计算机科学基础

——程序设计与算法语言

第一章 C++基础知识

戚隆宁

Email: longn_qi@seu.edu.cn

Tel: 13813839703

本章目录

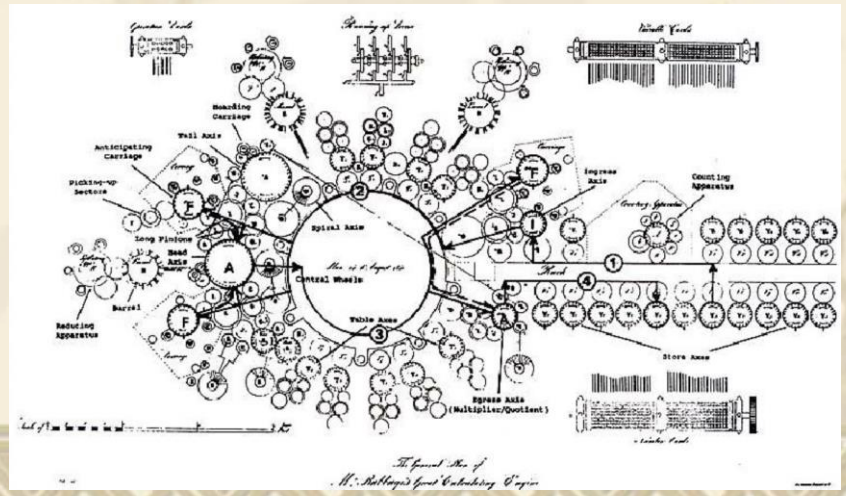
本章目录

计算机语言概述（1/6）
——发展历史

- 最早的程序（Program）设想
  - “算法”的提出
  - “流程”的描述

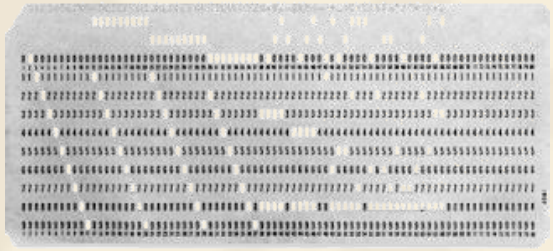
阿达·奥古斯塔（英国）
世界上第一位软件工程师
1842年，在巴贝奇分析机上
解伯努利方程

“分析机谈不上能创造什么东西。但它能做我们命令它做的任何工作。它能进行分析但不能预知任何分析关系或真理。它的本份是帮助我们去实现我们已知的事 情。”——Ada



计算机语言概述（2/6）
——发展历史

- 诞生：机器语言
  - 二进制指令
  - 早期以接线、穿孔卡片或穿孔纸带形式存储
  - 完全依赖于硬件



1935年
IBM穿孔卡片机（美）

计算机语言概述（3/6）
——发展历史

- 汇编语言（Assembly Language）
  - 助记符号
  - 与硬件关系密切
  - 早期人工汇编

```
MONITOR FOR 6802 1.4          9-14-80  TSC ASSEMBLER  PAGE    2

C000                    ORG     ROM+$0000 BEGIN MONITOR
C000 8E 00 70  START    LDS     #STACK

                ****************************************
                * FUNCTION: INITA - Initialize ACIA
                * INPUT: none
                * OUTPUT: none
                * CALLS: none
                * DESTROYS: acc A

0013            RESETA  EQU     %00010011
0011            CTLREG  EQU     %00010001

C003 86 13      INITA   LDA A   #RESETA   RESET ACIA
C005 B7 80 04           STA A   ACIA
C008 86 11              LDA A   #CTLREG   SET 8 BITS AND 2 STOP
C00A B7 80 04           STA A   ACIA

C00D 7E C0 F1           JMP     SIGNON    GO TO START OF MONITOR
```

