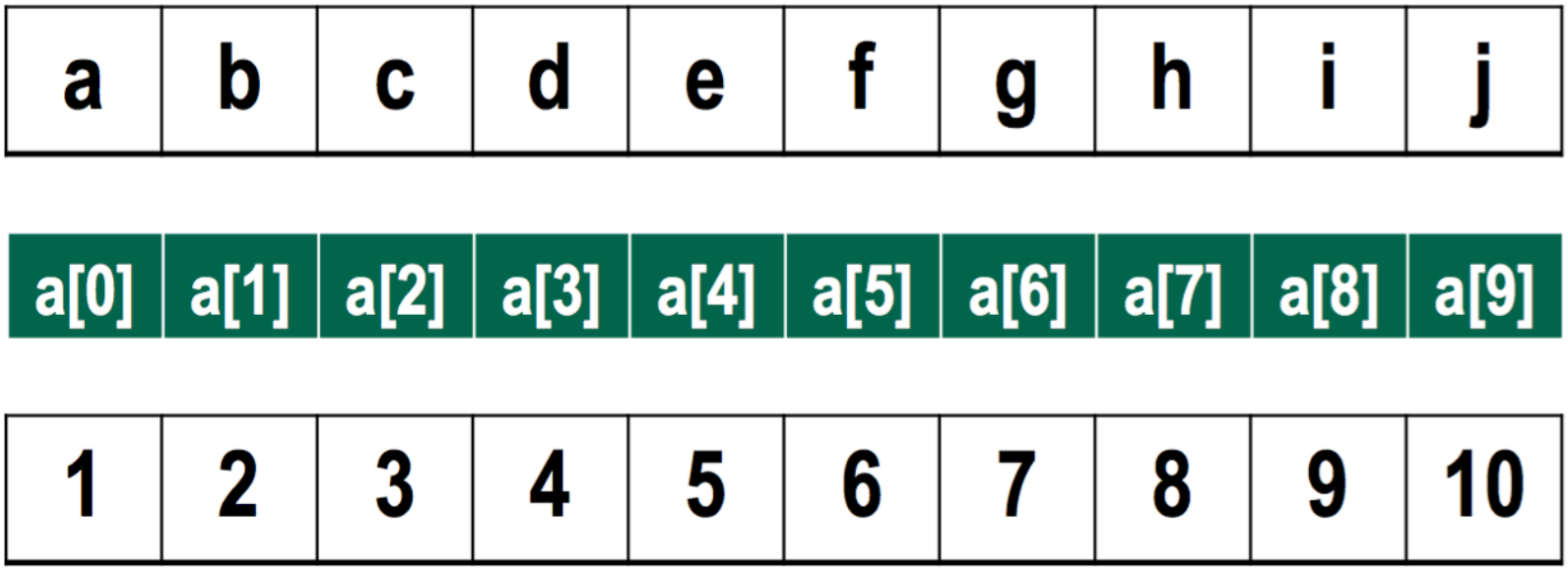
简单的程序⑥——循环结构

```
#include<iostream>
using namespace std;
int main()
{
  int i=0;
  cout<<" 从大到小输出 20 以内的奇数："<<endl;
  for (i=20;i>=0;i--)
    {
      if (i%2 != 0)
        cout<<i<<endl;
    }
  return 0;
}
```

简单的程序⑦——数组

```
#include<iostream>
using namespace std;
int main()
{
  int i=0;
  char a[10]={'a','b','c','d','e','f','g','h','i','j'};
  cout<<" 字母表中序号为奇数的前五个字母："<<endl;
  for (i=0;i<10;i=i+2)
    {
      cout<<a[i]<<endl;
    }
  return 0;
}
```

```
char a[10] = { 'a', 'b','c', 'd', 'e','f','g','h','i','j'};
```




```
int a[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
```

简单的程序⑧——综合

```
#include<iostream>
using namespace std;
int main()
{
  char a=' '; //用于存放用户输入的字母
  cout<<" 猜猜我是哪个字母，最多猜 5 次哦："<<endl;
  int i=0; //用于记录猜过多少次了
  for (i=0;i<5;i++)
    {
      cin>>a;
      if (a == 'G') //如果猜中
        {
          cout<<" 被你猜中了！"<<endl;
          break; //终止循环
        }
      else //如果没有猜中
        cout<<" 你猜错了！接着猜吧！"<<endl;
    }
  return 0;
}
```

什么样的程序是“好程序”？

我们重视

- √ “我的程序运行结果正确，解决了问题！”
- √ “我的程序更容易被别人看懂！”
- √ “我的程序结构更清楚，更漂亮！”

我们不重视

- × “我少使用了几个变量！”
- × “我的程序行数比较少！”
- × “我的程序运行起来快了一些！”

不积跬步

无以至千里

不积小流

无以成江海

荀子（约公元前313 - 前238）

——《劝学》

内容提要 I

1. 感性认识计算机程序
   - 说在前面的话
   - 程序是你告诉计算机的话
   - 如果你的大脑是台计算机
   - 如果你来设计一门编程语言
2. 快步走进C程序
   - 最简单的程序—“模仿”
   - 很简单的程序
   - 简单的程序
3. 从现实问题到计算机程序
   - 没有解决方案就没有程序
   - 先有构想再写程序
   - 体验结构化的程序

