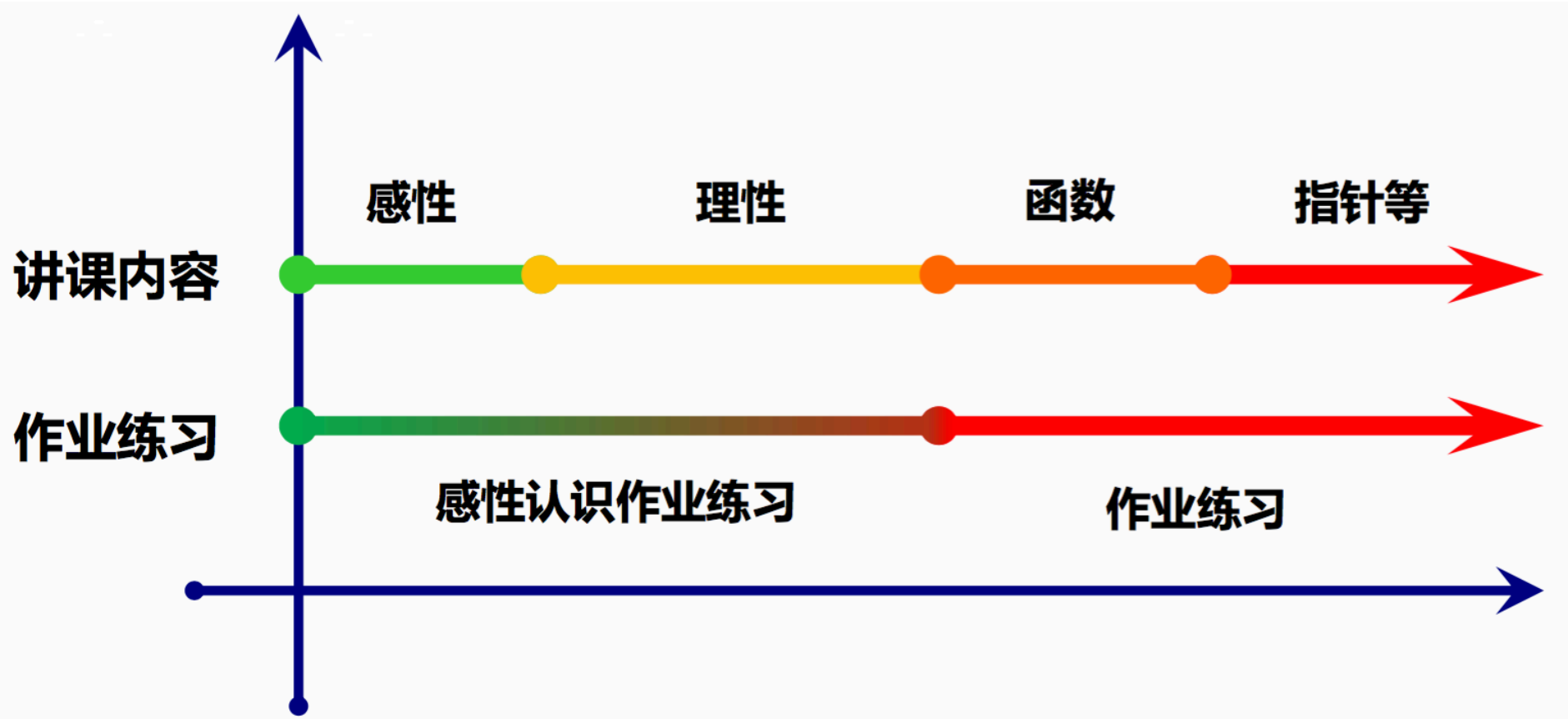
2 理性认识计算机程序

郑海永
zhenghaiyong@ouc.edu.cn

中国海洋大学 电子工程学院

接下来的学习进度



目录 I

目录 II

目录 III

内容提要 I

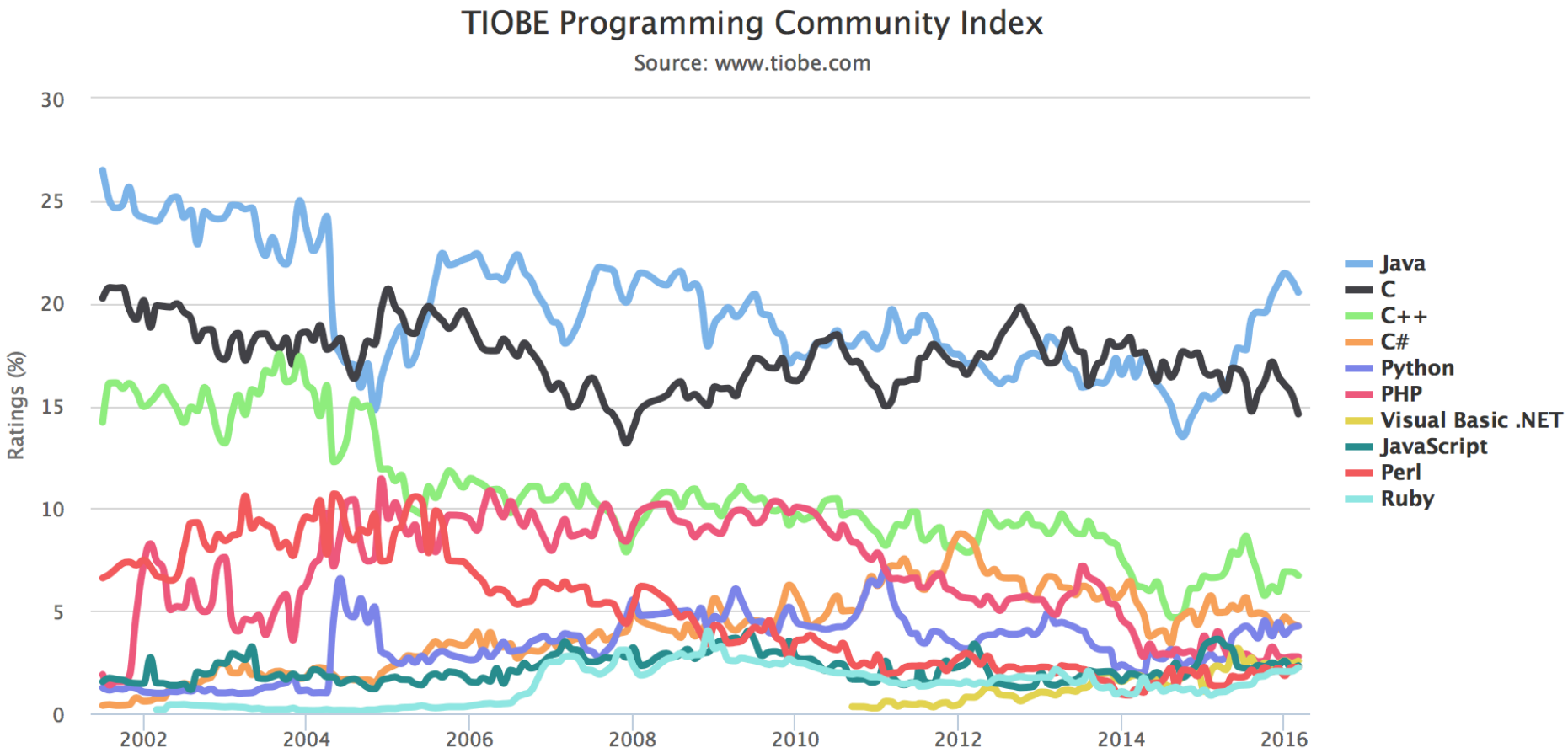
内容提要 II

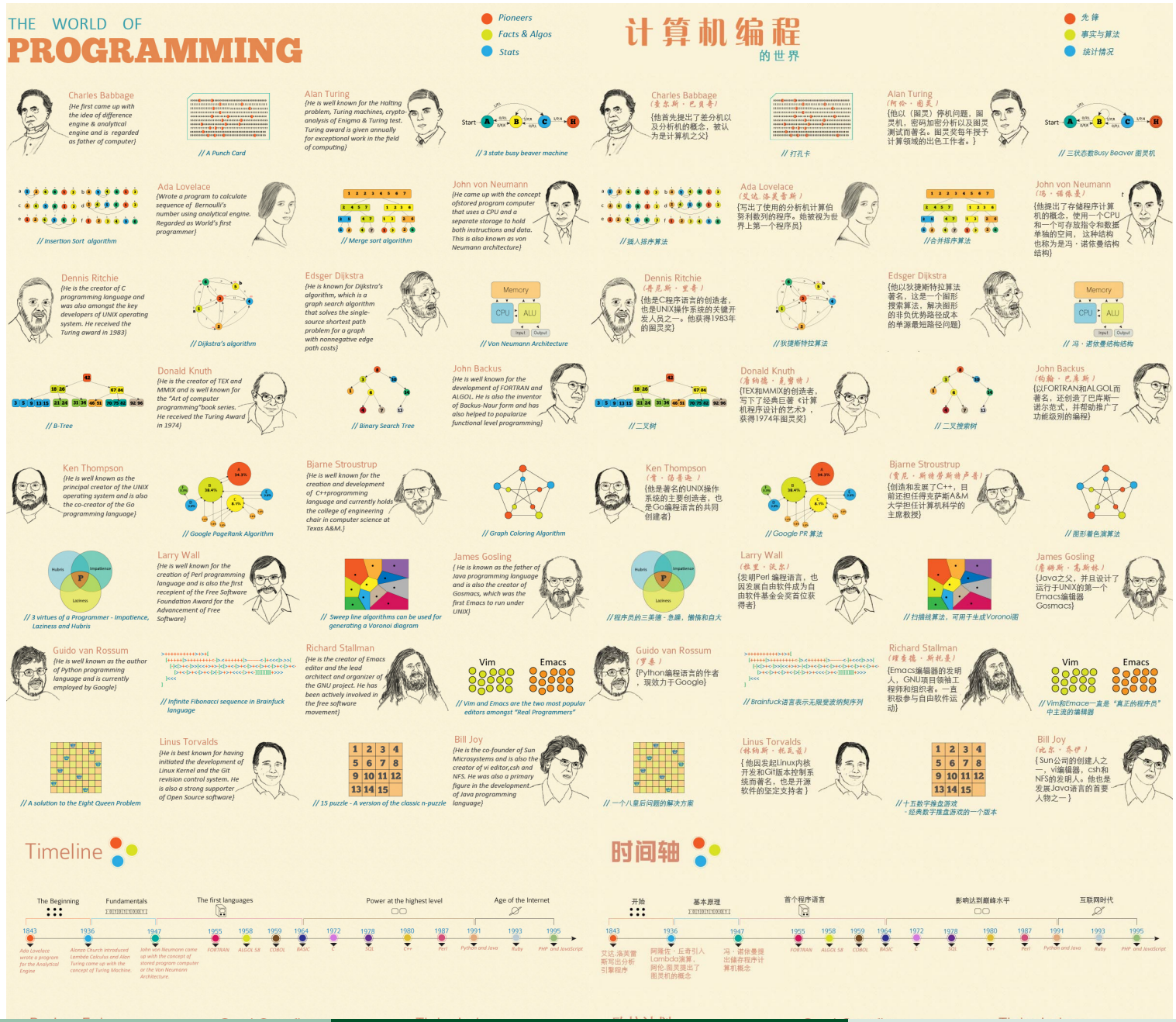
内容提要 III

- 字符串数组
- 输入与输出
- 示例


Microsoft
Language
Pascal
ABAP
C/AL
Cg
SYMPL
Cesil
Scala
Rust
GDL
Bistro
JOSS
Blue
Genie
Inform
SenseTalk
HAL/S
SIMSCRIPT
Revolution
Sawzall
Chapel
Rapira
CSKA
SIMPLE
ALGOL
DATATRIEVE
PL/I
ELAN
Ajax
Agora
Clojure
Not
Trac
Object
Autolt
Microcode
code
Lynx
SAM76
Redcode
Logo
SALSA
ATS
Fjölnir
Fuxi
Assembly
TACPOL
FP
C
shell
language
E
A+
Lisp
VB
MicroScript
Squirrel
Windows/Dos
Kaleidoscope
IMP
Fancy
SuperTalk
P
C++
LC-3
ABC
K
Visual
JavaScript
FLJavaFX
R
Forth
b
T
Newspeak
L
XOTcl
dc
Cayenne
Go
Delphi
DASL
D
Caml
Z
GameMonkey
ECMAScript
Prolog
GJ
bc
GM
Constraint
S
EusLisp
ObjectScript
JASS
Oxygene
G
PEARL
Programming
Java
BLISS
notation
Ruby
EXEC
RPG
M
Cyclone
RPL
Script
PHP
CorVision
Action
REXX
F
Little
NET
Q
Simula
Boomerang
COBOL
GraphTalk
A
ChucK
Common
Agena
Strand
Turing
ISWIM
PL/SQL
Arc
Moby
ORCA/Modula-2
Uniface
Python
rexx
HaXe
Eiffel
S/SL
J
Robot
SP/k
Coral
WATFOR

http://www.tiobe.com/tiobe_index




THE WORLD OF
PROGRAMMING
Pioneers
Facts & Algos
Stats
计算机编程
的世界
先锋
事实与算法
统计情况
Charles Babbage
{He first came up with the idea of difference engine & analytical engine and is regarded as father of computer}
// A Punch Card
Alan Turing
{He is well known for the Halting problem, Turing machines, crypto-analysis of Enigma & Turing test. Turing award is given annually for exceptional work in the field of computing}
// 3 state busy beaver machine
Charles Babbage
{查尔斯·巴贝奇}
{他首先提出了差分机以及分析机的概念，被认为是计算机之父}
// 打孔卡
Alan Turing
{阿伦·图灵}
{他以（图灵）停机问题，图灵机，密码加密分析以及图灵测试而著名。图灵奖每年授予计算领域的出色工作者。}
// 三状态数Busy Beaver 图灵机
// Insertion Sort algorithm
Ada Lovelace
{Wrote a program to calculate sequence of Bernoulli's number using analytical engine. Regarded as World's first programmer}
// Merge sort algorithm
John von Neumann
{He came up with the concept ofstored program computer that uses a CPU and a separate storage to hold both instructions and data. This is also known as von Neumann architecture}
// 插入排序算法
Ada Lovelace
{艾达·洛芙莱斯}
{写出了使用的分析机计算伯努利数列的程序。她被视为世界上第一个程序员}
//合并排序算法
John von Neumann
{冯·诺依曼}
{他提出了存储程序计算机的概念。使用一个CPU和一个可存放指令和数据单独的空间，这种结构也称为是冯·诺依曼结构结构}
Dennis Ritchie
{He is the creator of C programming language and was also amongst the key developers of UNIX operating system. He received the Turing award in 1983}
// Dijkstra's algorithm
Edsger Dijkstra
{He is known for Dijkstra's algorithm, which is a graph search algorithm that solves the single-source shortest path problem for a graph with nonnegative edge path costs}
Memory
CPU
ALU
Input
Output
// Von Neumann Architecture
Dennis Ritchie
{丹尼斯·里奇}
{他是C程序语言的创造者，也是UNIX操作系统的关键开发人员之一。他获得1983年的图灵奖}
//狄捷斯特拉算法
Edsger Dijkstra
{他以狄捷斯特拉算法著名，这是一个图形搜索算法，解决图形的非负优势路径成本的单源最短路径问题}
// 冯·诺依曼结构结构
// B-Tree
Donald Knuth
{He is the creator of TEX and MMIX and is well known for the "Art of computer programming"book series. He received the Turing Award in 1974}
// Binary Search Tree
John Backus
{He is well known for the development of FORTRAN and ALGOL. He is also the inventor of Backus-Naur form and has also helped to popularize functional level programming}
// 二叉树
Donald Knuth
{唐纳德·克努特}
{TEX和MMIX的创造者，写下了经典巨著《计算机程序设计的艺术》，获得1974年图灵奖}
// 二叉搜索树
John Backus
{帕科·巴库斯}
{以FORTRAN和ALGOL而著名，还创造了巴库斯—诺尔范式，并帮助推广了功能级别的编程}
Ken Thompson
{He is well known as the principal creator of the UNIX operating system and is also the co-creator of the Go programming language}
// Google PageRank Algorithm
Bjarne Stroustrup
{He is well known for the creation and development of C++programming language and currently holds the college of engineering chair in computer science at Texas A&M.}
// Graph Coloring Algorithm
Ken Thompson
{肯·汤普逊}
{他是著名的UNIX操作系统的主要创造者，也是Go编程语言的共同创建者}
// Google PR 算法
Bjarne Stroustrup
{贾尼·斯特劳斯特卢普}
{创造和发展了C++，目前还担任得克萨斯A&M大学担任计算机科学的主席教授}
// 图形着色演算法
Hubris
Impatience
Laziness
// 3 virtues of a Programmer - Impatience, Laziness and Hubris
Larry Wall
{He is well known for the creation of Perl programming language and is also the first recepient of the Free Software Foundation Award for the Advancement of Free Software}
// Sweep line algorithms can be used for generating a Voronoi diagram
James Gosling
{ He is known as the father of Java programming language and is also the creator of Gosmacs, which was the first Emacs to run under UNIX}
Hubris
Impatience
Laziness
//程序员的三美德：急躁，懒惰和自大
Larry Wall
{拉里·沃尔}
{发明Perl 编程语言，也因发展自由软件成为自由软件基金会奖首位获得者}
// 扫描线算法，可用于生成Voronoi图
James Gosling
{詹姆斯·高斯林}
{Java之父，并且设计了运行于UNIX的第一个Emacs编辑器Gosmacs}
Guido van Rossum
{He is well known as the author of Python programming language and is currently employed by Google}
// Infinite Fibonacci sequence in Brainfuck language
Richard Stallman
{He is the creator of Emacs editor and the lead architect and organizer of the GNU project. He has been actively involved in the free software movement}
Vim
Emacs
// Vim and Emacs are the two most popular editors amongst "Real Programmers"
Guido van Rossum
{罗萨}
{Python编程语言的作者，现效力于Google}
// Brainfuck语言表示无限斐波纳契序列
Richard Stallman
{理查德·斯托曼}
{Emacs编辑器的发明人，GNU项目领袖工程师和组织者。一直积极参与自由软件运动}
Vim
Emacs
// Vim和Emacs一直是“真正的程序员”中主流的编辑器
// A solution to the Eight Queen Problem
Linus Torvalds
{He is best known for having initiated the development of Linux Kernel and the Git revision control system. He is also a strong supporter of Open Source software}
// 15 puzzle - A version of the classic n-puzzle
Bill Joy
{He is the co-founder of Sun Microsystems and is also the creator of vi editor,csh and NFS. He was also a primary figure in the development of Java programming language}
// 一个八皇后问题的解决方案
Linus Torvalds
{林纳斯·托瓦兹}
{ 他因发起Linux内核开发和Git版本控制系统而著名，也是开源软件的坚定支持者 }
// 十五数字推盘游戏
·经典数字推盘游戏的一个版本
Bill Joy
{比尔·乔伊}
{ Sun公司的创建人之一，vi编辑器，csh和NFS的发明人。他也是发展Java语言的首要人物之一 }
Timeline
The Beginning
Fundamentals
The first languages
Power at the highest level
Age of the Internet
1843
1936
1947
1955
1958
1959
1964
1972
1978
1980
1987
1991
1993
1995
Ada Lovelace wrote a program for the Analytical Engine
Alonzo Church introduced Lambda Calculus and Alan Turing came up with the concept of Turing Machine.
John von Neumann came up with the concept of stored program computer or the Von Neumann Architecture.
FORTRAN
ALGOL 58
COBOL
BASIC
C
SQL
C++
Perl
Python and Java
Ruby
PHP and JavaScript
时间轴
开始
基本原理
首个程序语言
影响达到巅峰水平
互联网时代
艾达·洛芙莱斯写出分析引擎程序
阿隆佐·丘奇引入Lambda演算，阿伦·图灵提出了图灵机的概念
冯·诺依曼提出储存程序计算机概念

程序员技术练级攻略[1]

前言

- 不要乱买书，不要乱追新技术新名词。
- 回顾一下历史，你才能明白明天会是什么样。
- 一定要动手，例子不管多简单，都建议至少自己手敲一遍。
- **一定要学会思考**，思考为什么要这样，而不是那样。
- **还要举一反三的思考！**

[1] 来源：http://CoolShell.cn

程序员技术练级攻略[1]

Unix/Linux VS. Windows

- 现在的用户界面主宰：Web 和移动设备 iOS 或 Android
- 低成本高性能 Linux 及各种开源技术架构胜过高成本 Windows
- 微软产品变化太快，很不持久～玩弄程序员？
- 趋势：**前端 Web+ 移动，后台 Linux+ 开源**。

[1] 来源：http://CoolShell.cn

程序员技术练级攻略[1]

启蒙入门

1. 学习一门脚本语言（如 Python/Ruby）
2. 用熟一种程序员的编辑器（不是 IDE）和一些基本工具
3. 熟悉 Unix/Linux Shell 和常见的命令行
4. 学习 Web 基础（HTML/CSS/JS）+ 服务器端技术（LAMP）

[1] 来源：http://CoolShell.cn

程序员技术练级攻略[1]

启蒙入门

1. 学习一门脚本语言（如 Python/Ruby）
   - 处理文本文件，读一个本地文件，逐行处理。
   - 遍历本地文件系统（如写程序统计一个目录下所有文件大小并按各种条件排序）。
   - 跟数据库打交道（如统计数据库里条目数量）。
   - 学会用各种`print`简单粗暴方式进行调试
   - 学会用 Google
2. 用熟一种程序员的编辑器（不是 IDE）和一些基本工具
3. 熟悉 Unix/Linux Shell 和常见的命令行
4. 学习 Web 基础（HTML/CSS/JS）+ 服务器端技术（LAMP）

[1] 来源：http://CoolShell.cn

程序员技术练级攻略[1]

