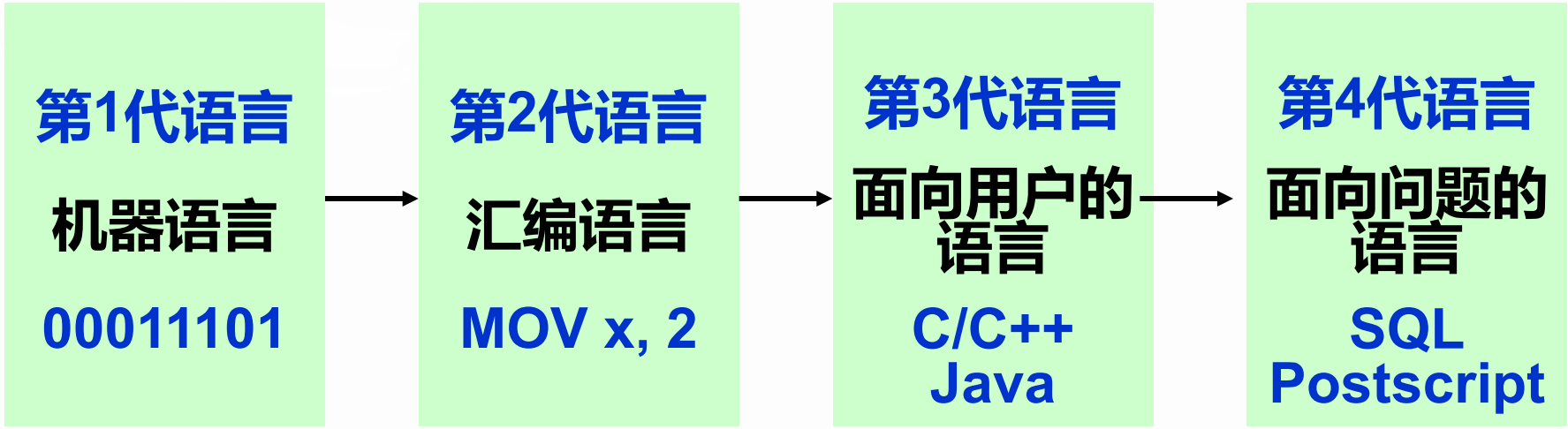
程序设计部分 前言

关于 C/C++ 程序设计语言

北京大学 信息科学技术学院

Courtesy 李戈《计算概论》

程序设计语言的发展



低级语言

高级语言

Courtesy 梁云《编译技术》

低级语言 vs. 高级语言

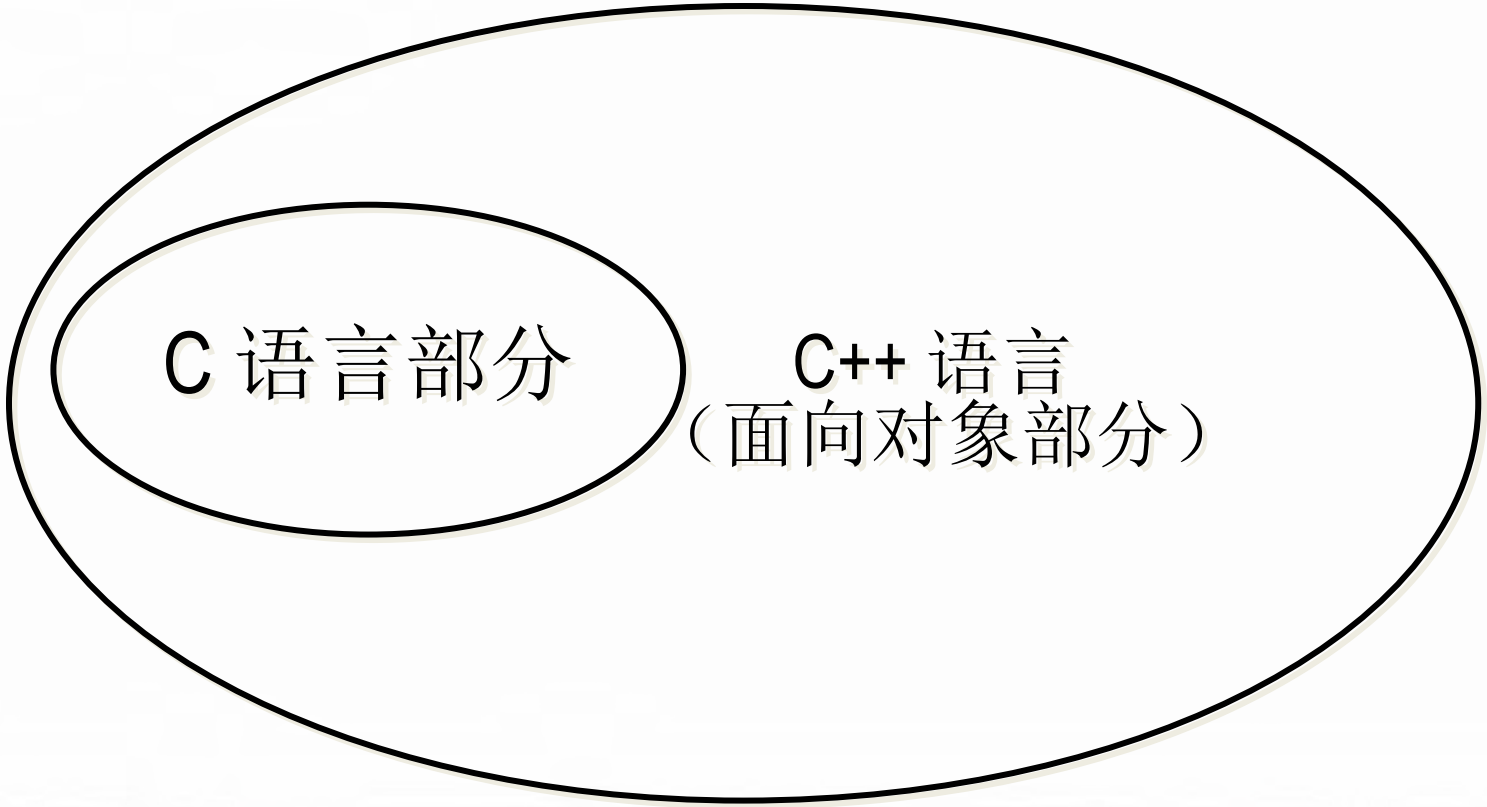
- 低级语言（Low level language)
  - 字位码、机器语言、汇编语言
  - 特点：与特定的机器有关，功效高，但使用复杂、繁琐、费时、易出错
- 高级语言（High level language）
  - Fortran、Pascal、C/C++、Java 语言等
  - 特点：不依赖具体机器，移植性好、对用户要求低、易使用、易维护等