1949年 EDSAC汇编程序

```
MOV   AL, 7
ADD   AL, 10
HLT
```

计算机语言概述（4/6）
——发展历史

- 高级语言
  - 符号语法更利于阅读和设计，更加规范
  - 需要强大的语言转换工具

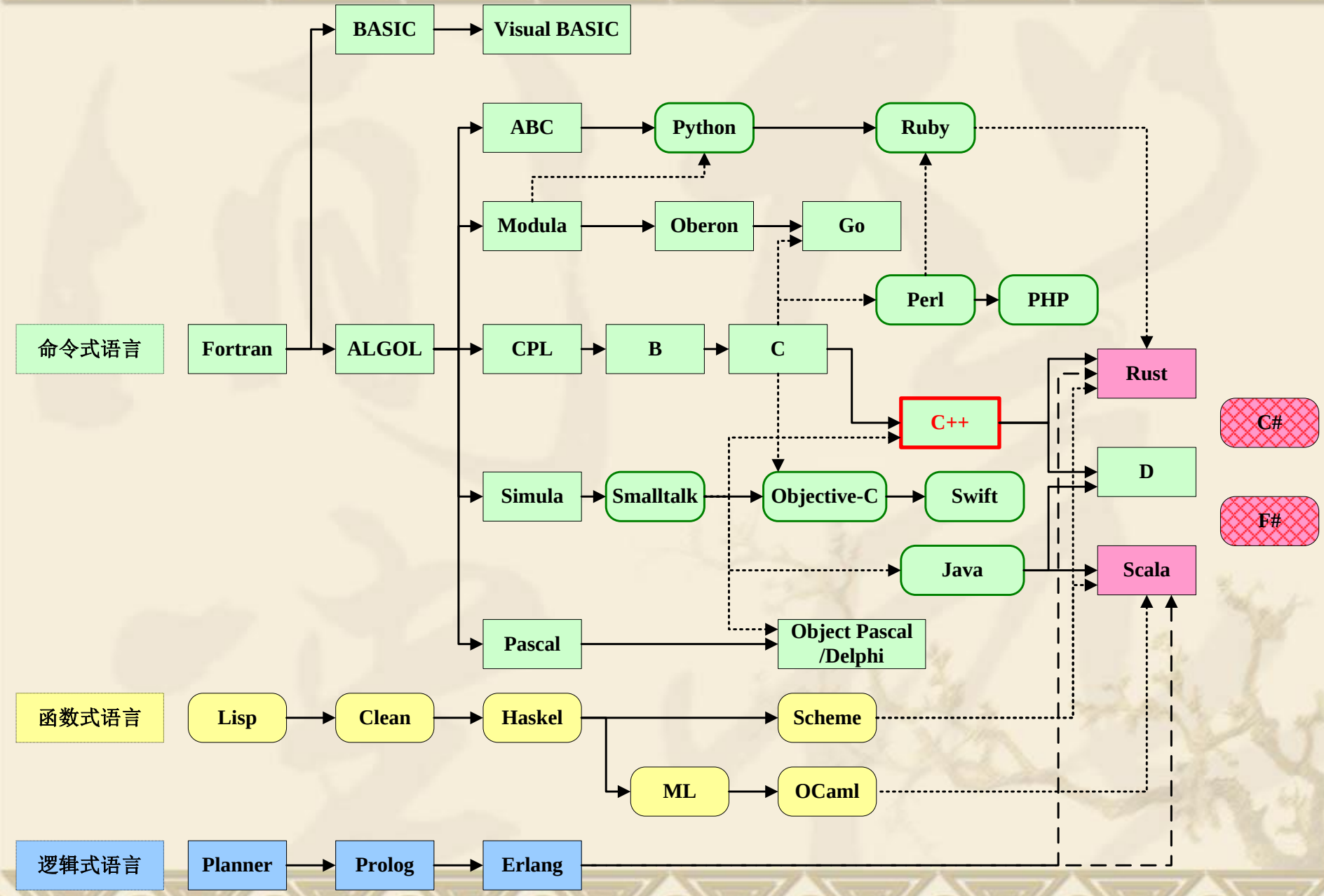
约翰·巴克斯（美国）
第一个高级语言FORTRAN的设计者

葛丽丝·霍普（美国）
第一个编译器和高级语言COBOL的设计者

1957 FORTRAN
1960 COBOL
1960 LISP
1965 BASIC
1971 PASCAL
1972 Prolog
1973 C
1975 Scheme
1975 SmallTalk
1983 C++
1986 Objective-C
1995 Java
2000 C#
2009 Go