启蒙入门

1. 学习一门脚本语言（如 Python/Ruby）
2. 用熟一种程序员的编辑器（不是 IDE）和一些基本工具
   - Vim / Emacs / Notepad++，学会如何配置代码补全、外观、外部命令等。
   - Source Insight（或 ctag）
   - Not Only Cool! 更有效率！
3. 熟悉 Unix/Linux Shell 和常见的命令行
4. 学习 Web 基础（HTML/CSS/JS）+ 服务器端技术（LAMP）

[1] 来源：http://CoolShell.cn

程序员技术练级攻略[1]

启蒙入门

1. 学习一门脚本语言（如 Python/Ruby）
2. 用熟一种程序员的编辑器（不是 IDE）和一些基本工具
3. 熟悉 Unix/Linux Shell 和常见的命令行
   - 一定要少用少用图形界面
   - 学会使用 man 来查看帮助
   - 文件系统结构和基本操作`ls chmod chown rm find ln cat mount mkdir tar gzip`
   - 学会使用一些文本操作命令`sed awk grep tail less more`
   - 学会使用一些管理命令`ps top lsof netstat kill tcpdump iptables dd`
   - 了解/etc 目录下的各种配置文件，学会查看/var/log 下的系统日志以及/proc 下的系统运行信息
   - 了解正则表达式，使用正则表达式来查找文件。
4. 学习 Web 基础（HTML/CSS/JS）+ 服务器端技术（LAMP）

程序员技术练级攻略[1]

进阶加深

1. C 语言和操作系统调用（指针、内存模型，数据结构与算法）
2. 学习 Java
3. Web 的安全与架构
4. 学习关系型数据库（MySQL）
5. 一些开发工具（Git 或 SVN、gdb、Makefile 等）

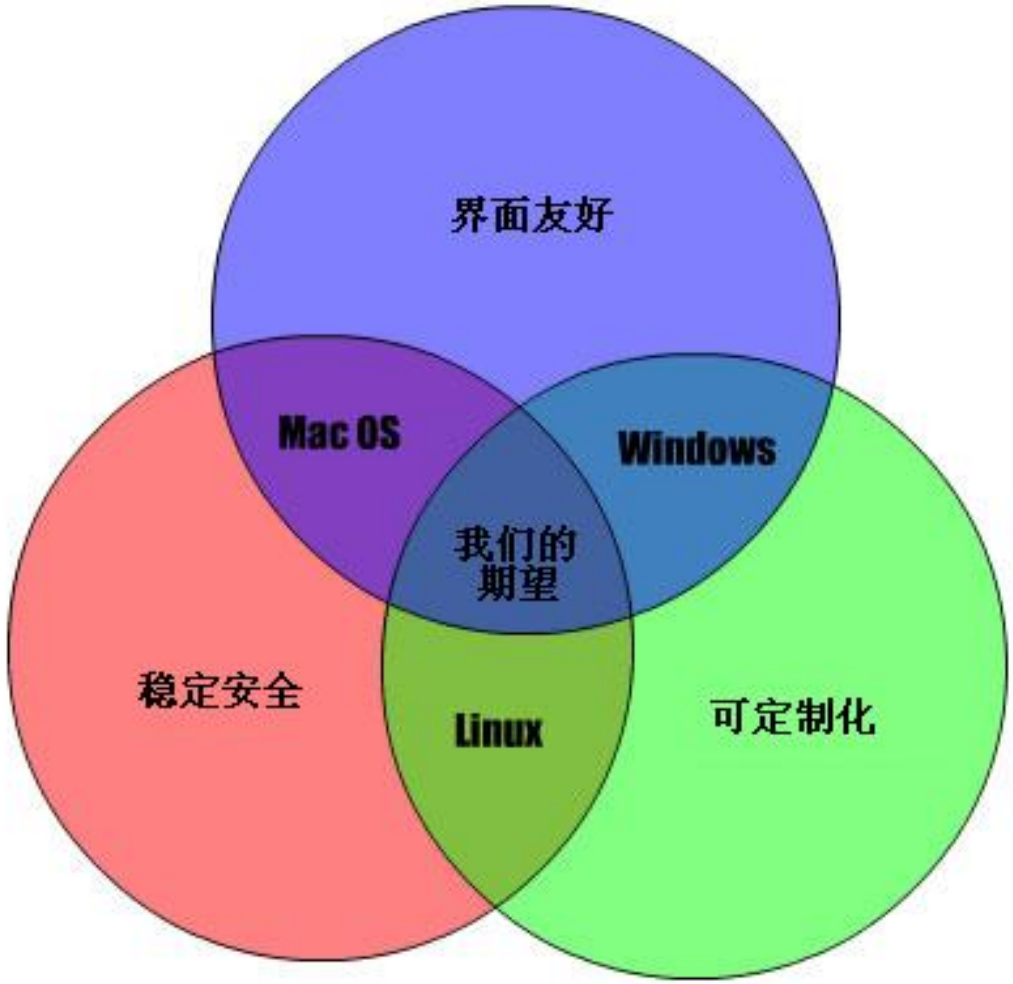
[1] 来源：http://CoolShell.cn

程序员技术练级攻略[1]

高级深入

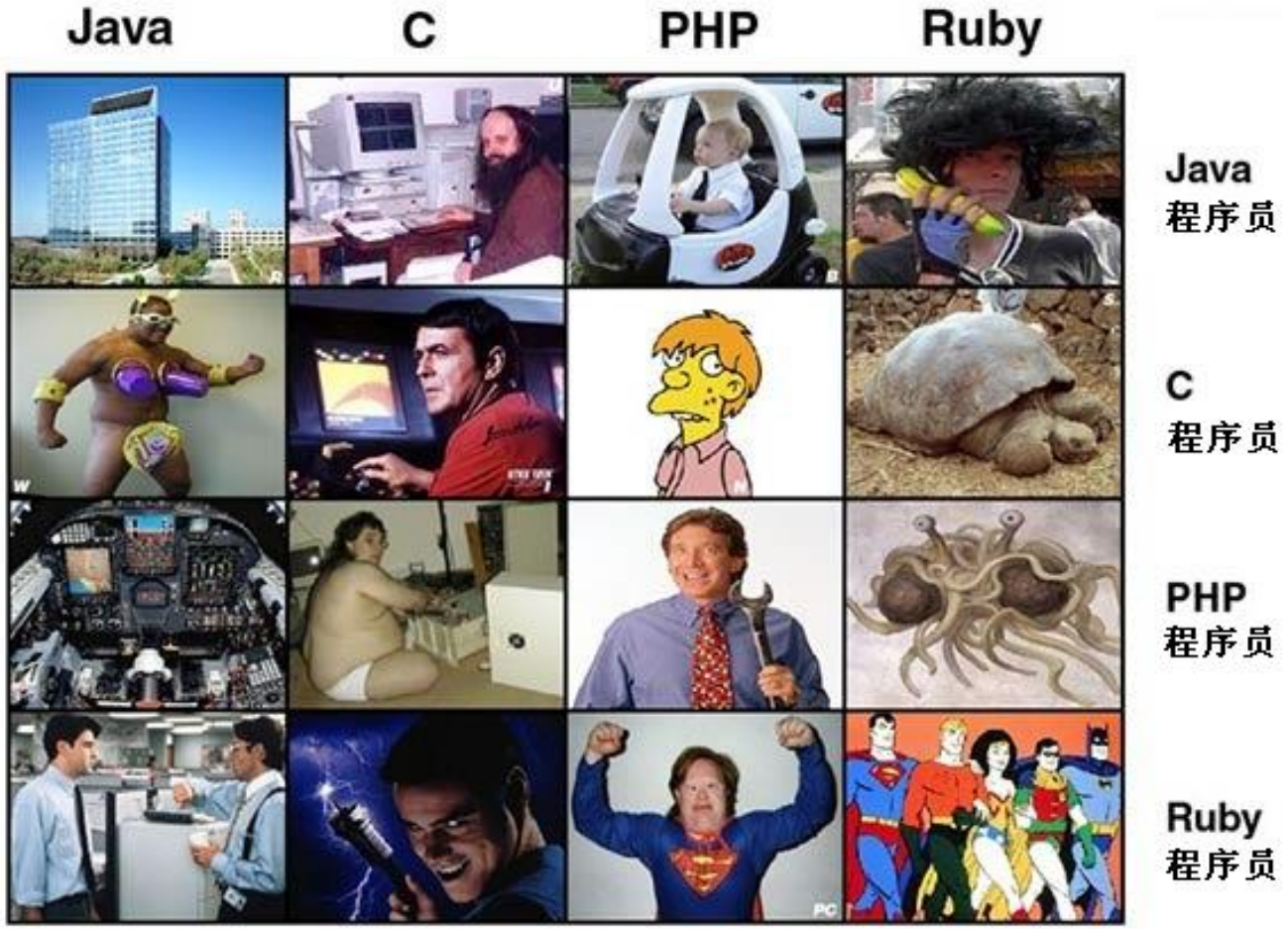
1. C++/Java 和面向对象
2. 加强系统了解（Unix 编程、网络编程，TCP/IP 详解，等）
3. 系统架构（负载均衡，多层分布式系统，P2P 式，服务器备份，虚拟化，Hadoop，NoSQL，等）

[1] 来源：http://CoolShell.cn


界面友好
Mac OS
Windows
我们的
期望
稳定安全
Linux
可定制化

粉丝眼中的操作系统

程序员眼中的编程语言



内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

它为什么叫“C”语言？


evaluated
statement
standard
case
void
contains
array
INITIALIZED
PROGRAM
include
TYPE
return
CALLED
size
POINTER
function
variables
const
argument
zero
define
int
char
read
USED
CHARACTER
operator
declaration
external
expression
RESULT
NULL
structure
write
String
specifier
SPACE
STRUCT
input
library
error
object
double
time
left

程序设计语言的分类

低级语言之机器语言

00000001000000001000	数据装入寄存器 0
00000001000100001010	数据装入寄存器 1
00000101000000000001	寄存器 0 与 1 的数据乘
00000001000100001100	数据装入寄存器 1
00000100000000000001	寄存器 0 与 1 的数据加
00000010000000001110	保存寄存器 0 里的数据

程序设计语言的分类

低级语言之汇编语言

```
load 0 a    数据装入寄存器 0
load 1 b    数据装入寄存器 1
mult 0 1    寄存器 0 与 1 的数据乘
load 1 c    数据装入寄存器 1
add 0 1     寄存器 0 与 1 的数据加
save 0 d    保存寄存器 0 里的数据
```

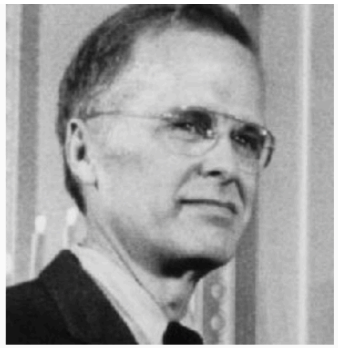
程序设计语言的分类

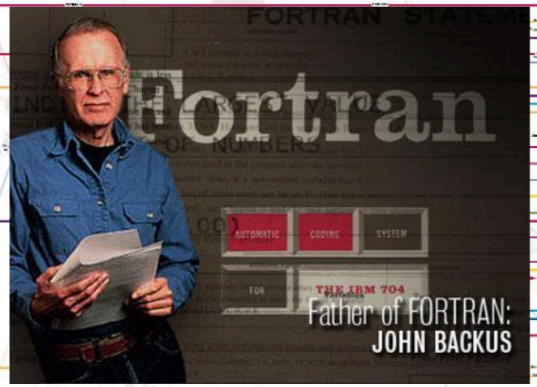
高级语言之 C

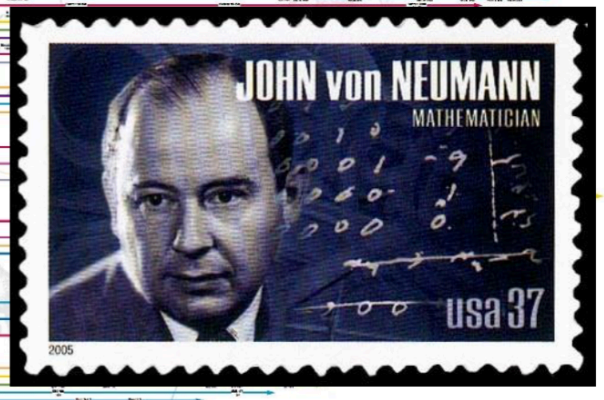
```
                d = a * b + c
00000001000000001000   load 0 a
00000001000100001010   load 1 b
00000101000000000001   mult 0 1
00000001000100001100   load 1 c
00000100000000000001   add 0 1
00000010000000001110   save 0 d
```

高级程序设计语言

1954 -1956年 IBM 的John Backus 和他的研究小组研发了FORTRAN(FORmula TRANslation)

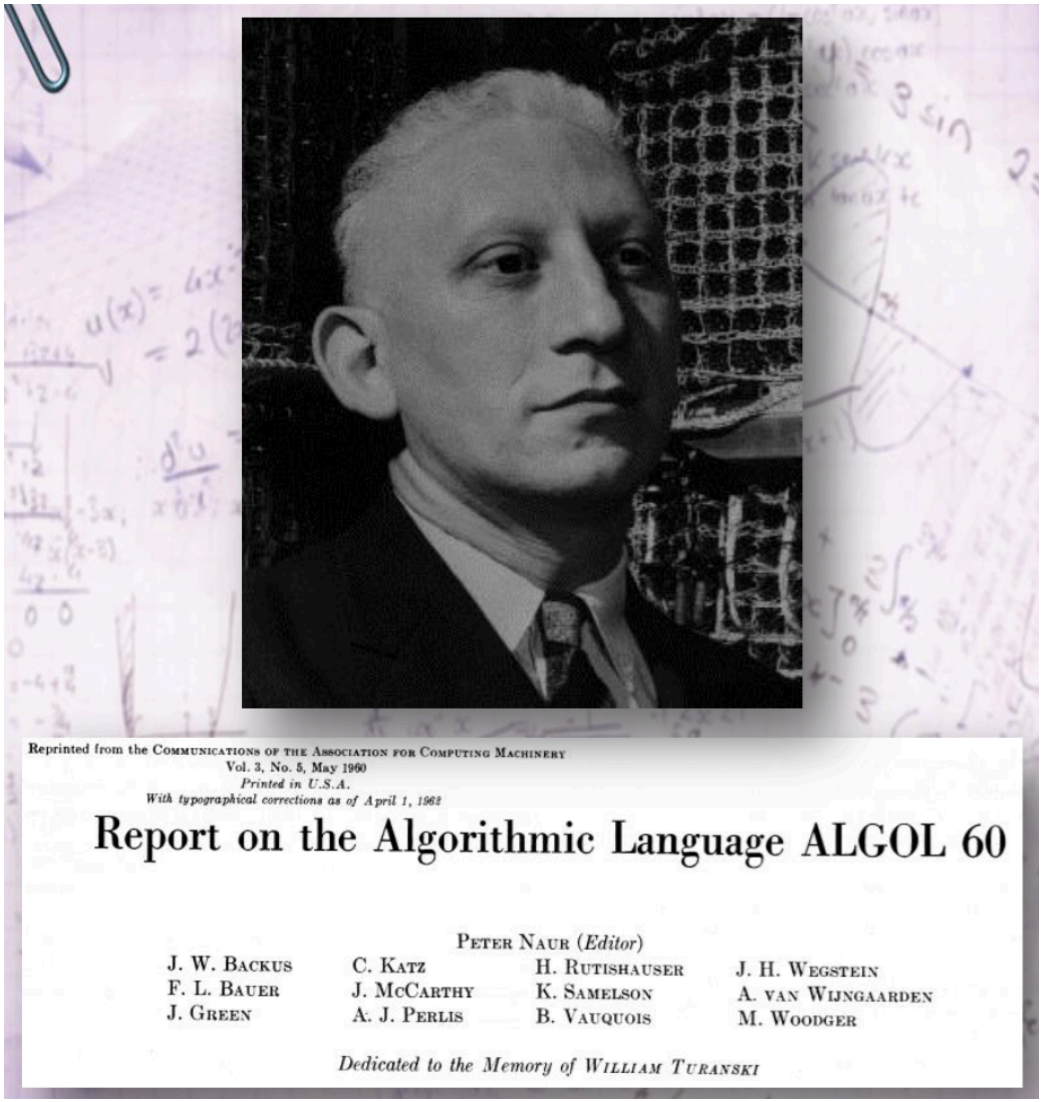






http://staff.pausd.org/~cbly/1web_design/12b_final/daniel/history.html

C 程序设计语言的历史



Reprinted from the COMMUNICATIONS OF THE ASSOCIATION FOR COMPUTING MACHINERY
Vol. 3, No. 5, May 1960
Printed in U.S.A.
With typographical corrections as of April 1, 1962

Report on the Algorithmic Language ALGOL 60

PETER NAUR (*Editor*)

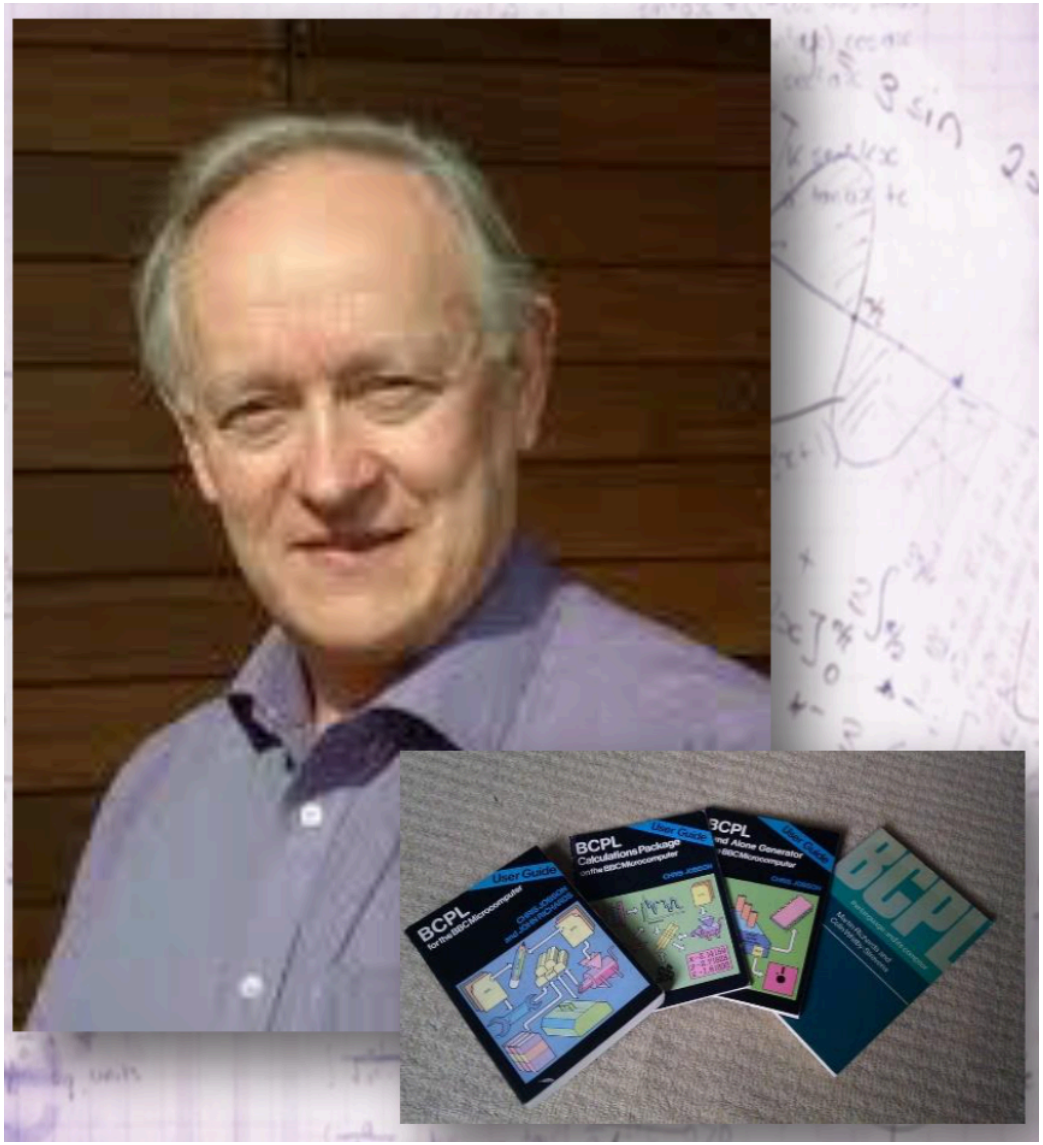
J. W. BACKUS, F. L. BAUER, J. GREEN, C. KATZ, J. MCCARTHY, A. J. PERLIS, H. RUTISHAUSER, K. SAMELSON, B. VAUQUOIS, J. H. WEGSTEIN, A. VAN WIJNGAARDEN, M. WOODGER

Dedicated to the Memory of WILLIAM TURANSKI

1960 年 1 月

- Alan J. Perlis
- 软件专家讨论会（巴黎）
- 发表“算法语言 Algol 60 报告”
- 宣告程序设计语言 Algol 60 诞生
- 计算科学里程碑
- (A 语言)