用高级语言编写的程序，计算机不能立即执行，必须通过一个“翻译程序”进行加工，转化为与其等价的机器语言程序，机器才能执行。

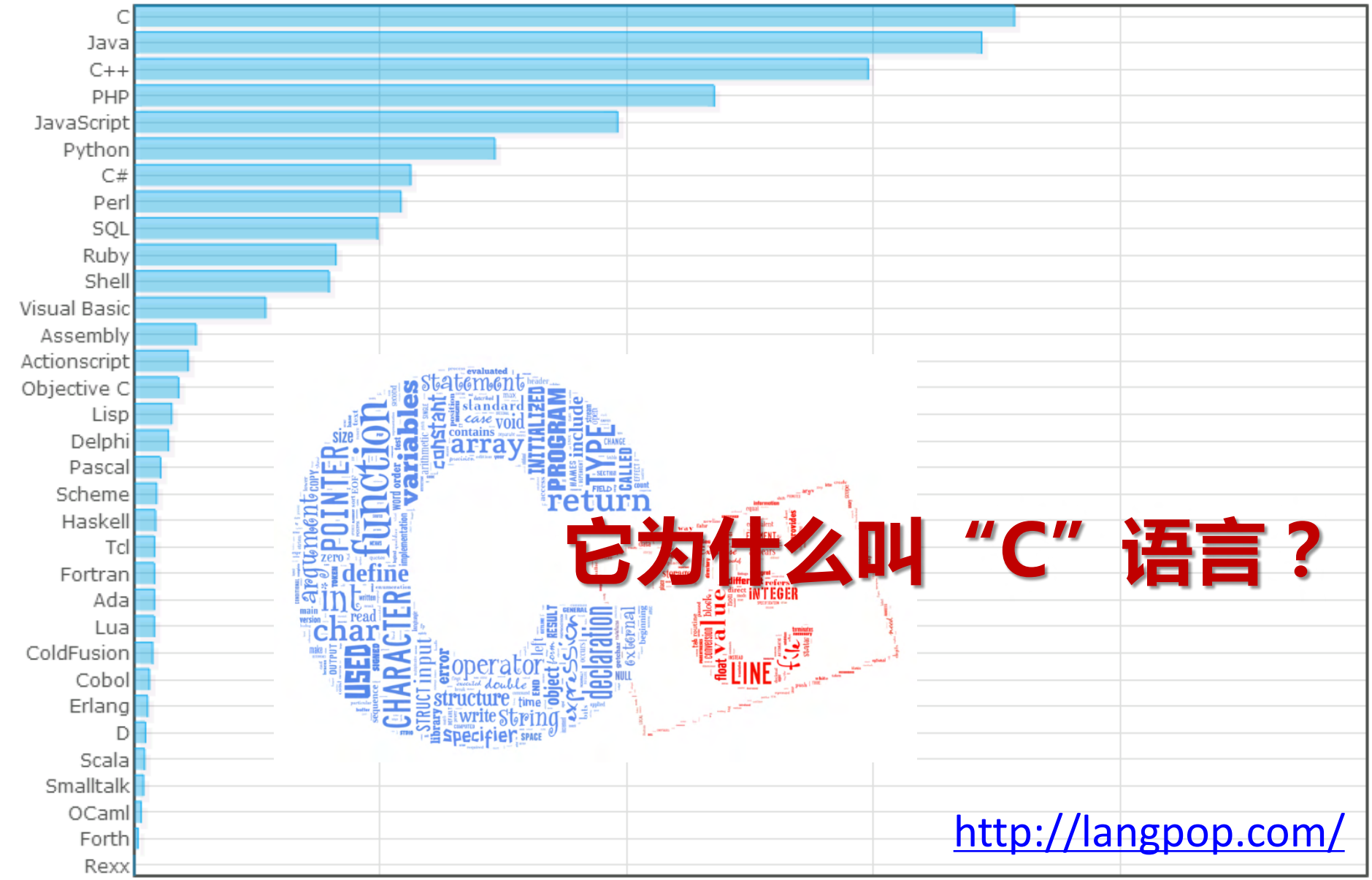
这种翻译程序，被称为“编译程序”。

Co

我们学的是什么高级语言?



C 程序设计语言



它为什么叫“C”语言？

http://langpop.com/

程序设计语言的分类

- 低级语言 之 机器语言

```
0000000100000001000 数据装入寄存器0
0000000100010001010 数据装入寄存器1
0000010100000000001 寄存器0与1的数据乘
0000000100010001100 数据装入寄存器1
0000010000000000001 寄存器0与1的数据加
0000001000000001110 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

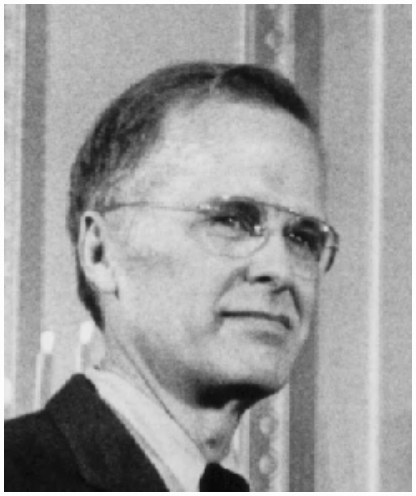
```
0000000100000001000     load  0 a
0000000100010001010     load  1 b
0000010100000000001     mult  0 1
0000000100010001100     load  1 c
0000010000000000001     add   0 1
0000001000000001110     save  0 d
```

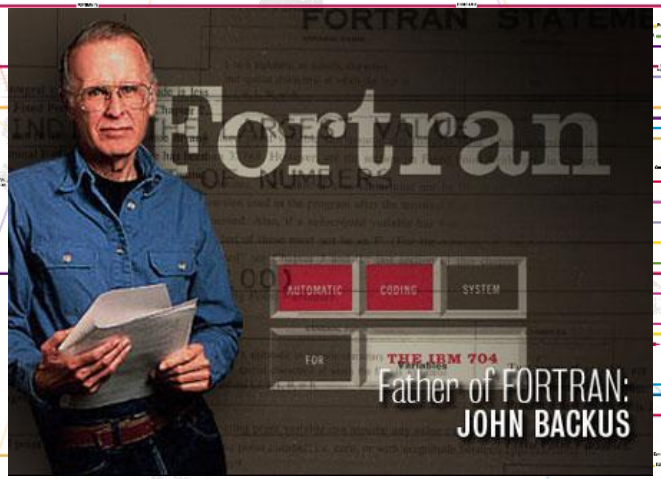
- C语言写同样的程序：