内容提要 I

1 感性认识计算机程序
- 说在前面的话
- 程序是你告诉计算机的话
- 如果你的大脑是台计算机
- 如果你来设计一门编程语言

2 快步走进C程序
- 最简单的程序—“模仿”
- 很简单的程序
- 简单的程序

3 从现实问题到计算机程序
- 没有解决方案就没有程序
- 先有构想再写程序
- 体验结构化的程序

程序—是你告诉计算机的话

计算机其实很笨

- **它可以按照“你告诉它的话去执行”**
- **它却不能帮你“想出”解决问题的办法！**

程序—是你告诉计算机的话

计算机其实很笨

- **它可以按照“你告诉它的话去执行”**
- **它却不能帮你“想出”解决问题的办法！**

程序—是你告诉计算机的话

计算机其实很笨

- 它可以按照“你告诉它的话去执行”
- 它却不能帮你“想出”解决问题的办法！

面对一个问题，你必须要先找到解决这个问题的办法，然后才有可能写出相应的程序！

从一个例子开始说起

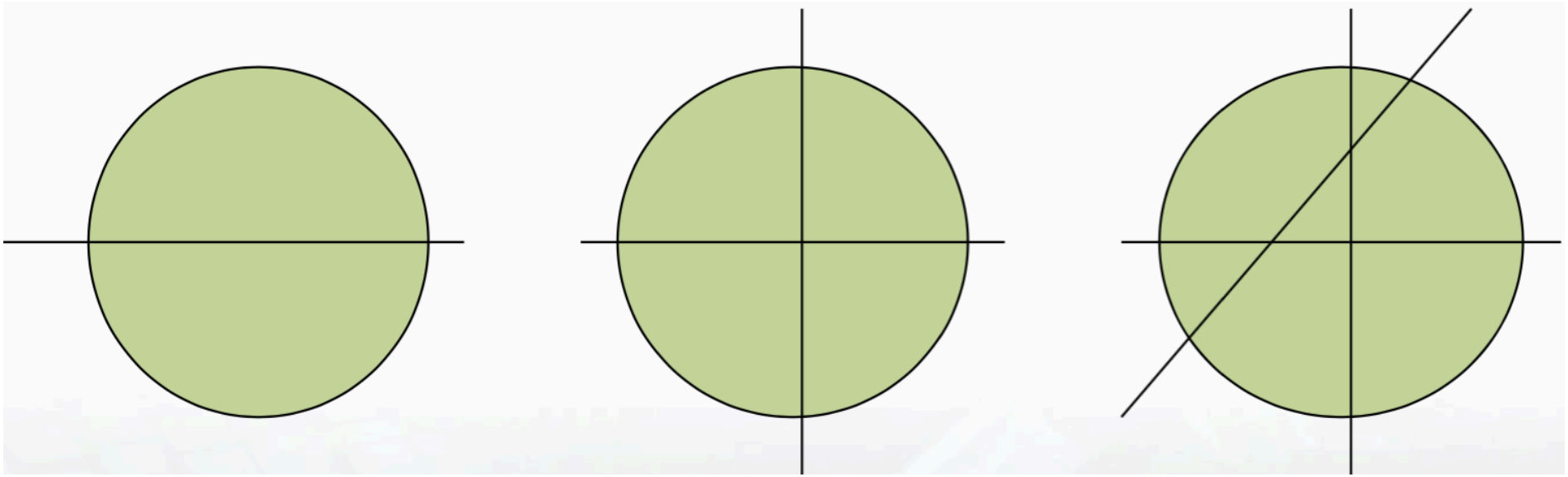
切饼

- 假设：有一张很大很大的饼，给你一把足够长的刀；
- 要求：每次在饼上切一刀；
- 问题：n 刀，最多能切出多少块饼？

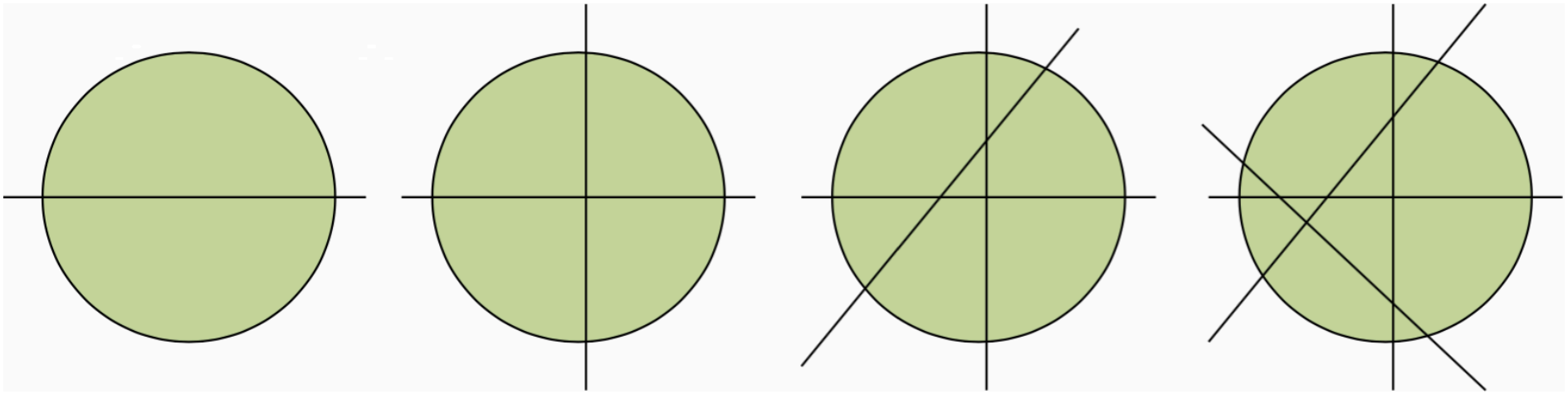
从一个例子开始说起

切饼

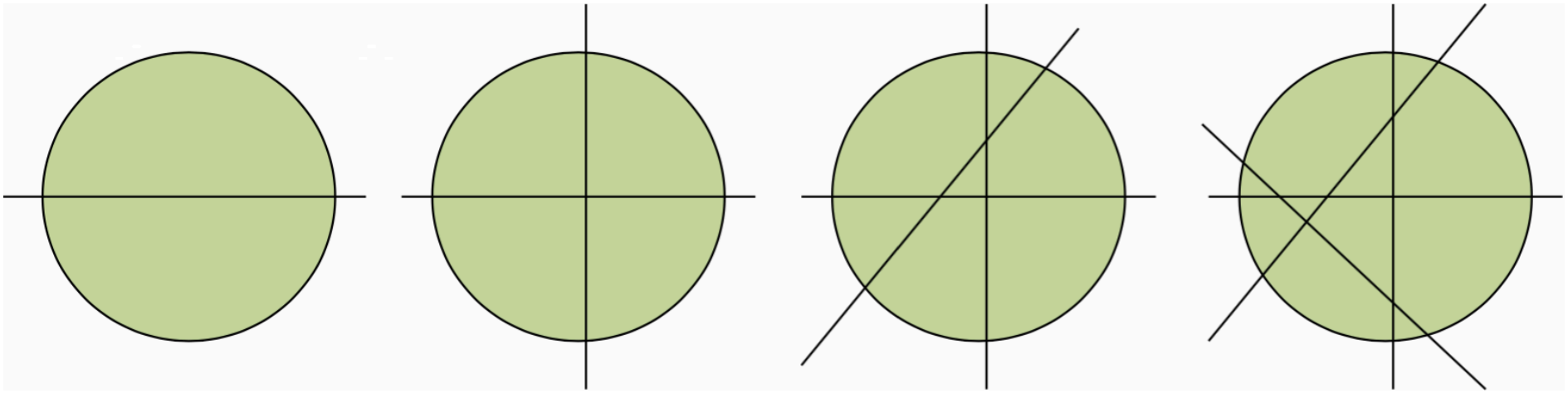
- 假设：有一张很大很大的饼，给你一把足够长的刀；
- 要求：每次在饼上切一刀；
- 问题：n 刀，最多能切出多少块饼？



从一个例子开始说起

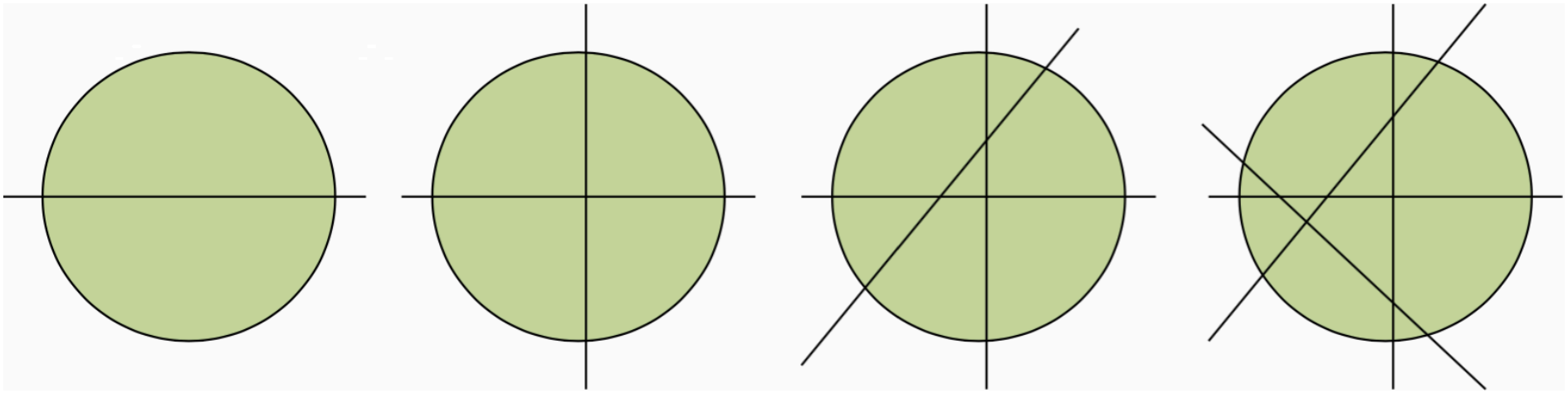


从一个例子开始说起



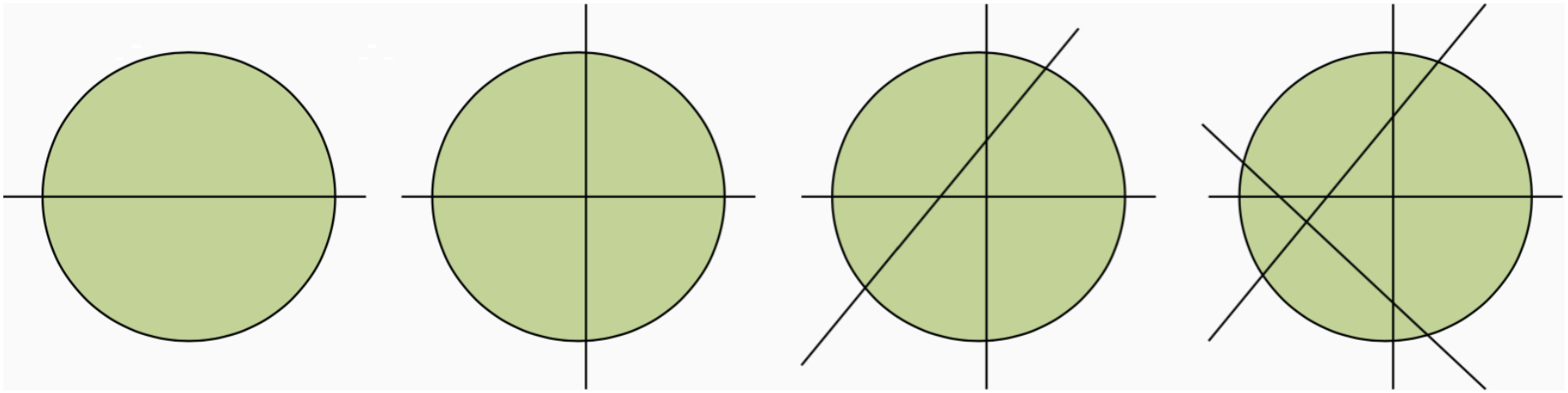
- $q(1) = 1 + 1 = 2$
- $q(2) = 1 + 1 + 2 = 4$
- $q(3) = 1 + 1 + 2 + 3 = 7$
- $q(4) = 1 + 1 + 2 + 3 + 4 = 11$
- $q(n) = q(n-1) + n, q(0) = 1$

从一个例子开始说起



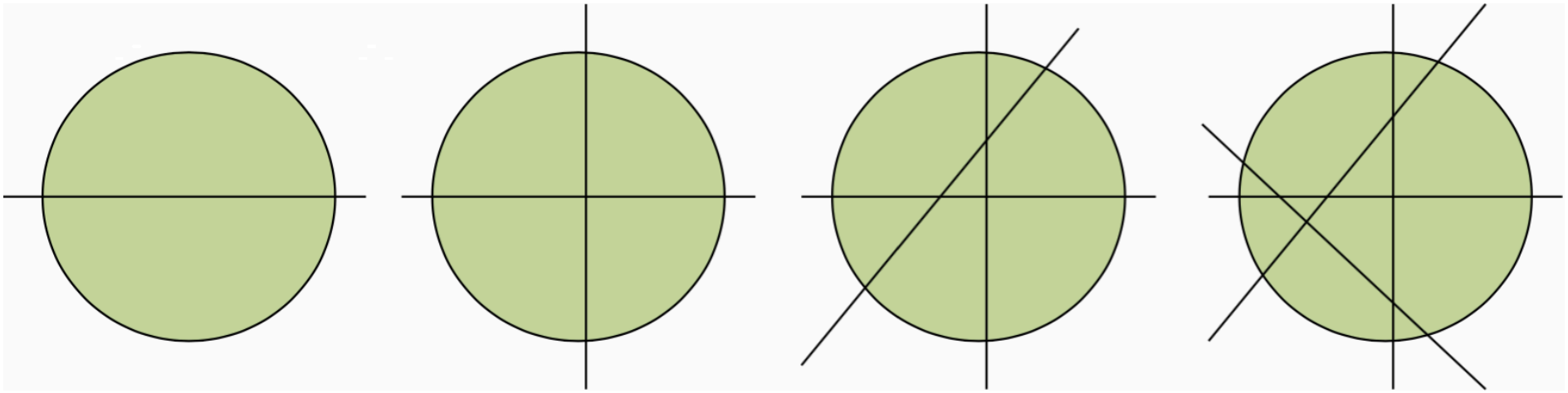
- $q(1) = 1 + 1 = 2$
- $q(2) = 1 + 1 + 2 = 4$
- $q(3) = 1 + 1 + 2 + 3 = 7$
- $q(4) = 1 + 1 + 2 + 3 + 4 = 11$
- $q(n) = q(n-1) + n, q(0) = 1$

