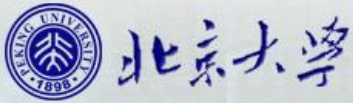
《计算概论A》课程 程序设计部分

C++语言基本成分 —— 数据成分

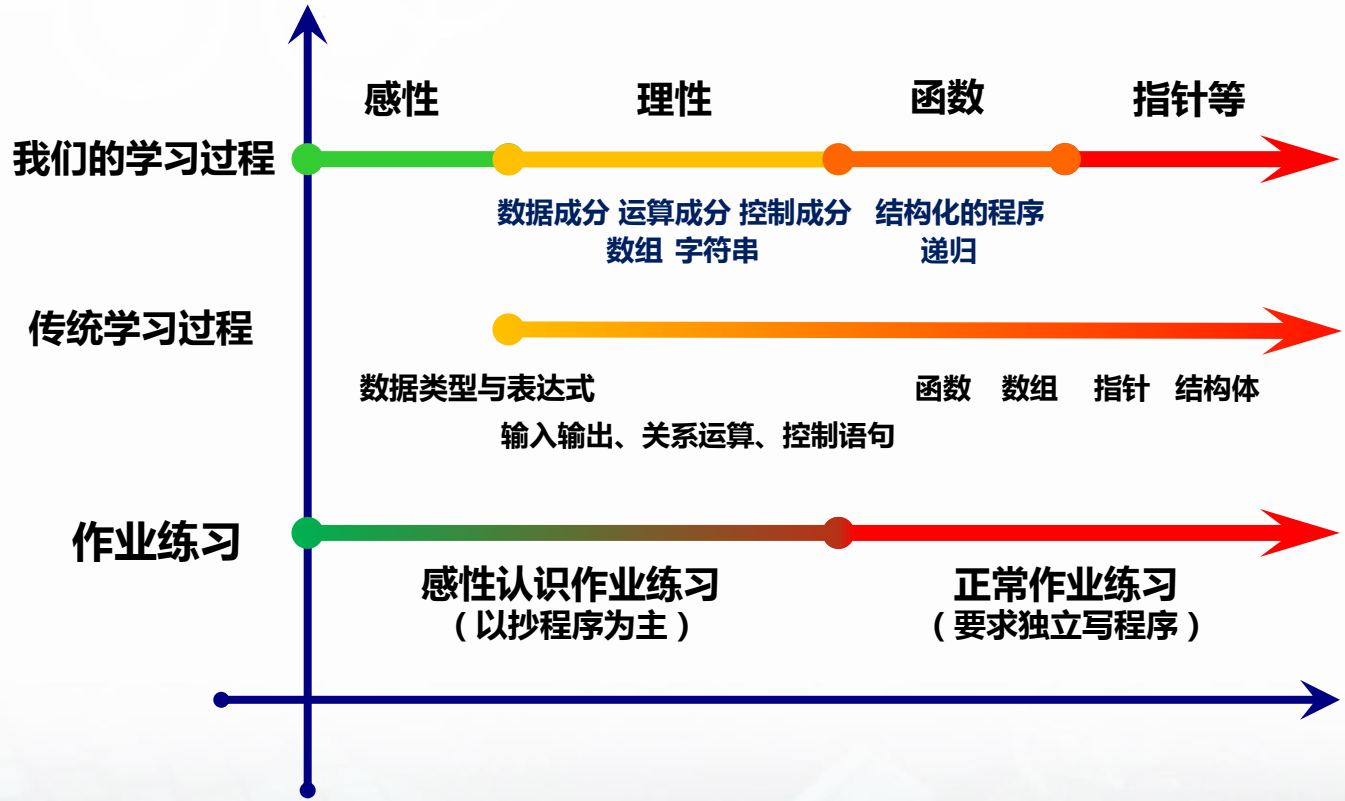
李 戈

北京大学 信息科学技术学院 软件研究所

lige@sei.pku.edu.cn



再谈，我们的进度安排


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）

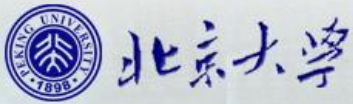
程序设计语言的构成

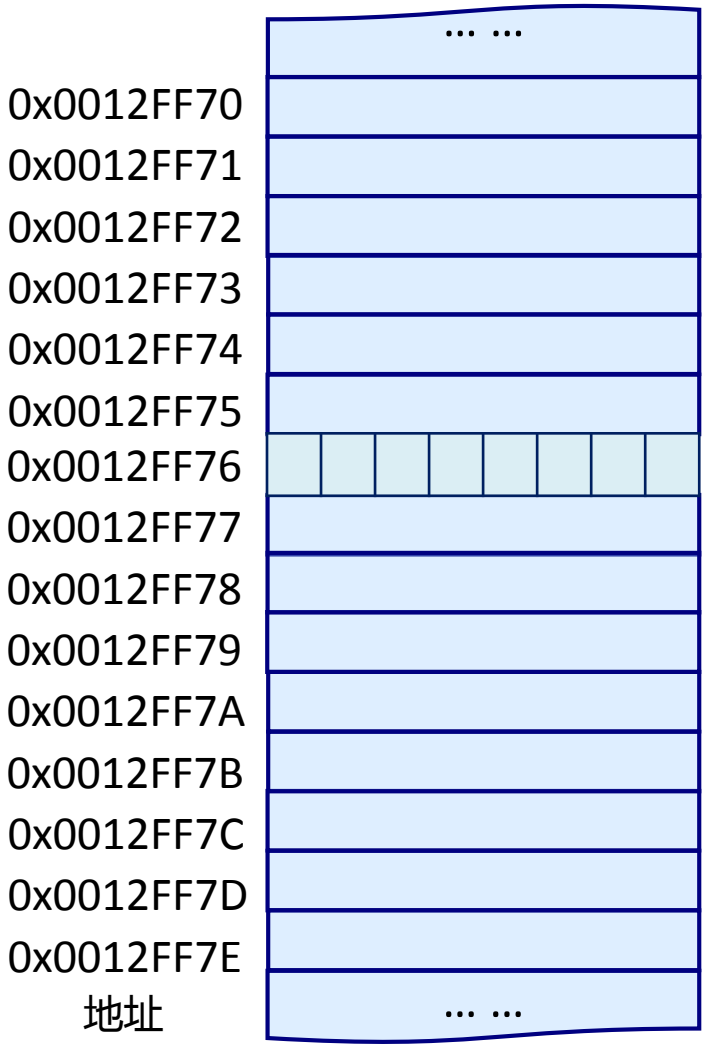
- **语言的种类千差万别，但是，一般说来，基本成分不外四种：**
  - **数据成分**，用以描述程序中所涉及的数据；
  - **运算成分**，用以描述程序中所包含的运算；
  - **控制成分**，用以表达程序中的控制构造；
  - **传输成分**，用以表达程序中数据的传输；