计算机语言概述（5/6）
——发展历史

- 编程范式（Program Paradigm）
  - 命令式语言（Imperative Language）：程序侧重描述过程，怎么做，按照既定步骤执行指令序列。
  - 声明式语言（Declarative Language）：程序侧重描述问题，做什么。
    - 函数式语言（Functional Language）：类似数学计算，程序只依赖于输入的参数，只要参数相同，结果必然相同。
    - 逻辑式语言（Logical Language）：类似数学证明，基于陈述的事实和制定的规则，推断出新的事实。

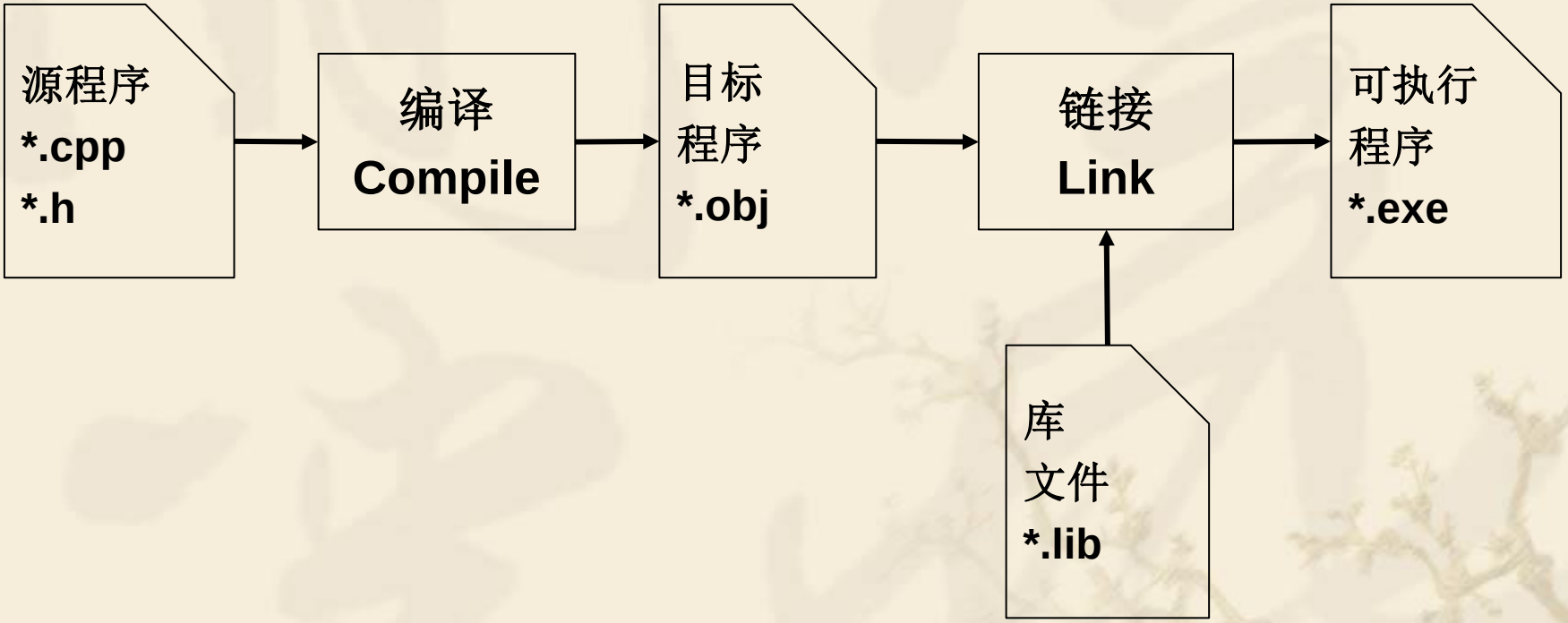

BASIC
Visual BASIC
ABC
Python
Ruby
Modula
Oberon
Go
Perl
PHP
命令式语言
Fortran
ALGOL
CPL
B
C
Rust
C++
C#
D
Simula
Smalltalk
Objective-C
Swift
F#
Java
Scala
Pascal
Object Pascal
/Delphi
函数式语言
Lisp
Clean
Haskel
Scheme
ML
OCaml
逻辑式语言
Planner
Prolog
Erlang

计算机语言概述（6/6）
——语言的处理模式

- 编译模式（Compile）
  - 方式：源程序（Source Code）在运行前一次性转换为可执行程序
  - 工具：编译器（Compiler）/汇编器（Assembler），链接器（Linker）
- 解释模式（Translate）
  - 方式：源程序在运行时逐条语句翻译为可执行指令
  - 工具：翻译器（Translator）
- 动态编译模式（Dynamic Compile）
  - 方式：源程序在运行时根据策略逐段转换为可执行指令
  - 工具：动态编译器（Dynamic Compiler）

编译模式

C++
Windows平台



本章目录

C++语言概述（1/3）

- 发展历史
  - 1983年诞生，改进C语言，增加了类
  - 1994年ANSI标准化
  - 1998年ISO标准化（ISO14882），C++98
  - 每隔几年修订一版，C++03/C++11/C++14/C++17
  - 最新标准，C++20

Bjarne Stroustrup（丹麦）
C++之父

C++语言概述（2/3）

- 特点
  - 语法部分兼容C语言（C99），编译执行
  - 多重范式（Multi-paradigm）编程语言，支持多种设计方法
    - 过程型编程（Procedural Programming）
    - 面向对象编程（Object-Oriented Programming）
    - 泛型编程（Generic programming）
