



# 《计算概论A》课程 程序设计部分

## C++语言基本成分 (1)

李 戈

北京大学 信息科学技术学院 软件研究所

2010年10月15日



北京大学



# 问 题

- 说说以下几个概念之间的关系：
  - ◆ 内存、地址、存储单元、字节
- 计算机程序为什么要先编译再运行？
- 说说程序中“`#include <iostream>`”的含义。
- 判断题：
  - ◆ 函数是C程序的基本组成单位。
  - ◆ 每个程序中只能包含一个函数。
  - ◆ 每个程序中只能包含一个Main（ ）函数。
  - ◆ 每个程序只能由一个文件组成。





# 程序设计语言的构成

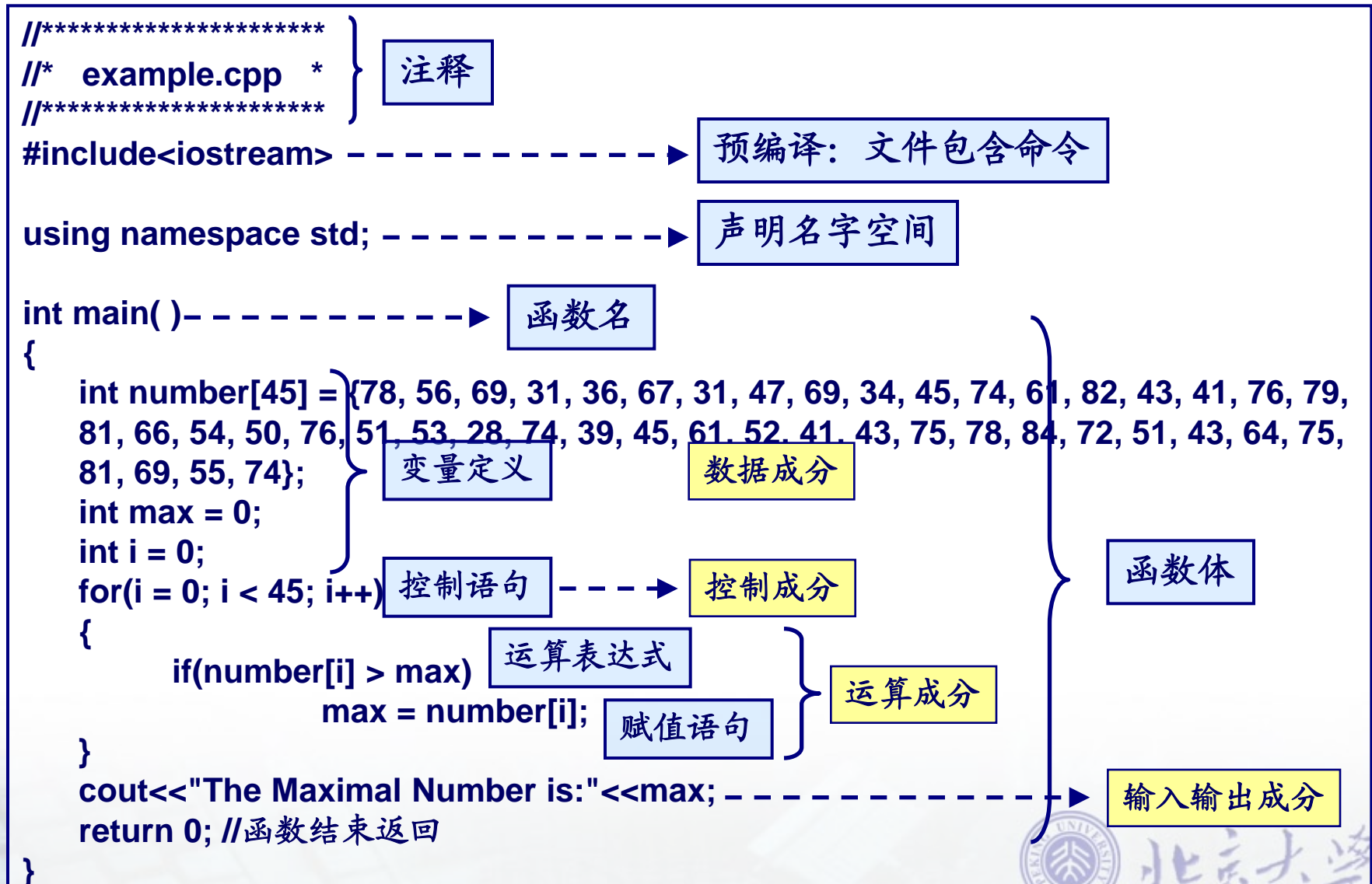
- 语言的种类千差万别，但是，一般说来，基本成分不外四种：
  - ◆ 数据成分，用以描述程序中所涉及的数据；
  - ◆ 运算成分，用以描述程序中所包含的运算；
  - ◆ 控制成分，用以表达程序中的控制构造；
  - ◆ 传输成分，用以表达程序中数据的传输；