从一个例子开始说起



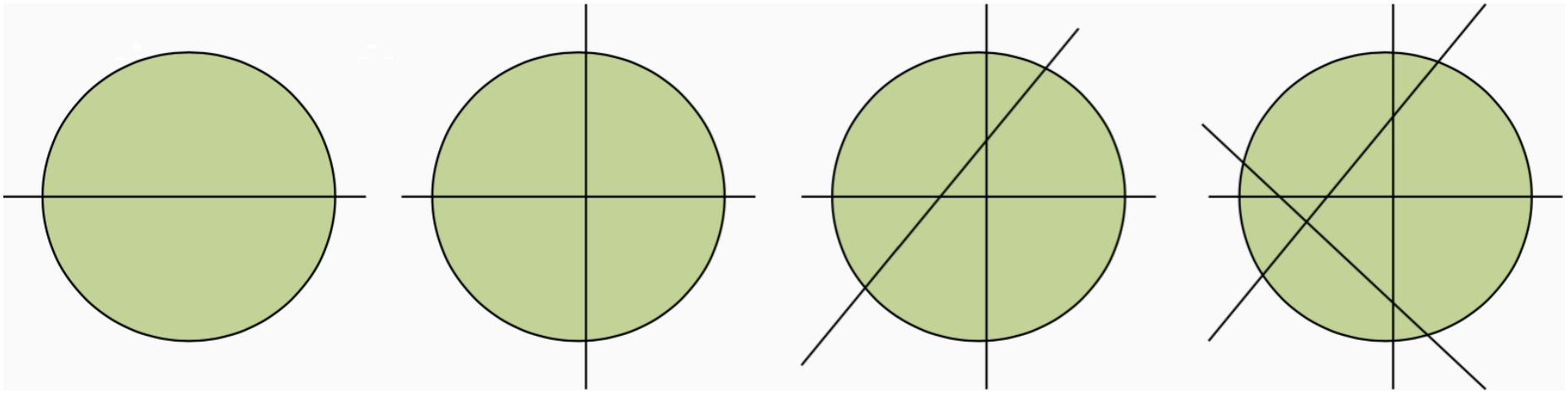
- $q(1) = 1 + 1 = 2$
- $q(2) = 1 + 1 + 2 = 4$
- $q(3) = 1 + 1 + 2 + 3 = 7$
- $q(4) = 1 + 1 + 2 + 3 + 4 = 11$
- $q(n) = q(n-1) + n, q(0) = 1$

从一个例子开始说起



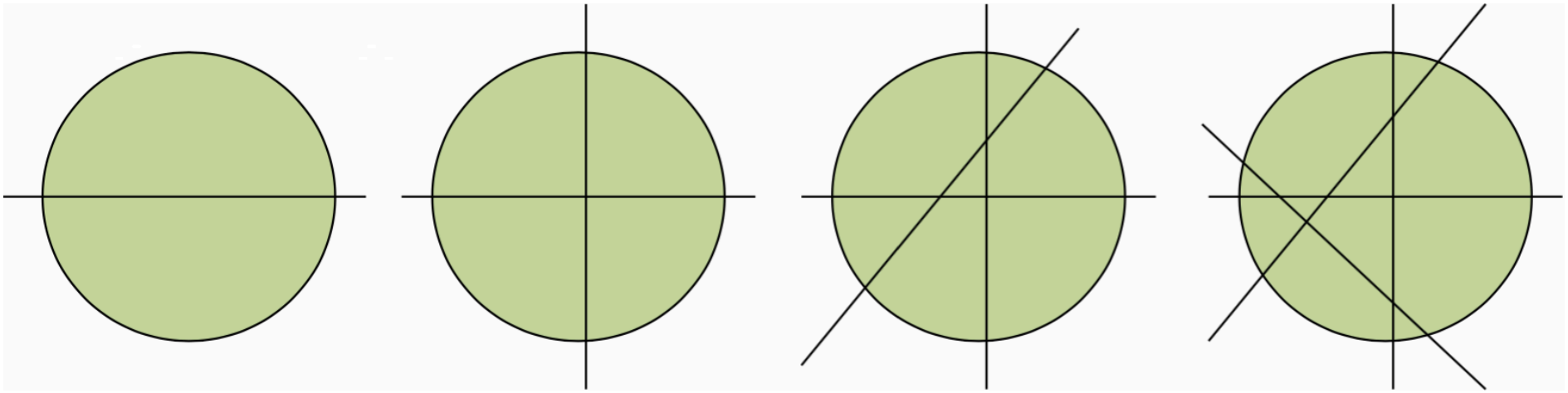
- $q(1) = 1 + 1 = 2$
- $q(2) = 1 + 1 + 2 = 4$
- $q(3) = 1 + 1 + 2 + 3 = 7$
- $q(4) = 1 + 1 + 2 + 3 + 4 = 11$
- $q(n) = q(n-1) + n, q(0) = 1$

从一个例子开始说起



- $q(1) = 1 + 1 = 2$
- $q(2) = 1 + 1 + 2 = 4$
- $q(3) = 1 + 1 + 2 + 3 = 7$
- $q(4) = 1 + 1 + 2 + 3 + 4 = 11$
- $q(n) = q(n-1) + n, q(0) = 1$

从一个例子开始说起



- $q(1) = 1 + 1 = 2$
- $q(2) = 1 + 1 + 2 = 4$
- $q(3) = 1 + 1 + 2 + 3 = 7$
- $q(4) = 1 + 1 + 2 + 3 + 4 = 11$
- $q(n) = q(n-1) + n, q(0) = 1$

- 第 n 刀切下去，最多比切之前多出 n 块；
- 第 1 刀切下去，得到 2 块。

在你还没有想到解决方案的时候，不要急着动手去写程序！

是不是有了解决方案就有程序了？

内容提要 I

1 感性认识计算机程序
- 说在前面的话
- 程序是你告诉计算机的话
- 如果你的大脑是台计算机
- 如果你来设计一门编程语言

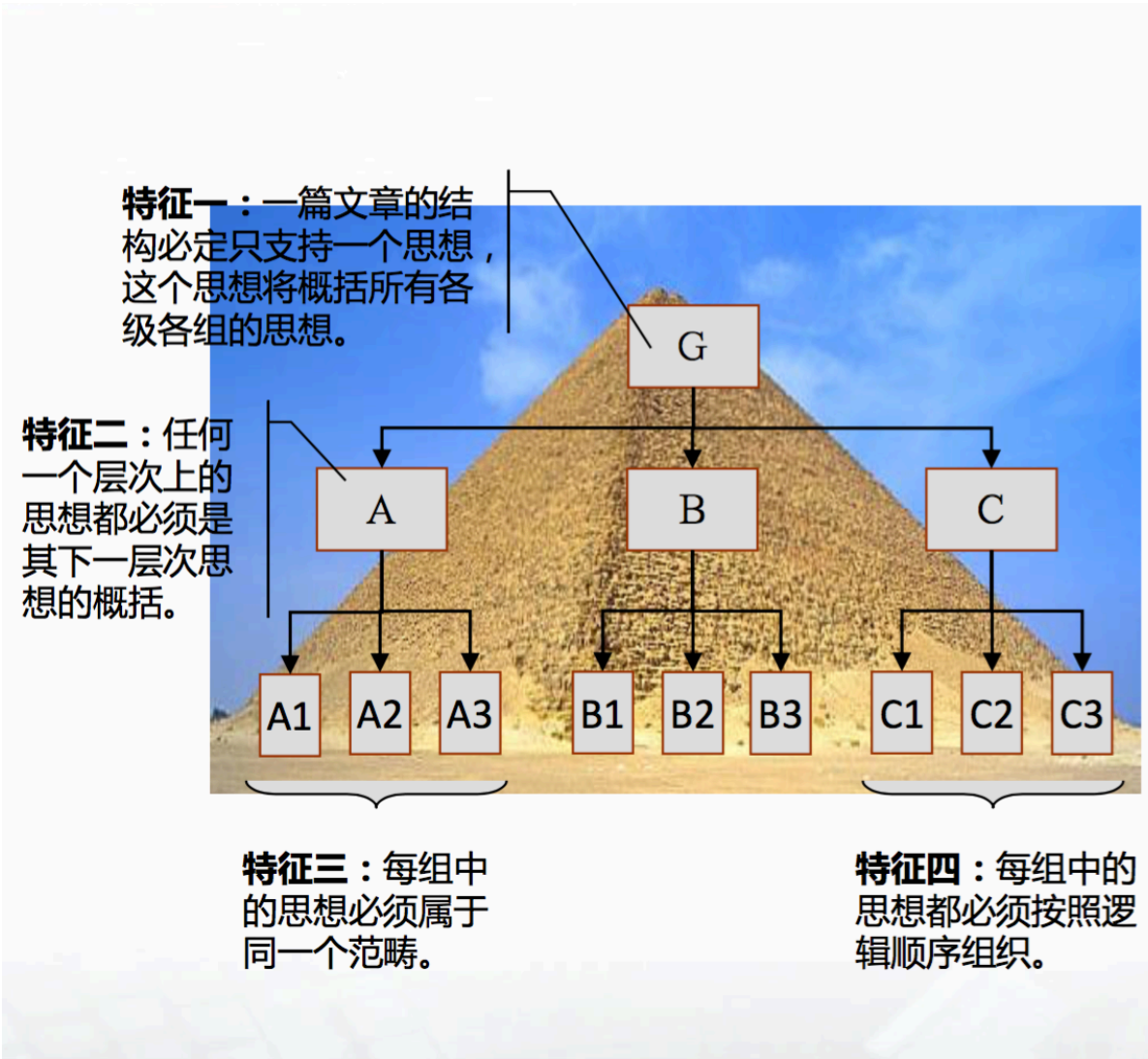
2 快步走进C程序
- 最简单的程序—“模仿”
- 很简单的程序
- 简单的程序

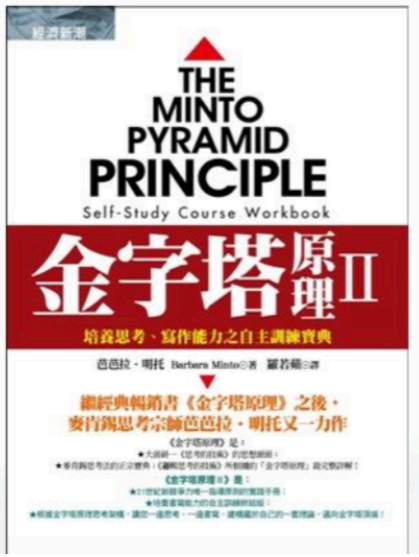
3 从现实问题到计算机程序
- 没有解决方案就没有程序
- 先有构想再写程序
- 体验结构化的程序