——计算机科学技术百科全书

C 程序设计语言的基本构成

—— 数据成分





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

```
#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;
}
```

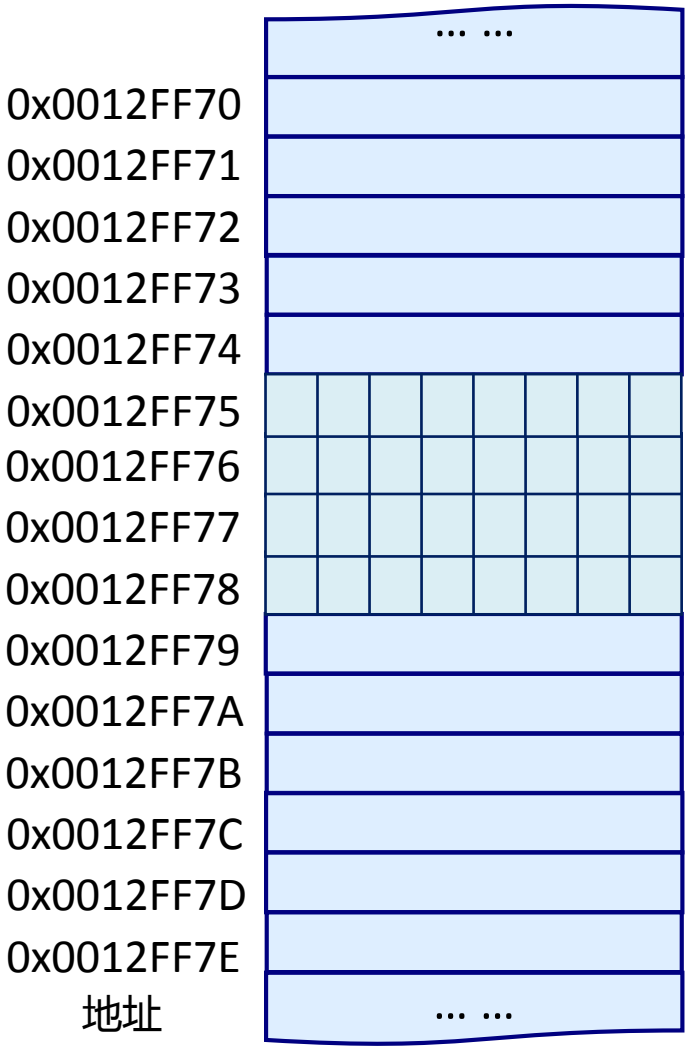
变量：先定义 再使用

- “值可以变化的量”
- 变量的定义格式

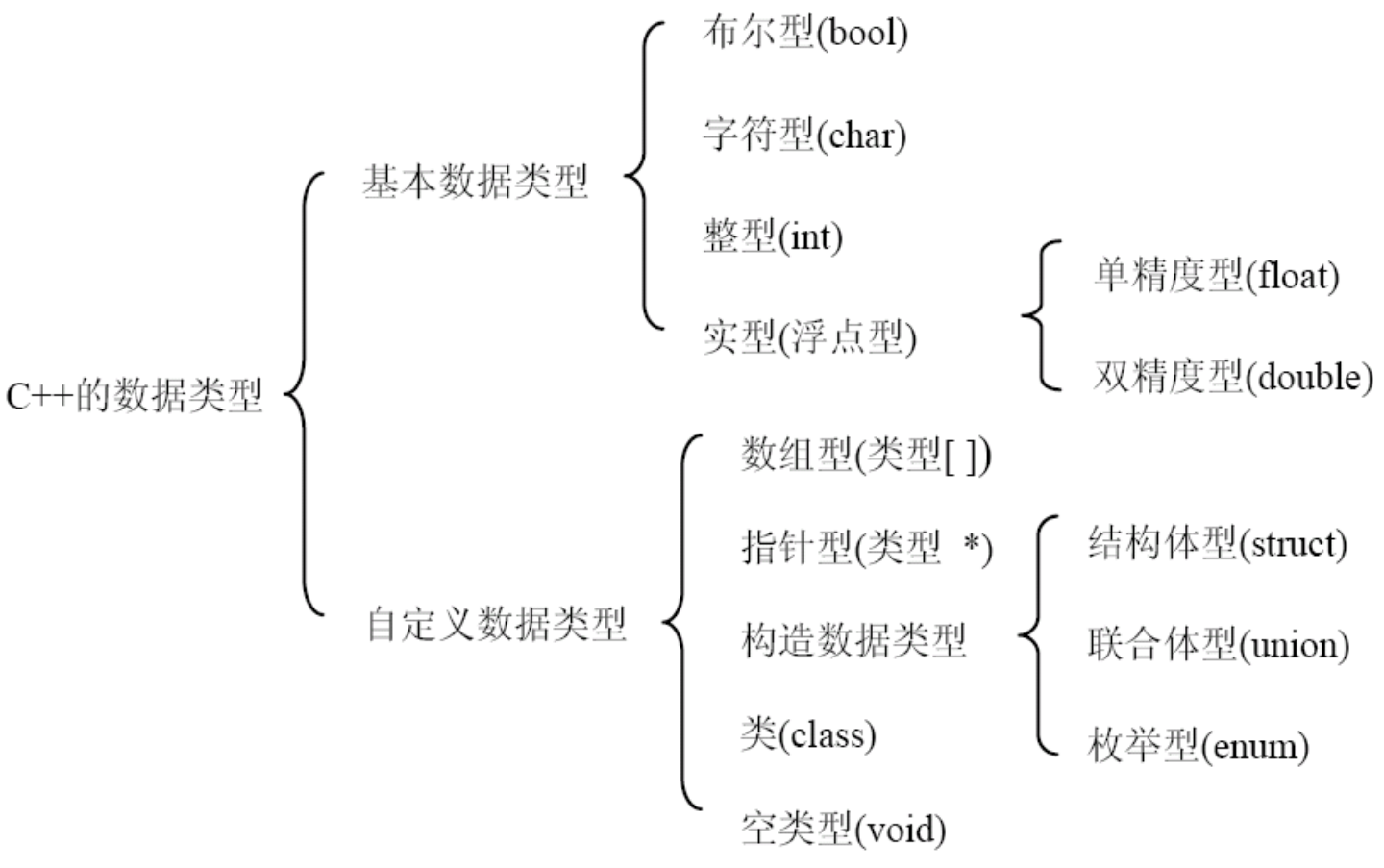
（变量类型）（变量标识符）；

```
int     Max;
int     Max = 0;
char    character;
char    character = ‘A’ ;
double  Result  = 12.345 ;
```