——计算机科学技术百科全书



北京大学

# 简单程序的组成





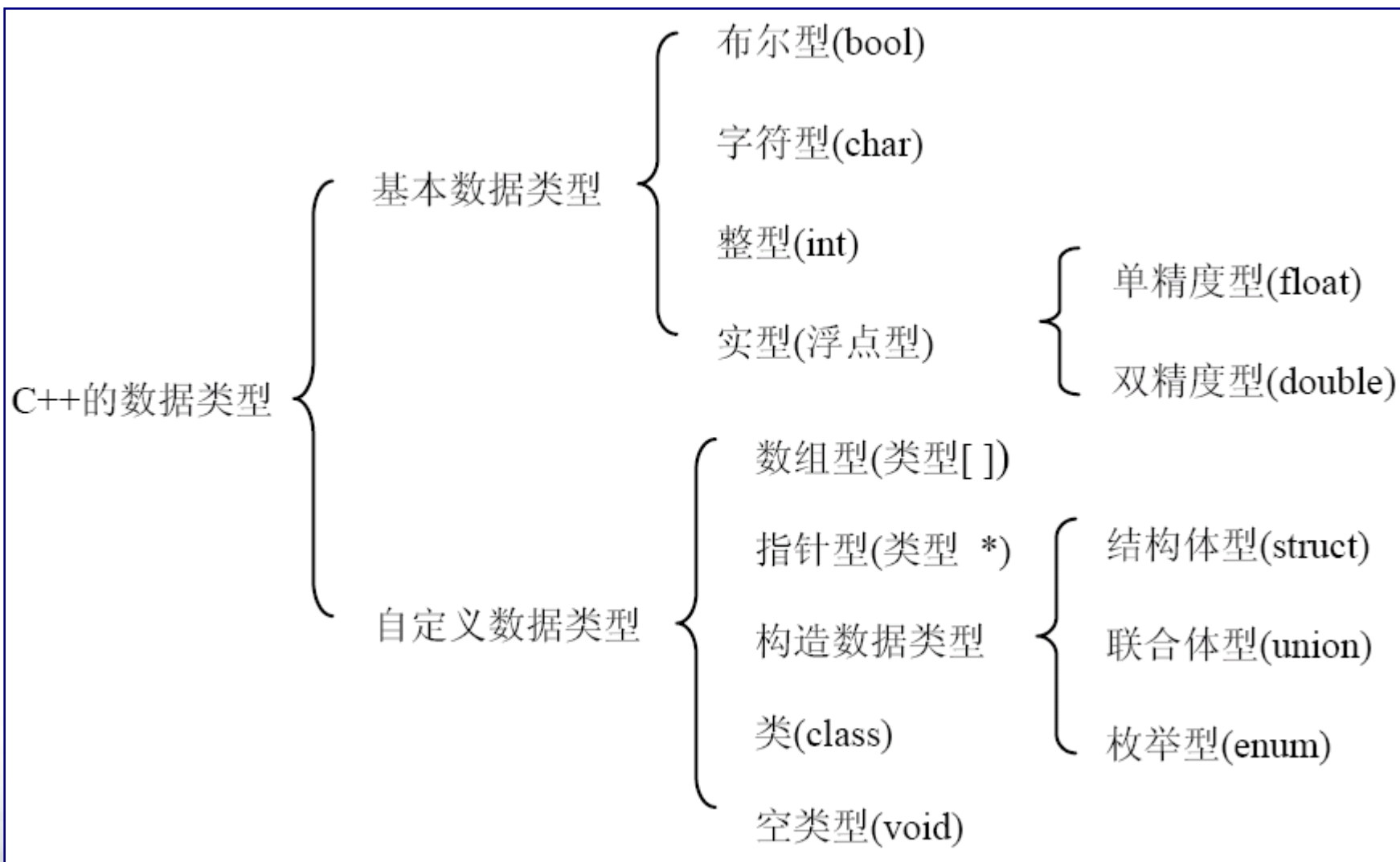
# C++程序设计的基本构成

## —— 数据成分（基本数据成分）



北京大学

# 计算机能够哪种类型的数据？







# 整型数据的分类

## ■ 按照数据的表示范围分类

- ◆ 基本型：以int表示
- ◆ 短整型：以short int或short表示
- ◆ 长整型：以long int 或long 表示

## ■ 按照数据有无符号分类

- ◆ 有符号型：表示某个范围内的整数
  - [signed] int; [signed] short; [signed] long
- ◆ 无符号型：表示某个范围内的正整数
  - unsigned int; unsigned short; unsigned long





# 整型数据的范围

- C/C++标准没有具体规定以上各类数据所占内存字节数，只要求long型数据长度不短于int型，short型不长于int型。
- VC中每种类型所占内存空间和表示的范围：

short [int]	2	-32 768~32 767
signed short [int]	2	-32 768~32 767
unsigned short [int]	2	0~65 535
int	4	-2 147 483 648~2 147 483 647
signed int	4	-2 147 483 648~2 147 483 647
unsigned int	4	0~4 294 967 295
long [int]	4	-2 147 483 648~2 147 483 647
signed long [int]	4	-2 147 483 648~2 147 483 647
unsigned long [int]	4	0~4 294 967 295



# 如何知道某种类型的数占多少字节?

## ■ sizeof 运算符

◆ 用于计算某种类型的对象在内存中所占的字节数。

```
#include <iostream>
using namespace std;