从解决方案到程序

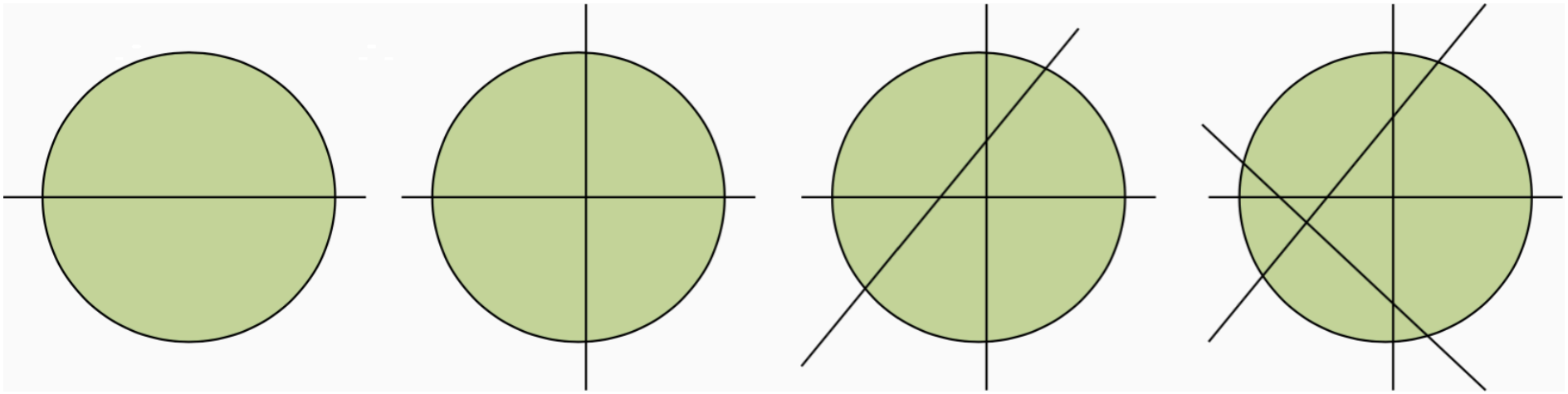
在**结构化程序设计**中，总是按照“**先粗后细**、**先抽象后具体**”的办法，对所要描述的解决方案进行穷尽分解，直到分解为**顺序**、**分支**、**循环**三种结构。

自然语言写作的规律

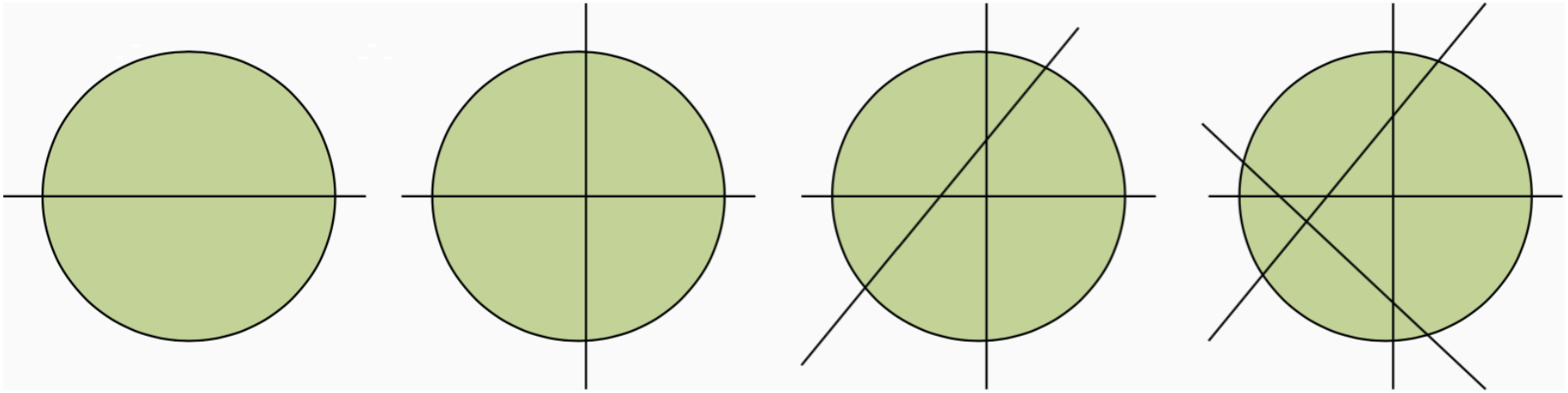

特征一：一篇文章的结构必定只支持一个思想，这个思想将概括所有各级各组的思想。
特征二：任何一个层次上的思想都必须是其下一层次思想的概括。
G
A
B
C
A1
A2
A3
B1
B2
B3
C1
C2
C3
特征三：每组中的思想必须属于同一个范畴。
特征四：每组中的思想都必须按照逻辑顺序组织。


THE MINTO PYRAMID PRINCIPLE
Self-Study Course Workbook
金字塔原理II
培養思考、寫作能力之自主訓練寶典

从一个例子开始说起

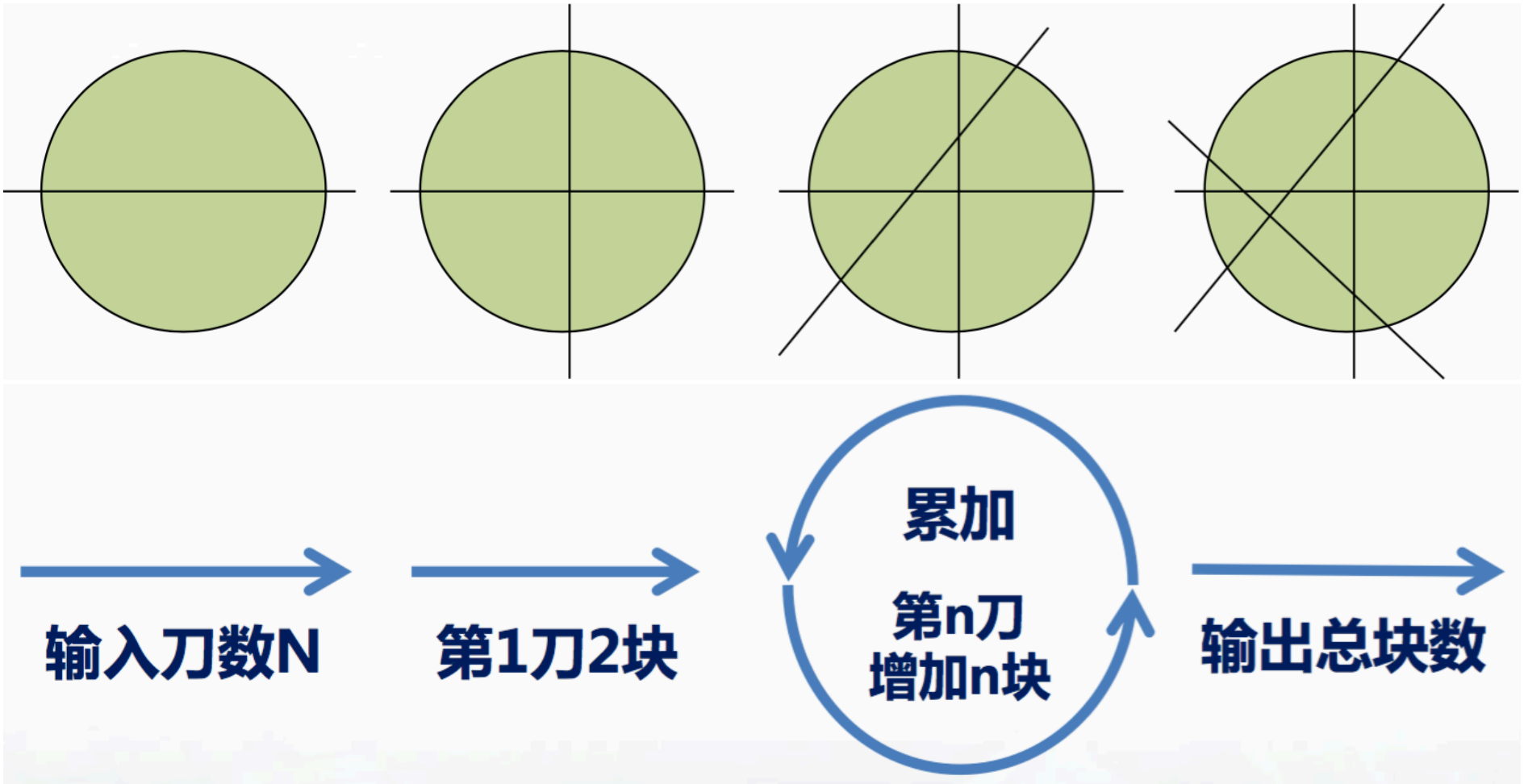


从一个例子开始说起



- $q(1) = 1 + 1 = 2$
- $q(2) = 1 + 1 + 2 = 4$
- $q(3) = 1 + 1 + 2 + 3 = 7$
- $q(4) = 1 + 1 + 2 + 3 + 4 = 11$
- $q(n) = q(n-1) + n, q(0) = 1$

从一个例子开始说起



从一个例子开始说起

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5    int blockCount = 0;
6    int i = 0, N = 0;
7    cin>>N;
8    blockCount = 1;
9    for (i=1;i<=N;i++)
10     blockCount = blockCount+i;
11   cout<<" 最多可切"<<blockCount<<" 块"<<endl;
12   return 0;
13 }
```

从这个例子我们知道

- **没有想好解决方案，不要急于动手写程序！**
- 有了解决方案以后，可以按照“**先粗后细**、**先抽象后具体**”的办法，先有程序的轮廓，如有必要可以借助“**建模工具**”画一些图，然后再动手写程序。
- 写程序时，可以**先写出程序轮廓**，而后再补变量定义等细节。

示例①：鸡兔同笼问题

问题描述

一个笼子里面关了鸡和兔子（鸡有 2 只脚，兔子有 4 只脚，没有例外）。已经知道了笼子里面脚的总数 a，问笼子里面至少有多少只动物，至多有多少只动物？

示例①：鸡兔同笼问题

问题描述

一个笼子里面关了鸡和兔子（鸡有 2 只脚，兔子有 4 只脚，没有例外）。已经知道了笼子里面脚的总数 a，问笼子里面至少有多少只动物，至多有多少只动物？

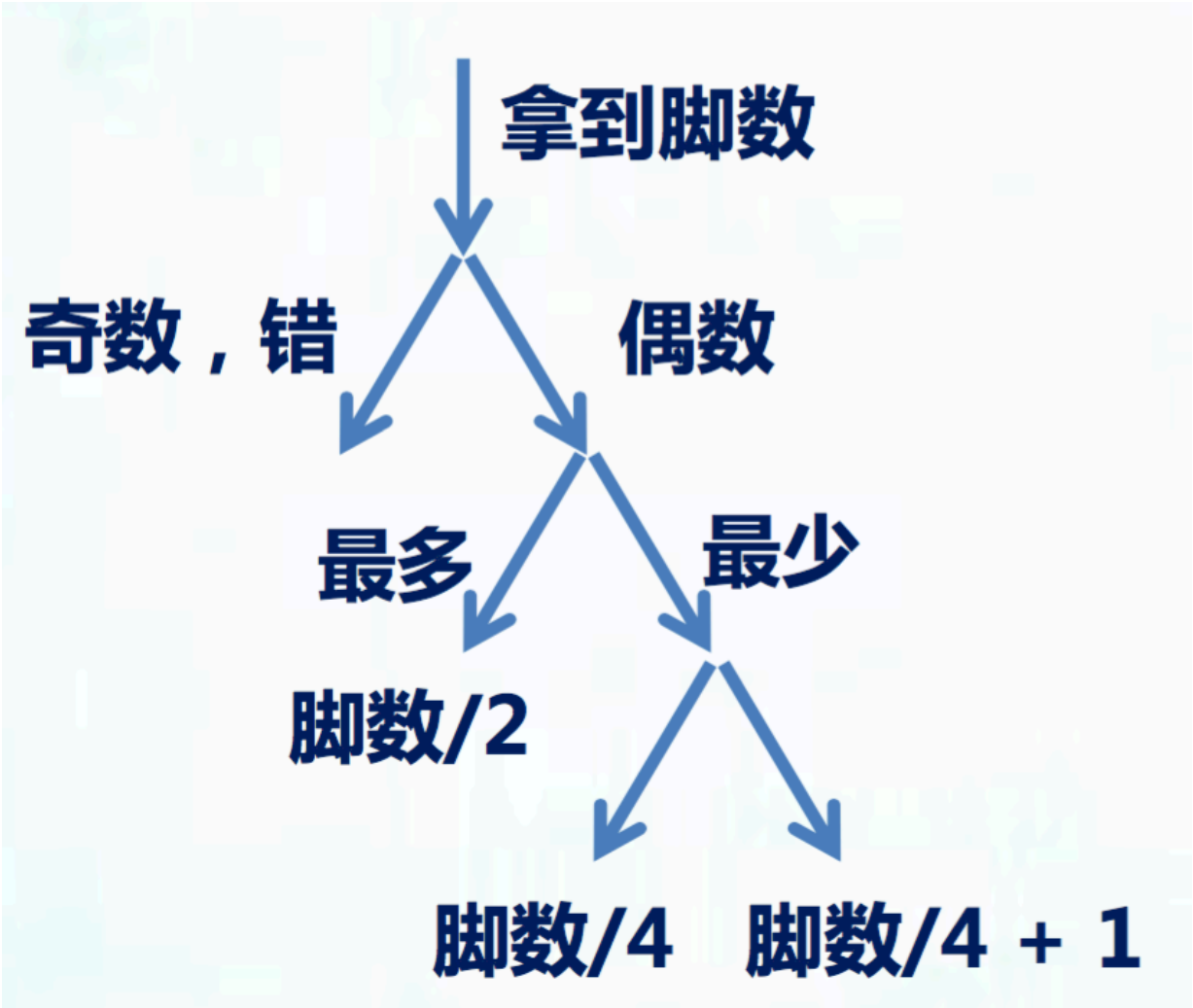
输入样例