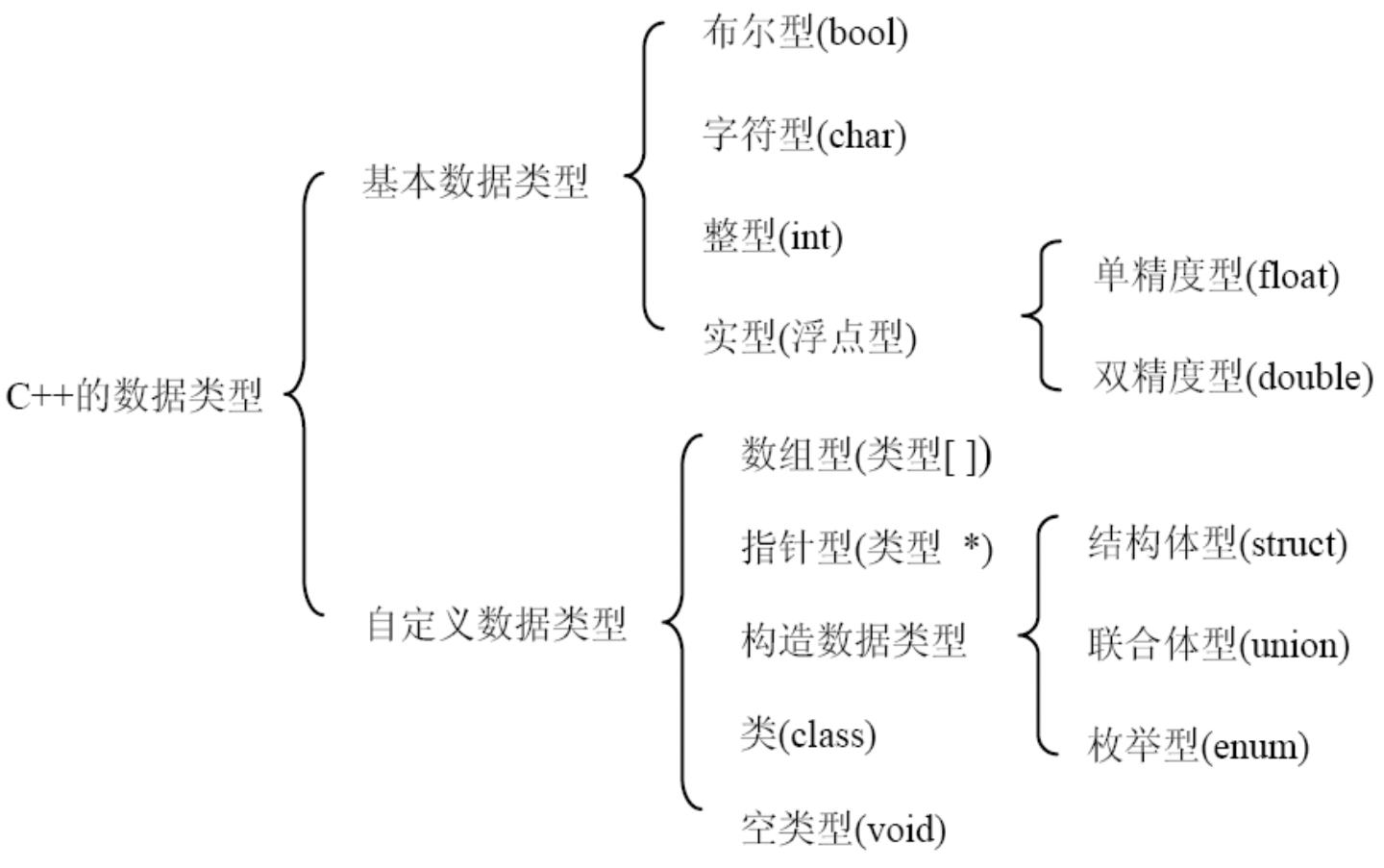
请这样来 想象 内存


… …
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)

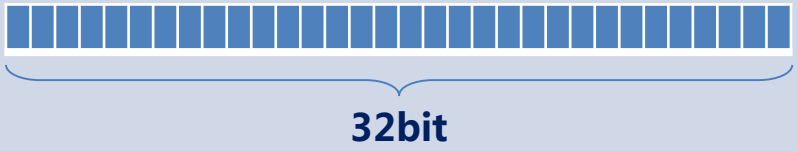
C/C++ 程序中的数据类型


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)