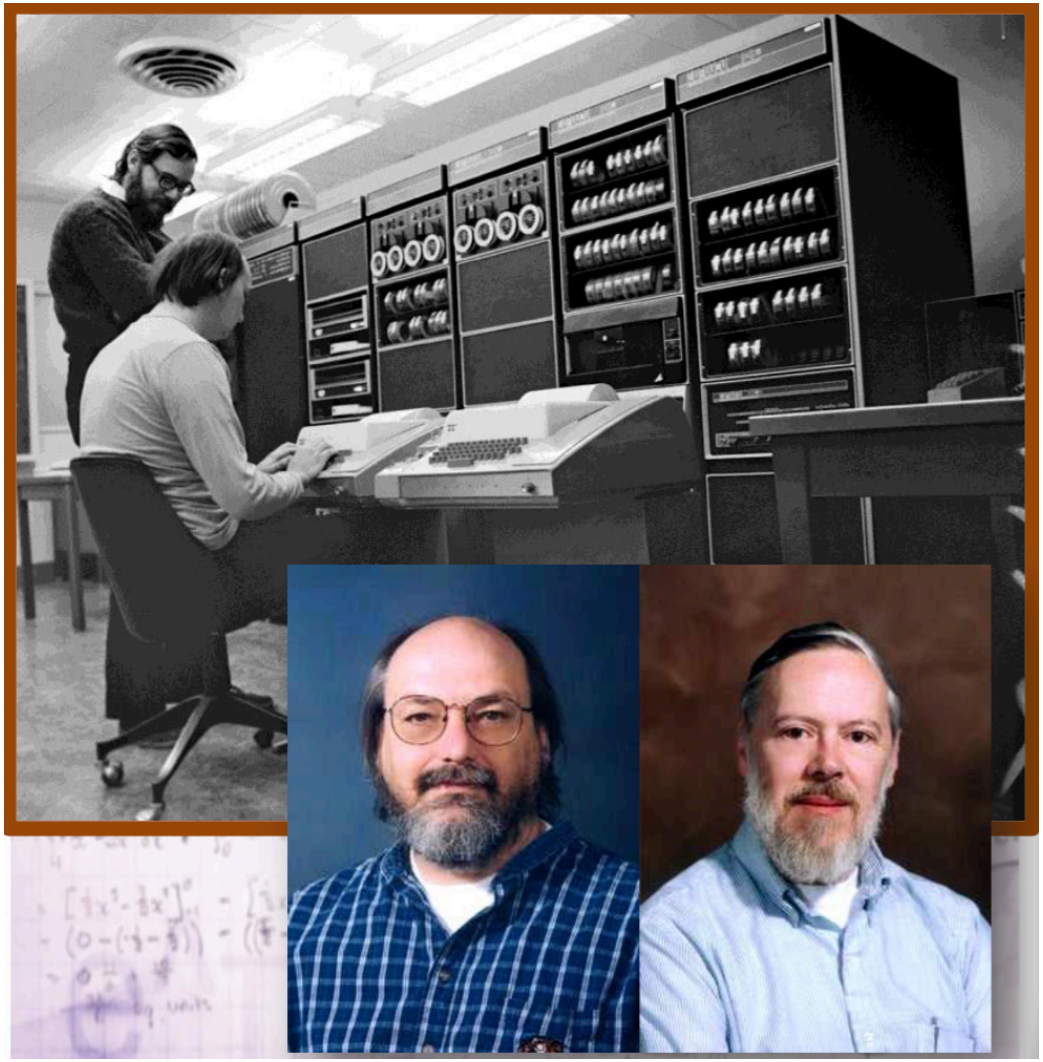
C 程序设计语言的历史



1963-1967 年

- BCPL 语言
- 1963 年，剑桥大学在 Algol 60 基础上推出 CPL（Combined Programming Language）语言，但规模比较大，难以实现。
- 1967 年，剑桥大学 Matin Richards 对 CPL 语言作了简化，推出了 BCPL（Basic Combined Programming Language）语言。

C 程序设计语言的历史



1972-1973 年

- B 语言：贝尔实验室 Ken Thompson 设计出 B 语言，并用 B 语言写第一个 UNIX 操作系统，在 PDP-7 上实现。
- C 语言：1972-1973 年间，Dennis Ritchie 和 Ken Thompson 在 B 语言基础上发展和完善出 C 语言，并重写 UNIX。

C 程序设计语言的历史

1979 年

- Bjarne Stroustrup http://www.stroustrup.com
- C with Classes（带类的 C）⇒ C++
- C++ 语言可以被看作是 C 语言的超集
- Objective-C、C#、Java、Python、Ruby、Perl、PHP、...

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

C 程序设计语言的版本

K&R C

- 1978 年，Kernighan 和 Ritchie 的《The C Programming Language》第一版出版，一直被广泛作为 C 语言事实上的规范，称“**K&R C**”。

ANSI C 和 ISO C

- 1989 年，C 语言被 ANSI 标准化，对 K&R C 进行了扩展，包括了一些新的特性，也规定了一套标准函数库。
- ISO 成立 WG14 工作组来规定国际标准的 C 语言；通过对 ANSI 标准的少量修改，最终通过了**ISO 9899:1990**；随后 ISO 标准被 ANSI 采纳。

C99

- 在 ANSI 标准化后，WG14 小组继续致力于改进 C 语言；新的标准很快推出：**ISO 9899:1999**；这个标准就是通常提及的**C99**；它被 ANSI 于 2000 年 3 月采用。

C11

- 2011 年 12 月 8 日，ISO 正式公布 C 语言新的国际标准草案：**ISO/IEC 9899:2011**，即**C11**。
- http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.

C 程序设计语言的版本

K&R C

- 1978 年，Kernighan 和 Ritchie 的《The C Programming Language》第一版出版，一直被广泛作为 C 语言事实上的规范，称“**K&R C**”。

ANSI C 和 ISO C

- 1989 年，C 语言被 ANSI 标准化，对 K&R C 进行了扩展，包括了一些新的特性，也规定了一套标准函数库。
- ISO 成立 WG14 工作组来规定国际标准的 C 语言；通过对 ANSI 标准的少量修改，最终通过了**ISO 9899:1990**；随后 ISO 标准被 ANSI 采纳。

C99

- 在 ANSI 标准化后，WG14 小组继续致力于改进 C 语言；新的标准很快推出：**ISO 9899:1999**；这个标准就是通常提及的**C99**；它被 ANSI 于 2000 年 3 月采用。

C11

- 2011 年 12 月 8 日，ISO 正式公布 C 语言新的国际标准草案：**ISO/IEC 9899:2011**，即**C11**。
- http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.

C 程序设计语言的版本

K&R C

- 1978 年，Kernighan 和 Ritchie 的《The C Programming Language》第一版出版，一直被广泛作为 C 语言事实上的规范，称“**K&R C**”。

ANSI C 和 ISO C

- 1989 年，C 语言被 ANSI 标准化，对 K&R C 进行了扩展，包括了一些新的特性，也规定了一套标准函数库。
- ISO 成立 WG14 工作组来规定国际标准的 C 语言；通过对 ANSI 标准的少量修改，最终通过了**ISO 9899:1990**；随后 ISO 标准被 ANSI 采纳。

C99

- 在 ANSI 标准化后，WG14 小组继续致力于改进 C 语言；新的标准很快推出：**ISO 9899:1999**；这个标准就是通常提及的**C99**；它被 ANSI 于 2000 年 3 月采用。

C11

- 2011 年 12 月 8 日，ISO 正式公布 C 语言新的国际标准草案：**ISO/IEC 9899:2011**，即**C11**。
- http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.

C 程序设计语言的版本

K&R C

- 1978 年，Kernighan 和 Ritchie 的《The C Programming Language》第一版出版，一直被广泛作为 C 语言事实上的规范，称“**K&R C**”。

ANSI C 和 ISO C

- 1989 年，C 语言被 ANSI 标准化，对 K&R C 进行了扩展，包括了一些新的特性，也规定了一套标准函数库。
- ISO 成立 WG14 工作组来规定国际标准的 C 语言；通过对 ANSI 标准的少量修改，最终通过了**ISO 9899:1990**；随后 ISO 标准被 ANSI 采纳。

C99

- 在 ANSI 标准化后，WG14 小组继续致力于改进 C 语言；新的标准很快推出：**ISO 9899:1999**；这个标准就是通常提及的**C99**；它被 ANSI 于 2000 年 3 月采用。

C11

- 2011 年 12 月 8 日，ISO 正式公布 C 语言新的国际标准草案：**ISO/IEC 9899:2011**，即**C11**。
- http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.

注意

C 语言规范定义的非常宽泛：

- long 型数据长度不短于 int 型
- short 型不长于 int 型

经常导致：

- 相同的程序在不同编译器上具有不同解释
- 相同的程序在不同平台上运行结果不同
  - 例如：整型变量定义；对 ++、--的解释；输入输出赋值顺序的不同；浮点数计算精度的不同；……

注意

C 语言规范定义的非常宽泛：

- long 型数据长度不短于 int 型
- short 型不长于 int 型

经常导致：

- **相同的程序在不同编译器上具有不同解释**
- **相同的程序在不同平台上运行结果不同**
  - 例如：整型变量定义；对 ++、--的解释；输入输出赋值顺序的不同；浮点数计算精度的不同；……

For Example

C89

```
1 int i;
2 for (i=0;i<9;i++)
```

```
1 $ gcc -std=c89
```

C99

```
1 for (int i=0;i<9;i++)
```

```
1 $ gcc -std=c99
```

C++

- 1979 年，贝尔实验室 Bjarne Stroustrup 开发了一种语言，被称为“C with Classes”，后来演化为“C++”。
- 1985 年 10 月，Bjarne 博士完成经典巨著《The C++ Programming Language》第一版。
- 1998 年 11 月 ISO 颁布了 C++ 程序设计语言的国际标准 ISO/IEC 14882-1998。
- ISO 于 2011 年 9 月 1 日发布了 ISO/IEC 14882:2011 即 C++ 2011。

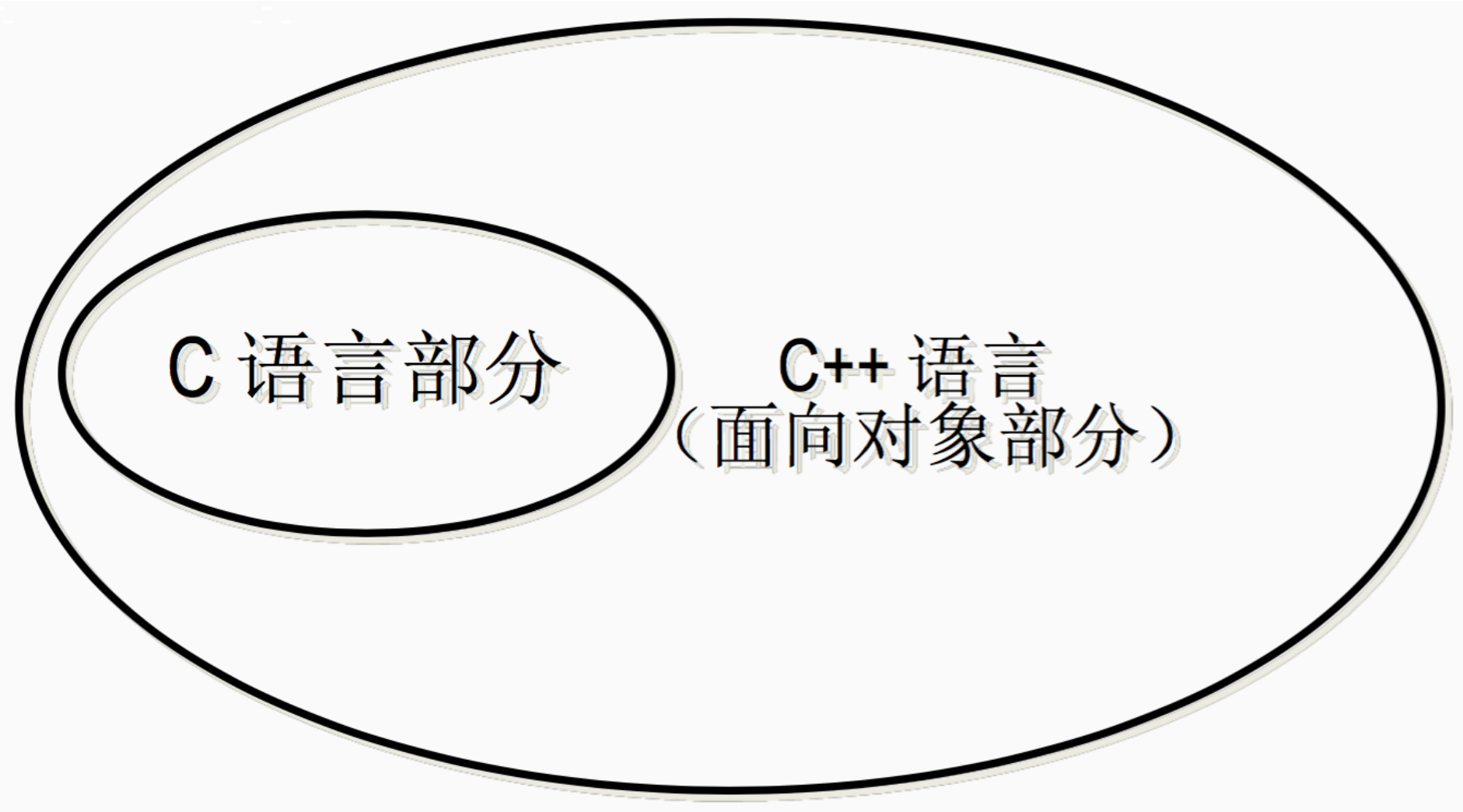
内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

我们学的是什么高级语言？



应该如何学习编程语言

程序设计语言的构成

语言的种类千差万别，但是，一般来说，基本成分不外四种：

数据成分 用以描述程序中所涉及的数据；

运算成分 用以描述程序中所包含的运算；

控制成分 用以表达程序中的控制构造；

传输成分 用以表达程序中数据的传输。

——计算机科学技术百科

如果让我来学习一门程序设计语言

数据成分

- 有哪些数据类型？如何使用？

运算成分

- 有哪些运算符号？如何使用？

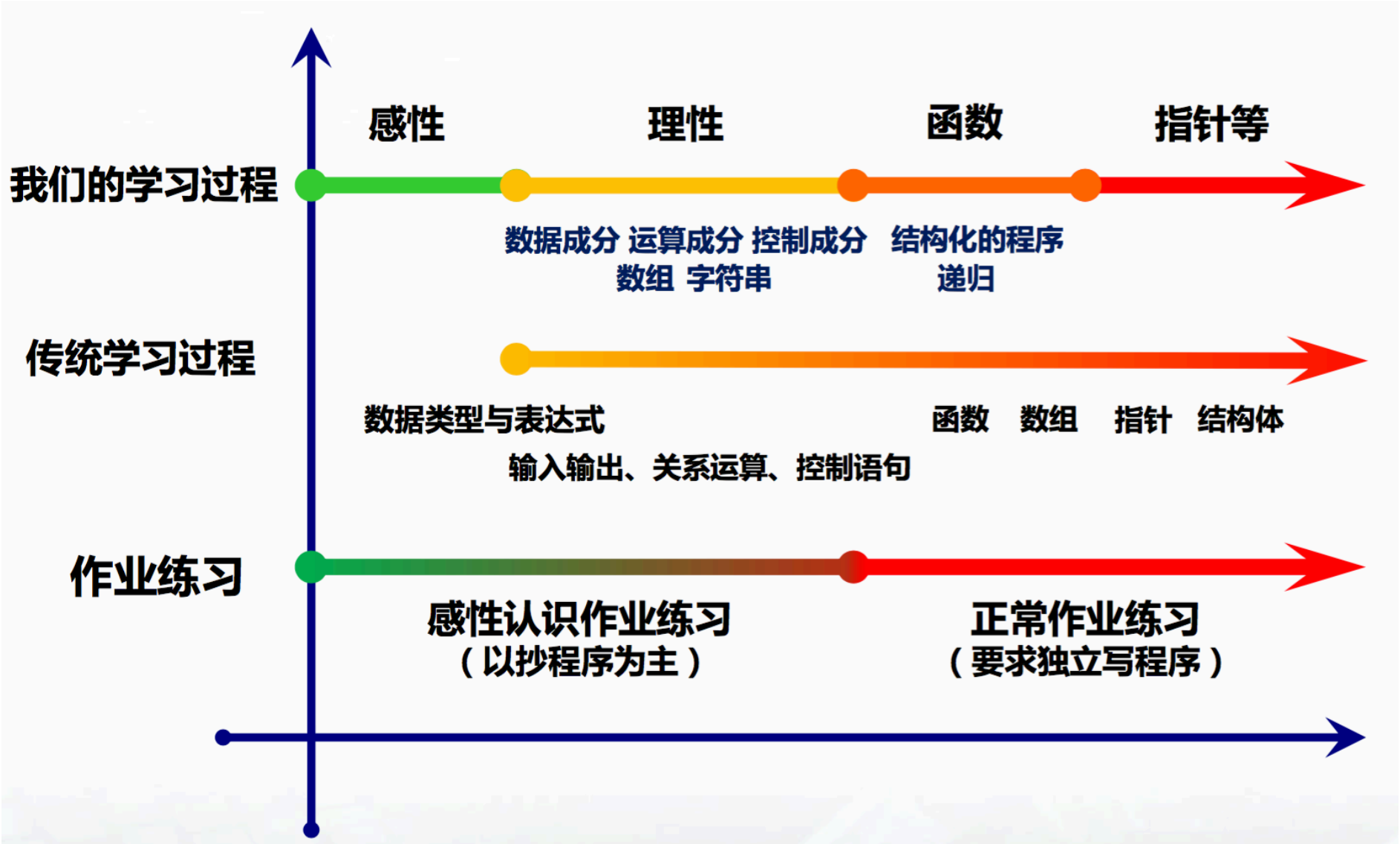
控制成分

- 三种类型的控制语句是如何写的？

传输成分

- 在程序中如何输入和输出数据？

```
#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;
}
```


感性
理性
函数
指针等
我们的学习过程
数据成分 运算成分 控制成分 结构化的程序
数组 字符串
递归
传统学习过程
数据类型与表达式
函数 数组 指针 结构体
输入输出、关系运算、控制语句
作业练习
感性认识作业练习
（以抄程序为主）
正常作业练习
（要求独立写程序）

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

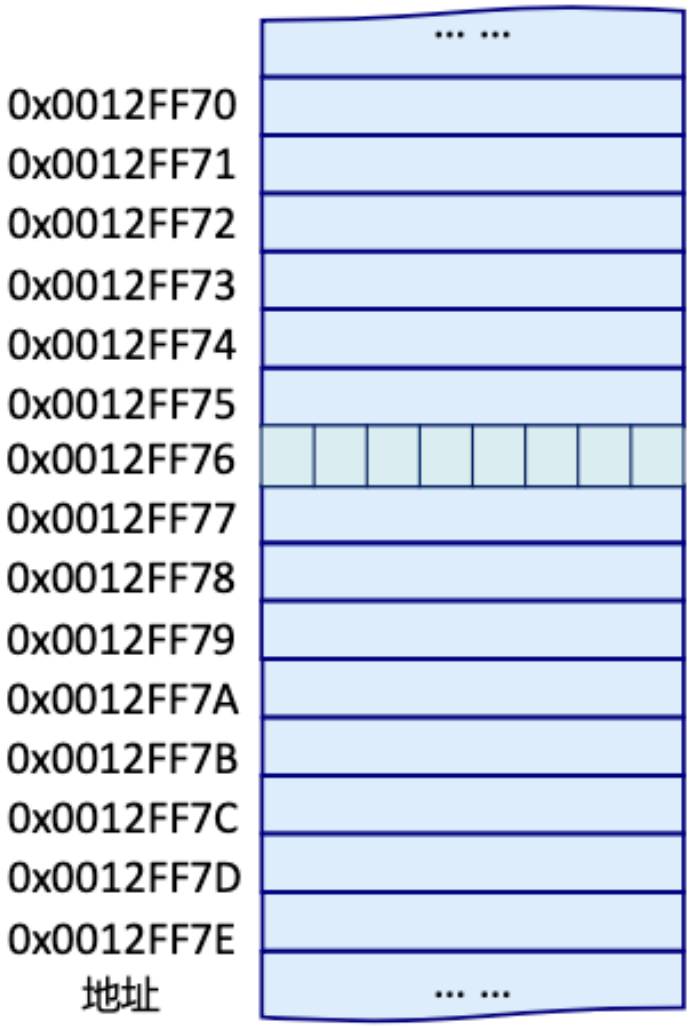
内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

数据与存储



请这样来 想象 内存

- **存储空间单位**

2^{10}=1024

1B (Byte) = 8 b (bit)