```
3
20
22
```

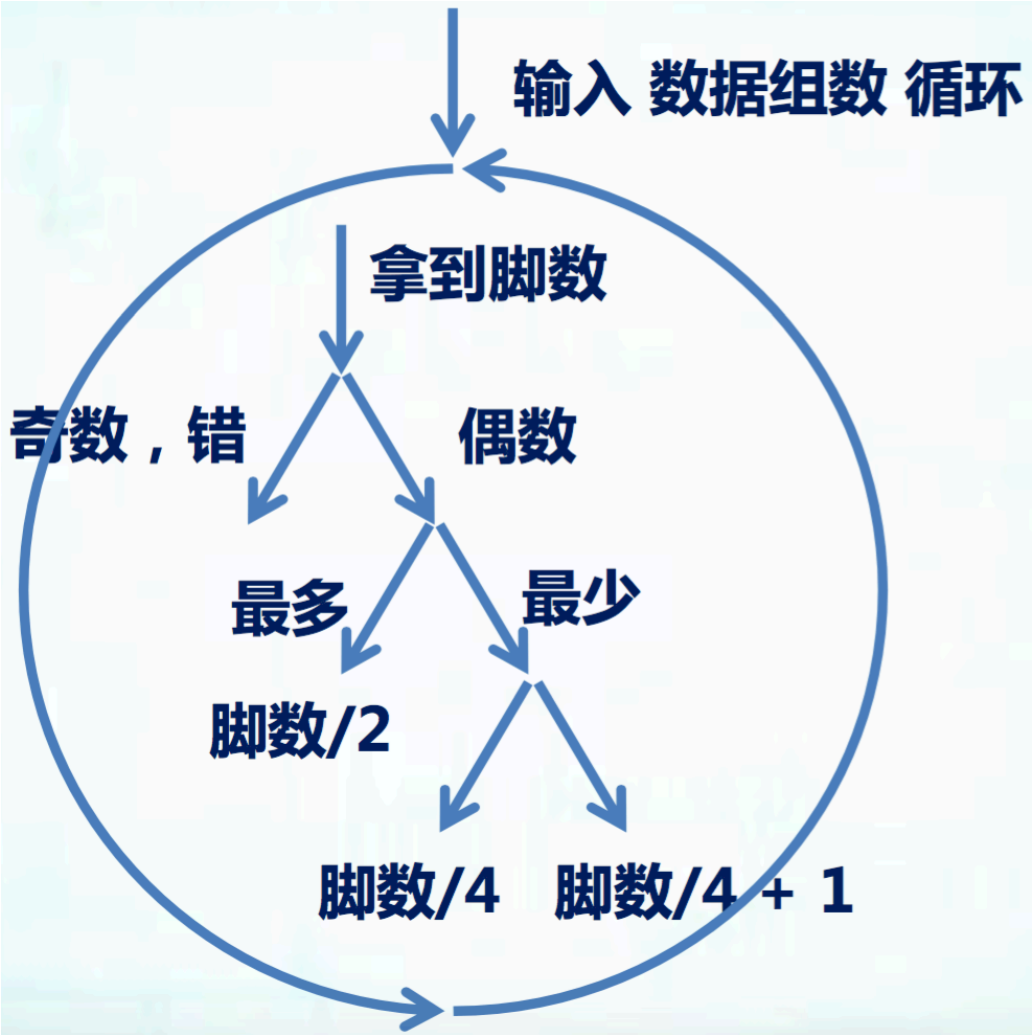
输出样例

```
0 0
5 10
6 11
```

示例①：鸡兔同笼问题



示例①：鸡兔同笼问题



```cpp
#include <iostream>
using namespace std;
int main()
{
  int nCases, i, nFeet;
  //nCases 表示输入测试数据的组数, nFeet 表示输入的脚数。
  cin >> nCases;
  for (i=0;i<nCases;i++) {
    cin >> nFeet;
    if (nFeet % 2 != 0) // 如果有奇数只脚, 则输入不正确,
                        // 因为不论 2 只还是 4 只, 都是偶数
      cout << "0 0" << endl;
    else if (nFeet % 4 != 0)
      //若要动物数目最少, 使动物尽量有 4 只脚
      //若要动物数目最多, 使动物尽量有 2 只脚
      cout << nFeet / 4 + 1 << " " << nFeet / 2 << endl;
    else
      cout << nFeet / 4 << " " << nFeet / 2 << endl;
  }
  return 0;
}
```

在思考解决方案的时候，记得利用计算机的特性——不怕啰嗦！☺

示例②：百元买百鸡问题

问题描述

假定母鸡每只 3 元，公鸡每只 2 元，小鸡每只 5 角。现在有 100 元钱要求买 100 只鸡，编程列出所有可能的购鸡方案。

示例②：百元买百鸡问题

问题描述

假定母鸡每只 3 元，公鸡每只 2 元，小鸡每只 5 角。现在有 100 元钱要求买 100 只鸡，编程列出所有可能的购鸡方案。

$$\begin{cases} x+y+z=100 \\ 3x+2y+0.5z=100 \end{cases}$$

示例②：百元买百鸡问题

问题描述

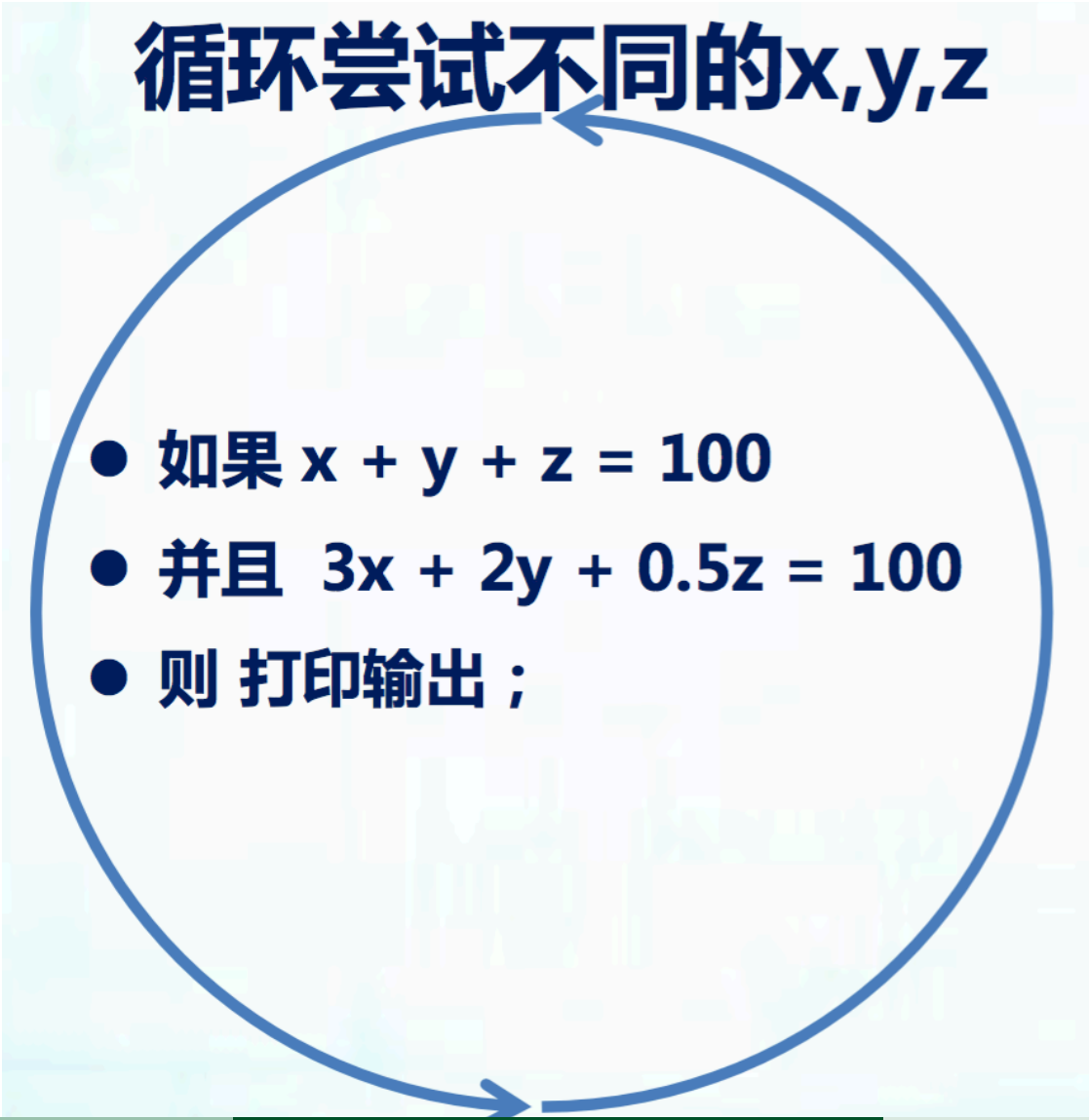
假定母鸡每只 3 元，公鸡每只 2 元，小鸡每只 5 角。现在有 100 元钱要求买 100 只鸡，编程列出所有可能的购鸡方案。

$$\begin{cases} x+y+z=100 \\ 3x+2y+0.5z=100 \end{cases}$$

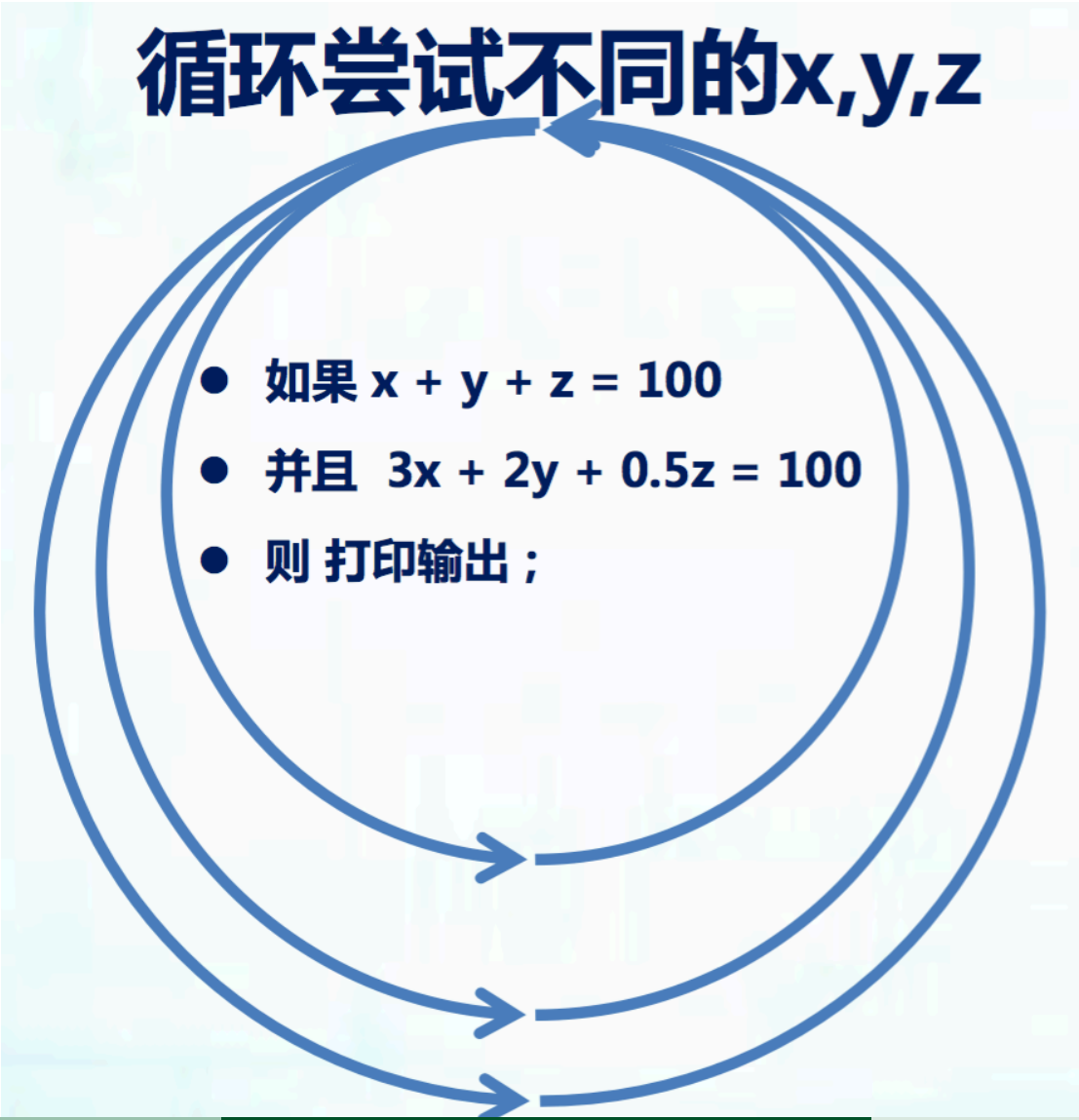
穷举

将可能出现的各种情况一一测试，判断是否满足条件。

示例②：百元买百鸡问题



示例②：百元买百鸡问题



解决方案

1: **for** each $x <= 33$ **do**
2: **for** each $y <= 50$ **do**
3: **for** each $z <= 100$ **do**
4: **if** $x + y + z = 100$ **and** $3x + 2y + 0.5z = 100$ **then**
5: output (x, y, z)
6: **end if**
7: **end for**