整型数据的分类

	整型
基本型	**int**
短整型	**short** short int
长整型	**long** long int

整型数据的分类

	整型	内存空间
基本型	**int**	
短整型	**short** short int	
长整型	**long** long int	

【注】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。

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;
int main()
{
    cout << "sizeof(short int)=" << sizeof(short int) << endl;
    cout << "sizeof(int)=" << sizeof(int) << endl;
    cout << "sizeof(long int)=" << sizeof(long int) << endl;
    return 0;
}
```

整型数据的分类

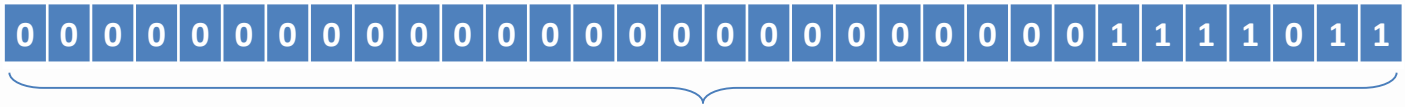
	有符号	无符号
基本型	**int** signed int	**unsigned int**
短整型	**short** short int signed short signed short int	**unsigned short** unsigned short int
长整型	**long** long int signed long signed long int	**unsigned long** unsigned long int

整型数据的分类

	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

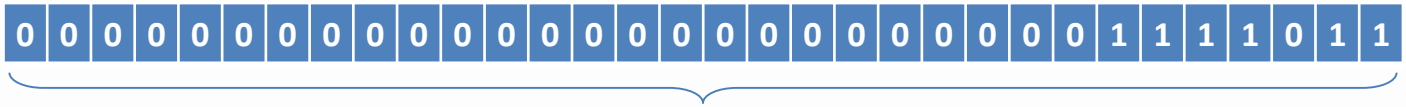
unsigned int i = 123;



signed int i = -123;

signed vs. unsigned

unsigned int i = 123;



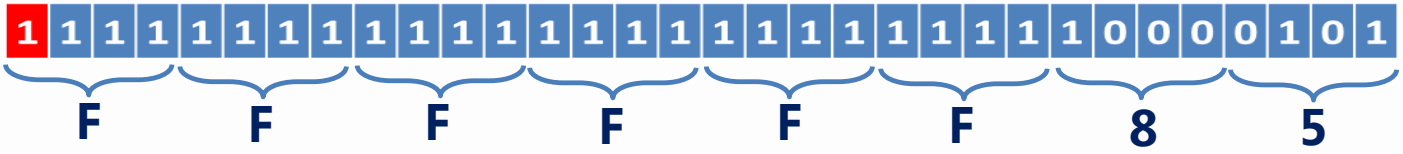
32bit

signed int i = -123;

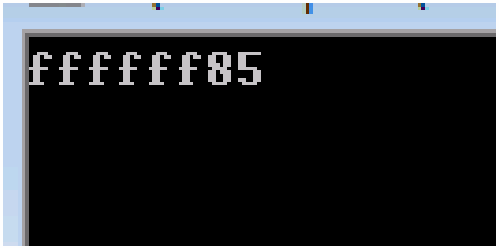
32bit

打印一个数的十六进制表示

signed int i = -123;

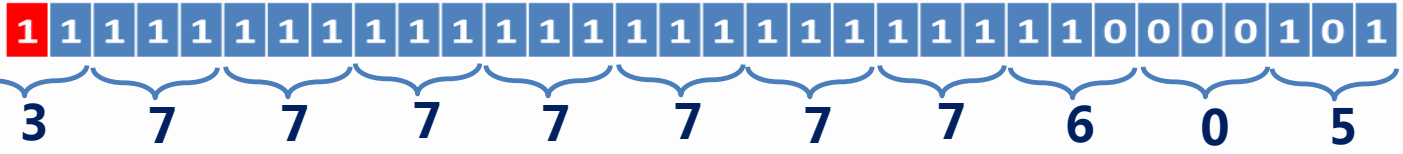



```
#include <iostream>
using namespace std;
int main()
{
        int a = -123;
        cout << hex << a << endl;
        return 0;
}
```

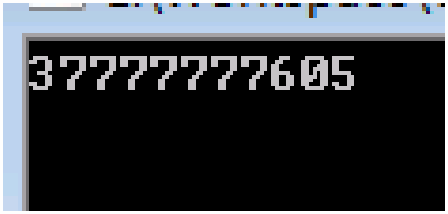


打印一个数的八进制

signed int i = -123;

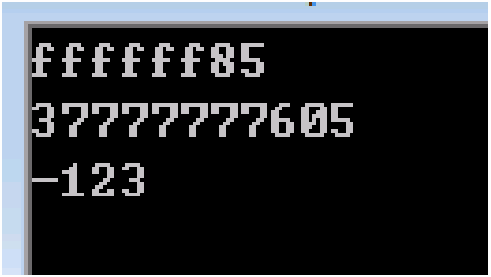



```
#include <iostream>
using namespace std;
int main()
{
        int a = -123;
        cout << oct << a << endl;
        return 0;
}
```

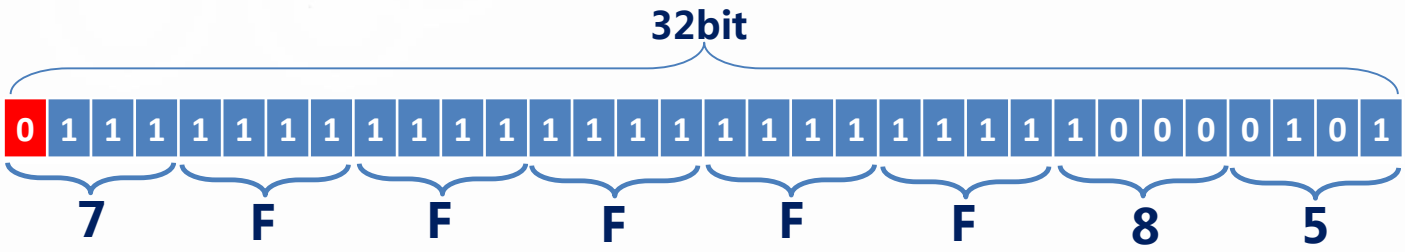


以不同的方式输出整数

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



把一个十六进制数输入程序

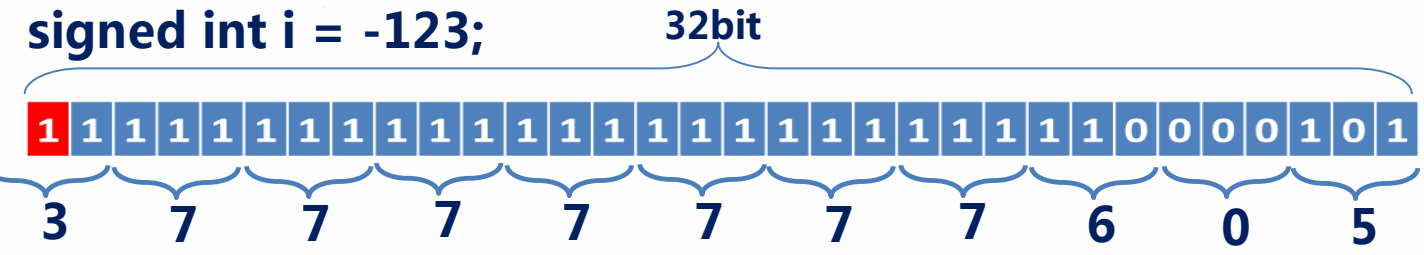



```
#include <iostream>
using namespace std;
int main()
{
        int a = 0x7FFFFF85;
        cout << dec << a << endl;
        cout << oct << a << endl;
        return 0;
}
```

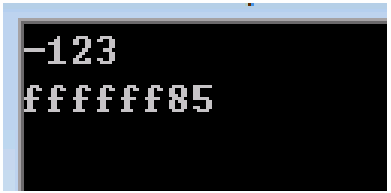
```
2147483525
17777777605
```

把一个八进制数输入程序

signed int i = -123;

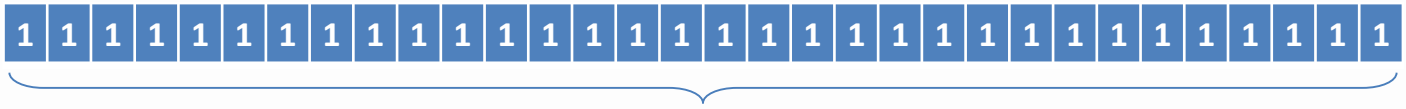