void main()
{
    cout<<"sizeof(short int)="<<sizeof(short int)<<endl;
    cout<<"sizeof(signed short int)="<<sizeof(signed short int)<<endl;
    cout<<"sizeof(unsigned short int)="<<sizeof(unsigned short int)<<endl;
    cout<<"sizeof(int)="<<sizeof(int)<<endl;
    cout<<"sizeof(signed int)="<<sizeof(signed int)<<endl;
    cout<<"sizeof(unsigned int)="<<sizeof(unsigned int)<<endl;
    cout<<"sizeof(long int)="<<sizeof(long int)<<endl;
    cout<<"sizeof(signed long int)="<<sizeof(signed long int)<<endl;
    cout<<"sizeof(unsigned long int)="<<sizeof(unsigned long int)<<endl;
}
```





# 以不同的方式输出整数

```
#include <iostream>
using namespace std;
int main()
{
    int x = 010, y = 10, z = 0x10;
    cout<<"以十进制形式显示: ";
    cout<<"x="<<x<<", y="<<y<<", z="<<z<<endl;
    cout<<"以八进制形式显示: ";
    cout<<oct<<"x="<<x<<", y="<<y<<", z="<<z<<endl;
    cout<<"以十六进制形式显示: ";
    cout<<hex<<"x="<<x<<", y="<<y<<", z="<<z<<endl;
    return 0;
}
```