8: **end for**
9: **end for**

```cpp
#include <iostream>
using namespace std;
int main()
{
  int x, y, z;
  cout << "\t 母鸡\t\t 公鸡\t\t 小鸡" << endl;
  for (x = 0; x <= 33; x++)
    for (y = 0; y <= 50; y++)
      for (z = 0; z <= 100; z++)
        {
          if ((x + y + z) == 100)
            if ((3 * x + 2 * y + 0.5*z) == 100)
              cout << "\t" << x << "\t\t"
                   << y << "\t\t" << z << endl;
        }
  return 0;
}
```

稍作简化的解决方案

$$\begin{cases} x+y+z=100 \\ x<=33, y<=50, z<100 \end{cases}$$

稍作简化的解决方案

$$\begin{cases} x+y+z=100 \\ x<=33, y<=50, z<100 \end{cases}$$

```
1: for each x <= 33 do
2:     for each y <= 50 do
3:         z = 100 − x − y
4:         if 3x + 2y + 0.5z = 100 then
5:             output (x, y, z)
6:         end if
7:     end for
8: end for
```

```cpp
#include <iostream>
using namespace std;
int main()
{
  int x, y, z;
  cout << "\t 母鸡\t\t 公鸡\t\t 小鸡" << endl;
  for (x = 0; x <= 33; x++)
    for (y = 0; y <= 50; y++)
      {
        z = 100 - x - y;
        if ((3 * x + 2 * y + 0.5*z) == 100)
          cout << "\t" << x << "\t\t"
               << y << "\t\t" << z << endl;
      }
  return 0;
}
```

内容提要 I

1 感性认识计算机程序
- 说在前面的话
- 程序是你告诉计算机的话
- 如果你的大脑是台计算机
- 如果你来设计一门编程语言

2 快步走进C程序
- 最简单的程序—“模仿”
- 很简单的程序
- 简单的程序

3 从现实问题到计算机程序
- 没有解决方案就没有程序
- 先有构想再写程序
- 体验结构化的程序

示例③：分出奇偶数

问题描述

从键盘上输入10个整数，请将其中的奇数和偶数识别出来，分别放入不同的数组，并输出。

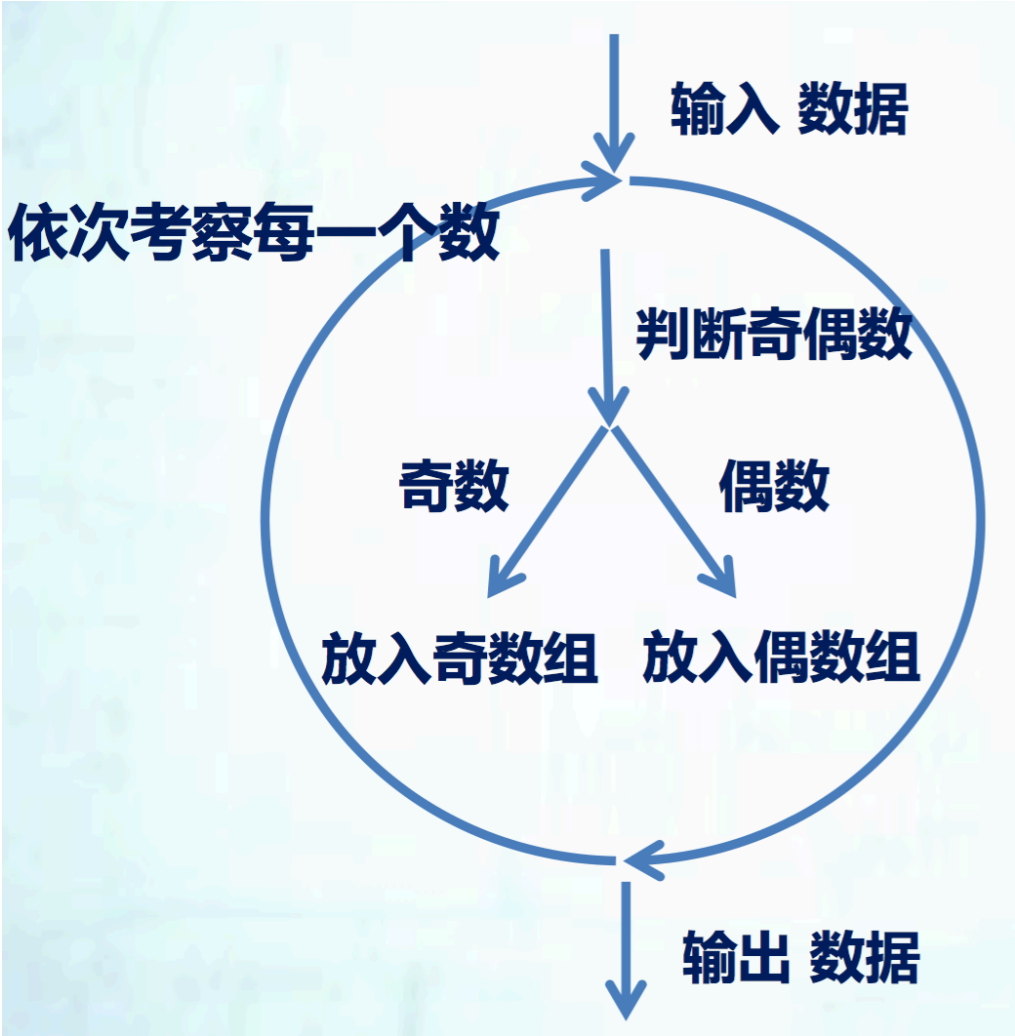
输入样例