1KB = 1024Byte

1MB = 1024KB

1GB= 1024MB

1TB = 1024GB

1PB = 1024TB

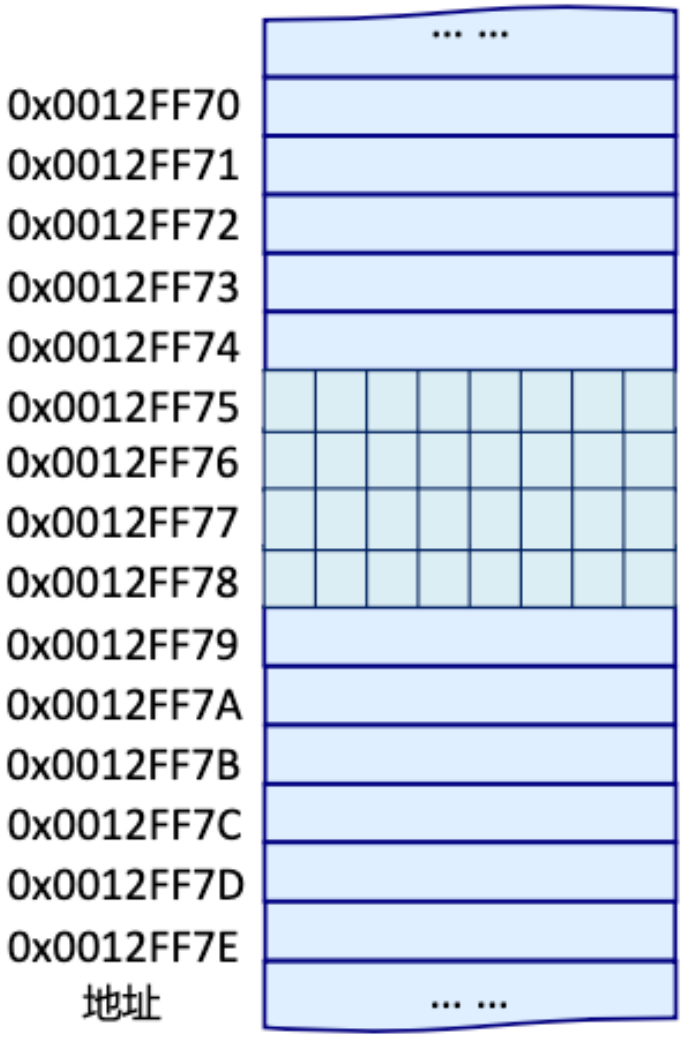
```cpp
#include<iostream>
using namespace std;
int main( )
{
  int number[45] = {78, 56, 69, 31, 36, 67, 31, 47, 69,
  ↪  34, 45, 74, 61, 82, 43, 41, 76, 79, 81, 66, 54, 50,
  ↪  76, 51, 53, 28, 74, 39, 45, 61, 52, 41, 43, 75, 78,
  ↪  84, 72, 51, 43, 64, 75, 81, 69, 55, 74};
  int max = 0;
  int i =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

★ 变量 ★

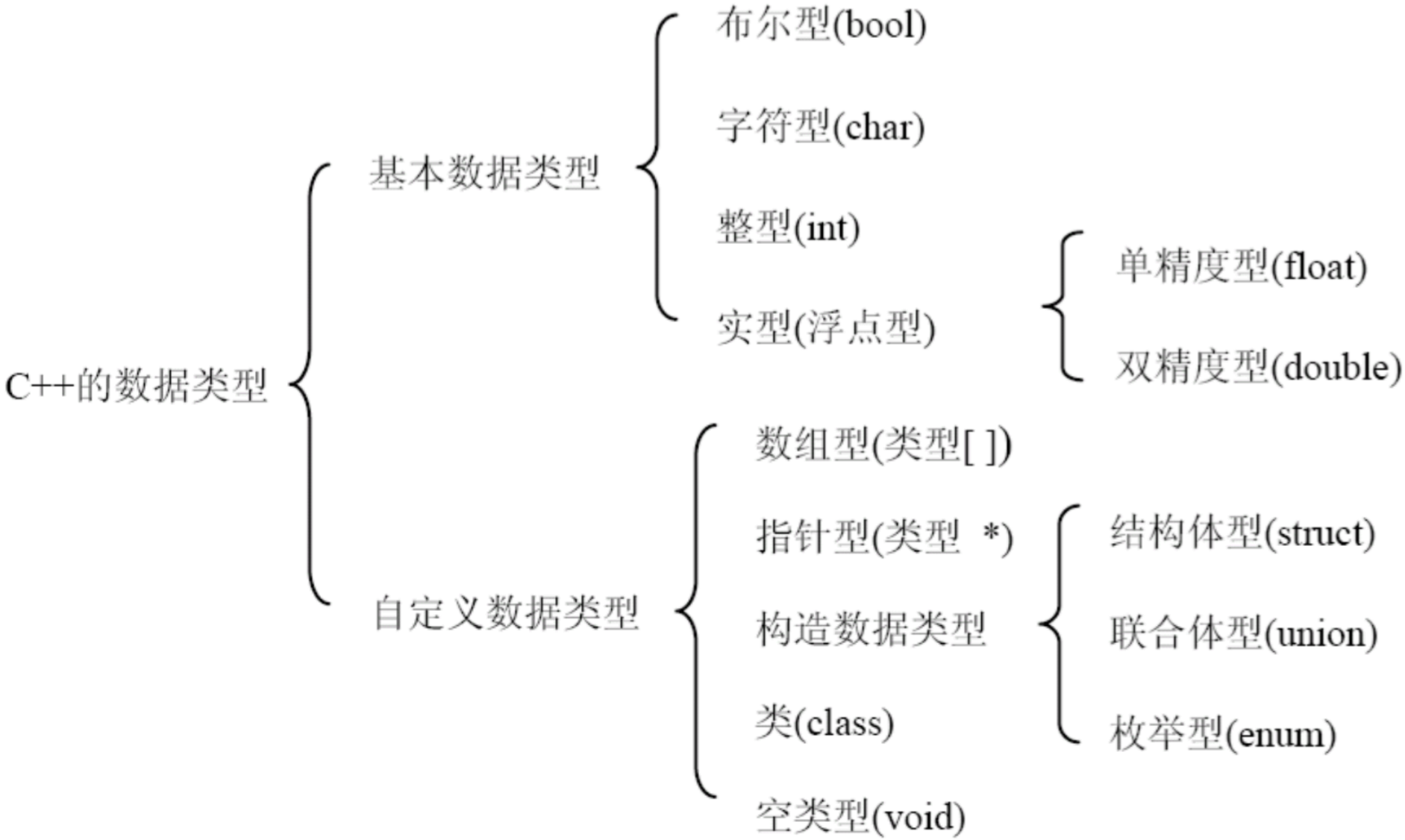
- 先定义、再使用
- 值可以变化的量
- 变量的定义格式

```
1 变量类型 变量标识符;
2 int Max;
3 int Max=0;
4 char character;
5 char character='A';
6 double Result=12.345;
```

内存


... ...
0x0012FF70
0x0012FF71
0x0012FF72
0x0012FF73
0x0012FF74
0x0012FF75
0x0012FF76
0x0012FF77
0x0012FF78
0x0012FF79
0x0012FF7A
0x0012FF7B
0x0012FF7C
0x0012FF7D
0x0012FF7E
地址
... ...

C/C++ 程序中的数据类型


C++的数据类型
基本数据类型
布尔型(bool)
字符型(char)
整型(int)
实型(浮点型)
单精度型(float)
双精度型(double)
自定义数据类型
数组型(类型[ ])
指针型(类型 *)
构造数据类型
结构体型(struct)
联合体型(union)
枚举型(enum)
类(class)
空类型(void)

如何知道某种类型的数占多少内存？

sizeof 运算符

```
1 #include<iostream>
2 using namespace std;
3 int main()
4 {
5   cout<<"The size of int is "<<sizeof(int)<<"
    ↪  Bytes."<<endl;
6   cout<<"The size of short int is "<<sizeof(short)<<"
    ↪  Bytes."<<endl;
7   cout<<"The size of long int is "<<sizeof(long)<<"
    ↪  Bytes."<<endl;
8   return 0;
9 }
```

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

① 整型

	有符号	无符号
基本型	**int** signed int	**unsigned int**
短整型	**short** short int signed short signed short int	**unsigned short** unsigned short int
长整型	**long** long int signed long signed long int	**unsigned long** unsigned long int

signed vs. unsigned

unsigned int i = 123;

32bit

signed int i = -123;

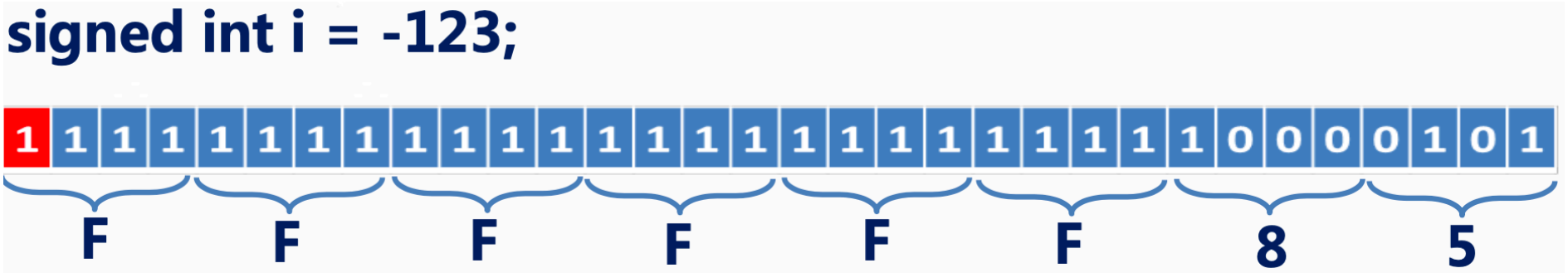
32bit

signed vs. unsigned

unsigned int i = 123;

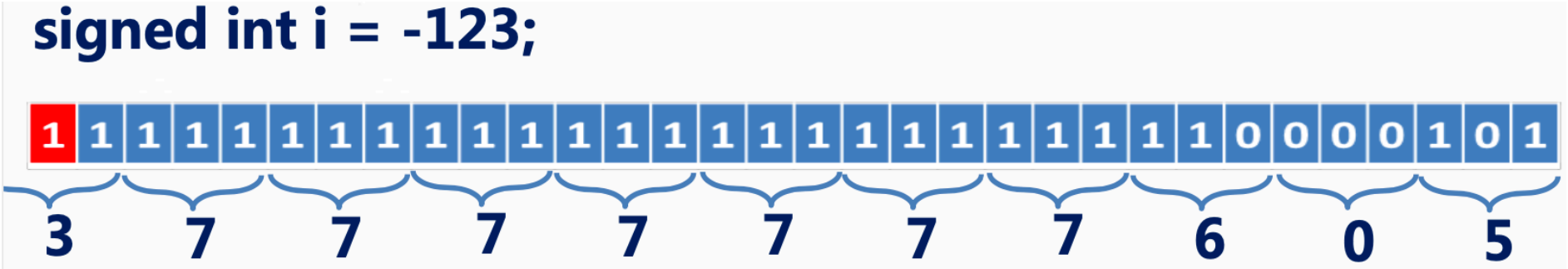
signed int i = -123;

数的十六进制表示




```
1 #include<iostream>
2 using namespace std;
3 int main()
4 {
5   int a = -123;
6   cout<<hex<<a<<endl;
7   return 0;
8 }
```

数的八进制表示




```
1 #include<iostream>
2 using namespace std;
3 int main()
4 {
5   int a = -123;
6   cout<<oct<<a<<endl;
7   return 0;
8 }
```

数的进制转换

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5    int a = -123;
6    cout<<hex<<a<<endl;
7    cout<<oct<<a<<endl;
8    cout<<dec<<a<<endl;
9    return 0;
10 }
```

数的进制转换

```
1  #include<iostream>
2  using namespace std;
3
4  int main()
5  {
6      int a=0x7FFFFF85;
7      cout<<dec<<a<<endl;
8      cout<<oct<<a<<endl;
9      return 0;
10 }
```

数的进制转换

```
1  #include<iostream>
2  using namespace std;
3
4  int main()
5  {
6      int a=037777777605;
7      cout<<dec<<a<<endl;
8      cout<<hex<<a<<endl;
9      return 0;
10 }
```

Max & Min

unsigned int i = Max;

signed int i = Max;

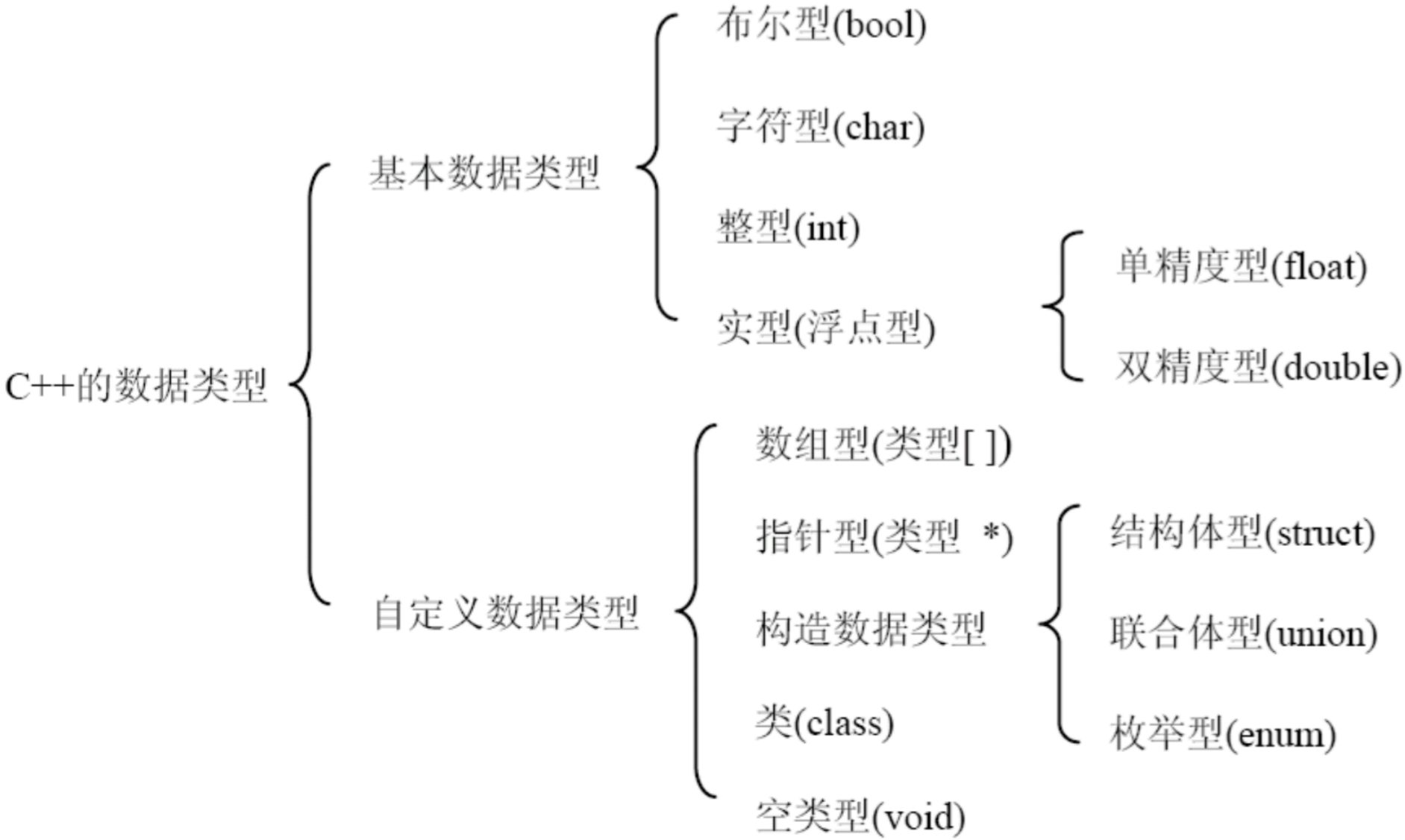
Max & Min

```
1    unsigned int a=0xFFFFFFFF;
2    cout<<"Max unsigned int: "<<dec<<a<<endl;
3    a=a+1;
4    cout<<"Min unsigned int: "<<dec<<a<<endl;
5    signed int b=0x7FFFFFFF;
6    cout<<"Max signed int: "<<dec<<b<<endl;
7    b=b+1;
8    cout<<"Min signed int: "<<dec<<b<<endl;
```

使用须知

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

C/C++ 程序中的数据类型


C++的数据类型
基本数据类型
布尔型(bool)
字符型(char)
整型(int)
实型(浮点型)
单精度型(float)
双精度型(double)
自定义数据类型
数组型(类型[ ])
指针型(类型 *)
构造数据类型
结构体型(struct)
联合体型(union)
枚举型(enum)
类(class)
空类型(void)

内容提要 I

内容提要 II

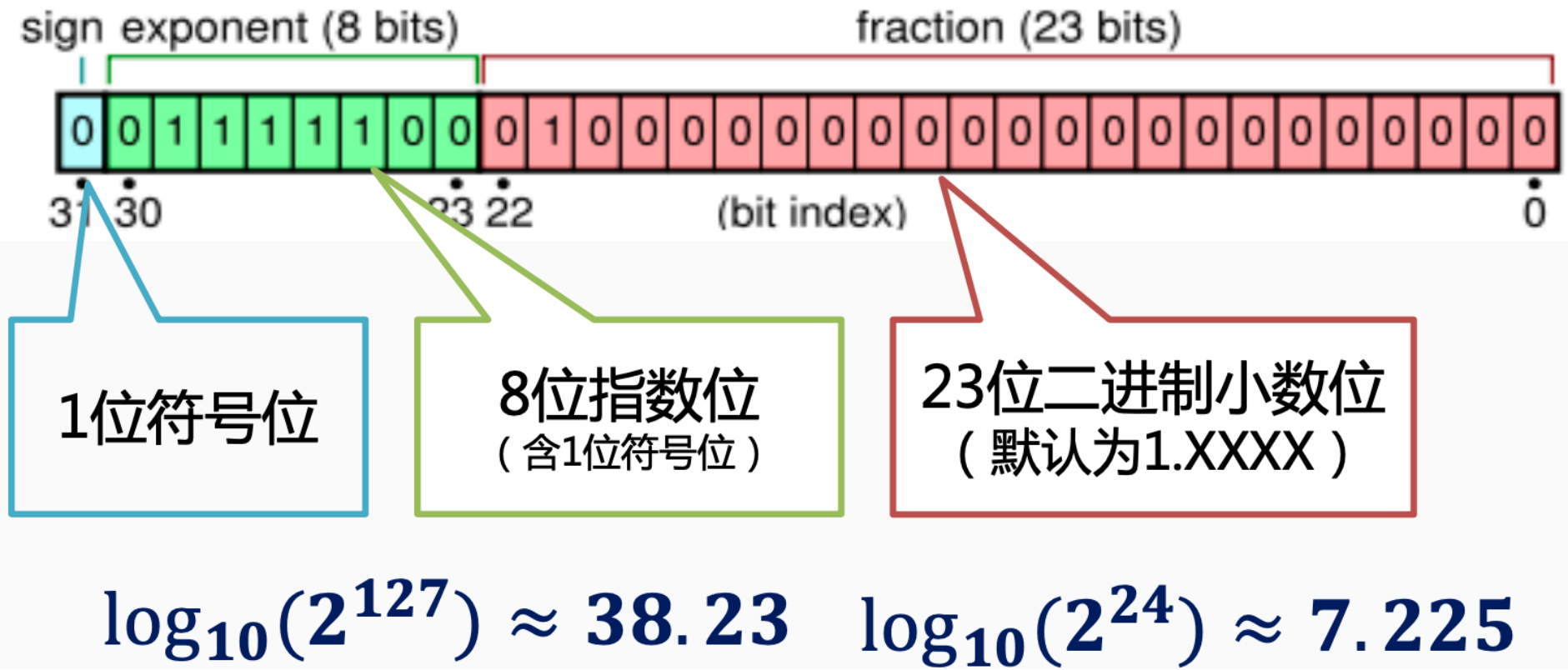
内容提要 III

- 字符串数组
- 输入与输出
- 示例

② 浮点型

```
#include<iostream>
#include<iomanip>
using namespace std;

int main()
{
    float a=3.14159265358979323846264338327950238419716939937510582097494459230781640628620899862803482534211706793214808;
    double b=3.14159265358979323846264338327950238419716939937510582097494459230781640628620899862803482534211706793214808;
    long double c=3.14159265358979323846264338327950238419716939937510582097494459230781640628620899862803482534211706793214808;
    cout<<a<<endl;
    cout<<setprecision(100)<<a<<endl;
    cout<<b<<endl;
    cout<<c<<endl;
    return 0;
}
```

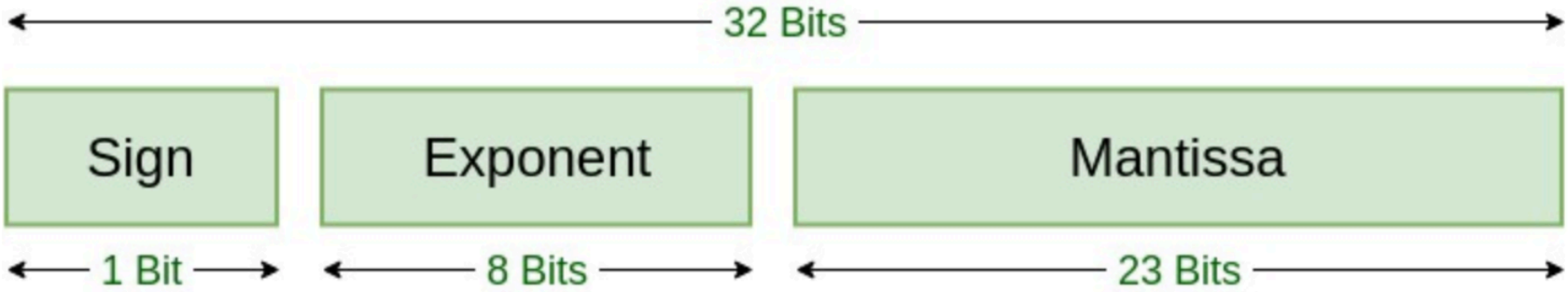
浮点数的表示



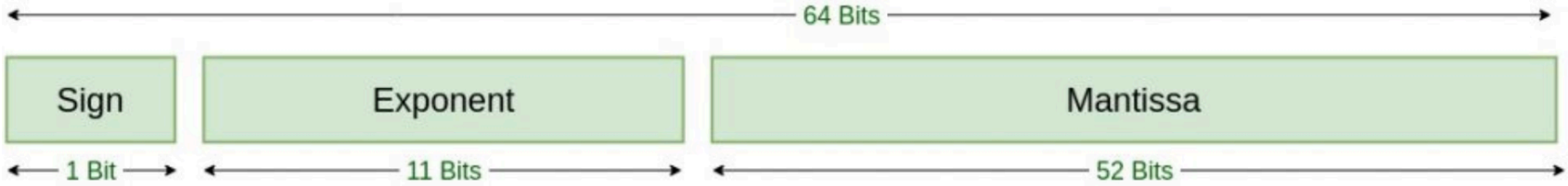
$$\log_{10}(2^{127}) \approx 38.23 \qquad \log_{10}(2^{24}) \approx 7.225$$