# 整型数据的初始值

- 在VC中，定义一个int类型的变量，而不给它赋值，会出现什么情况？
- 例：

```
#include <iostream>
using namespace std;
int main()
{
    int a;
    cout<<a;
    return 0;
}
```

当输出是个很大的负数时，检查一下是否忘记赋值。





# 整型数据的溢出

- 在VC中一个int型变量的最大允许值为2147483647，如果再加1，会出现什么情况？

- 例：

```
int main()
{
    int a, b;
    a=2147483647;
    b=a+1;
    cout<<a<<" "<<b;
    return 0;
}
```

a:

0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

b:

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



北京大学

# 整型数据的存储

## ■ 存储空间的分配

有符号整数变量:

0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

无符号整数变量:

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

## ■ 整数补码的定义

$$[X]_{\text{补}} = \begin{cases} X & \text{if } 0 \leq X < 2^n \\ 2^{n+1} + X & \text{if } -2^n \leq X < 0 \end{cases}$$



北京大學

```
#include <iostream>
using namespace std;
int main()
{
    int x = 0x7fffffff;
    cout<<"以十六进制形式显示: ";
    cout<<hex<<"x="<<x<<endl;
    cout<<"以十进制形式显示: ";
    cout<<dec<<"x="<<x<<endl;
    int y = 0xffffffff;
    cout<<"以十六进制形式显示: ";
    cout<<hex<<"y="<<y<<endl;
    cout<<"以十进制形式显示: ";
    cout<<dec<<"y="<<y<<endl;
    int z = 0x80000000;
    cout<<"以十六进制形式显示: ";
    cout<<hex<<"z="<<z<<endl;
    cout<<"以十进制形式显示: ";
    cout<<dec<<"z="<<z<<endl;
    return 0;
}
```



# 实型数据的分类及范围

## ■ 实型数据

- ◆ 单精度型 (float型)
- ◆ 双精度型 (double型)
- ◆ 长双精度型 (long double型)

## ■ 实型数据的范围 (VC中)

- ◆ float: 浮点型, 占4个字节
  - 范围:  $-3.4 \times 10^{38} \sim 3.4 \times 10^{38}$ , 有效为7位
- ◆ double: 双精度, 占8个字节
  - 范围:  $-1.7 \times 10^{308} \sim 1.7 \times 10^{308}$ , 有效为15位
- ◆ long double: 长双精度, 占8个字节
  - 范围:  $-1.7 \times 10^{308} \sim 1.7 \times 10^{308}$ , 有效为15位





# 实型数据的舍入误差

- 由于实型变量是由有限的存储单元组成的，因此能提供的有效数字总是有限的，在有效位以外的数字将被舍去。

- 例：实型数据的舍入误差

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

- $a+20$ 的理论值应是12345678920，而一个实型变量只能保证的有效数字是7位有效数字；
- 运行程序得到的a和b的值是1.23457e+010

- 应当避免将一个很大的数和一个很小的数直接相加或相减，否则就会“丢失”小的数。



# 布尔型

- 布尔类型(bool)是C++新增的类型，只能取true和false两个值，分别对应整数的1和0。

```
void main()
{
    bool b1=true,b2=false;
    cout<<"b1=true 时， b1="<<b1<<endl;
    cout<<"b2=false 时， b2="<<b2<<endl;
    int i;
    cout<<"请输入一个正整数i: ";
    cin>>i;
    b1=i>3;
    cout<<"b1=i>3 时， b1="<<b1<<endl;
    b2=(bool)-100;
    cout<<"b2=-100 时， b2="<<b2<<endl;
}
```