    - 元编程（Meta-programing）
    - 函数式编程（Functional programing）

C++语言概述（3/3）

- 开发工具（x86 Windows平台）
  - Microsoft Visual C++ 6.0~9.0/Visual Studio 2003~2020
  - Intel C++ 9.0~19.0
  - GNU GCC 4.0~10.2
  - LLVM CLANG 1.0~11.0
  - Bloodshed Dev C++ 4.0/5.11
  - Borland C++ Builder 5.0/6.0

本章目录

本章目录

参考资料：
1）微软MSDN帮助文档（C++ 语言参考）
https://msdn.microsoft.com/zh-cn/library/3bstk3k5.aspx
2）C++ 语言参考
https://zh.cppreference.com/w/cpp

C++源程序组成

- 语句（statement）
  - 以分号（;）表示结束，内容可以为空
    - 表达式语句，控制语句，声明语句
    - 复合语句，即以大括号（{}）组合的一组语句
- 注释（comment）
  - 以双斜杠（//）表示注释行开始
  - 以（/*）和（*/）表示注释块的起始
  - 不可嵌套
- 编译预处理（preprocessing）
  - 以（#）表示编译预处理指令行开始

字符集（character set）

英文字母	a～z　　A～Z
数字字符	0～9
运算字符	+ − * / % = ~ & ^ ! <> [] () , . ? :
特殊字符	# _ \ ’ ” ; {}
空白字符	空格　制表　换行回车

没有
@和$

词法单位

- 标点符号
- 关键字（keyword）
- 字面符（literal）
- 标识符（identifier）
- 运算符（operator）

词法单位（1/5）
——标点符号

分隔符号	空格（Space）、换行回车（Enter）、制表（Tab）等
语句符号	, ; : () {}…
注释符号	// /* */
预处理符号	#

词法单位（2/5）
——关键字keyword

- 系统保留，不可它用，区分大小写

控制符	if else switch case default do while for break goto contiue return …
说明符与修饰符	char int short long float double signed unsigned extern static register volatile const public private protected class struct …
运算符与逻辑值	new delete sizeof true false …