```
#include <iostream>
using namespace std;
int main()
{
        int a = 037777777605;
        cout << dec << a << endl;
        cout << hex << a << endl;
        return 0;
}
```



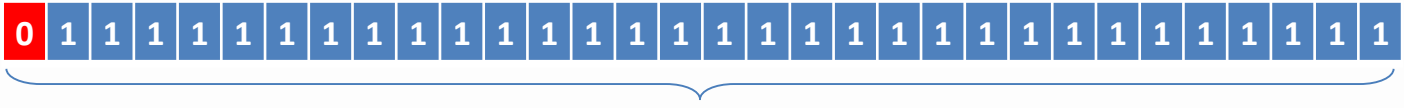
Max & Min

unsigned int i = Max;



32bit

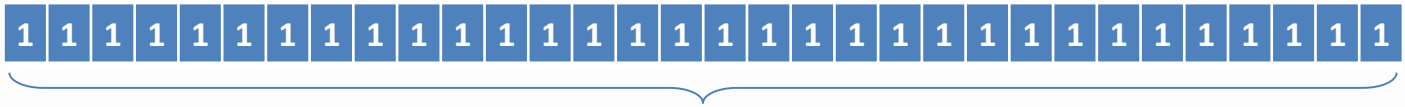
signed int i = Max;



32bit

Max & Min

unsigned int i = Max;



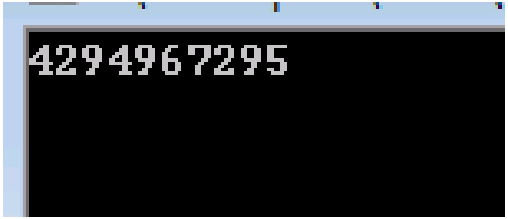
32bit

si

1

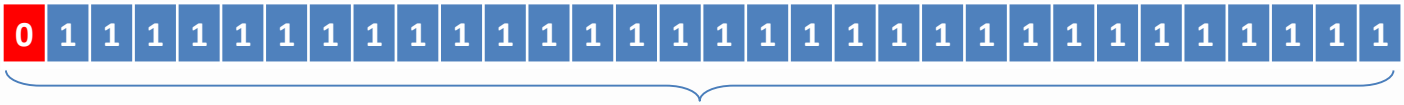
1

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

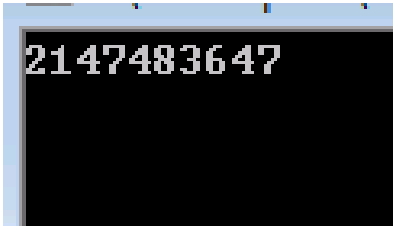


Max & Min

signed int i = Max;

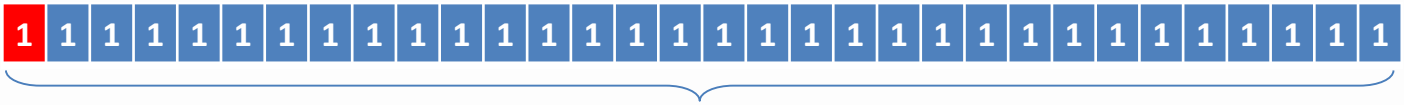



```
#include <iostream>
using namespace std;
int main()
{
        signed int a = 0x7FFFFFFF;
        cout << dec << a << endl;
        return 0;
}
```



Max & Min

signed int i = Min;



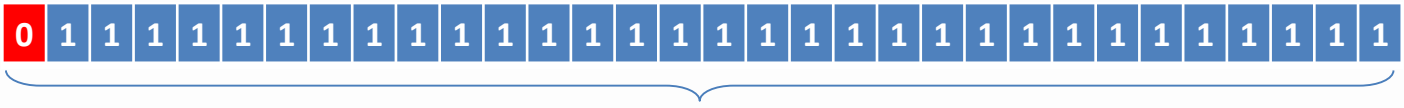
Right ? No!

```
#include <iostream>
using namespace std;
int main()
{
        signed int a = 0x7FFFFFFF;
        a = a + 1;
        cout << dec << a << endl;
        return 0;
}
```