# 字符型数据

## ■ 字符型常量

- ◆ 用单引号括起来的单个字符。如 ‘a’, ‘x’, ‘D’, ‘#’
- ◆ 一些特殊的字符常量，以 \ 开头的字符序列，有特殊的含义，故叫转义符，如： \n 表示换行

```
#include <iostream>
using namespace std;
int main( )
{
    char a = 'a';
    cout<<a<<endl;
    cout<<"This is the first line ! \n";
    cout<<'a'<<'\n';
    return 0;
}
```





# C++中的转义字符

转义字符	描 述
\a	响铃(audible bell)
\b	退格(backspace)
\f	换页(formfeed)
\n	换行(newline)
\r	回车(carriage return)
\t	水平制表(horizontal tab)
\v	垂直制表(vertical tab)
\\	反斜线(backslash)
\'	单引号(single quote)
\"	双引号(double quote)
\DDD	八进制数 DDD 对应的字符(octal number)
\xHH	十六进制数 HH 对应的字符(hexadecimal number)





# 字符型数据

## ■ 一个字符占一个字节

- ◆ 字符要转化成数字存储;
- ◆ 一个字符变量一般存放一个字符;
- ◆ 字符与字符串是不同的数据类型;
  - 不能用字符变量来存放字符串;

## ■ 字符数据的存储形式

- ◆ 在内存中字符变量的存储形式是ASCII码





# ASCII码

- **ASCII码(America Standard Code for Information Interchange, 美国信息交换标准码)**
  - ◆ 美国国家标准学会 (American National Standards Institute - ANSI) 制定
  - ◆ 早期使用7位表示, 只规定了128个最常用的英文字符;
  - ◆ 现在则使用8个位, 共可表示256个不同的文字与符号;
  - ◆ 在此基础上发展了16bits的Unicode







# 字符型数据

<b>L \ H</b>	<b>0000</b>	<b>0001</b>	<b>0010</b>	<b>0011</b>	<b>0100</b>	<b>0101</b>	<b>0110</b>	<b>0111</b>
<b>0000</b>	<b>NUL</b>	<b>DLE</b>	<b>SP</b>	<b>0</b>	<b>@</b>	<b>P</b>	<b>,</b>	<b>p</b>
<b>0001</b>	<b>SOH</b>	<b>DC1</b>	<b>!</b>	<b>1</b>	<b>A</b>	<b>Q</b>	<b>a</b>	<b>q</b>
<b>0010</b>	<b>STX</b>	<b>DC2</b>	<b>"</b>	<b>2</b>	<b>B</b>	<b>R</b>	<b>b</b>	<b>r</b>
<b>0011</b>	<b>ETX</b>	<b>DC3</b>	<b>#</b>	<b>3</b>	<b>C</b>	<b>S</b>	<b>c</b>	<b>s</b>
<b>0100</b>	<b>EOT</b>	<b>DC4</b>	<b>\$</b>	<b>4</b>	<b>D</b>	<b>T</b>	<b>d</b>	<b>t</b>
<b>0101</b>	<b>ENQ</b>	<b>NAK</b>	<b>%</b>	<b>5</b>	<b>E</b>	<b>U</b>	<b>e</b>	<b>u</b>
<b>0110</b>	<b>ACK</b>	<b>SYN</b>	<b>&amp;</b>	<b>6</b>	<b>F</b>	<b>V</b>	<b>f</b>	<b>v</b>
<b>0111</b>	<b>BEL</b>	<b>ETB</b>	<b>,</b>	<b>7</b>	<b>G</b>	<b>W</b>	<b>g</b>	<b>w</b>
<b>1000</b>	<b>BS</b>	<b>CAN</b>	<b>)</b>	<b>8</b>	<b>H</b>	<b>X</b>	<b>h</b>	<b>x</b>
<b>1001</b>	<b>HT</b>	<b>EM</b>	<b>(</b>	<b>9</b>	<b>I</b>	<b>Y</b>	<b>i</b>	<b>y</b>
<b>1010</b>	<b>LF</b>	<b>SUB</b>	<b>*</b>	<b>:</b>	<b>J</b>	<b>Z</b>	<b>j</b>	<b>z</b>
<b>1011</b>	<b>VT</b>	<b>ESC</b>	<b>+</b>	<b>;</b>	<b>K</b>	<b>[</b>	<b>k</b>	<b>{</b>
<b>1100</b>	<b>FF</b>	<b>FS</b>	<b>'</b>	<b>&lt;</b>	<b>L</b>	<b>\</b>	<b>l</b>	<b> </b>
<b>1101</b>	<b>CR</b>	<b>GS</b>	<b>-</b>	<b>=</b>	<b>M</b>	<b>]</b>	<b>m</b>	<b>}</b>
<b>1110</b>	<b>SO</b>	<b>RS</b>	<b>.</b>	<b>&gt;</b>	<b>N</b>	<b>^</b>	<b>n</b>	<b>~</b>
<b>1111</b>	<b>SI</b>	<b>US</b>	<b>/</b>	<b>?</b>	<b>O</b>	<b>_</b>	<b>o</b>	<b>DEL</b>