词法单位（2/5）
——关键字：基本数据类型（1/4）

- 基本类型（Basic Type）
  - 布尔型：bool
  - 字符型：char
  - 整型：int
  - 实型（浮点型）：float、double
  - 无值型（未定型）：void
  - 隐式推断型（自动型）：auto （C++11）

词法单位（2/5）
——关键字：基本数据类型（2/4）

类型	名称	存储字节	取值范围
bool	布尔型	1	true、false
signed char	有符号字符型	1	-128~127
unsigned char	无符号字符型	1	0~255
char	字符型	1	由编译器确定

词法单位（2/5）
——关键字：基本数据类型（3/4）

类型	名称	存储字节	取值范围
(signed) short (int)	短整型	2	-2^{15}~$2^{15}-1$
unsigned short (int)	无符号短整型	2	0~$2^{16}-1$
(signed) int	整型	4*	-2^{31}~$2^{31}-1$
unsigned int	无符号整型	4*	0~$2^{32}-1$
(signed) long (int)	长整型	4*	-2^{31}~$2^{31}-1$
unsigned long (int)	无符号长整型	4*	0~$2^{32}-1$

*取决于不同编译器定义，long >= int >= short

词法单位（2/5）
——关键字：基本数据类型（4/4）

类型	名称	存储字节	取值范围
float	单精度浮点型	4	-2^{128}～2^{128} 精度23 bit
double	双精度浮点型	8	-2^{1024}～2^{1024} 精度52 bit
long double	长双精度浮点型	8*	-2^{1024}～2^{1024} 精度52 bit

IEEE 754规范化格式：数符（1b）+阶码（移码）+尾数（原码）

*取决于不同编译器定义8/10/12/16，精度高于double

词法单位（3/5）
——字面符（1/4）

- 字面符/文字常量（Literal）：由程序员自定义的固定不变的直接数据
  - 数值常量
  - 字符常量
  - 字符串常量
- 特征
  - 预先确定
  - 固化在程序中
  - 只读数据

词法单位（3/5）
——字面符（2/4）：数值常量

- 整型常量（默认int类型）
  - 123，-123，1E2，-12e3
- 实型常量（默认double类型）
  - 1.2，.23，-1.，1.234E2，-.1e-3
- 进制前缀
  - 十六进制：0X或0x开头，0xBAD
  - 八进制：0开头，0123
- 范围、精度后缀
  - U、L、UL、F（不区分大小写）

词法单位（3/5）
——字面符（3/4）：字符常量

- 单引号（ ‘’ ）标注，表示单个ASCII字符，char类型
  - 绝大部分可显示字符（20H~7EH）
    - ‘A’、’ 0’、’ +’、’ @’、’ ’
    - 除了单引号（ ‘’ ）、双引号（“”）和反斜杠（\）
  - 其他字符用（\）转义表示
    - ’ \n’、’ \r’、’ \t’、 ‘\”、’ \””、 ‘\\’ 、‘\0’、’ \x1B’

词法单位（3/5）
——字面符（4/4）：字符串常量

- 双引号（” ”）标注，表示连续字符
  - 以NUL空字符（ ‘\0’）表示字符串结束
  - 中英文都可以
    - “中国ok”
  - 支持转义字符
    - “123\nabc”

”Hello”

‘H’	‘e’	‘l’	‘l’	‘o’	‘\0’

词法单位（4/5）
——标识符identifier（1/3）

- 由程序员自定义的变量（variable）、类型（type）、成员（member）和宏（micro）的名称
- 命名规则
  - 由英文字母、数字和下划线组成
  - 至少包含一个英文字母
  - 首字母必须是英文字母或下划线
  - 英文字母区分大小写
  - 唯一性