```
-2147483648
```

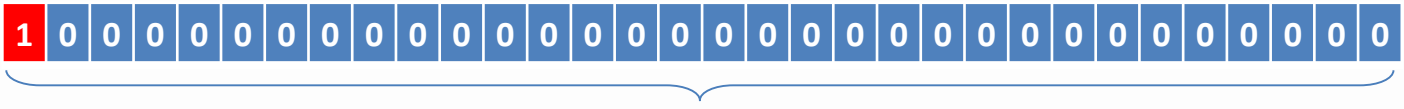
Max & Min

signed int i = Max;



32bit

2147483648



32bit

1

- **当最高位是1其他位为0（-0）时，最高位既表示负号，也表示整数最高位1.**

= -2147483648

整型数据的范围

- VC中每种类型所占内存空间和表示的范围：

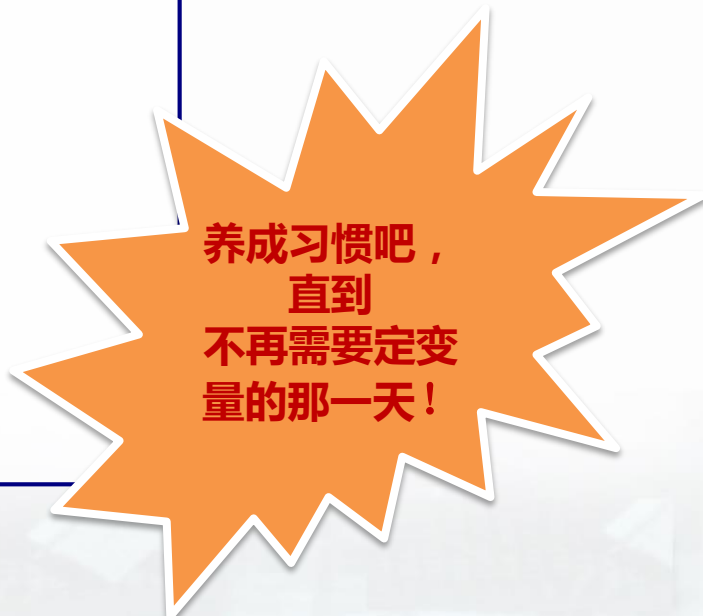
short [int]	2	−32 768～32 767
signed short [int]	2	−32 768～32 767
unsigned short [int]	2	0～65 535
int	4	−2 147 483 648～2 147 483 647
signed int	4	−2 147 483 648～2 147 483 647
unsigned int	4	0～4 294 967 295
long [int]	4	−2 147 483 648～2 147 483 647
signed long [int]	4	−2 147 483 648～2 147 483 647
unsigned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

- 总结：**其实，就用 int 就行了！**

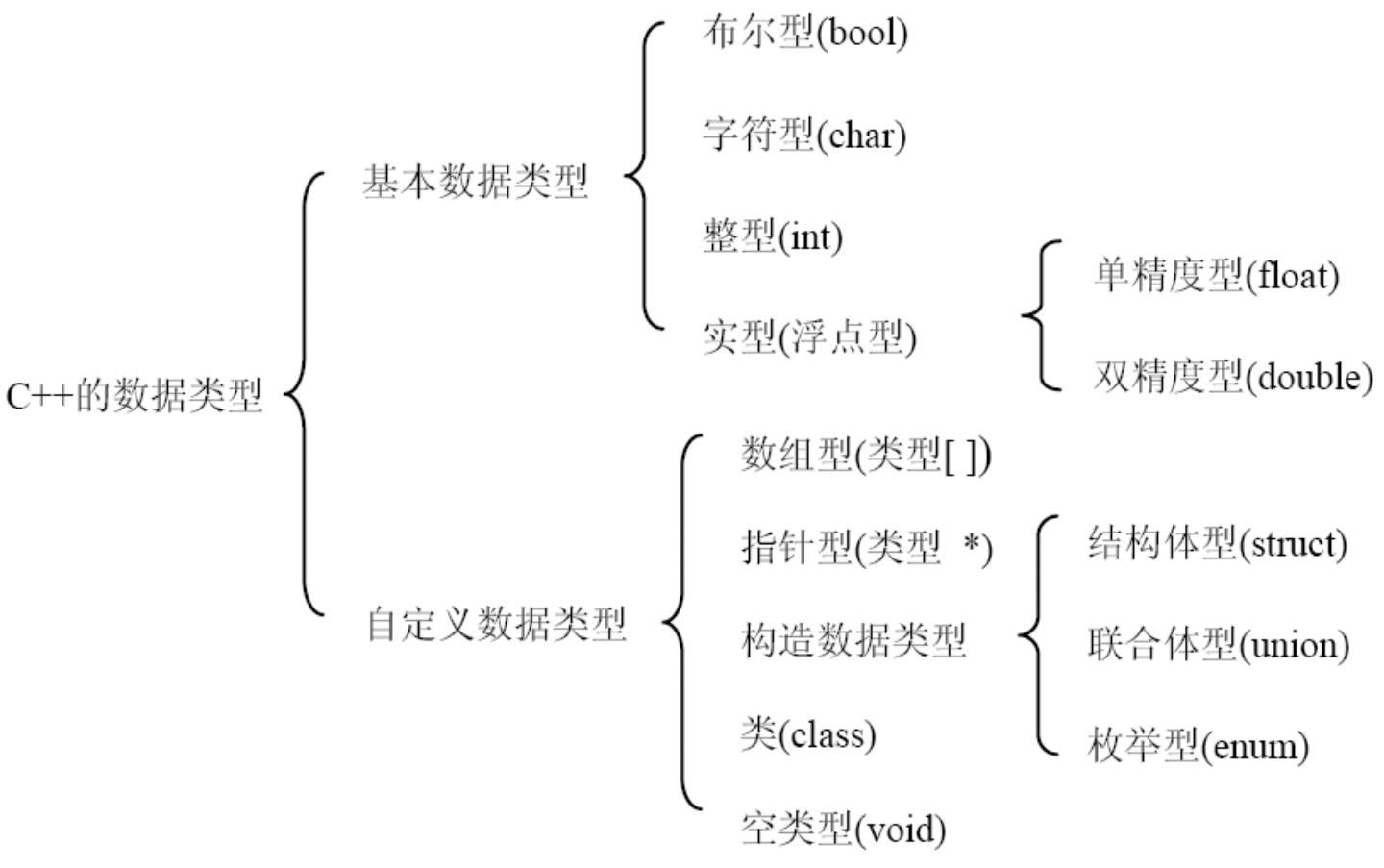
使用须知

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

```
number is: 4201998
```



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)

浮点型

浮点型 = 实型

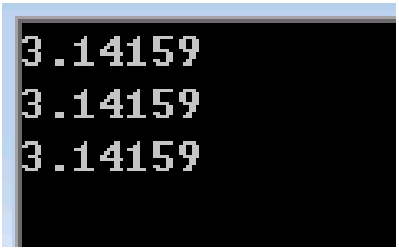
浮点型	长度	有效位	范围
float	32bit	7位	$-3.4*10^{38}$～$3.4*10^{38}$
double	64bit	15位	$-1.7*10^{308}$～$1.7*10^{308}$
long double	64bit	15位	$-1.7*10^{308}$～$1.7*10^{308}$

感受浮点型

```
#include <iostream>
using namespace std;
int main()
{
        float a = 3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706 79;

        double b=3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706 79;

        long double c =3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706 79;
        cout << a << endl;
        cout << b << endl;
        cout << c << endl;
        return 0;
}
```

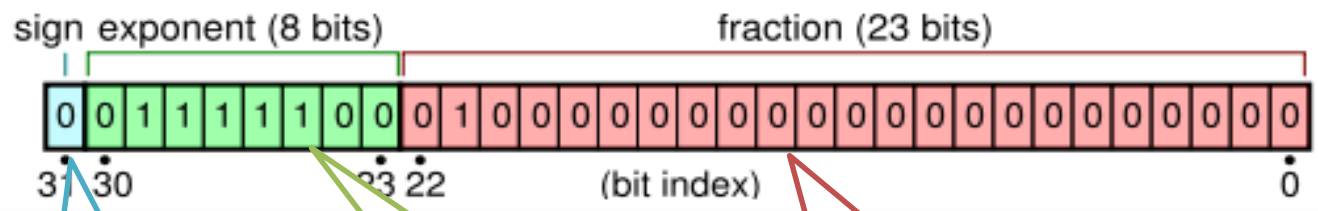


感受浮点型

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
        float a = 3.14159265358979323846264338327950288419716939937510
582097494459230781640628620899862803482534211706 79;
        double b=3.14159265358979323846264338327950288419716939937510
582097494459230781640628620899862803482534211706 79;
        long double c =3.14159265358979323846264338327950288419716939
937510582097494459230781640628620899862803482534211706 79;
        cout << setprecision(100) << a << endl;
        cout << b << endl;
        cout << c << endl;
        return 0;
}
```

```
3.1415927410125732000000000000000000000000000000000000000000000000000000000000000000000000000000
00
3.1415926535897931000000000000000000000000000000000000000000000000000000000000000000000000000000
00
3.1415926535897931000000000000000000000000000000000000000000000000000000000000000000000000000000
00
```