浮点数的表示

- Single Precision



- Double Precision

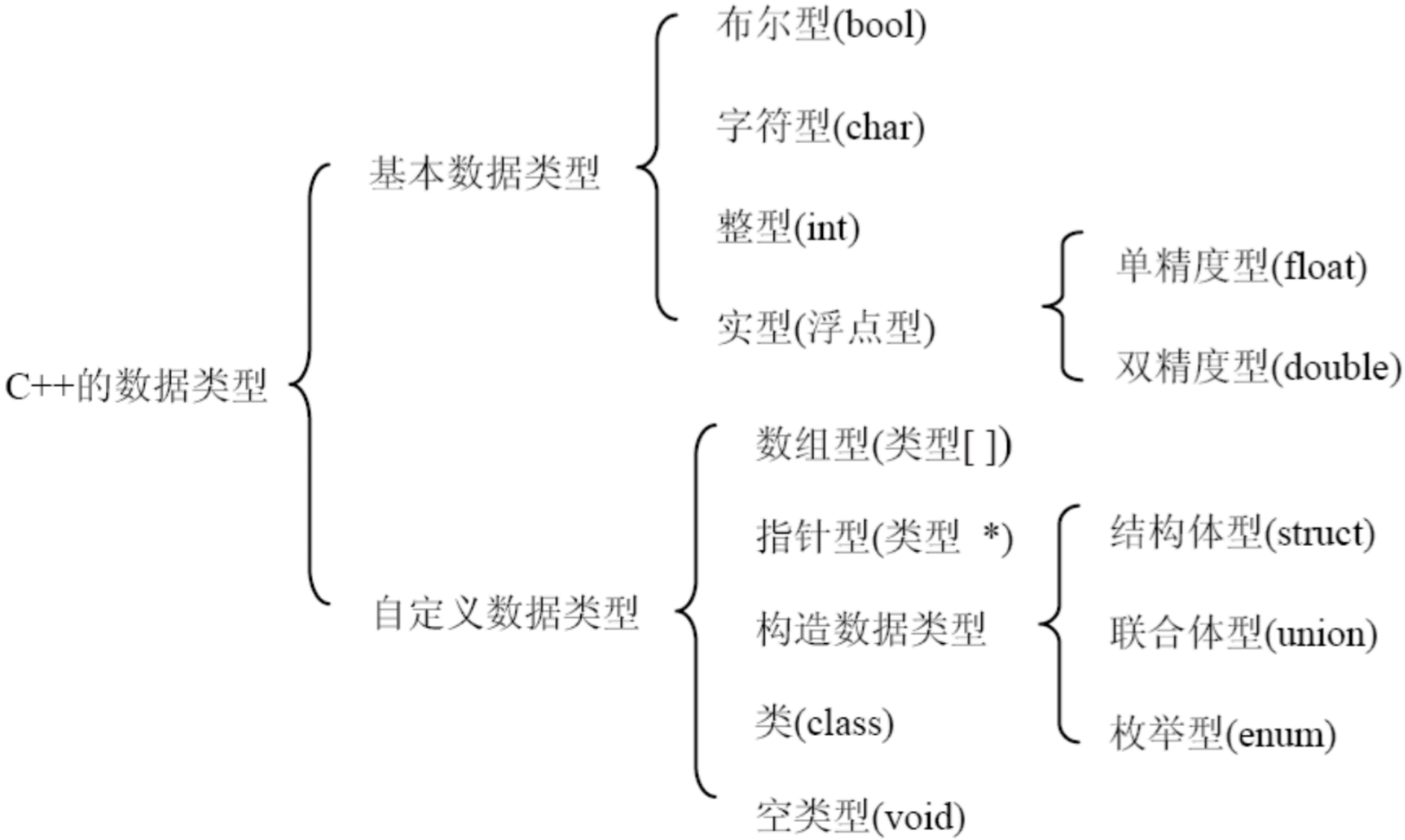


- IEEE 754 Floating-Point Standard
- IEC 60559 Floating-Point Standard

使用须知

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
    float a=0,b=0;
    a=123456.789e5;
    cout<<setprecision(20)<<a<<endl;
    b=a+20;
    cout<<setprecision(20)<<b<<endl;
    return 0;
}
```

C/C++ 程序中的数据类型


C++的数据类型
基本数据类型
布尔型(bool)
字符型(char)
整型(int)
实型(浮点型)
单精度型(float)
双精度型(double)
自定义数据类型
数组型(类型[ ])
指针型(类型 *)
构造数据类型
结构体型(struct)
联合体型(union)
枚举型(enum)
类(class)
空类型(void)

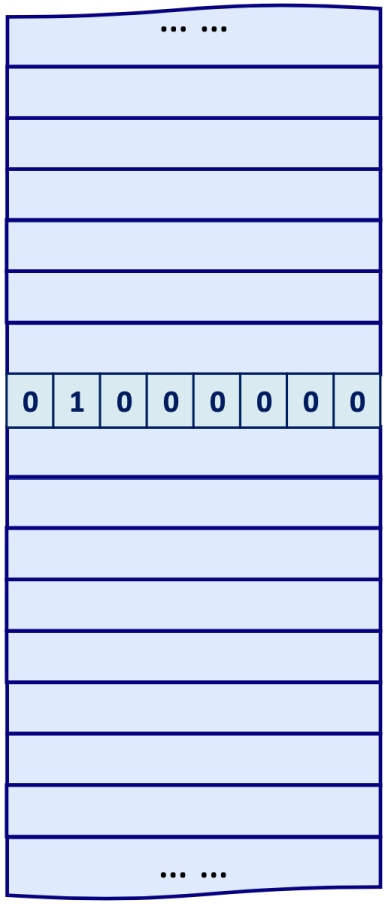
内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

③ 字符型

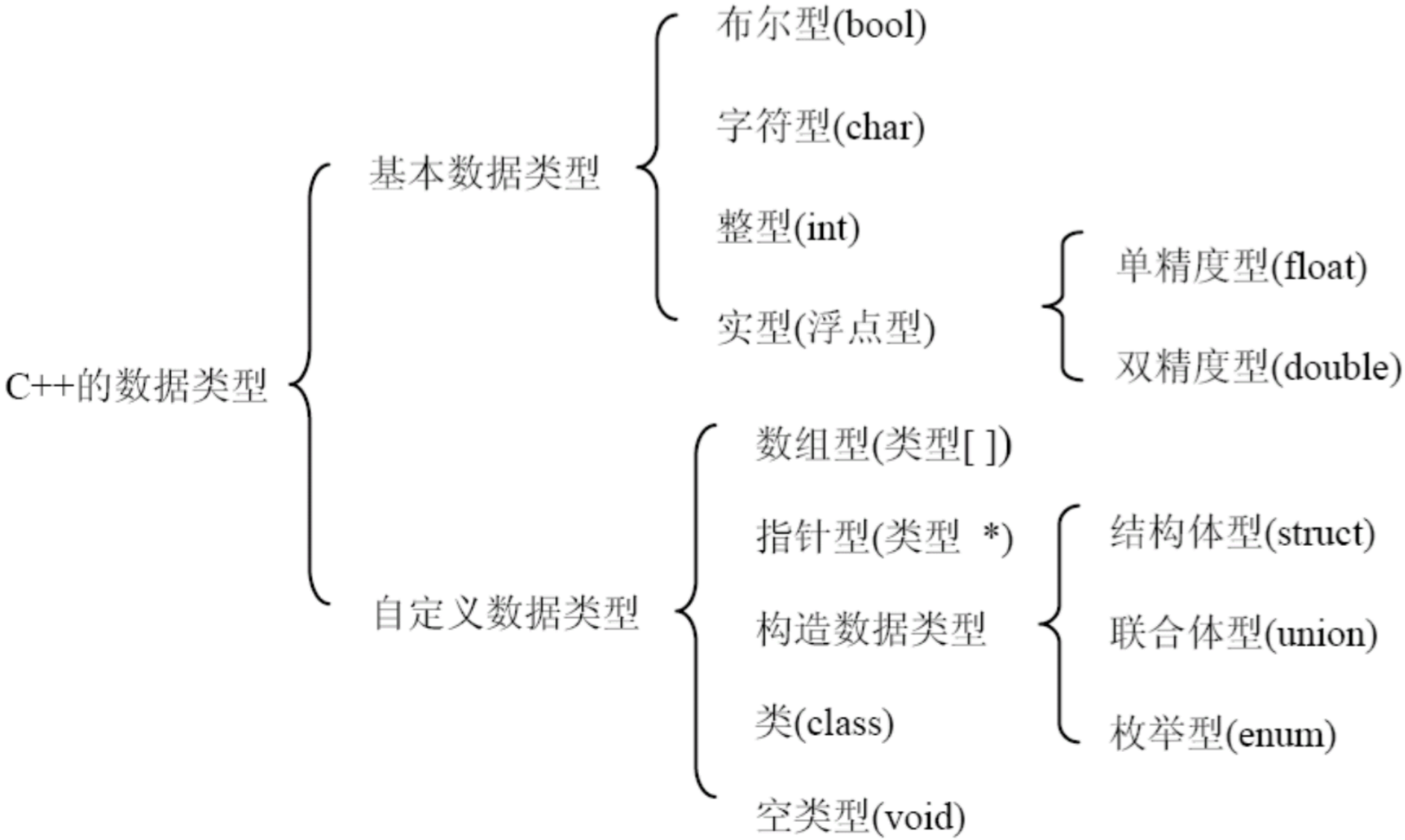


- 一个字符型占一个字节
- `char a='@';`

使用须知

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
    char a=64;
    int b='Z';
    int c=b-a;
    char d=6+256;
    cout<<a<<" "<<b<<" "<<c<<" "<<d<<endl;
    cout<<endl;
    cout<<"This is the first line!\n";
    cout<<'\a'<<'\\'<<'\n';
    return 0;
}
```

C/C++ 程序中的数据类型


C++的数据类型
基本数据类型
布尔型(bool)
字符型(char)
整型(int)
实型(浮点型)
单精度型(float)
双精度型(double)
自定义数据类型
数组型(类型[ ])
指针型(类型 *)
构造数据类型
结构体型(struct)
联合体型(union)
枚举型(enum)
类(class)
空类型(void)

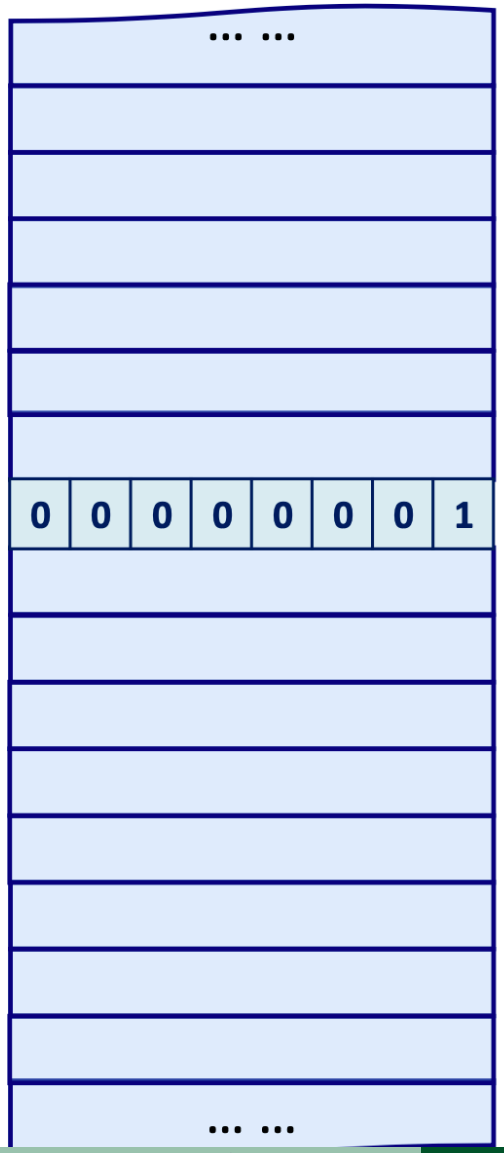
内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

④ 布尔型



布尔型

- 用于存储“真”和“假”的变量
  - 占一个字节
  - 其值 只能为 1 或 0
    - 1 代表 True
    - 0 代表 False
- 赋给布尔型变量的值
  - 可以赋任何值给它，但
    - 赋 0 存 0，表示False
    - 赋 非零 存1，表示True

布尔型

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5      bool b1=true,b2=false;
6      cout<<"b1=true 时, b1="<<b1<<endl;
7      cout<<"b2=false 时, b2="<<b2<<endl;
8      b1=7>3;
9      b2=-100;
10     cout<<"b1=7>3 时, b1="<<b1<<endl;
11     cout<<"b2=-100 时, b2="<<b2<<endl;
12     return 0;
13 }
```

★常量★

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5      const double PI=3.14159;
6      float r,area;
7      cin>>r;
8      area=r*r*PI;
9      cout<<"Area="<<area;
10     return 0;
11 }
```

常量

- 常量：程序运行过程中其值保持不变
- 字面常量：`-1`，`0`，`123`，`4.6`，`-1.23` ⇐ `const`
- 符号常量：用一个标识符代表一个常量 ⇐ *#define*
- 所有“数”都有类型 ⇒ 常量也一样

★ 定义须知 ★

标识符 C/C++

- 用来标识符号常量名、变量名、函数名、数组名、类型名、文件名的有效字符序列（identifier）。
- 只能由字母、数字和下划线组成，且第一个字符必须为字母或下划线，且不可与保留字相同。

保留字 关键字

命名法

- 匈牙利命名法：开头小写字母指定数据类型、其后首字母大写单词指出变量用途 `bOnOff`
- 驼峰命名法：第一个单词全部小写、其后单词全部首字母大写 `myLastName`

内容提要 I

内容提要 II

- 条件运算符
- 强制类型转换
- 运算符优先级

4 C 语言中的控制成分
- 分支结构
- 循环结构

5 C 语言中的传输成分
- stdio.h
- iostream

6 C 程序中的数组
- 定义
- 初始化
- 二维数组

7 C 程序中的字符串
- 定义
- 初始化

内容提要 III

- 字符串数组
- 输入与输出
- 示例

运算符

- 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 `(type)`
- 分量运算符 `. ->`

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

① 赋值运算

- 两边类型不同 ⇒ 自动完成类型转换
- 长数赋给短数 ⇒ 截取长数的低位送给短数
- 短数赋给长数 ⇒ 原来是什么数、现在还是什么数
- 符号位的赋值处理 ⇒ 直接赋值、不管符号位还是数字位

表达式

- 程序中由运算符、操作数和括号等所组成的计算式
- 计算求值的基本单位

```
1 a*b+c;
2 123 < 10;
3 'a' * 3.14f==1;
4 a=b;
```

- 表达式是有“值”的，赋值语句也不例外。

```
1 int main()
2 {
3   int i=0;
4   cout<<(i=10)<<endl;
5   cout<<(i=i+i)<<endl;
6   return 0;
7 }
```

复合赋值运算

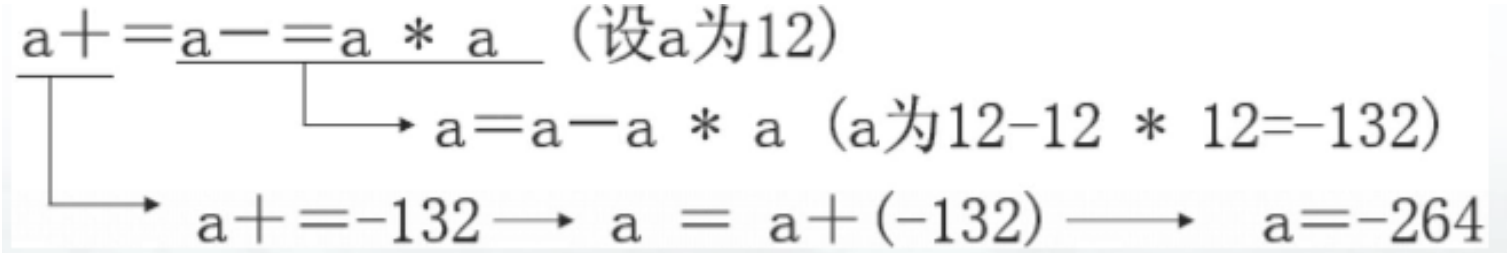
- 在赋值符号前加上其它运算符号则构成**复合赋值运算**

```
a += 3;
a = a+3;

x *= y+8;
x = x*(y+8);

x %= 3;
x = x%3;
```

连续赋值运算

- 自由而左的结合顺序

```
1 a = b = 5;
2 a = b = c = 5;
3 int a=b=c=5;
4
5 a = (b = 4) + (c=6);
```



赋值运算符

- = 给赋值号左边的变量赋予数值
- 在变量定义的同时可以为变量赋初值

```
1 int a=3;
2 int a; a=3;
3
4 int a,b,c=5;
5 int a,b,c; c=5;
6
7 int a=b=c=5; //right?
```

赋值运算符：两边类型不同

- 若 = 两边的类型不一致，赋值时要进行**类型转换**。
- 不管 = 右边的操作数是什么类型，都要转换为 = 左边的类型

```
1 int main()
2 {
3   int int_i=64.12345;
4   char char_i=int_i;
5   float float_i=char_i;
6   bool bool_i=float_i;
7   cout<<showpoint<<int_i<<" "<<char_i<<" "<<float_i<<"
    ↪  "<<bool_i<<endl;
8   return 0;
9 }
```

赋值运算符：长数赋给短数

- 截取长数的**低 n 位**送给短数

```
1 int main()
2 {
3   char char_a='';
4   int int_i=0x361;
5   cout<<hex<<int_i<<endl;
6   char_a=int_i;
7   cout<<char_a<<endl;
8   return 0;
9 }
```

赋值运算符：长数赋给短数

short=long

- 截取长整型数的低16位送给**short**字符。
- 如果最高位为1，则得到负数，否则得到正数。

```
1 int main()
2 {
3   long int long_i=0x2AAAAAAA;
4   cout<<long_i<<endl;
5   short short_j=long_i;
6   cout<<hex<<short_j<<endl;
7   cout<<dec<<short_j<<endl;
8   return 0;
9 }
```

赋值运算符：短数赋给长数

- 最好处理的情况：原来是什么数、现在还是什么数！
- `short int a=-1; int b=a;`

计算机的处理过程：

- 若`short`型数为**无符号数**：
  - `short`型 16 位到`long`型低 16 位，`long`型高 16 位补 0；
- 若`short`型数为**有符号数**：
  - `short`型 16 位到`long`型低 16 位；
  - 若`short`型最高位为 0，则`long`型高 16 位补 0；
  - 若`short`型最高位为 1，则`long`型高 16 位补 1。

赋值运算符：短数赋给长数

- 最好处理的情况：原来是什么数、现在还是什么数！
- `short int a=-1; int b=a;`

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
  short short_i=-123;
  cout<<hex<<short_i<<endl;
  int int_j=short_i;
  cout<<hex<<int_j<<endl;
  cout<<dec<<int_j<<endl;
  return 0;
}
```

赋值运算符：符号位的赋值处理

- 也很好处理的情况：直接搬运！
- 直接赋值，不管符号位还是数字位！

例如：

- `unsigned=int`：直接赋值，符号位当作数字。
- `int=unsigned int`：直接赋值，数字当作符号位。

赋值运算符：符号位的赋值处理

- 也很好处理的情况：直接搬运！
- 直接赋值，不管符号位还是数字位！

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
  unsigned int unsigned_int_i=0xAAAAAAAA;
  cout<<unsigned_int_i<<endl;
  signed int signed_int_i<<endl;
  cout<<hex<<signed_int_j<<endl;
  cout<<dec<<signed_int_j<<endl;
  return 0;
}
```

① 赋值运算

- 两边类型不同 ⇒ 自动完成类型转换
- 长数赋给短数 ⇒ 截取长数的低位送给短数
- 短数赋给长数 ⇒ 原来是什么数、现在还是什么数
- 符号位的赋值处理 ⇒ 直接赋值、不管符号位还是数字位

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

② 算术运算

算术运算符和算术表达式

- 基本的算术运算：+ - * / %
- % 是模运算，即求余运算，必须是整数。7%4=3

注意：

- 整数运算，结果仍为整数；
- 实数运算，结果为double型；
- 舍入的方向随编译器的不同而不同。

算术表达式

- 算术运算符的优先级

```
1 ( )
2 * / %
3 + -
```

- 在同一级别中，采取由左至右的结合方向
  - 如：a-b+c 相当于 (a-b)+c
  - 如：a%b*c/d 相当于 (((a%b)*c))/d
- 当数据类型非常复杂时，可采用剪刀法求表达式的值
  - 如：123%s+(i+'@')+i*u-f/d
- 计算过程中注意类型转换

算术表达式

- 自增、自减运算符：使变量的值加 1 或减 1
- ++i / --i：在使用 i 之前，先将 i 的值加/减 1
- i++ / i--：在使用 i 之后，再将 i 的值加/减 1
- 自增和自减运算符只能用于变量

```
1 int i=3;
2 j=++i;
3 j=i++;
4 cout<<++i;
5 cout<<i++;
```

```cpp
int main()
{
    int a=0,b=0,c=2,d=0,e=2,f=2;
    cout<<a<<" "<<a++<<" "<<endl;
    cout<<++b<<" "<<b++<<" "<<endl;
    cout<<c<<" "<<(c++)+(++c)<<" "<<endl;
    cout<<(d=f++)+(e=f)<<endl;
    cout<<f<<" "<<d<<" "<<e<<endl;
    return 0;
}
```

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

③ 关系运算

- 优先级高：< <= > >=
- 优先级低：== !=
- 关系运算表达式的值：“真”/“假”
  如：a=3;b=4;
  - a>b的值为0（假）
  - a!=b的值为1（真）
  - a==b的值为0（假）

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

④ 逻辑运算

- 三种逻辑运算符：逻辑与&&、逻辑或||、逻辑非!
- 若 A、B 为真，则F=A&&B为真。
- 若 A、B 之一为真，则F=A||B为真。
- 若 A 为真，则!A为假。

思考

```
#include<iostream>
using namespace std;
int main()
{
    int a=0,b=0;
    a=5>3&&2||8<4-(b=!0);
    cout<<a<<" "<<b<<endl;
    return 0;
}
```