词法单位（4/5）
——标识符（2/3）：变量（1/5）

- 变量（variable）：由程序员自定义的可变的数据
  - 有唯一的标识符，即变量名
  - 有数据类型说明，即变量类型
  - 存储在数据区，可读写，且有生命期
  - 使用前必须完成上述说明，即变量定义（variable definition），且只能定义一次
  - 一般需要先赋予初值再使用，即变量初始化（variable initialization）

词法单位（4/5）

——变量（2/5）：定义语法

- 语法格式

  [存储说明] 变量类型 变量名1[, 变量名2, 变量名n];

  例如：

  ```
  int a, b, c;
  double x, y;
  ```

词法单位（4/5）

——变量（3/5）：初始化语法

- 语法格式

  [存储说明] 变量类型 变量名 = 变量初值;

  例如：

  ```
  int a = 1;
  double x = 2.1, y = 3.3;
  ```

词法单位（4/5）
——变量（4/5）：常变量

- 常变量（constant variable）：用const修饰的变量
  - 变量的数据内容不可修改
  - 必须在定义时初始化

例如：

```
const int a = 10;
const double x = 1.0;
```

词法单位（4/5）
——变量（5/5）：命名风格

- 基本原则
  - 易读易懂，简洁而意义明确
  - 能够有效区分类型、存储等特性
- 常见风格
  - 匈牙利命名法：变量名=属性+类型+单词描述（单词首字母大写）
    - 例：**bool bTestFinished**; **const int c_nLength** = 0;
  - **K&R命名法：**下划线分割单词（全小写）
    - 例：**bool test_finished**;

词法单位（4/5）
——标识符（3/3）：自定义类型

- 复合类型（Compound Type）
  - 数组（***type* []**）、指针（***type* ***）、引用函数（***type* &**）、结构体（**struct**）、联合体（**union**）、枚举（**enum**）、类（**class**）
- 类型别名（**typedef**）
  - 例： **typedef unsigned char** BYTE;

词法单位（5/5）
——运算符（1/6）

- 运算符（operator）：对操作数（operand）进行运算处理的符号
- 运算符和操作数构成表达式（expression）
- 操作数可以是
  - 变量和文字常量
  - 类型名和类型成员名
  - 表达式

词法单位（5/5）
——运算符（2/6）：分类（1/2）

- 按运算性质分类
  - 算术运算符：+、-、*、/、%
  - 关系运算符：>、<、==、!=、<=、>=
  - 位运算符：&、|、^、~、<<、>>
  - 逻辑运算符：&&、||、！
  - 赋值运算符：=、+=、-=、*=、/=、%=、>>=、<<=
  - 其他
    - 顺序运算符（,）、括号运算符()、自增/自减运算符（++、--）
    - 下标运算符[ ]、地址运算符（*、&）
    - 成员运算符（.、->）、域运算符（::）
    - 动态变量运算符（new、delete）
    - 类型运算符（sizeof、typeid）等

词法单位（5/5）
——运算符（2/6）：分类（2/2）

- 按操作数分类
  - 单目运算符
    - 符号算术运算（+、-）、~、！、++、--、()、类型运算（sizeof、typeid）、地址运算（*、&）、new、delete等
  - 双目运算符
    - 大部分运算符
  - 三目运算符
    - 条件运算符（？：）

词法单位（5/5）
——运算符（3/6）：操作数要求与影响（1/2）

- 对操作数的要求
  - void类型操作数不能进行求值运算（编译错误），可以参与类型运算
  - /、/=、%、%=的右操作数不能是0 （3级编译警告和运行时错误）
  - %、%=和位运算的操作数不能是浮点型（编译错误）
  - <<、>>、<<=、>>=的右操作数≥ 0且<32（1级编译警告，计算结果未定义）
  - ++ 、-- 、=和复合赋值运算的左操作数必须是左值（编译错误）
  - 逻辑运算的操作数要求是bool型（无警告和错误，自动转换）