```
d = a * b + c;
```

高级程序设计语言

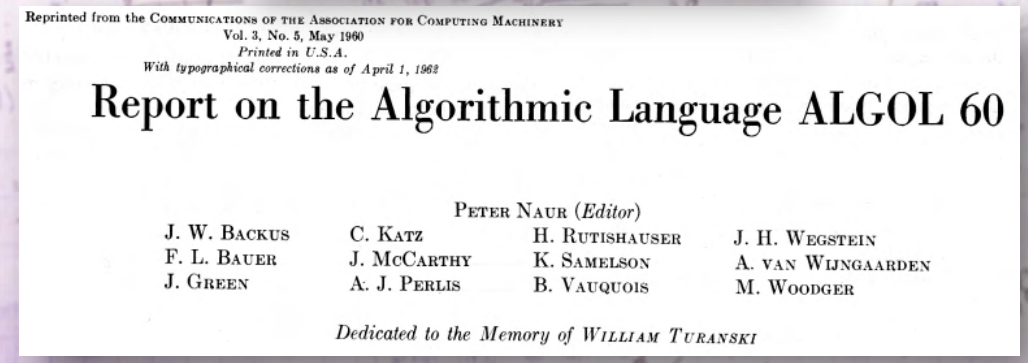
1954 -1956年 IBM 的John Backus 和他的研究小组研发了FORTRAN(FORmula TRANslation)





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

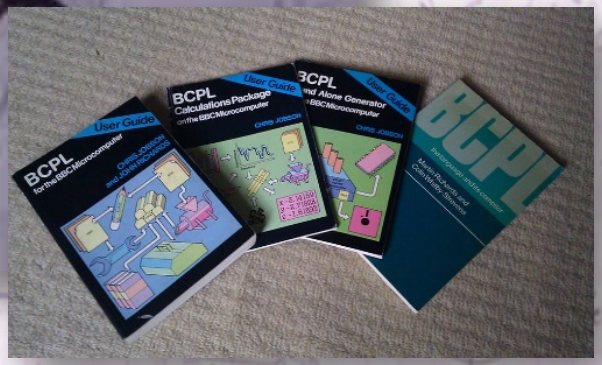
PETER NAUR (*Editor*)

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

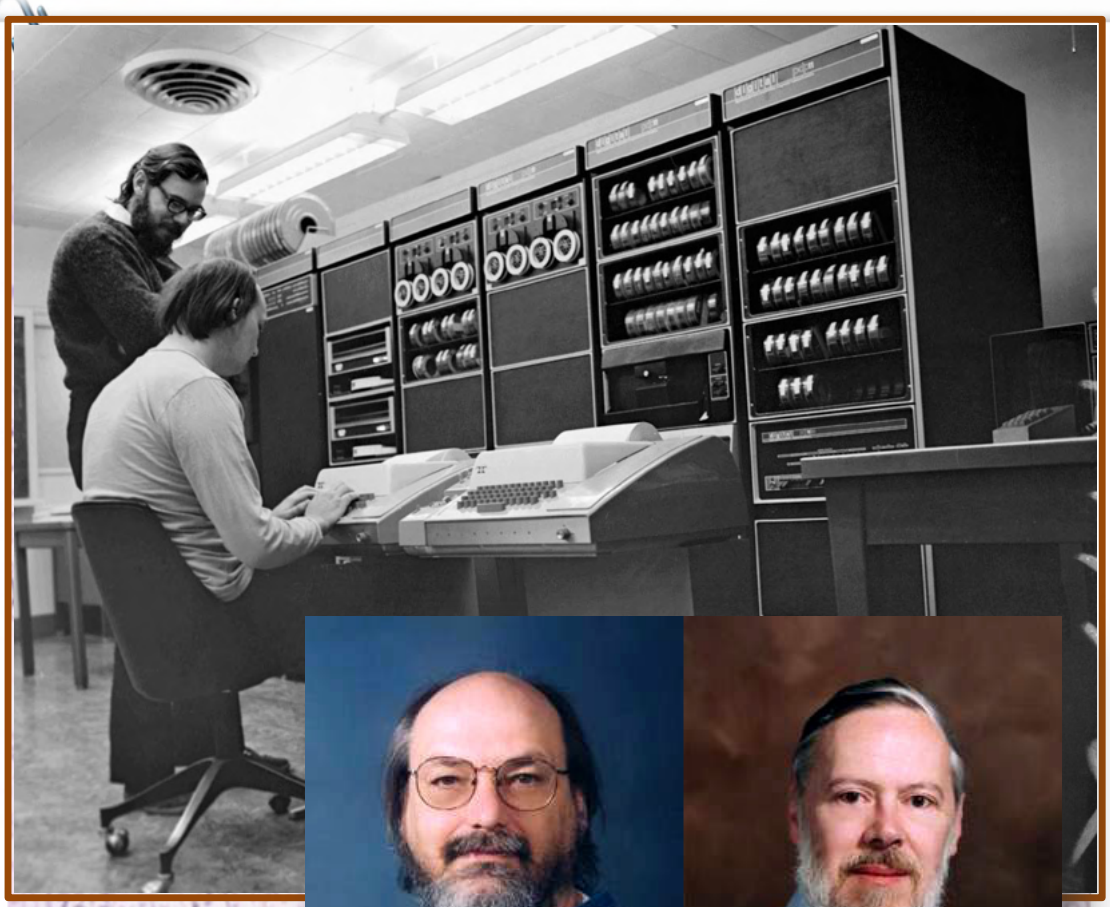
- 1960年1月
  - Alan J. Perlis
  - 软件专家讨论会（巴黎）
  - 发表“算法语言Algol 60报告”
  - 宣告了程序设计语言Algol 60的诞生
  - 计算科学里程碑
  - A 语言

C 程序设计语言的历史



- BCPL语言
- 1963年，剑桥大学在ALGOL 60的基础上推出了CPL（Combined Programming Language）语言，但规模比较大，难以实现。
- 1967年，剑桥大学的Martin 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



- B语言
- 贝尔实验室的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语言的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