逻辑运算的取舍

- 逻辑表达式求解中，并不总是执行所有的运算
- 只有在必须执行下一个逻辑运算符才能求出表达式的解时，才执行该运算符！
- 如：对于表达式a&&b&&c
  - 只有 a 为真（非 0）时，才需要判别 b 的值；
  - 只有 a 和 b 都为真的情况下才需要判别 c 的值。
- 如：对于表达式a||b||c
  - 只要 a 为真（非 0），就不必判断 b 和 c；
  - 只有 a 为假，才判别 b；
  - a 和 b 都为假时才判别 c。

示例

```
#include<iostream>
using namespace std;
int main()
{
    int i=0,x=1,y=2,z=3;
    i=++x||++y||z++;
    cout<<i<<" "<<x<<" "<<y<<" "<<z<<endl;
    return 0;
}
```

运算对象的扩展

- 逻辑运算符的两侧可以是任何类型，如字符型、实型或指针型等。
- 系统最终以 0 和非 0 来判定它们，如：`'c'&&'d'`。
- 思考：`'a'=='b'&&!'c'`

示例

```
#include<iostream>
using namespace std;
int main()
{
    int a=0,b=1;
    a = 8>4-(b=!'c')&&5>3+'a'%6=='b';
    cout<<a<<" "<<b<<endl;
    return 0;
}
```

应用

- 问题：要判别某一年 year 是否闰年。闰年的条件是符合下面二者之一：
  1. 能被 4 整除，但不能被 100 整除。
  2. 能被 100 整除，又能被 400 整除。
- 求解：可以用如下的逻辑表达式来判别 year 是否为闰年：
  1. `(year%4==0&&year%100!=0)||year%400==0`
  2. `(year%4!=0)||(year%100==0&&year%400!=0)`

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

⑤ 逗号运算

- 用逗号将两个表达式连起来：
  - 表达式 1，表达式 2，表达式 3，…，表达式 n
  - 先求表达式 1，再求表达式 2，…，再求表达式 n
  - 整个表达式的值为表达式 n 的值，如：`a=3*5,a*4;`
- 下式是否相同：
  - `x=(a=3,6*3);`
  - `x=a=3,6*3;`

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

⑥ 条件运算

- 表达式 1 ? 表达式 2 : 表达式 3
- 如果表达式 1 的值为真，则以表达式 2 的值作为条件表达式的值；
- 否则以表达式 3 的值作为整个条件表达式的值。
- max = (a>b) ? a : b; 相当于

```
1 if(a>b)
2     max=a;
3 else
4     max=b;
```

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

⑦ 强制类型转换

- 形式：(类型名) 表达式
- 举例：
  - (double)a 将 a 的值转换成 double 类型
  - (int)(x+y) 将 x+y 的值转换成 int 类型
  - (float)(5/3) 将 5/3 的值转换成 float 类型
- 注意：强制类型转换后，被转换的量的类型并没有发生变化。

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

⑧ 运算符优先级

- 同一优先级的运算符，运算次序由结合方向决定。
- 不同的运算符要求由不同的运算对象个数，如单目、双目、三目等。
- 各类运算符的优先级大致可归纳为：
  初等运算符 > 单目运算符 > 算术运算符（先乘除、后加减）> 关系运算符 > 逻辑运算符（不包括!）> 条件运算符 > 赋值运算符 > 逗号运算符

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

计算机程序的基本结构

- 什么样的结构才能支持程序运行的逻辑？
- 论文：C. Bohm and G. Jacopini, Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules, *Communications of the ACM*, 1966.
- 从理论上证明了“任何具有单入口单出口的程序都可以用三种基本结构表达”：
  - 顺序结构
  - 分支结构
  - 循环结构

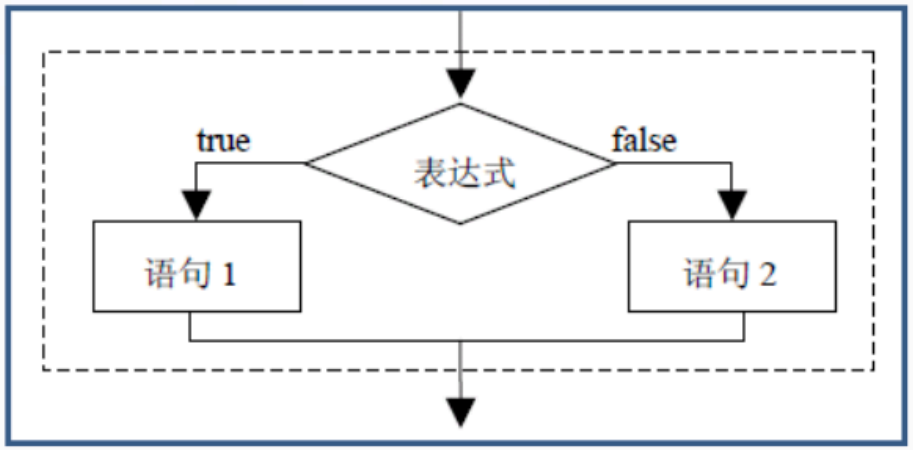
内容提要 I

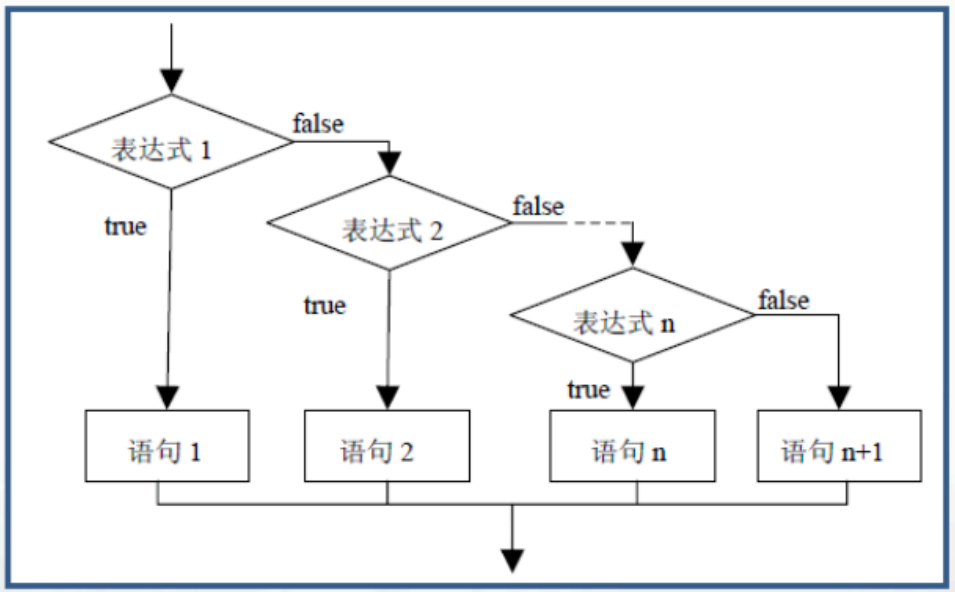
内容提要 II

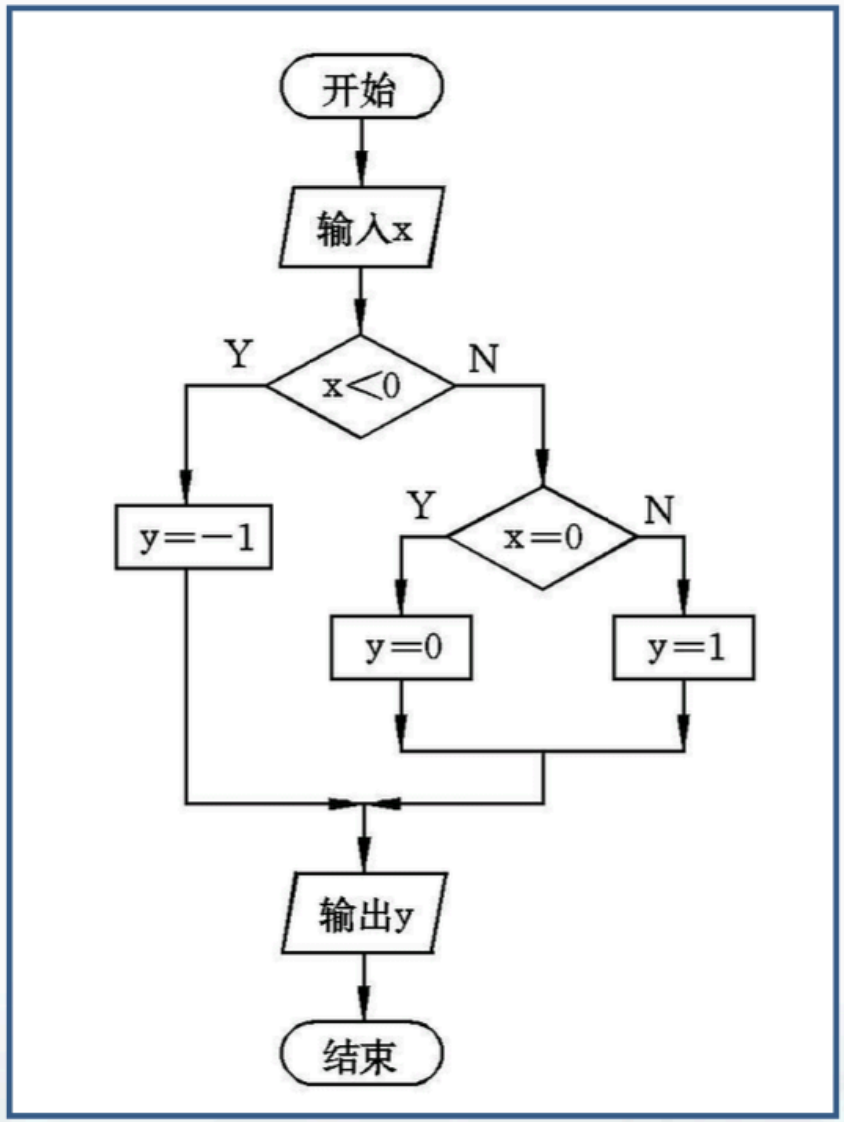
内容提要 III

- 字符串数组
- 输入与输出
- 示例

① 分支语句


true
表达式
false
语句 1
语句 2


表达式 1
false
true
表达式 2
false
true
表达式 n
false
true
语句 1
语句 2
语句 n
语句 n+1


开始
输入x
Y
x<0
N
y=−1
Y
x=0
N
y=0
y=1
输出y
结束

```cpp
#include<iostream>
using namespace std;
int main()
{
    float weight=0,height=0,healthRate=0;
    cin>>weight>>height;//weight=>kg height=>m
    healthRate=weight/(height*height);
    if((18<=healthRate)&&(healthRate<=25))
    cout<<" 体重适中！"<<endl;
    else if((25<healthRate)&&(healthRate<=30))
    cout<<" 超重！注意控制！"<<endl;
    else if((30<healthRate)&&(healthRate<=35))
    cout<<" 肥胖！减肥吧！"<<endl;
    else if((35<healthRate)&&(healthRate<=40))
    cout<<" 重度肥胖！别吃了！"<<endl;
    else
    cout<<" 请直接拨打 120！"<<endl;
    return 0;
}
```

```cpp
#include<iostream>
using namespace std;
int main()
{
    int year=0;
    cin>>year;
    if(year%4==0)
    {
    if(year%100==0)
    {
        if(year%400==0)
        cout<<"Y";
        else
        cout<<"N";
    }
    else
        cout<<"Y";
    }
    else
    cout<<"N";
    return 0;
}
```

if 语句

- 在执行 if 语句前先对表达式求解
- if()内可以是任意的数值类型
  包括整型、实型、字符型、指针型数据
  - `if('a') cout<<'a'<<endl;`
  - `if(3) cout<<"OK"<<endl;`
- 若表达式的值为 0，按“假”处理；
- 若表达式的值为非 0，按“真”处理。

switch 语句

```
1 switch(表达式)
2 {
3     case 常量表达式1: 语句1;
4     case 常量表达式2: 语句2;
5     ...
6     case 常量表达式n: 语句n;
7     default: 语句n+1;
8 }
```

- 当 表达式的值与某一个 case 后面的 常量表达式的值相等时，就执行此 case 后面的语句；
- 若所有的 case 中的 常量表达式的值都没有与表达式的值匹配的，就执行 default 后面的语句。

```cpp
#include<iostream>
using namespace std;
int main()
{
    char grade='a';
    cin>>grade;
    switch(grade)
    {
    case 'a': cout<<"85~100"<<endl;
    case 'b': cout<<"70~84"<<endl;
    case 'c': cout<<"60~69"<<endl;
    case 'd': cout<<"<60"<<endl;
    default: cout<<"Error"<<endl;
    }
    return 0;
}
```

```cpp
#include<iostream>
using namespace std;
int main()
{
    char grade='a';
    cin>>grade;
    switch(grade)
    {
    case 'a': cout<<"85~100"<<endl;break;
    case 'b': cout<<"70~84"<<endl;break;
    case 'c': cout<<"60~69"<<endl;break;
    case 'd': cout<<"<60"<<endl;break;
    default: cout<<"Error"<<endl;
    }
    return 0;
}
```

```
#include<iostream>
using namespace std;
int main()
{
    char grade='a';
    cin>>grade;
    switch(grade)
    {
    case 'a': cout<<"85~100"<<endl;
    case 'b': cout<<"70~84"<<endl;
    case 'c':
    case 'd':
    case 'e':
    case 'f': cout<<"60~69"<<endl;
    case 'g': cout<<"<60"<<endl;
    default: cout<<"Error"<<endl;
    }
    return 0;
}
```

```cpp
#include<iostream>
using namespace std;
int main()
{
    char grade='a';
    cin>>grade;
    switch(grade)
    {
    case 'a': cout<<"85~100"<<endl;
    default: cout<<"Error"<<endl;
    case 'b': cout<<"70~84"<<endl;
    case 'c':
    case 'd':
    case 'e':
    case 'f': cout<<"60~69"<<endl;
    case 'g': cout<<"<60"<<endl;
    }
    return 0;
}
```

练习

- 学校要求实行成绩等级制度。现在已经有同学们的百分制成绩，要求按照百分制成绩输出相应的等级成绩。其中 90–100 分为'A'，80–90 分为'B'，70–80 分为'C'，60–70 分为'D'，60 分以下为'E'。
- 最直接的办法（方案）：
  1. 输入成绩；
  2. 判定成绩是否：90<= 成绩 <=100 · · ·
  3. 判定成绩是否：80<= 成绩 <=90 · · ·
  4. · · ·
- 如果用 `switch` 语句，如何解决？

```
#include<iostream>
using namespace std;
int main()
{
    int score=0,num=0;
    cout<<"Please give the score:"<<endl;
    cin>>score;
    num=score/10;
    switch(num)
    {
    case 10: cout<<"A"<<endl;break;
    case 9: cout<<"A"<<endl;break;
    case 8: cout<<"B"<<endl;break;
    case 7: cout<<"C"<<endl;break;
    case 6: cout<<"D"<<endl;break;
    default: cout<<"E"<<endl;
    }
    return 0;
}
```

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

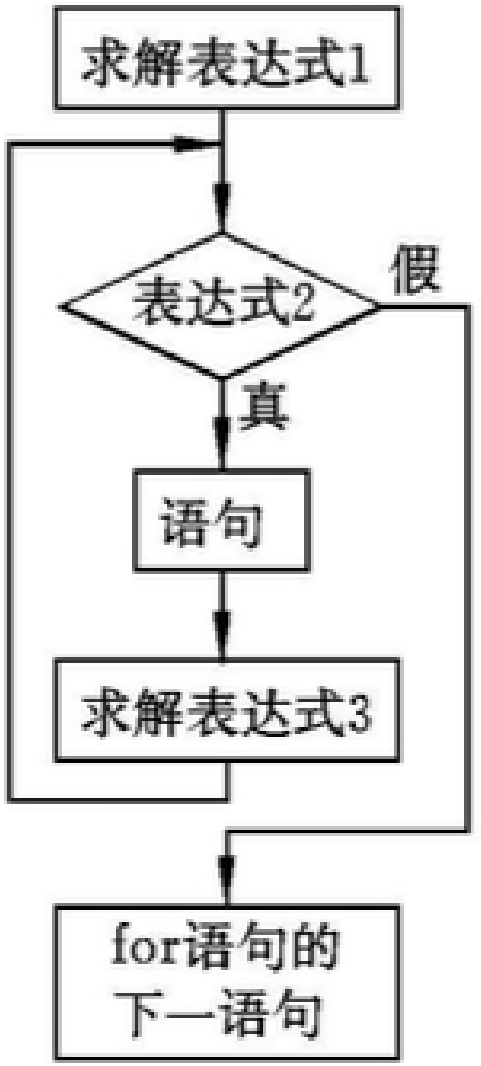
② 循环语句

- 循环结构
  - `for` 语句
  - `while` 语句
  - `do ... while` 语句
  - `goto ... if` 语句
- 循环中止或跳出语句
  - `continue` 语句跳出**本次循环**
  - `break` 语句跳出**本层循环**

for 语句

```
for(表达式1; 表达式2;
 ↪  表达式3)
    语句;
```

```
for(i=0; i<10; i++)
{
    a = a*a;
}
```



while 语句

```
while(条件语句)
{
    执行语句;
}
```

```
#include<iostream>
using namespace std;
int main()
{
    int i,sum=0;
    i=1;
    while(i<=100)
    {
        sum=sum+i;
        i++;
    }
    cout<<sum<<endl;
    return 0;
}
```

示例

- 小红 10 岁，父亲 33 岁，问多少年之后，父亲的年龄是小红的二倍？

```
#include<iostream>
using namespace std;
int main()
{
    int ageOfHong=10,ageOfFather=33,count=0;
    while(2*ageOfHong != ageOfFather)
    {
        ageOfHong++;
        ageOfFather++;
        count++;
    }
    cout<<count;
    return 0;
```

do ... while 语句

```
do
{
    执行语句;
} while(条件语句);
```

```
#include<iostream>
using namespace std;
int main()
{
    int i,sum=0;
    i=1;
    do
    {
        sum=sum+i;
        i++;
    } while(i<=100);
    cout<<sum<<endl;
    return 0;
}
```

示例

```
#include<iostream>
using namespace std;
int main()
{
    int num; int count=0;
    cout<<"Please enter an integer:"<<endl;
    cin>>num;
    do {
            cout<<num%10;
            num=num/10;
            count++;
    } while(num!=0);
    cout<<count<<"digits"<<endl;
    return 0;
}
```

循环语句嵌套

```
(1) while( )
      {…
       while( )
        {…}
      }
```

```
(2) do
      {…
        do
          {… }
        while( );
      }
    while( );
```

```
(3) for(;;)
      {
        for(; ;)
          {…}
      }
```

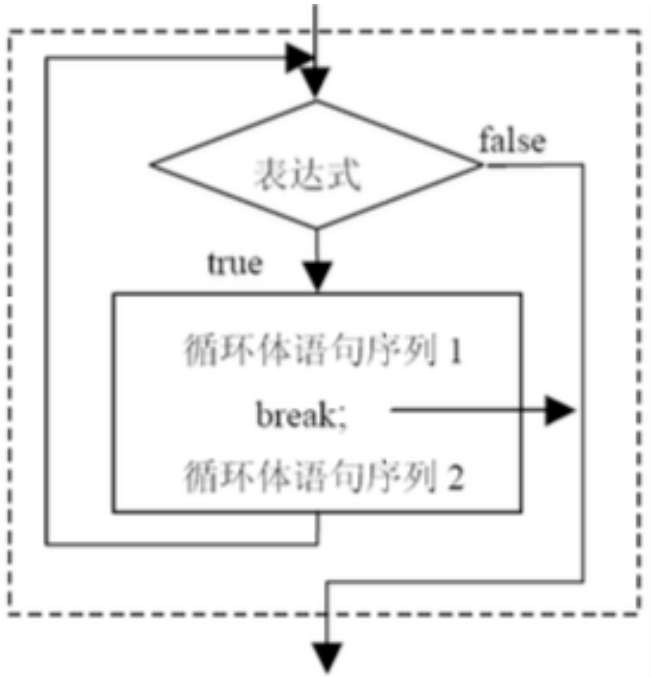
```
(4) while( )
      {…
       do
         {…}
       while( );
…
      }
```

```
(5) for(; ;)
      {…
         while( )
           { }
…
      }
```

```
(6) do
      {
…
          for (; ;)
            { }
      }
    while( );
```

转向控制语句 break

- 在switch语句、while语句、do-while语句、for语句中使用；
- 以跳出switch语句或内层循环，继续执行逻辑上的下一条语句。



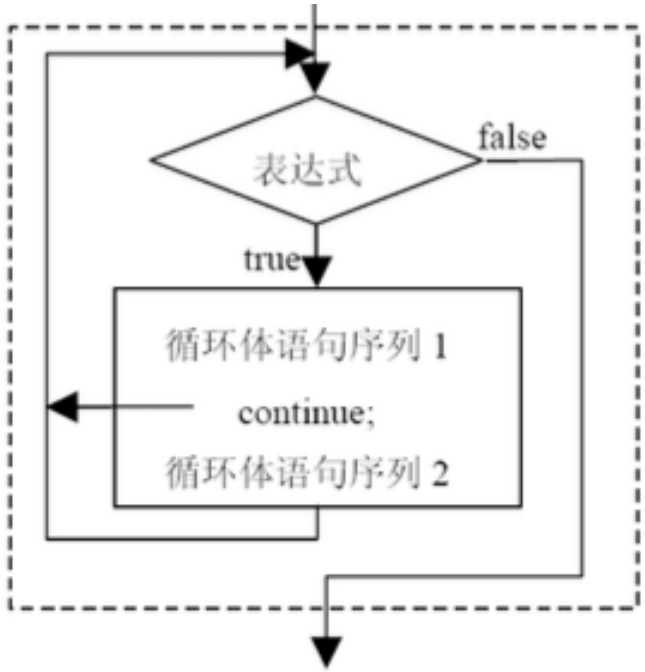
示例

```cpp
#include<iostream>
using namespace std;
int main()
{
    int n=0;
    for(;;)
    {
        cin>>n;
        if(n==0)
            break;
    }
    return 0;
}
```

```cpp
#include<iostream>
using namespace std;
int main()
{
    int n=0;
    while(true)
    {
        cin>>n;
        if(n==0)
            break;
    }
    return 0;
}
```

转向控制语句 continue

- 用于循环语句中；
- 结束本次循环，接着立即测试循环控制表达式，判断是否继续执行下一次循环。



示例

```
#include<iostream>
using namespace std;
int main()
{
    int n,counter=0;
    for(n=1;n<=100;n++)
    {
        if(n%3==0||n%5==0||n%7==0)
            continue;
        cout<<n<<'\t';
        counter++;
        if(counter%10==0)
            cout<<endl;
    }
    cout<<endl;
    return 0;
}
```

goto 语句

- 无条件转向语句
- goto 语句标号;
- 语句标号：标识符（命名规则与变量名相同）
- goto loop1;

示例

汇编语言

```
load 0 a 数据装入寄存器0
load 1 b 数据装入寄存器1
loop:
mult 0 1
 ↪  寄存器0与1的数据相乘
load 1 c 数据装入寄存器1
add 0 1
 ↪  寄存器0与1的数据相加
goto loop
save 0 d 保存寄存器0里的数据
```

```cpp
#include<iostream>
using namespace std;
int main()
{
    int i=0,sum=0;
    i=1;
loop:
    sum=sum+i;
    i++;
    goto loop;
    cout<<sum<<endl;
    return 0;
}
```

关于 goto 语句的讨论

- 60 年代末至 70 年代，关于 goto 语句的争论非常激烈
- **正方**：从高级语言中去掉 goto 语句
  - 包含 goto 语句的程序难以阅读，难以查错；
  - 去掉 goto 语句后，可以直接从程序结构上反映程序的运行过程，使程序的结构清晰、便于阅读，便于查错，而且也有利于程序正确性的证明。
- **反方**：goto 语句无害，应该保留
  - goto 语句使用起来比较灵活，而且有些情形能够提高程序的效率。
  - 如果一味强调删除 goto 语句，有些情形反而会使程序过于复杂，增加一些不必要的计算量。

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

文件描述符

- File Descriptor (FD) is an abstract indicator (handle) used to access a file or other input/output resource, such as a pipe or network socket.
- File Descriptors form part of the POSIX application programming interface.
- A File Descriptor is a non-negative integer, generally represented in the C programming language as the type `int`.