```
1 23 34 65 43 67 12 67 341 61 34
```

输出样例

```
1 奇数：23 65 43 67 67 341 61
2 偶数：34 12 34
```

示例③：分出奇偶数



```cpp
#include<iostream>
using namespace std;
int main() {
  int all[10], odd[10], even[10]; //odd 记录奇数、even 记录偶数
  int i = 0, j = 0; //i, j 为循环计数变量
  for (; i < 10; i++) //输入 10 个数
    cin >> all[i];
  int numOdd = 0; //numOdd, numEven 分别记录奇数、偶数的个数
  int numEven = 0;
  for (i = 0; i < 10; i++) //遍历数组 all, 奇数放入 odd, 偶数放入 even
    {
      if (all[i] % 2 != 0) {//奇数
        odd[numOdd] = all[i];
        numOdd++;
      } else {//偶数
        even[numEven] = all[i];
        numEven++;
      }
    }
  for (i = 0; i < numOdd; i++)//输出奇数
    cout << odd[i] << " ";
  cout << endl;
  for (i = 0; i < numEven; i++)//输出偶数
    cout << even[i] << " ";
  return 0;
}
```

示例④：整数排序

问题描述

从键盘上输入10个整数，请按照从大到小的顺序将它们排列好，并按新的次序输出到屏幕上。

输入样例

```
23 34 65 43 67 12 67 341 61 34
```

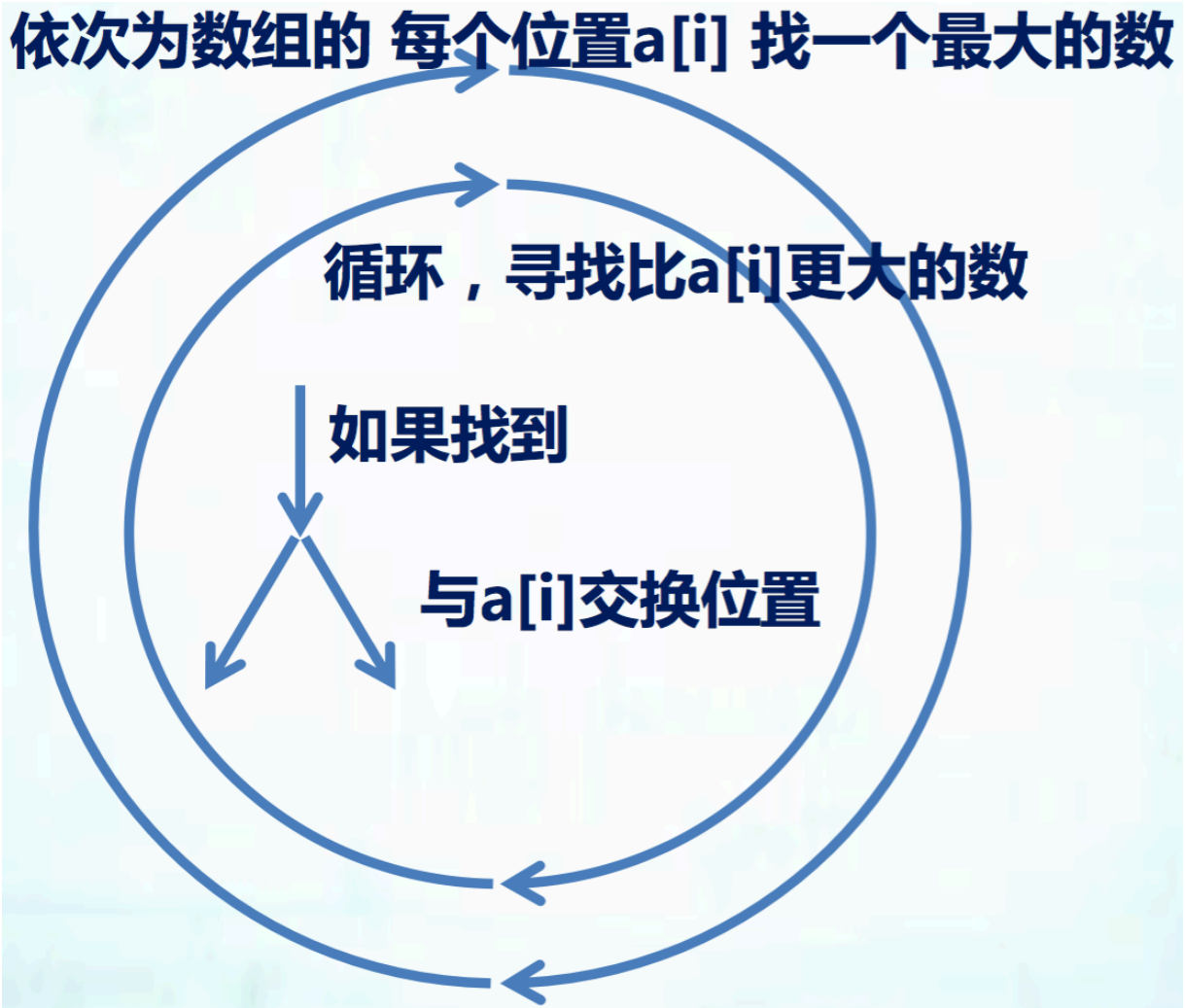
输出样例

```
341 67 67 65 61 43 34 34 23 12
```

示例④：整数排序

4	2	7	1	6	9	5	3	8	3
7	2	4	1	6	9	5	3	8	3
9	2	4	1	6	7	5	3	8	3
9	4	2	1	6	7	5	3	8	3
9	6	2	1	4	7	5	3	8	3
9	7	2	1	4	6	5	3	8	3
9	8	2	1	4	6	5	3	7	3

示例④：整数排序



```cpp
#include <iostream>
using namespace std;
int main() {
  int a[10]; //用于存放输入的数据
  int i = 0, j = 0; //用于循环计数
  int temp = 0; //临时变量，用于暂存要交换的数据
  for (i = 0; i < 10; i++) //依次输入 10 个待排序的数据
    cin >> a[i];
  for (i = 0; i < 9; i++) //依次为数组的第 i 个元素选择最大的数
    {
      for (j = i + 1; j < 10; j++) //从第 i+1 个元素开始寻找比 a[i] 更大的数
        {
          if (a[j] > a[i]) {
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;
          }
        }
    }
  //如果找到比 a[i] 更大的数，就将它与 a[i] 互换
  for (i = 0; i < 10; i++) //输出最终的排序结果
    cout<< a[i]<<" ";
  return 0;
}
```

示例⑤：整数奇偶排序

问题描述

输入10个0～100之间的不同整数，彼此以空格分隔，重新排序以后输出（也按空格分隔），要求：

1. 先输出其中的奇数，并按从大到小排列；
2. 然后输出其中的偶数，并按从小到大排列。

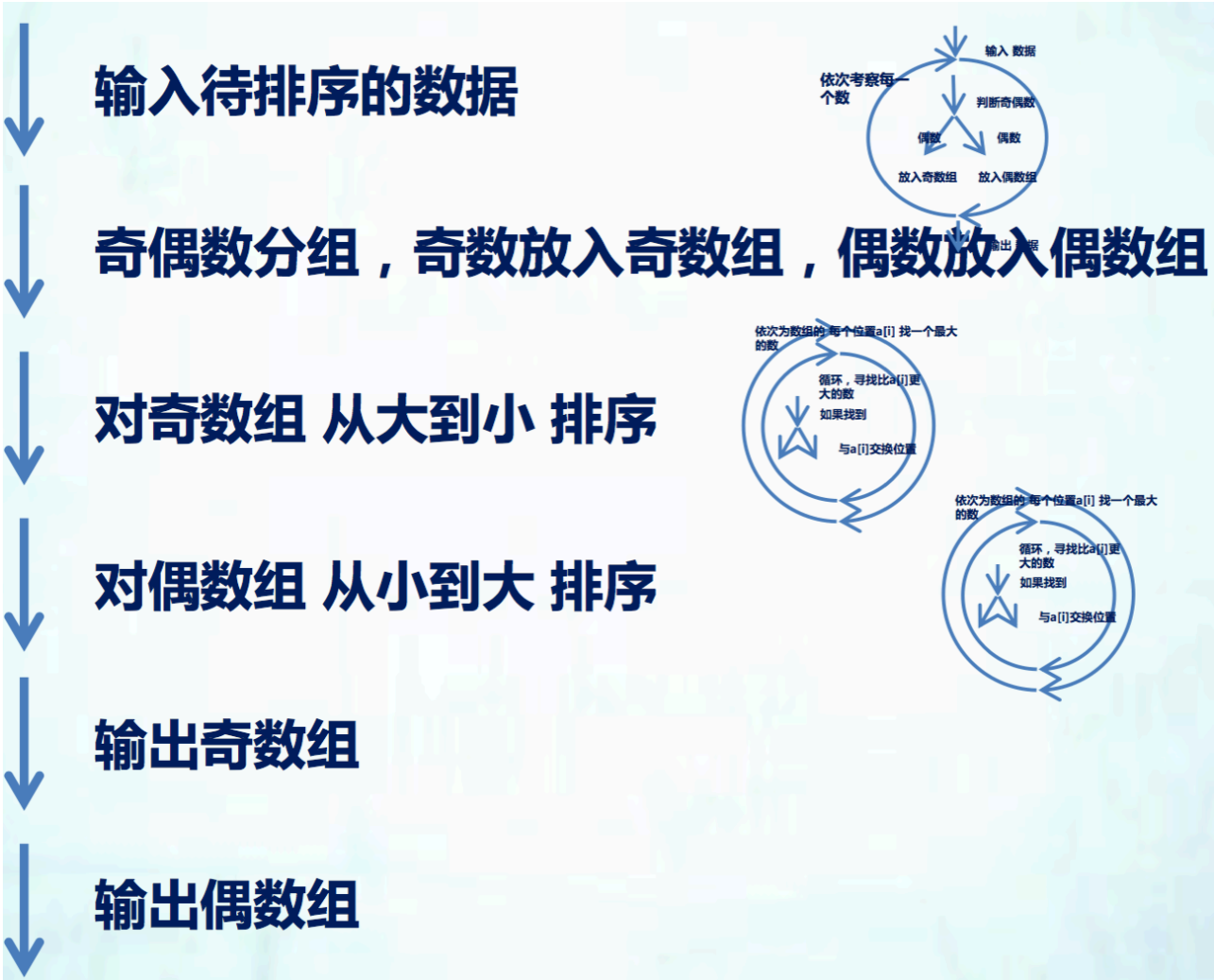
输入

任意排序的10个整数（0～100），彼此以空格分隔

输出

按照要求排序后输出，由空格分隔

示例⑤：整数奇偶排序


输入待排序的数据
奇偶数分组，奇数放入奇数组，偶数放入偶数组
对奇数组 从大到小 排序
对偶数组 从小到大 排序
输出奇数组
输出偶数组
输入 数据
依次考察每一个数
判断奇偶数
偶数
偶数
放入奇数组
放入偶数组
输出 数据
依次为数组的 每个位置a[i] 找一个最大的数
循环，寻找比a[i]更大的数
如果找到
与a[i]交换位置
依次为数组的 每个位置a[i] 找一个最大的数
循环，寻找比a[i]更大的数
如果找到
与a[i]交换位置