Thompson (left) and Ritchie (center) receiving the National Medal of Technology from President Clinton in 1999

http://v.youku.com/v_show/id_XMjMyMTYwODc2.html

C 程序设计语言的版本

- K&R C
  - 在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，就是ISO9899:1999（1999年出版）。这个版本就是通常提及的C99。它被ANSI于2000年三月采用。
- 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.htm?csnumber=57853

需要特别说明的情况

- **C语言规范定义得非常宽泛：**
  - 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....

C程序设计语言的历史

Bjarne Stroupstrup
本贾尼•斯特劳斯特卢普

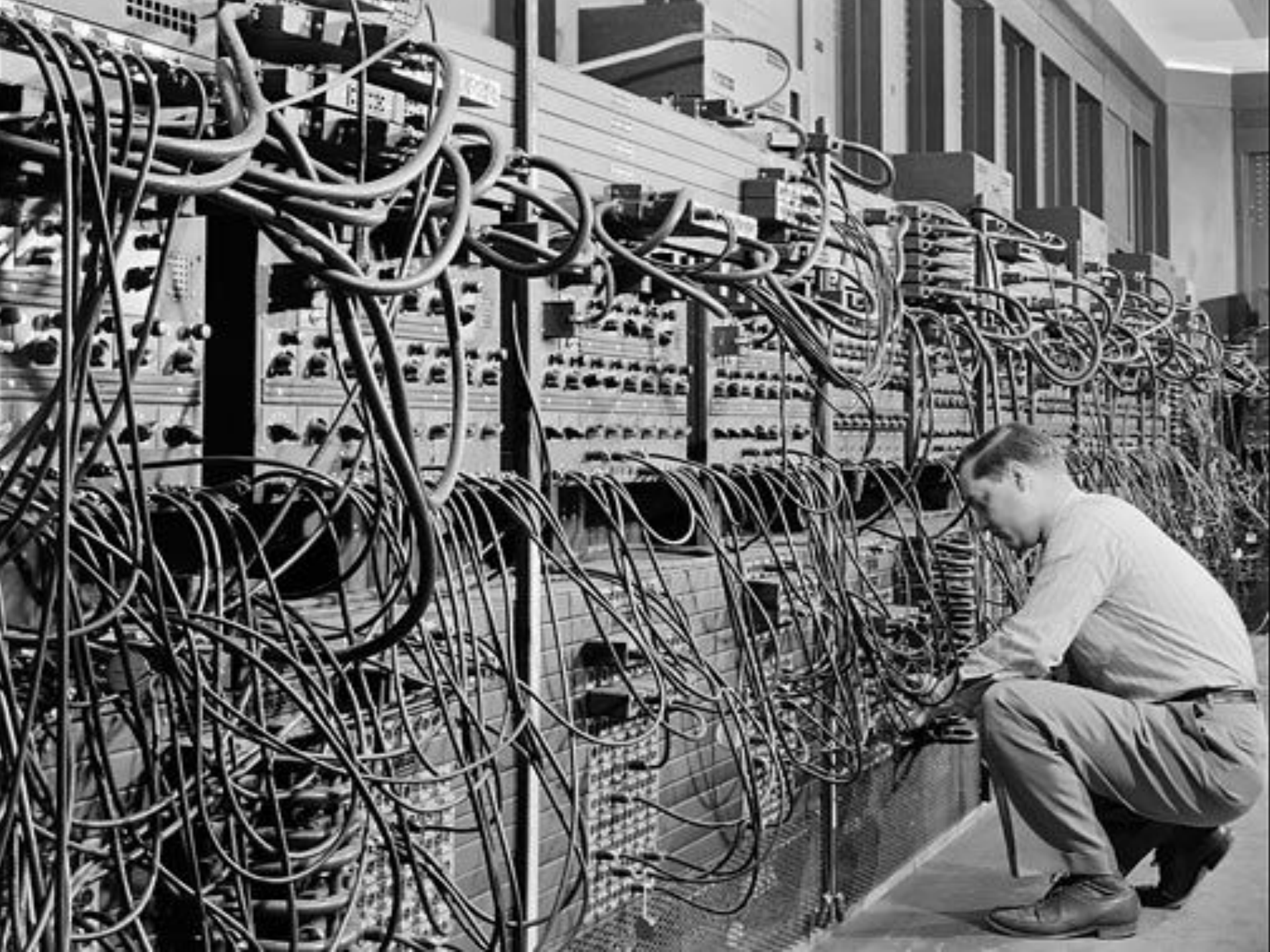
http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=50372

- 1979年，贝尔实验室的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博士完成了经典巨著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，即人们常说的C++2011.

感性认识C++程序