文件描述符	通道名	描述	默认连接	用途
0	`stdin`	标准输入	键盘	read only
1	`stdout`	标准输出	终端	write only
2	`stderr`	标准错误	终端	write only
3 及以上	`filename`	其他文件	none	read and/or write

标准输入输出

理解**重定向**与**管道**

```
$ cat > test1
hello
hi
^C
$ cat > test2
hi
hello
^C
$ cat > test1 < test2
```

```
$ ls -l
$ ls -l > ls-l
$ cat ls-l

$ dddd
$ dddd > /dev/null
$ dddd 2> /dev/null
$ dddd > /dev/null 2>&1
```

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

stdio.h

- Standard Input/Output
- C 的标准 I/O 库
- 以函数的方式向 *buffer* 写入或读取字符
- `scanf()` `printf()`

`iostream`

- Input/Output Stream
- C++ 的标准 I/O 库（类库）
- 引入了输入/输出流的概念
- 以类的方法从 *streambuf* 中写入或读取字符
- `cin>>` `cout<<`

scanf

- scanf是从标准输入中读取字符，然后进行相关转化；
- 如果是空格或者换行，直接跳过；
- 如果不符合格式化要求，不处理同时也不减少缓冲区中的信息。

```
1  #include <stdio.h>
2  int main()
3  {
4      char ch1, ch2;
5  //    scanf("%c", &ch1);
6  //    scanf("%c", &ch2);
7      scanf("%c %c", &ch1, &ch2);
8      printf("%d %d\n", ch1, ch2);
9      return 0;
10 }
```

scanf

1. 将数据按照行缓冲送到键盘缓冲区；
2. 按照格式化要求从缓冲区中读取数据到相应内存空间。
3. scanf并不能保证后面的参数都被正确赋值。

```
1  #include <stdio.h>
2  int main()
3  {
4      char str1[20], str2[20];
5      scanf("%s", str1);
6      printf("%s\n", str1);
7      scanf("%s", str2);
8      printf("%s\n", str2);
9      return 0;
10 }
```

getchar

```c
#include <stdio.h>
int main()
{
    char ch1, ch2;
    ch1 = getchar();
    ch2 = getchar();
    printf("%d %d\n", ch1, ch2);
    return 0;
}
```

scanf vs. getchar

- scanf是格式输入函数（按用户指定的格式从键盘上把数据输入到指定的变量中）
- scanf输入时要用一个以上的空格或回车符作为每两个输入数之间的间隔
- scanf读取数字时会跳过空格、制表符和换行符

- getchar是键盘输入函数（从键盘上输入一个字符）
- getchar功能就是接收一个字符

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

iostream

- Input/Output Stream
- C++ 的标准 I/O 库（类库）
- 引入了输入/输出流的概念
- 以类的方法从 *streambuf* 中写入或读取字符
- `cin>>` `cout<<`

`stdio.h`

- Standard Input/Output
- C 的标准 I/O 库
- 以函数的方式向 *buffer* 写入或读取字符
- `scanf()` `printf()`

cin

- 如果cin读取整数，会自动忽略换行和空格。
- 遇到文件结束标记（EOF），cin>>a返回 False。
- cin读取字符串，会将\n作为字符串的分隔符。

```
1 int a;
2 while(cin>>a) {
3     cout<<a<<endl;
4 }
```

- cin读取字符串char *，读取完毕，会自动将\n转换成\0。

```
1 string a;
2 while(cin>>a) {
3     cout<<a;
4 }
```

scanf vs. cin

- scanf是格式化输入，printf是格式化输出。
- cin是输入流，cout是输出流。
- 格式化输出效率比较高，但是写代码麻烦。
- 流输出操作效率稍低，但书写简便。
- cout之所以效率低，是先把要输出的东西存入缓冲区再输出。

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

内容提要 I

内容提要 II

内容提要 III

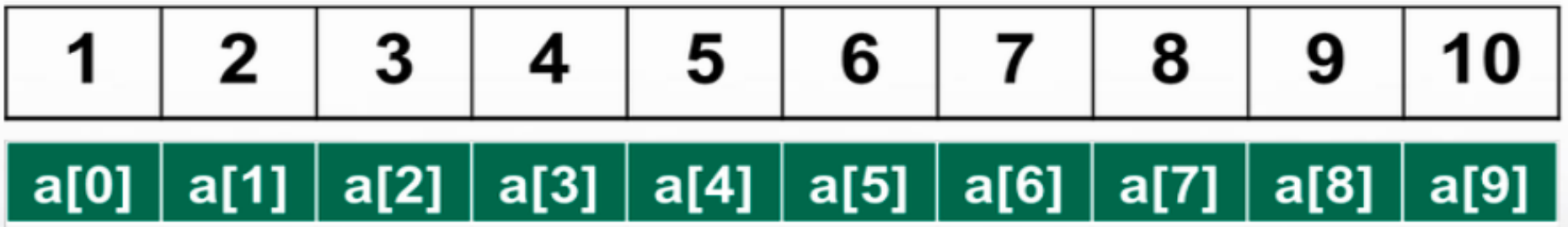
- 字符串数组
- 输入与输出
- 示例

数组的定义

```
1 类型 数组名[常量表达式];
2 float sheep[10];
3 int a2001[1000];
```

数组下标从 0 开始

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



关于数组的定义

- 常量表达式

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5      int n=10;
6      int a[n]={0};
7      for (int i=0;i<10;i++)
8          cout<<a[i];
9      return 0;
10 }
```

关于数组的定义

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5      const int i=4;
6      int a[i]={1,2,3,4};
7      cout <<"a[0]="<<a[0]<<endl
8           <<"a[1]="<<a[1]<<endl
9           <<"a[2]="<<a[2]<<endl
10          <<"a[3]="<<a[3]<<endl;
11     return 0;
12 }
```

关于数组的定义

```
1  #include<iostream>
2  using namespace std;
3  #define N 4
4  int main()
5  {
6      int a[N]={1,2,3,4};
7      cout <<"a[0]="<<a[0]<<endl
8           <<"a[1]="<<a[1]<<endl
9           <<"a[2]="<<a[2]<<endl
10          <<"a[3]="<<a[3]<<endl;
11     return 0;
12 }
```

内容提要 I

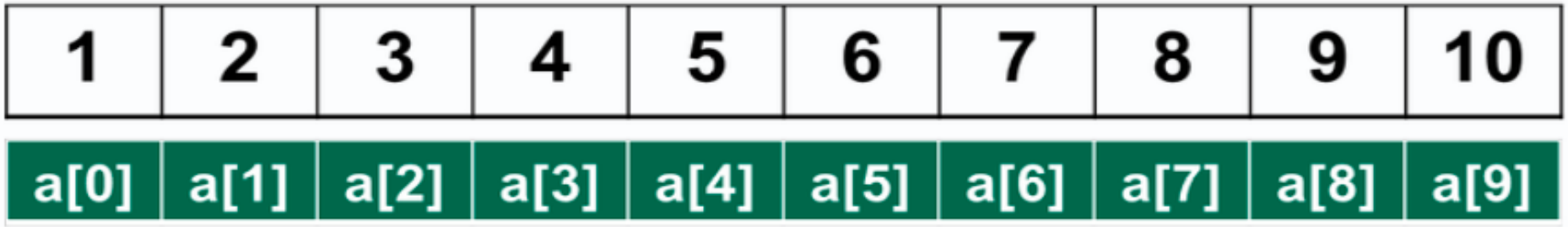
内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

数组的初始化

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



数组的初始化

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

数组的初始化

```
#include<iostream>
using namespace std;
int main()
{
    int a[]={1,2,3,4};
    cout<<a[0]<<a[1]<<a[2]<<a[3]<<endl;
    return 0;
}
```

数组的初始化

```
#include<iostream>
using namespace std;
int main()
{
    int a[4]={1,2};
    cout<<a[0]<<a[1]<<a[2]<<a[3]<<endl;
    return 0;
}
```

数组的初始化

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

数组的初始化

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

数组的初始化

```
#include<iostream>
using namespace std;
int main()
{
    int a[4]={1,2,3,d};
    cout<<a[0]<<a[1]<<a[2]<<a[3]<<endl;
    return 0;
}
```

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

从一维数组到二维数组

```
1 int a[12]={1,2,3,4,5,6,7,8,9,10,11,12};
```

1	2	3	4	5	6	7	8	9	10	11	12
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]

二维数组

```
1 int a[3][4];
```

	0列	1列	2列	3列
0行	**1** a[0][0]	**2** a[0][1]	**3** a[0][2]	**4** a[0][3]
1行	**5** a[1][0]	**6** a[1][1]	**7** a[1][2]	**8** a[1][3]
2行	**9** a[2][0]	**10** a[2][1]	**11** a[2][2]	**12** a[2][3]

二维数组的初始化

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
    int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
    for(int i=0;i<3;i++)
    {
        for(int j=0;j<4;j++)
            cout<<setw(3)<<a[i][j];
        cout<<endl;
    }
    return 0;
}
```

二维数组的初始化

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
    int a[3][4]={1,2,3,4,5,6,7,8,9,10,11,12};
    for(int i=0;i<3;i++)
    {
        for(int j=0;j<4;j++)
            cout<<setw(3)<<a[i][j];
        cout<<endl;
    }
    return 0;
}
```

二维数组的初始化

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
    int a[][4]={1,2,3,4,5,6,7,8,9,10,11,12};
    for(int i=0;i<3;i++)
    {
        for(int j=0;j<4;j++)
            cout<<setw(3)<<a[i][j];
        cout<<endl;
    }
    return 0;
}
```

二维数组的初始化

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
    int a[][4]={{1},{0,6},{0,0,11}};
    for(int i=0;i<3;i++)
    {
        for(int j=0;j<4;j++)
            cout<<setw(3)<<a[i][j];
        cout<<endl;
    }
    return 0;
}
```

二维数组的初始化

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
    int a[3][4]={0};
    for(int i=0;i<3;i++)
    {
        for(int j=0;j<4;j++)
            cout<<setw(3)<<a[i][j];
        cout<<endl;
    }
    return 0;
}
```

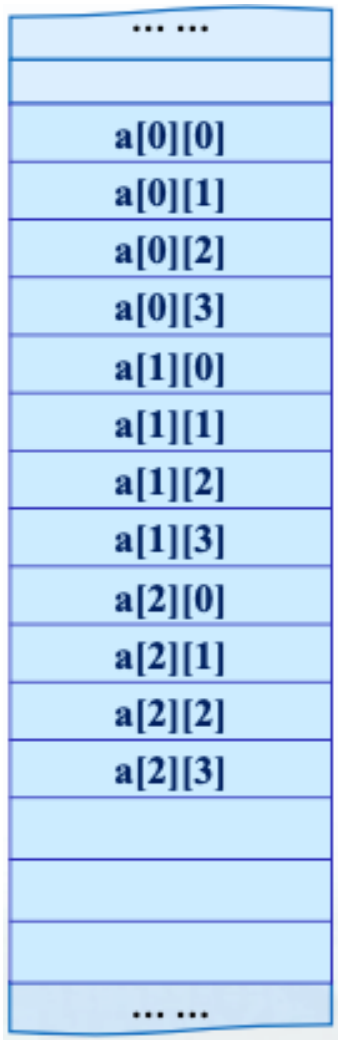
二维数组的初始化

```
1 int main()
2 {
3     int a[3][4]={0};
4     for(int i=0;i<3;i++)
5         for(int j=0;j<4;j++)
6             a[i][j]=4*i+j+1;
```

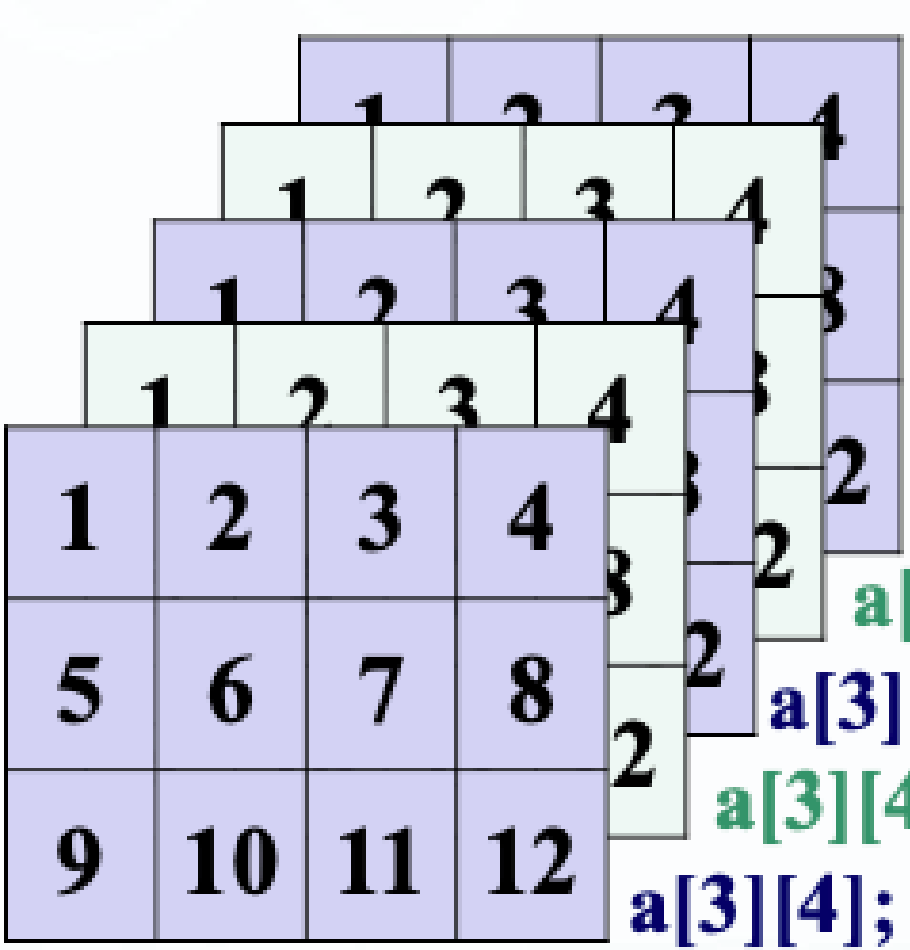
二维数组

```
1 int a[3][4];
```

	0列	1列	2列	3列
0行	**1** a[0][0]	**2** a[0][1]	**3** a[0][2]	**4** a[0][3]
1行	**5** a[1][0]	**6** a[1][1]	**7** a[1][2]	**8** a[1][3]
2行	**9** a[2][0]	**10** a[2][1]	**11** a[2][2]	**12** a[2][3]



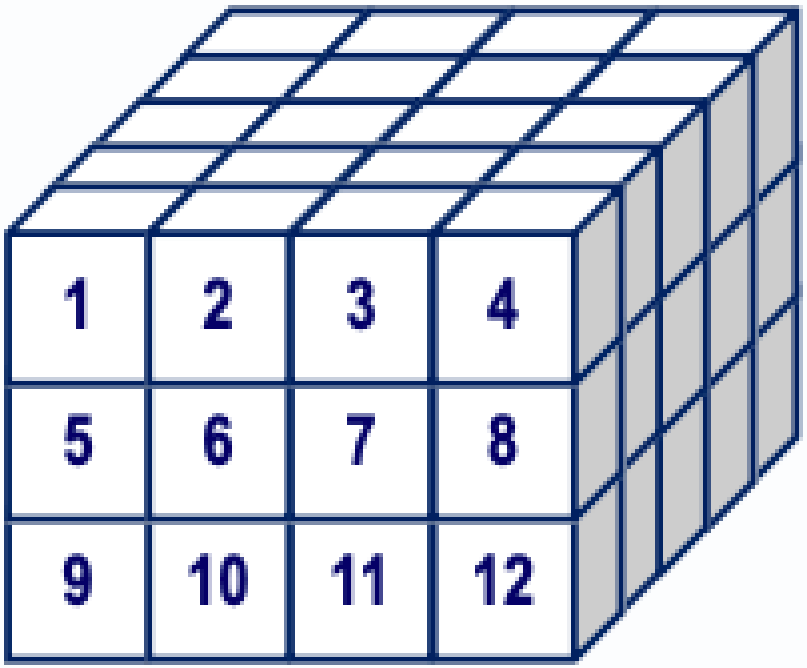
三维数组




```
int a[5][3][4];
```

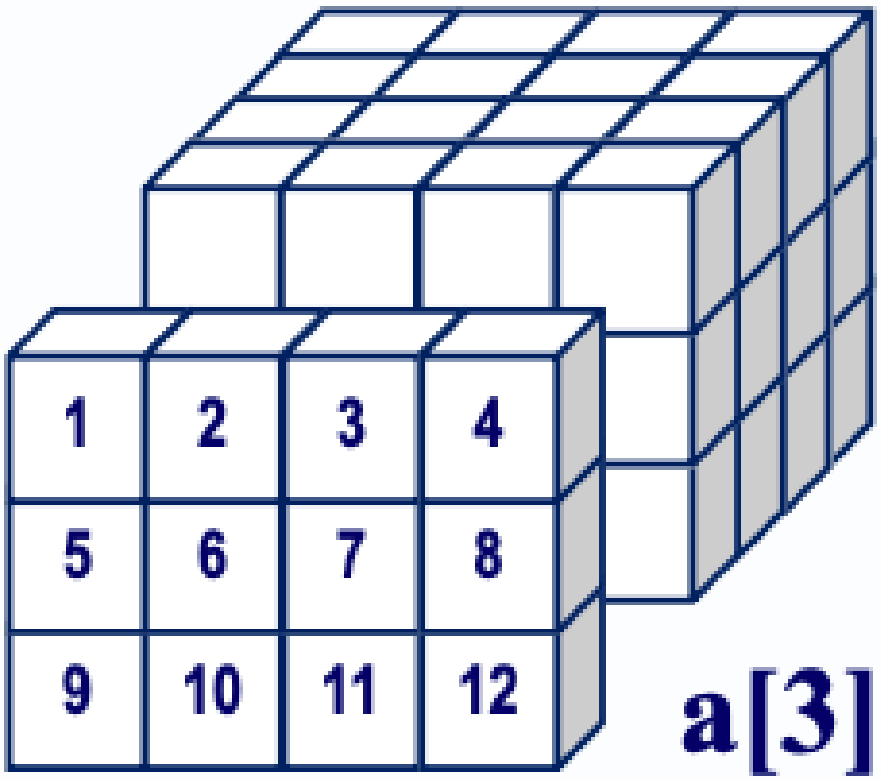
a[4][3][4];
a[3][3][4];
a[2][3][4];
a[1][3][4];
a[0][3][4];

三维数组



int a[5][3][4];

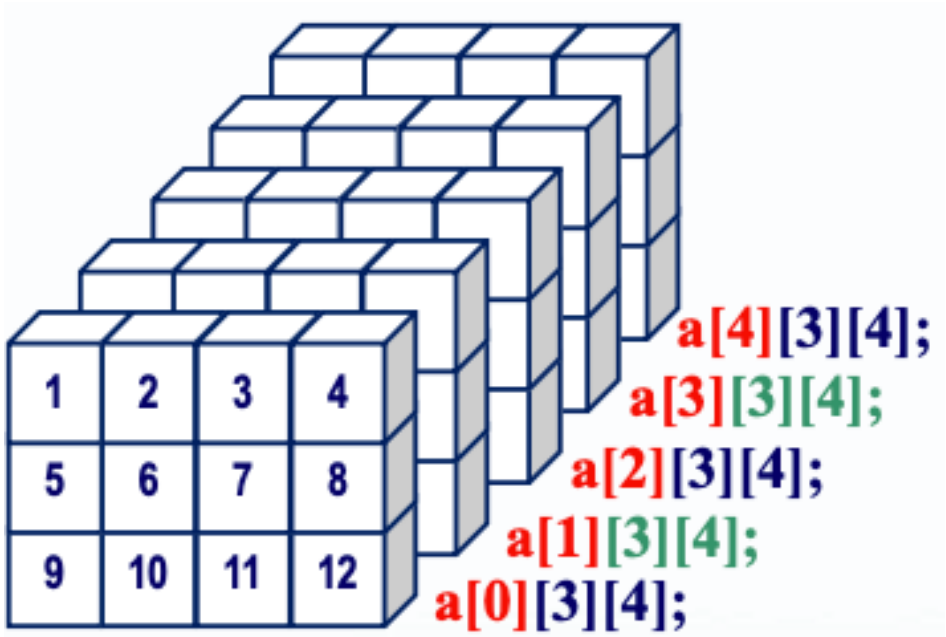
三维数组

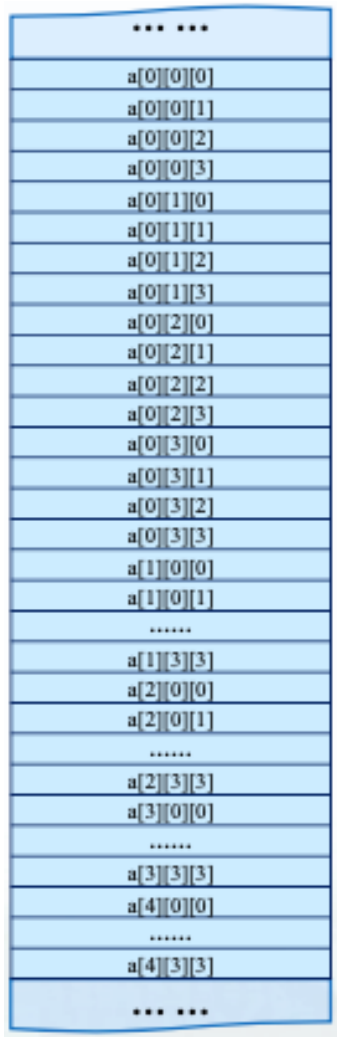


int a[5][3][4];

a[3][4];

三维数组


1
2
3
4
5
6
7
8
9
10
11
12
a[4][3][4];
a[3][3][4];
a[2][3][4];
a[1][3][4];
a[0][3][4];


… …
a[0][0][0]
a[0][0][1]
a[0][0][2]
a[0][0][3]
a[0][1][0]
a[0][1][1]
a[0][1][2]
a[0][1][3]
a[0][2][0]
a[0][2][1]
a[0][2][2]
a[0][2][3]
a[0][3][0]
a[0][3][1]
a[0][3][2]
a[0][3][3]
a[1][0][0]
a[1][0][1]
……
a[1][3][3]
a[2][0][0]
a[2][0][1]
……
a[2][3][3]
a[3][0][0]
……
a[3][3][3]
a[4][0][0]
……
a[4][3][3]
… …

三维数组