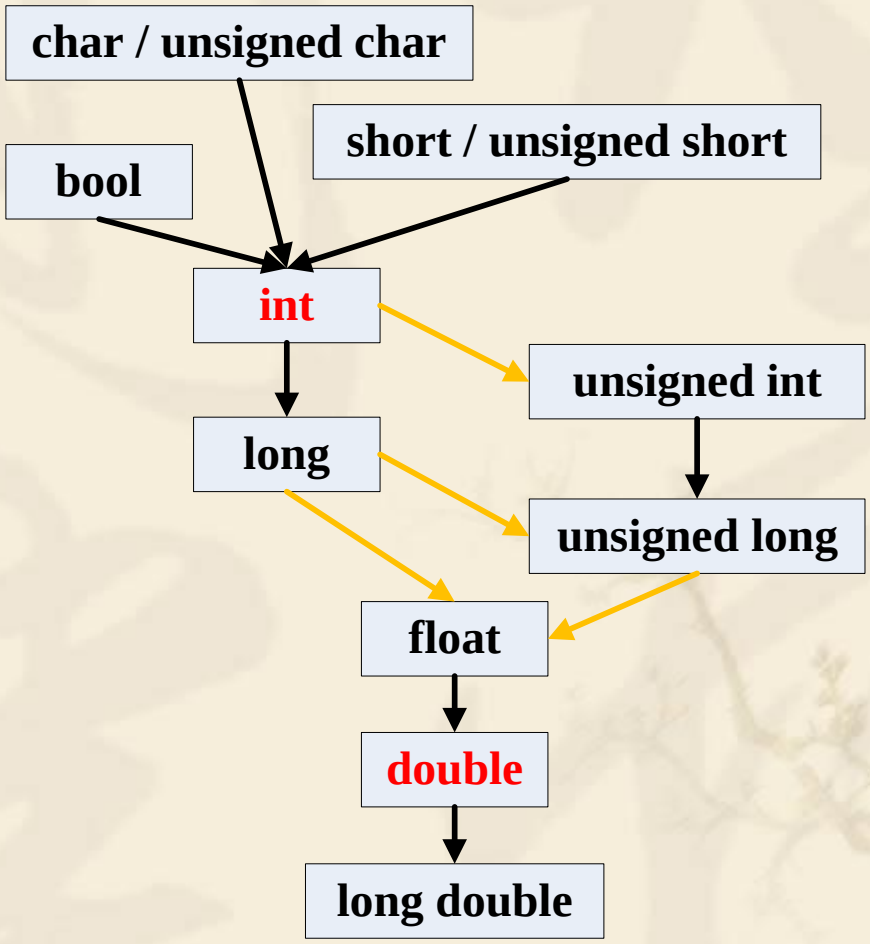
词法单位（5/5）
——运算符（3/6）：操作数要求与影响（2/2）

- 操作数的影响
  - /：整型（整除）和浮点型（小数除法）
  - %：余数正负符号与被除数一致
  - >>：有符号整数带符号右移，无符号整数高位补0

词法单位（5/5）
——运算符（4/6）：类型不匹配问题

- 数据类型不匹配问题（Type Mismatch）
  - 自动转换/隐式转换（implicit conversion）
    - 算术转换/类型提升：一般提升为范围更大、精度更高的数据类型（见提升规则）
    - bool值转换：逻辑运算的操作数、条件表达式的结果自动转为bool值（等价于x!=0）
    - 赋值转换：转换为赋值目标类型
      - 可能的问题：丢失精度、溢出（整数截断、浮点数无穷1.#INF）
  - 强制转换/显式转换（emplicit conversion）
    - 强制类型转换运算符（C风格）：(类型)表达式
    - static_cast、const_cast、reinterpret_cast和dynamic_cast：xxx_cast<类型>(表达式)

数据类型的提升规则


char / unsigned char
short / unsigned short
bool
int
unsigned int
long
unsigned long
float
double
long double