北京大学 信息科学技术学院

北京大学



什么是程序?

- 计算机也是机器！
  - 必须“设置”好才能运行；
    - ENIAC采用“手工插线”的方式“编程”；
  - “编程序” = 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；

问题：怎么告诉它呢？

- 告诉计算机一些什么东西，它才能运行？
- 以什么形式告诉它，它才能够明白？

先来做道题

- 清空一下大脑，来做个题：
  - 给你一个数列：

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

  - 请你找出其中最大的数字，快！

你怎么做的?

- 哪个数字最大？
  - 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；
  - 拿这个临时的“特别数字” 与其他数字相比较；
  - 如果有其他数字比临时的“特别数字”更大，就把“特别的数字”换成这个更大的数字；
  - 重复上述过程直到把所有的数字都比较完毕；
  - 那么大脑中这个“特别数字”就记录了最大的数字；

把这个过程稍做整理.....

- 更清楚的描述方式：
  - 在大脑中开辟一片“存储空间”存放输入的数字；
  - 使用了一个另一片“存储空间”存放“特别数字”；
  - 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：
    - 比较“存储空间中的数字”与“特别数字”；
    - 如果“存储空间中的数字”大于“特别数字”；
    - 那么，将“特别数字”换成“存储空间中的数字”；
  - 说出“特别数字”；

直接来读这个程序！