```
int main()
{
    int a[5][3][4]={0};
    for(int i=0;i<5;i++)
    for(int j=0;j<3;j++)
    for(int k=0;k<4;k++)
        a[i][j][k]=12*i+4*j+k+1;
    for(int i=0;i<5;i++)
    {
        for(int j=0;j<3;j++)
        {
            {
                for(int k=0;k<4;k++)
                cout<<setw(3)<<a[i][j][k];
            }
            cout<<endl;
        }
        cout<<endl;
    }
```

应用：①数字统计

- 输入 20 个 0 ～ 9 之间的整数，请统计每个数在输入数列中出现的次数。

应用：①数字统计

```
for(i=0;i<10;i++)
{
    cin>>num;
    for(j=0;j<10;j++)
    {
        if(num==j)
            count[j]++;
    }
}
```

应用：①数字统计

```
#include<iostream>
using namespace std;
int main()
{
    int num,count[10]={0};
    for(int i=0;i<20;i++)
    {
        cin>>num;
        for(int j=0;j<10;j++)
        {
            if(num==j) count[j]++;
        }
    }
    for(int i=0;i<10;i++)
    {
        if(count[i]!=0)
            cout<<i<<" 输入了"<<count[i]<<" 次"<<endl;
    }
    return 0;
```

应用：①数字统计

```
for(i=1;i<=20;i++)
{
    cin>>num;
    switch(num)
    {
        case 0: count[0]++;break;
        case 1: count[1]++;break;
        case 2: count[2]++;break;
        ...
    }
}
```

应用：①数字统计

```
int main()
{
    int num,count[10]={0};
    for(int i=1;i<=20;i++)
    {
        cin>>num;
        switch(num)
        {
        case 0: count[0]++;break;
        case 1: count[1]++;break;
        case 2: count[2]++;break;
        case 3: count[3]++;break;
        case 4: count[4]++;break;
        case 5: count[5]++;break;
        case 6: count[6]++;break;
        case 7: count[7]++;break;
        case 8: count[8]++;break;
        case 9: count[9]++;break;
        }
```

应用：①数字统计

```
1 for(i=0;i<20;i++)
2 {
3     cin>>num;
4     count[num]++;
5 }
```

应用：①数字统计

```
#include<iostream>
using namespace std;
int main()
{
    int num,count[10]={0};
    for(int i=0;i<20;i++)
    {
        cin>>num;
        count[num]++;
    }
    for(int i=0;i<10;i++)
    {
        if(count[i]!=0)
            cout<<i<<" 输入了"<<count[i]<<" 次"<<endl;
    }
    return 0;
}
```

应用：②数字统计

- 某学校有 1000 位老师，分布在 20 个不同的学院中，每个学院最多有 12 个系，请编写一个程序，输入每位老师的所在院、系编号（院编号 1 ～ 20，系编号 1 ～ 12），打印出各个系老师的数量。

应用：②数字统计

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
    int teacher[21][13];
    int school,department;
    int i,j;
    char name[30];
    for(i=0;i<1000;i++)
    {
        cin>>name>>school<<department;
        teacher[school][department]++
    }
    for(i=1;i<21;i++)
        for(j=1;j<13;j++)
            cout<<setw(4)<<teacher[i][j];
    cout<<endl;
    return 0;
```

应用：③找出素数

- 请编写程序，找出 100 以内的所有素数。

应用：③找出素数

- 请编写程序，找出 100 以内的所有素数。

典型的解决办法：

- 循环（i 从 2 至 100）
  - 设置一个标识 p=0
  - 循环（j 从 2 至 i-1）
    - 如果（i mode j == 0）p=1
  - 如果 p==0 输出 i

应用：③找出素数

```
#include<iostream>
using namespace std;
int main()
{
    bool prime=true;
    for(int i=0;i<100;i++)
    {
        prime=true;
        for(int j=2;j<i;j++)
        {
            if(i%j==0)
                prime=false;
        }
        if(prime==true)
            cout<<i<<endl;
    }
    return 0;
```

应用：③找出素数

- 请编写程序，输出 100 以内的所有素数。
- 思路：
  - 将数组中 1 的倍数、2 的倍数、3 的倍数、···、100 的倍数全部划掉；
  - 那么，剩下的数都是素数。

应用：③找出素数

- 解题思路：
  - 将数组所有元素设置为 0；
  - 筛出所有合数：
    - 分别计算 2、3、4、5、···、99 自我相加多次的数值；
    - 每次计算得到的结果都是一个合数，在数组中标记该数字被“筛掉”；
    - 每次计算过程中，只要相加结果没有达到 100 就继续自我相加；
  - 根据标记输出所有没有被筛掉的数字。

应用：③找出素数

```
#include<iostream>
using namespace std;
int main()
{
    int sum=0;a[100]={0};
    for(int i=2;i<100;i++)
    {
        sum=i;
        while(sum<100)
        {
            sum=sum+i;
            if(sum<100) a[sum]=1;
        }
    }
    for(int i=2;i<100;i++)
    {
        if(a[i]==0) cout<<i<<" ";
    }
    return 0;
```

应用：③找出素数

- 筛法求素数：
  - 埃拉托斯特尼（Eratosthenes，约公元前 274~194 年）发明，又称埃拉托斯特尼筛子。
  - 基本思路：不是挑选出所有的素数，而是筛掉所有的合数。

应用：③找出素数

- 稍作优化：
  - 可以让 2、3、4、5、$\cdots$、c 中的每个数自我相加多次，来获得 100 以内的所有合数；
  - c 如何确定？
  - 根据初等数论，若 n 为合数，则 n 的最小正因数 c 满足：$1 < c \leq \sqrt{n}$

应用：③找出素数

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    int sum=0,a[100]={0};
    for(int i=2;i<sqrt(100.0);i++)
    {
        sum=i;
        if(a[sum]==0)
        {
            while(sum<100)
            {
                sum=sum+i;
                if(sum<100) a[sum]=1;
            }
         }
    }
    for(int i=2;i<100;i++)
```

数组的作用

- 不仅当你有一些数据要进行存储时：用于存放一系列数据类型相同的数据；
- 还能当你的处理对象是连续的整数时：利用数据与下标间的对应关系解决问题。

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

字符数组的定义

```
1 #include<iostream>
2 using namespace std;
3 int main()
4 {
5     char a[10]={'a','b','c','d','e','f','g','h','i','j'};
6     for(int i=0;i<10;i++)
7         cout<<a[i];
8     return 0;
9 }
```

a	b	c	d	e	f	g	h	i	j
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]

字符数组的定义

```
1 #include<iostream>
2 using namespace std;
3 int main()
4 {
5     char a[10]={'a','b','c','d','e'};
6     for(int i=0;i<10;i++)
7         cout<<a[i];
8     return 0;
9 }
```

a	b	c	d	e	\0	\0	\0	\0	\0
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]

内容提要 I

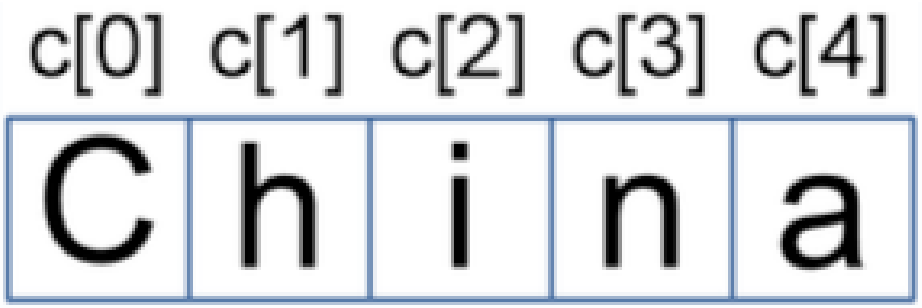
内容提要 II

内容提要 III

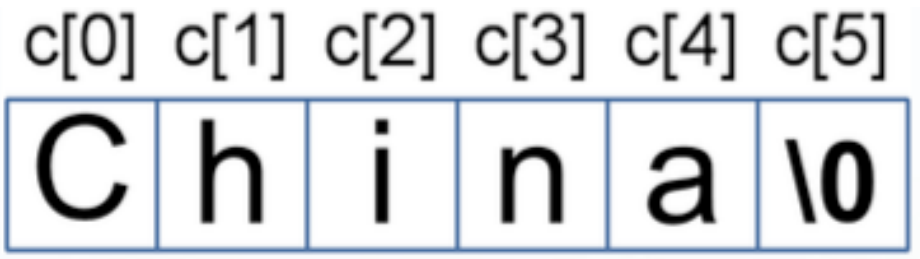
- 字符串数组
- 输入与输出
- 示例

字符数组的初始化

```
1 char c[]={'C','h','i','n','a'};
```




```
1 char c[]="China";
```



关于赋值

- 只可以
  - 在数组定义并初始化的时候赋值：`char c[6]="China";`
- 不可以
  - 用赋值语句将一个字符串常量或字符数组直接赋给另一个字符数组
  - `str1[]="China";`
  - `str1="China";`
  - `str2=str1;`

正确的赋值方式

```
#include<iostream>
using namespace std;
int main()
{
    char str1[]="C++ language",str2[20];
    int i=0;
    while(str1[i]!='\0')
    {
        str2[i]=str1[i];
        i++;
    }
    str2[i]='\0';
    cout<<"String1:"<<str1<<endl;
    cout<<"String2:"<<str2<<endl;
    return 0;
}
```

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

字符串数组

- 利用二维数组存储多个字符串

```
1 char weekday[7][11]={"Sunday","Monday","Tuesday",
2 "Wednesday","Thursday","Friday","Saturday"};
```

S	u	n	d	a	y	\0	\0	\0	\0	\0
M	o	n	d	a	y	\0	\0	\0	\0	\0
T	u	e	s	d	a	y	\0	\0	\0	\0
W	e	d	n	e	s	d	a	y	\0	\0
T	h	u	r	s	d	a	y	\0	\0	\0
F	r	i	d	a	y	\0	\0	\0	\0	\0
S	a	t	u	r	d	a	y	\0	\0	\0

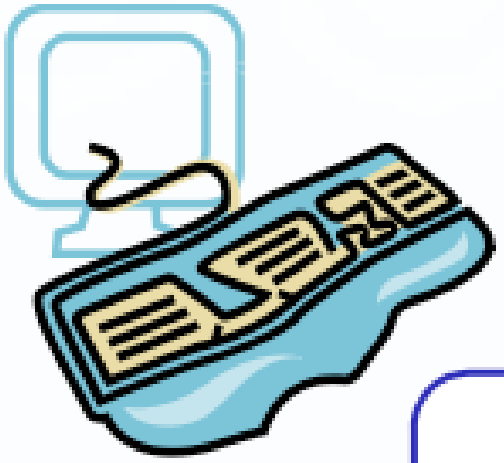
内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

输入缓冲区



How are you ?

```
1 cin>>str;
```

用 cin 输入数据

```
1 int a,b;
2 cin>>a>>b; //21 22[ENTER]
```

```
1 int a,b;
2 cin>>a>>b; //21 abc[ENTER]
```

```
1 int a,b,c;
2 cin>>a>>b>>c;//21 22[ENTER]23[ENTER]
```

用 cin 输入数据

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5      float grade;
6      cout<<"Enter grade:";
7      while(cin>>grade)
8      {
9          if(grade>=85)
10             cout<<grade<<"Good!"<<endl;
11         if(grade<60)
12             cout<<grade<<"Fail!"<<endl;
13         cout<<"Enter grade:";
14     }
15     return 0;
16 }
```

一个字符的输入 cin

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5      char c;
6      cout<<"Enter a sentence:"<<endl;
7      while(cin>>c)
8          cout<<c;
9      return 0;
10 }
```

一个字符的输入 cin.get()

cin.get()函数

- 可以用于读入一个字符
- 两种形式：无参数和一个参数
  - cin.get()
  - cin.get(char)

一个字符的输入 cin.get()

```
#include<iostream>
using namespace std;
int main()
{
    char c;
    cout<<"Enter a sentence:"<<endl;
    while((c=cin.get())!=EOF)
        cout<<c;
    return 0;
}
```

一个字符的输入 cin.get()

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5      char c;
6      cout<<"Enter a sentence:"<<endl;
7      while(cin.get(c))
8          cout<<c;
9      return 0;
10 }
```

一个字符的输入 getchar()

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5      char c;
6      cout<<"Enter a sentence:"<<endl;
7      while(c=getchar())
8          cout<<c;
9      return 0;
10 }
```

一串字符的输出 cout

```
1 #include<iostream>
2 using namespace std;
3 int main()
4 {
5     char a[10]="Computer";
6     cout<<a;
7     return 0;
8 }
```

C	o	m	p	u	t	e	r	\0	\0
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]

一串字符的输出 cout

```
1 #include<iostream>
2 using namespace std;
3 int main()
4 {
5     char a[8]={'C','o','m','p','u','t','e','r'};
6     cout<<a;
7     return 0;
8 }
```

C	o	m	p	u	t	e	r
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]

一串字符的输出 cout

```
#include<iostream>
using namespace std;
int main()
{
    char weekday[7][11]={"Sunday","Monday","Tuesday",
        "Wednesday","Thursday","Friday","Saturday"};
    for(int i=0;i<7;i++)
        cout<<weekday[i]<<endl;
    return 0;
}
```

一串字符的输出 cout

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

一串字符的输入 cin

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5      char str[10];
6      cout<<"Enter a sentence:"<<endl;
7      while(cin>>str)
8          cout<<str<<endl;
9      return 0;
10 }
```

一串字符的输入 cin.get()

带 3 个参数的get函数

```
1 cin.get(ch,10,'\n');
```

- 读取 10 − 1 个字符（包含空格），赋给指定的字符数组；
- 如果在读取 10 − 1 个字符之前，遇到指定的终止字符'\n'，则提前结束读取；
- 如果第 3 个参数没有指定，则默认为'\n'；
- 读取成功返回非 0 值（真），若失败（遇文件结束符），则返回 0 值（假）。

一串字符的输入 cin.get()

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5      char ch[20];
6      cout<<"Enter a sentence:"<<endl;
7      cin.get(ch,10,'o');
8      cout<<ch<<endl;
9      return 0;
10 }
```

一串字符的输入 cin.getline()

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5      char ch[20];
6      cout<<"Enter a sentence:"<<endl;
7      cin.getline(ch,10,'o');
8      cout<<ch<<endl;
9      return 0;
10 }
```

getline vs. get

- getline遇到终止标志字符时结束，缓冲区指针移到终止标志字符之后；
- get遇到终止标志字符时停止读取，指针不移动。

一串字符的输入 cin.getline()

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5      char weekday[7][11];
6      for(int i=0;i<7;i++)
7          cin.getline(weekday[i],11);
8      for(int i=0;i<7;i++)
9          cout<<weekday[i]<<endl;
10     return 0;
11 }
```

注意

```
#include<iostream>
using namespace std;
int main()
{
    char a[10][10];
    int n=0;
    cin>>n;
    for(int i=0;i<n;i++)
        cin.getline(a[i],10);
    for(int i=0;i<n;i++)
        cout<<a[i]<<endl;
    return 0;
}
```

注意

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5      char a[10][10];
6      int n=0;
7      cin>>n;
8      cin.get(); //Add this
9      for(int i=0;i<n;i++)
10         cin.getline(a[i],10);
11     for(int i=0;i<n;i++)
12         cout<<a[i]<<endl;
13     return 0;
14 }
```

内容提要 I

内容提要 II

内容提要 III

- 字符串数组
- 输入与输出
- 示例

示例：①字符串加密

- 输入一个字符串，把每个字符变成它后续字符，如果是'Z' 或者'z'，则变成'A' 或'a'；空格不变。然后将变换后的字符串输出。
- 要求能够接受连续输入。

示例：①字符串加密

思路：

- 读入字符串（想一想以什么方式输入？）
- 从字符头到尾循环：
  - 是'Z'则直接赋值'A'，跳过以下步骤；
  - 是'z'则直接赋值'a'，跳过以下步骤；
  - 空格不做处理，跳过以下步骤；
  - 其他字符 ++
- 输出新字符串。

示例：①字符串加密

```
1  int main()
2  {
3      char str[200];
4      while(cin.getline(str,200))
5      {
6          for(int i=0;str[i]!='\0';i++)
7          {
8              if(str[i]=='Z') {
9                  str[i]='A';continue;}
10             if(str[i]=='z') {
11                 str[i]='a';continue;}
12             if(str[i]==' ') {
13                 continue;}
14             str[i]++;
15         }
16         cout<<str<<endl;
17     }
18     return 0;
}
```

示例：②字符串连接

问题：

- 输入两个字符串，将其中较短的串接到较长的串后面。

要求：

- 不使用系统函数strcat；
- 每个输入串的长度不超过 20。

字符数组常用操作

```
1  #include<iostream>
2  #include<string>
3  using namespace std;
4  int main()
5  {
6      char str1[20],str2[20];
7      cin.getline(str1,20);
8      strcpy(str2,str1);
9      cout<<str1<<endl;
10     cout<<str2<<endl;
11     return 0;
12 }
```

示例：②字符串连接

定义：

```
1 char str1[40],str2[40];
2 cin.getline(str1,20);cin.getline(str2,20);
```

计算长度：

```
1 for(len1=0;str1[len1]!='\0';len1++);
2 for(len2=0;str2[len2]!='\0';len2++);
```

拼接：

- 第一个串的下标指向最后一个元素之后
- 第二个串的下标指向第一个元素

```
1 for(len2=0;str2[len2]!='\0';len2++)
2     str1[len1++]=str2[len2];
3 str1[len1]='\0'; //string
```

```
int main()
{
    int len1,len2; char str1[40],str2[40];
    cin.getline(str1,20);cin.getline(str2,20);
    for(len1=0;str1[len1]!='\0';len1++);
    for(len2=0;str2[len2]!='\0';len2++);
    if(len1>=len2)
    {
        for(len2=0;str2[len2]!='\0';len2++)
            str1[len1++]=str2[len2];
        str1[len1]='\0';
    }
    else
    {
        for(len1=0;str1[len1]!='\0';len1++)
            str2[len2++]=str1[len1];
        str2[len2]='\0';
    }
    cout<<str1<<endl; cout<<str2<<endl;
    return 0;
}
```

示例：③统计单词数

- 输入一个英文句子（不超过 80 个字母），统计其中有多少个单词，单词之间用空格分开。

示例：③统计单词数

- 输入一个英文句子（不超过 80 个字母），统计其中有多少个单词，单词之间用空格分开。

> Everything should be made as simple as possible, but not simpler.
>
> – Albert Einstein

示例：③统计单词数

```
#include<iostream>
using namespace std;
int main()
{
    char str[80];
    int num=0,flag=0;
    cin.getline(str,80);
    for(int i=0;str[i]!='\0';i++)
    {
        if(str[i]==' ') flag=0;
        else if(flag==0) {
            flag=1;num++;}
    }
    cout<<"We have "<<num<<" words."<<endl;
    return 0;
}
```