# 字符型数据

- 由于存储类型和整型相同
  - ◆ 字符数据和整型数据可以相互赋值
  - ◆ ASCII码可以和整数一样进行运算
  - ◆ 字符和整型均可以以两种形式输出

```
int a=32;
```

```
char b='a';
```

```
int c = b-a; //结果为 65
```

```
char a=6;
```

//溢出！ 因为a只能占用8位，最大256

//输出：♠ 与 a = 6; 相同



北京大学



# 数组

## ■ 定义其他类型的数组并附初始值

- ◆ `int a[26];`     *//定义数组必须指明数组大小*
- ◆ `float a[26];`
- ◆ `bool a[26];`
- ◆ `char a[26] = {'a', 'b', 'c', ..., 'z'};`

## ■ 数组的使用

- ◆ 要使用第n个数: `a[n-1];`    `a[n-1] = 1+2;`
- ◆ 注意: 对数组`a[n]`, 只能使用`a[0] – a[n-1]`;  
`a[n]`不能用;



# 字符类型与字符串常量

## ■ 字符串常量

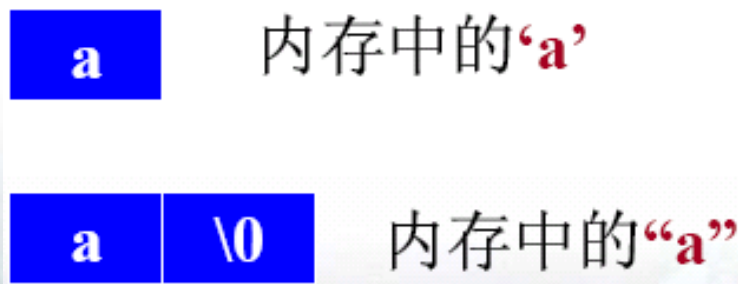
- ◆ 用双引号括起来的字符序列。如：

- ◆ “How do you do”, “CHINA”, “a”, “\$ 123.34” 等

## ■ 字符串可以一次输出，如：

- ◆ `cout<<“How do you do \n”;`

## ■ 注意 ‘a’是字符常量， “a”是字符串常量，二者不同



北京大学



# 看看这段程序

```
#include <iostream>
using namespace std;
int main( )
{
    int int_char_a = 'ab';
    int int_char_b = 'abcde';
    cout<<dec<<"int_char_a = "<<int_char_a<<endl;
    cout<<hex<<"int_char_a = "<<int_char_a<<endl;
    cout<<dec<<"int_char_b = "<<int_char_b<<endl;
    cout<<hex<<"int_char_b = "<<int_char_b<<endl;
    char char_c = "ab";
    return 0;
}
```