词法单位（5/5）
——运算符（5/6）：优先级与结合性

- 优先级高的先运算
  - 单目>双目（例外：域运算、成员运算、下标运算和类型转换运算）
  - 双目>三目（例外：赋值运算和顺序运算）
  - 单目：括号运算、后缀运算和类型运算>其他
  - 双目：域运算>成员运算>下标运算>类型转换运算 >算术运算>位运算和关系运算>逻辑运算>赋值运算>顺序运算
- 同优先级按结合性方向运算
  - 自左向右结合
    - 大部分运算
  - 自右向左结合
    - 单目运算（例外：右括号运算、后缀运算和类型运算）
    - 三目运算
    - 赋值运算

词法单位（5/5）
——运算符（6/6）：计算副作用问题

- 计算副作用问题（side effect）：不同编译器求值顺序的不同导致运算结果不同（增量和赋值）
  - 计算顺序点（sequence point）：在该点之前的所有副作用已经结束，并且后续的副作用还没发生。
    - 完整表达式末尾、（？：， && ||）运算的第一个操作数、完整声明末尾、函数调用和返回

本章目录

输入和输出

- 控制台（console）的输入和输出（字符形式）
  - 流方式（C++）
  - 标准库方式（C）

输入和输出
——流方式（1/4）

- 输出流：**cout**，将数据输出到控制台显示
  - 流插入运算符： <<

  例：cout << “a=” << a << ‘ ‘ << “a+b=” << (a+b) << endl;
- 输入流：**cin**，将从控制台获得数据输入
  - 流提取运算符：>>

  例：cin >> a >> b;
- 需要包含必要的头文件或库名

  (1) #include <iostream.h>

  (2) #include <iostream>

  using namespace std;

输入和输出
——流方式（2/4）：基本格式控制

- 整数进制控制（当前程序有效）
  - 输出进制控制
    - hex（十六进制）、oct（八进制）、dec（十进制，默认）
    - 进制前缀：showbase/noshowbase（默认）
    - 八进制和十六进制用补码显示负数

    例：cout << oct << a << endl;

    例：cout << showbase << hex << a << endl;
  - 输入进制控制
    - hex（十六进制）、oct（八进制）、dec（十进制，默认）
    - 不必输入进制前缀
    - 可输入负数

    例：cin >> oct >> b;

输入和输出
——流方式（3/4）：基本格式控制

- 浮点数格式控制（当前程序有效，默认6位精度）

格式算子	显示格式	
	$10^{-4} \le$ 数值$< 10^{6}$ 例：123，1.23	数值$<10^{-4}$ 或$\ge 10^{6}$ 例：12345678，0.0000123
noshowpoint (默认)	小数形式，最高6位有效数字 123, 1.23	科学计数法，最高6位有效数字 1.23457e+007, 1.23e-005
showpoint	小数形式，固定6位有效数字 123.000, 1.23000	科学计数法，固定6位有效数字 1.23457e+007, 1.23000e-005
fixed	小数形式，小数点后固定6位数字 123.000000, 1.230000, 12345678.000000, 0.000012	
scientific	科学计数法，小数点后固定6位数字 1.230000e+002, 1.230000e+000, 1.234568e+007, 1.230000e-005	

例：cout << scientific << a << b << endl;

输入和输出
——流方式（4/4）：基本格式控制

- 数据显示宽度/间隔控制：仅对后一个数有效
  - setw()

  例：cout << setw(8) << a << setw(4) << b << endl;
- 数据精度控制：当前程序有效
  - setprecision()

  例：cout << setprecision(6) << a << b << endl;
- 需要包含

  (1) #include <iomanip.h>

  (2) #include <iomanip>

  using namespace std;

本章要点

- 单文件程序基本框架和编译模式流程：主函数main()
- 类型及修饰关键字：存储大小和格式、表示范围和精度
- 常量定义规则
- 变量的命名规则、定义规则和初始化规则
- 运算符的分类和使用规则（包括运算优先级规则、类型提升和转换规则）：计算表达式的值，运用表达式表示一般数学公式（四则混合、逻辑、幂乘）
- 基本输入和输出方式：表达式输出和变量输入、简单格式控制