浮点数的表示



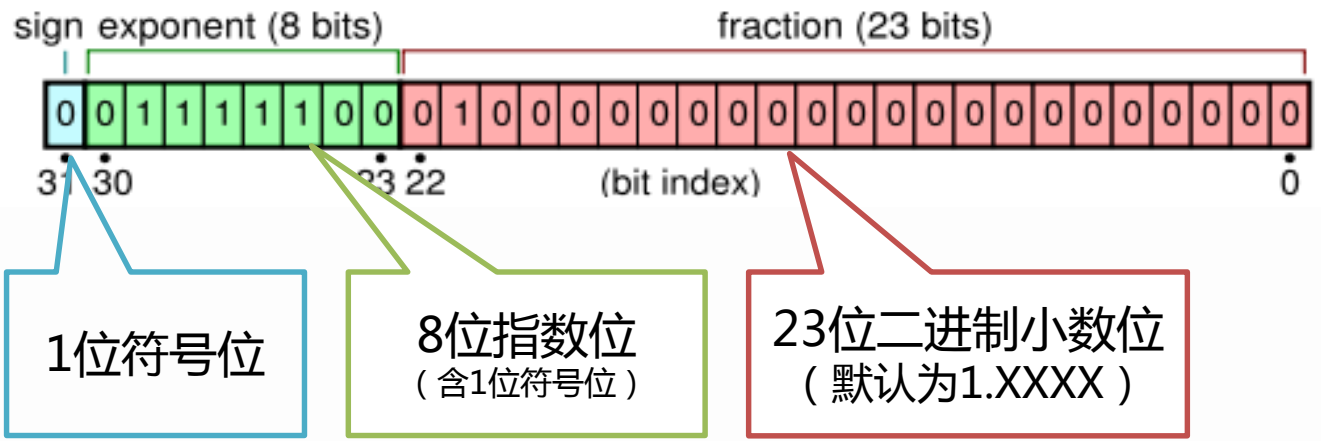
1位符号位

8位指数位
（含1位符号位）

23位二进制小数位
（默认为1.XXXX）

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

浮点数的表示



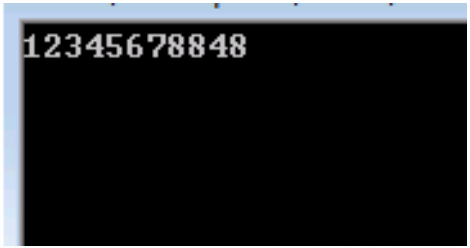
$$\log_{10}(2^{127}) \approx 38.23 \quad \log_{10}(2^{24}) \approx 7.225$$

- **IEEE Standard for Floating-Point Arithmetic (IEEE 754)**
  - **http://grouper.ieee.org/groups/754/**
- **ISO/IEC/IEEE 60559:2011 - Information technology - Microprocessor Systems - Floating-Point arithmetic**
  - **http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=57469**