# C++基本数据类型

数据类型标识符	字节数	数值范围
bool	1	false, true
char	1	-128~127
signed char	1	-128~127
unsigned char	1	0~255
short [int]	2	-32 768~32 767
signed short [int]	2	-32 768~32 767
unsigned short [int]	2	0~65 535
int	4	-2 147 483 648~2 147 483 647
signed int	4	-2 147 483 648~2 147 483 647
unsigned int	4	0~4 294 967 295
long [int]	4	-2 147 483 648~2 147 483 647
signed long [int]	4	-2 147 483 648~2 147 483 647
unsigned long [int]	4	0~4 294 967 295
float	4	3.4e-38~3.4e38
double	8	1.7e-308~1.7e308
long double	8	1.7e-308~1.7e308





# 变量的定义

- 变量必须**先定义，再使用**
- 变量的定义格式

( 变量类型 ) ( 变量标识符 ) ;

**int           Max;**

**int           Max = 0;**

**char          character;**

**char          character = 'A' ;**

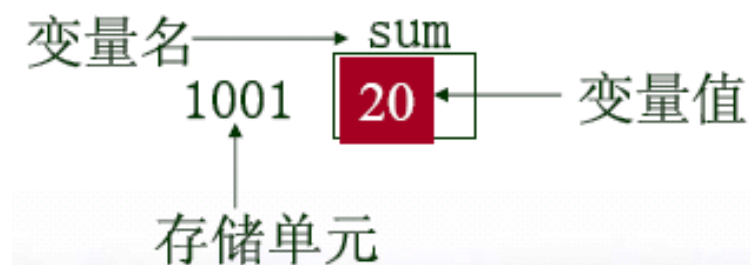
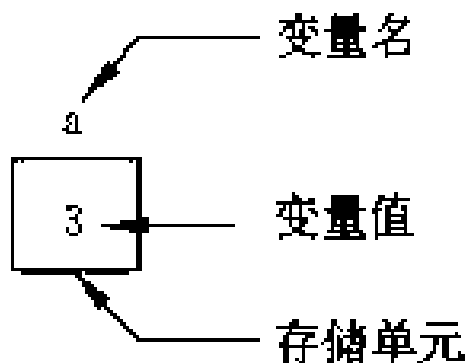
**double       Result = 12.345 ;**



# 变量的存储

## ■ 变量

- ◆ 每个变量在内存中占用一定的存储单元
- ◆ 变量的名字对应于存储单元的地址
- ◆ 变量的类型对应于存储单元的大小



## ■ 变量的定义

- ◆ 类型说明符 变量名标识符, 变量名标识符, ...;



北京大学





# C++ 程序的 标识符

## ■ 什么是标识符

- ◆ 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**称为**标识符**(identifier)。
- ◆ C++语言规定标识符只能由字母、数字和下划线三种字符组成，且第一个字符必须为**字母**或**下划线**，且不可与保留字（关键字）相同。

## ■ 合法的标识符:

sum, average, -total, class, day, month,  
student-name, tan, lotus-1-2-3, basic, li-ling

## ■ 不合法的标识符:

M.D.John, ¥ 123, # 33, 3D64, a>b



北京大學



# C++ 语言的 保留字

- 在程序里具有语言**预先定义好的特殊意义**，因此不能用于其他目的，不能作为普通的名字使用。

◆ C++保留字：

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





# 变量的命名

## ■ 匈牙利命名法

- ◆ 由Microsoft的著名开发人员、Excel的主要设计者**查尔斯·西蒙尼**在他的博士论文中提出来的，由于西蒙尼的国籍是匈牙利，所以这种命名法叫**匈牙利命名法**。

## ■ C标识符的命名规则：

1. 标识符的名字以一个或者多个**小写字母**开头，用这些字母来**指定数据类型**。
2. 在标识符内，前缀以后就是一个或者多个**第一个字母大写的**单词，这些单词清楚地指出了源代码内那个**对象的用途**。

■ 如： `chGrade; nLength; bEnable; strStudentName`



北京大学



好好想想，有没有问题？

谢谢！



清华大学