```cpp
#include<iostream>
using namespace std;
int main() {
  //定义变量
  //all 为全部十个数：odd 记录奇数、even 记录偶数，odd、even 至多 10 个
  int all[10], odd[10], even[10];
  //i, j 为循环变量
  int i = 0, j = 0;
  //依次输入 10 个数至 all，i 为 all 的下标
  for (; i < 10; i++)
    cin >> all[i];
  //numOdd, numEven 分别记录奇数、偶数的个数
  int numOdd = 0;
  int numEven = 0;
  //遍历数组 all，如果当前 all[i] 为奇数则放入 odd[numOdd],
  //偶数放入 even[numEven]
  for (i = 0; i < 10; i++) {
    if (all[i] % 2 != 0) {//奇数
      odd[numOdd] = all[i];
      numOdd++;
    } else {//偶数
      even[numEven] = all[i];
      numEven++;
    }
  }
```

```
  //对 odd 选择排序
  for (i = 0; i < numOdd - 1; i++) {
    for (j = i; j < numOdd; j++) {
      if (odd[j] > odd[i]) {
        //tmp 为临时变量
        int tmp = odd[i];
        odd[i] = odd[j];
        odd[j] = tmp;
      }
    }
  }
  //对 even 选择排序
  for (i = 0; i < numEven - 1; i++) {
    for (j = i; j < numEven; j++) {
      if (even[j] < even[i]) {
        //tmp 为临时变量
        int tmp = even[i];
        even[i] = even[j];
        even[j] = tmp;
      }
    }
  }
  //输出奇数
  for (i = 0; i < numOdd; i++) {
    cout << odd[i] << " ";
  }
  cout << endl;
  //输出偶数
  for (i = 0; i < numEven; i++) {
    cout << even[i] << " ";
  }
  cout << endl;
  return 0;
}
```

通过这些例子我们知道

写程序的过程

按照由大到小、由粗到精、由抽象到具体的方法分析、编写程序

通过这些例子我们知道

写程序的过程

按照由大到小、由粗到精、由抽象到具体的方法分析、编写程序

程序的结构

- 程序由若干个“模块”组成；
- 模块之内“高内聚”；
- 模块之间“低耦合”。

通过这些例子我们知道

写程序的过程

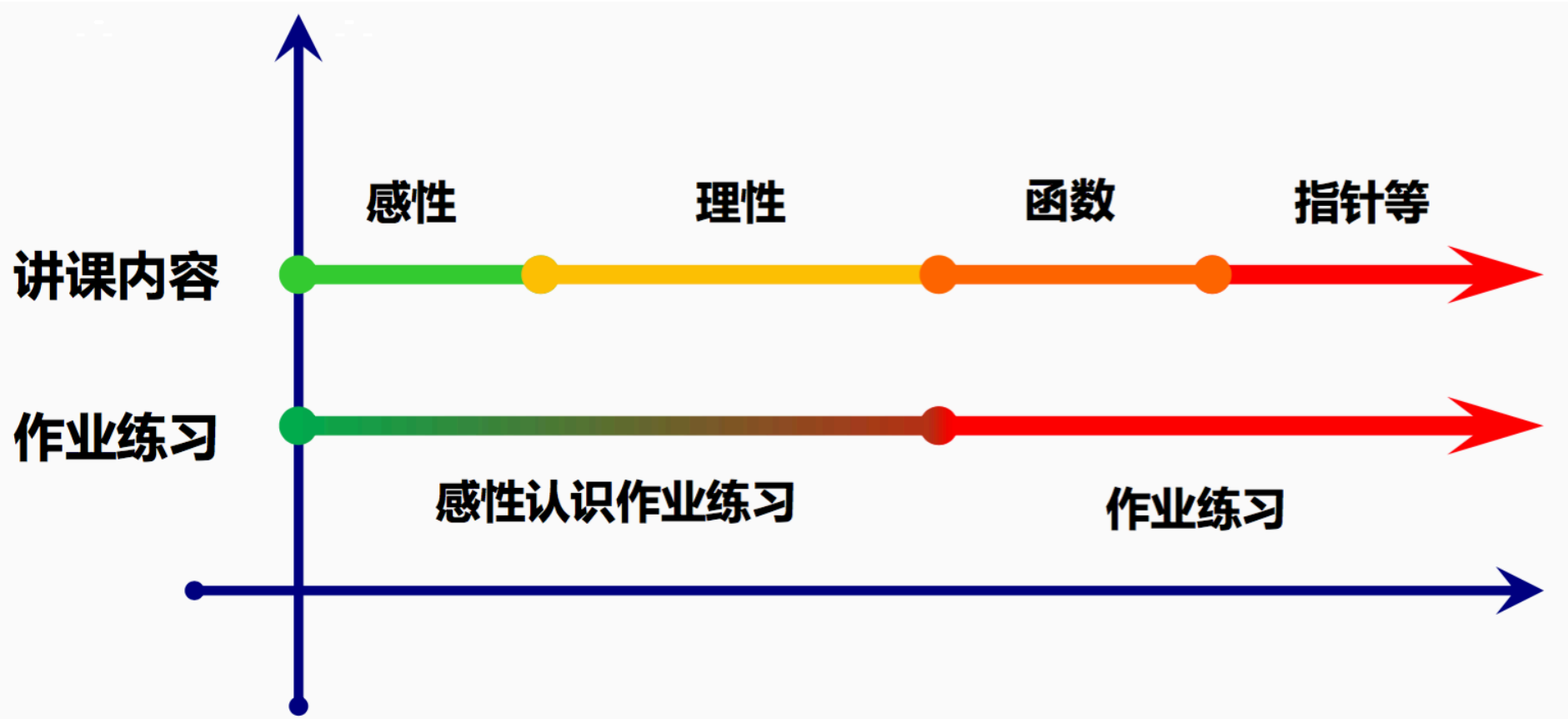
按照由大到小、由粗到精、由抽象到具体的方法分析、编写程序

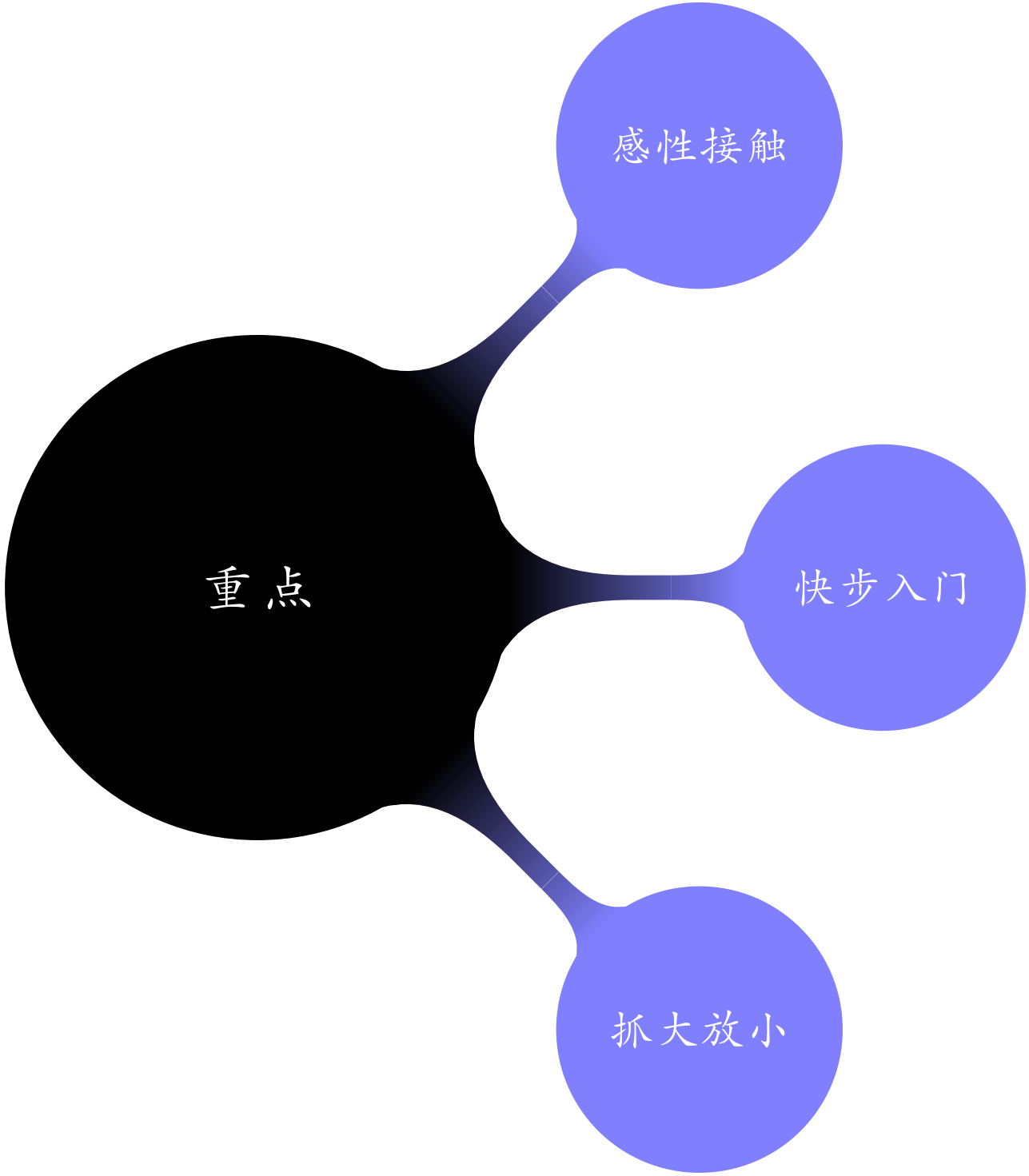
程序的结构

- 程序由若干个“模块”组成；
- 模块之内“高内聚”；
- 模块之间“低耦合”。

“结构化程序设计”的基本思想！

接下来的学习进度




感性接触
重点
快步入门
抓大放小