```
#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;
}
```

- 提个要求：

如果你要 创造 一门“程序设计语言”

你将如何设计呢？

北京大学

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？

```
#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;
}
```

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？

- **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

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

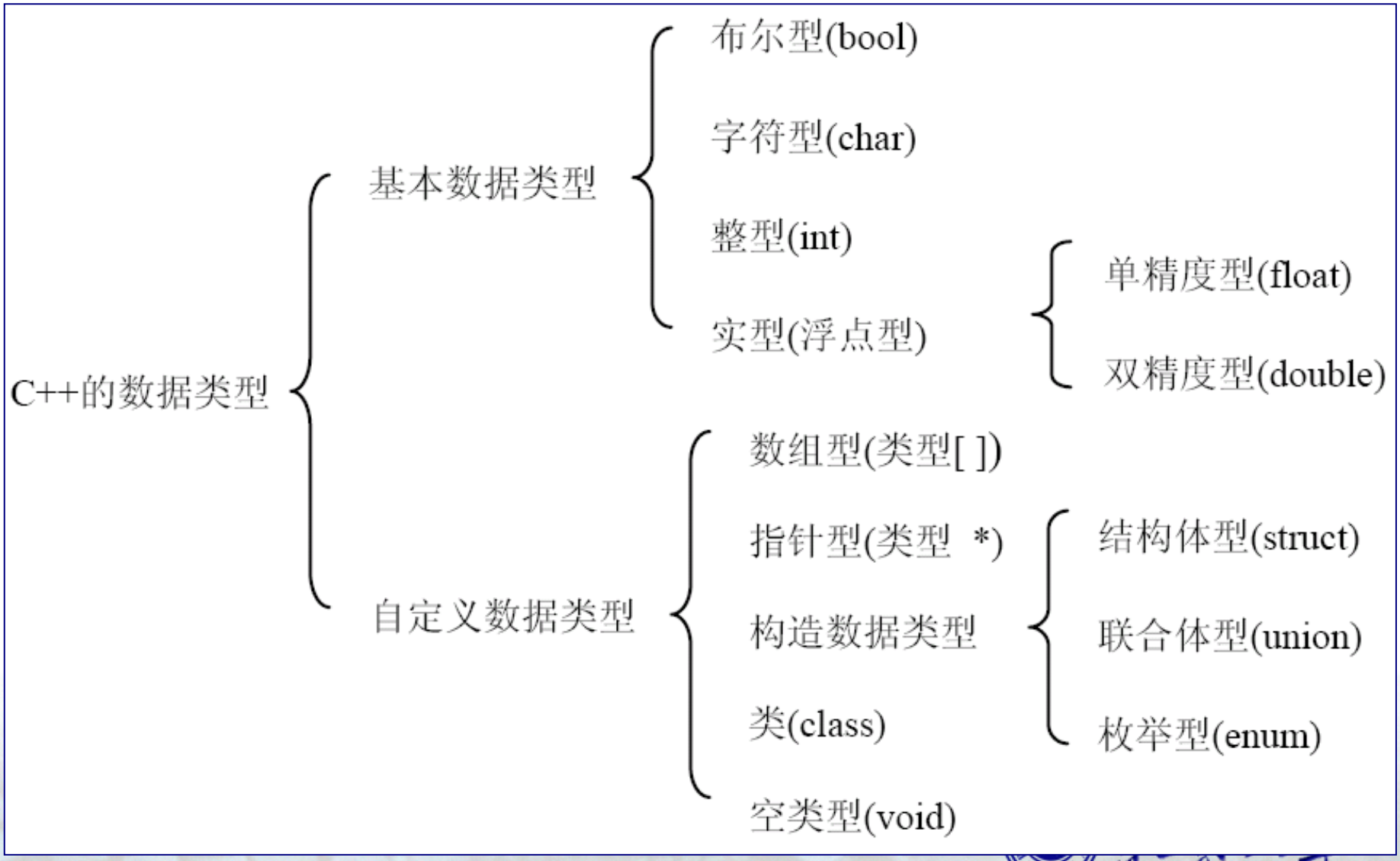
```
#include<iostream>
using namespace std;
int main()
{
    enum day{Mon, Tue, Wed, Thu, Fri, Sat, Sun};
    day  workDay;
    double  times, wages, hourlyRate, hours;
    cout<<"Enter the hourly wages rate.”<<endl;
    cin>>hourlyRate;
    cout<<"Enter hours worked daily\n";
    for (workDay=mon; workDay<=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？

- **NO！**
- **计算机只能“看懂”某些类型的数据，这些“数据的类型”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

- C++语言的运算符
  - 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: (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？

- **不多！三种而已！**

- C. Bohm & G. Jacopini, "Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules," Communications of the ACM, vol9(5) May 1966, pp 366-371.

如何表达纷繁复杂的计算逻辑?

' (# $

% & # $

! " # $

我们要学的编程语言

30几个关键字

十几种基本数据类型 + 30几个运算符号

三种基本的逻辑语句

怎么学？

从“简单的”开始

从“抄”程序开始

从“模仿”开始

抓大放小，学习狗熊

练习！练习！练习！

程序设计=K + S

好好想想，有没有问题?

谢　谢　！

北京大学