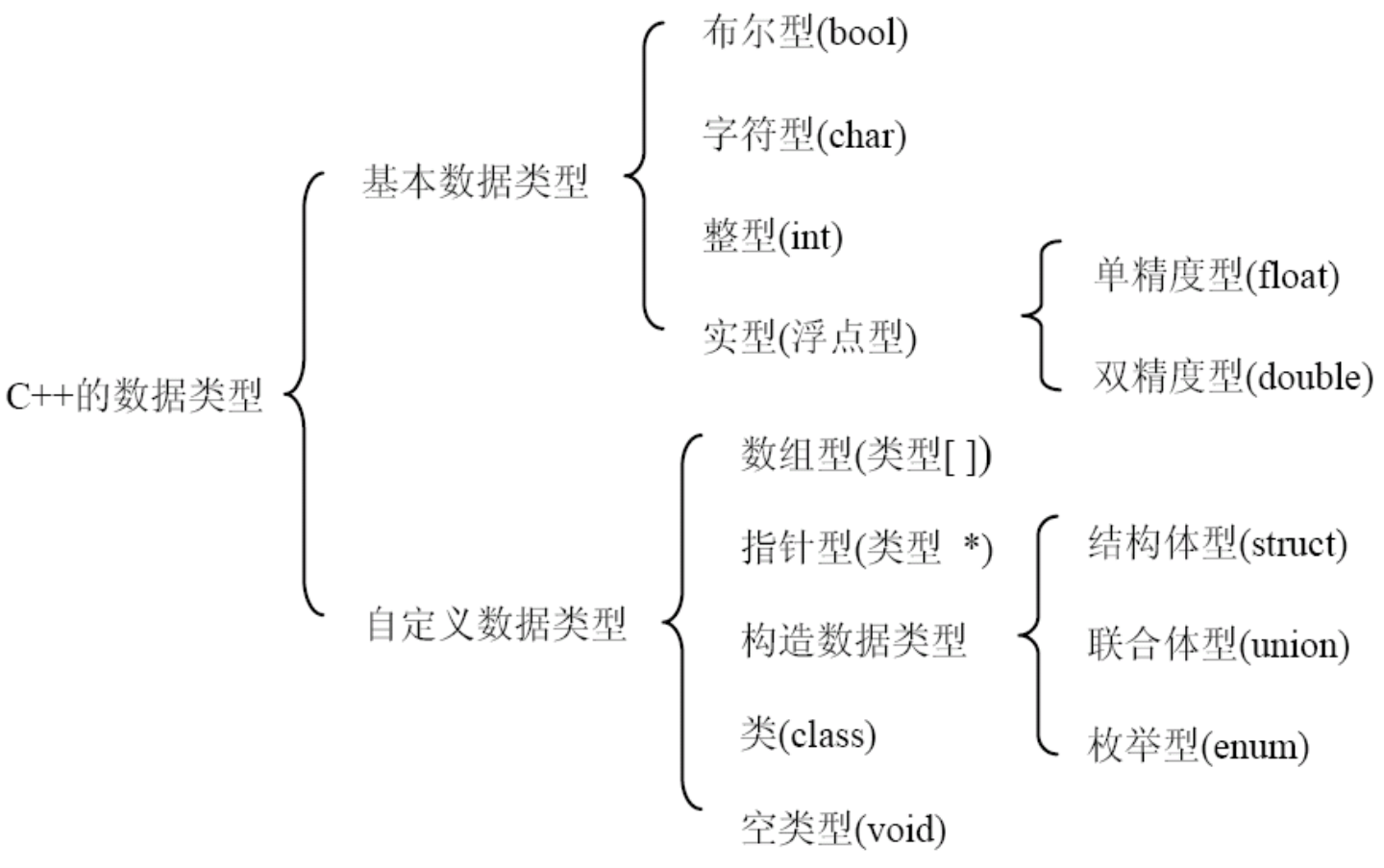
使用须知

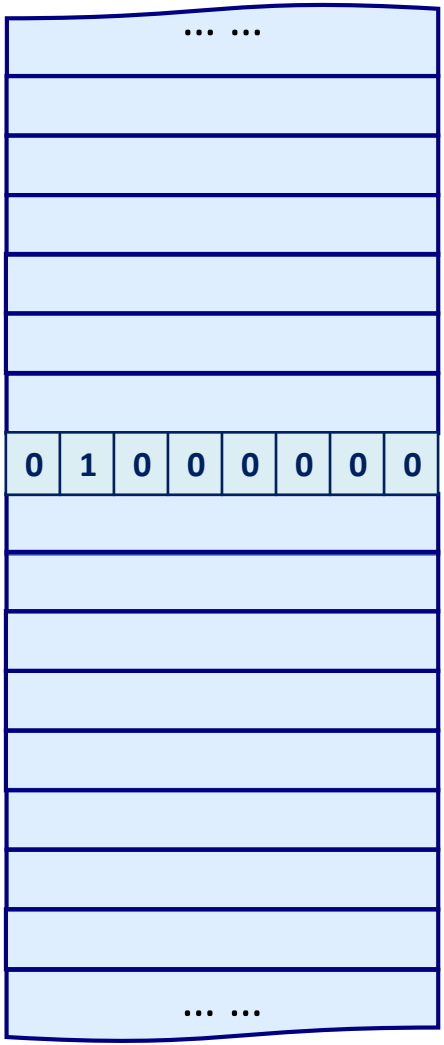
```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
        float a, b;
        a = 123456.789e5;
        b = a + 20;
        cout << setprecision(20) << b << endl;
        return 0;
}
```

避免将一个很大的数与一个很小的数直接相加或相减，否则就会“丢失”小的数。



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)



字符型数据

- 一个字符型占一个字节；
  - 其值可以是任何“可以出现在C/C++语言中的字符”；
  - 最多可以表示_____个字符

- 例如

  char a = ‘@’；

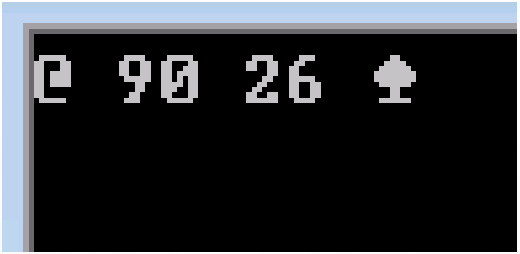
  @ ⟷ 64 ⟷ 01000000

0	<NUL>	32	<SPC>	64	@	96	`	128	Ä	160	†	192	¿	224	‡
1	<SOH>	33	!	65	A	97	a	129	Å	161	°	193	¡	225	·
2	<STX>	34	"	66	B	98	b	130	Ç	162	¢	194	¬	226	‚
3	<ETX>	35	#	67	C	99	c	131	É	163	£	195	√	227	„
4	<EOT>	36	$	68	D	100	d	132	Ñ	164	§	196	ƒ	228	‰
5	<ENQ>	37	%	69	E	101	e	133	Ö	165	•	197	≈	229	Â
6	<ACK>	38	&	70	F	102	f	134	Ü	166	¶	198	Δ	230	Ê
7	<BEL>	39	'	71	G	103	g	135	á	167	ß	199	«	231	Á
8	<BS>	40	(	72	H	104	h	136	à	168	®	200	»	232	Ë
9	<TAB>	41	)	73	I	105	i	137	â	169	©	201	…	233	È
10	<LF>	42	*	74	J	106	j	138	ä	170	™	202		234	Í
11	<VT>	43	+	75	K	107	k	139	ã	171	´	203	À	235	Î
12	<FF>	44	,	76	L	108	l	140	å	172	¨	204	Ã	236	Ï
13	<CR>	45	-	77	M	109	m	141	ç	173	≠	205	Õ	237	Ì
14	<SO>	46	.	78	N	110	n	142	é	174	Æ	206	Œ	238	Ó
15	<SI>	47	/	79	O	111	o	143	è	175	Ø	207	œ	239	Ô
16	<DLE>	48	0	80	P	112	p	144	ê	176	∞	208	–	240	
17	<DC1>	49	1	81	Q	113	q	145	ë	177	±	209	—	241	Ò
18	<DC2>	50	2	82	R	114	r	146	í	178	≤	210	“	242	Ú
19	<DC3>	51	3	83	S	115	s	147	ì	179	≥	211	”	243	Û
20	<DC4>	52	4	84	T	116	t	148	î	180	¥	212	‘	244	Ù
21	<NAK>	53	5	85	U	117	u	149	ï	181	µ	213	’	245	ı
22	<SYN	54	6	86	V	118	v	150	ñ	182	∂	214	÷	246	ˆ
23	<ETB>	55	7	87	W	119	w	151	ó	183	∑	215	◊	247	˜
24	<CAN>	56	8	88	X	120	x	152	ò	184	∏	216	ÿ	248	¯
25	<EM>	57	9	89	Y	121	y	153	ô	185	π	217	Ÿ	249	˘
26	<SUB>	58	:	90	Z	122	z	154	ö	186	∫	218	⁄	250	˙
27	<ESC>	59	;	91	[	123	{	155	õ	187	ª	219	€	251	°
28	<FS>	60	<	92	\	124	|	156	ú	188	º	220	‹	252	¸
29	<GS>	61	=	93	]	125	}	157	ù	189	Ω	221	›	253	˝
30	<RS>	62	>	94	^	126	~	158	û	190	æ	222	fi	254	˛
31	<US>	63	?	95	_	127	<DEL>	159	ü	191	ø	223	fl	255	ˇ

使用须知

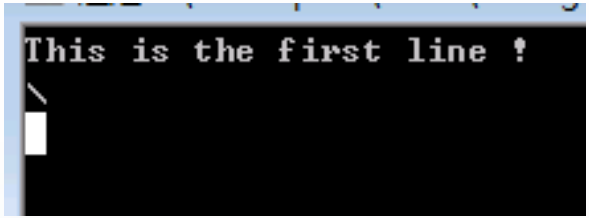
```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
        char a = 64;
        int b = 'Z';
        int c = b - a;
        char d = 6 + 256;
        cout << a << " " << b << " " << c << " " << d << endl;
        return 0;
}
```

- 由于存储类型和整型相同
  - 可以与整型数据相互赋值
  - 可以和整数一样进行运算



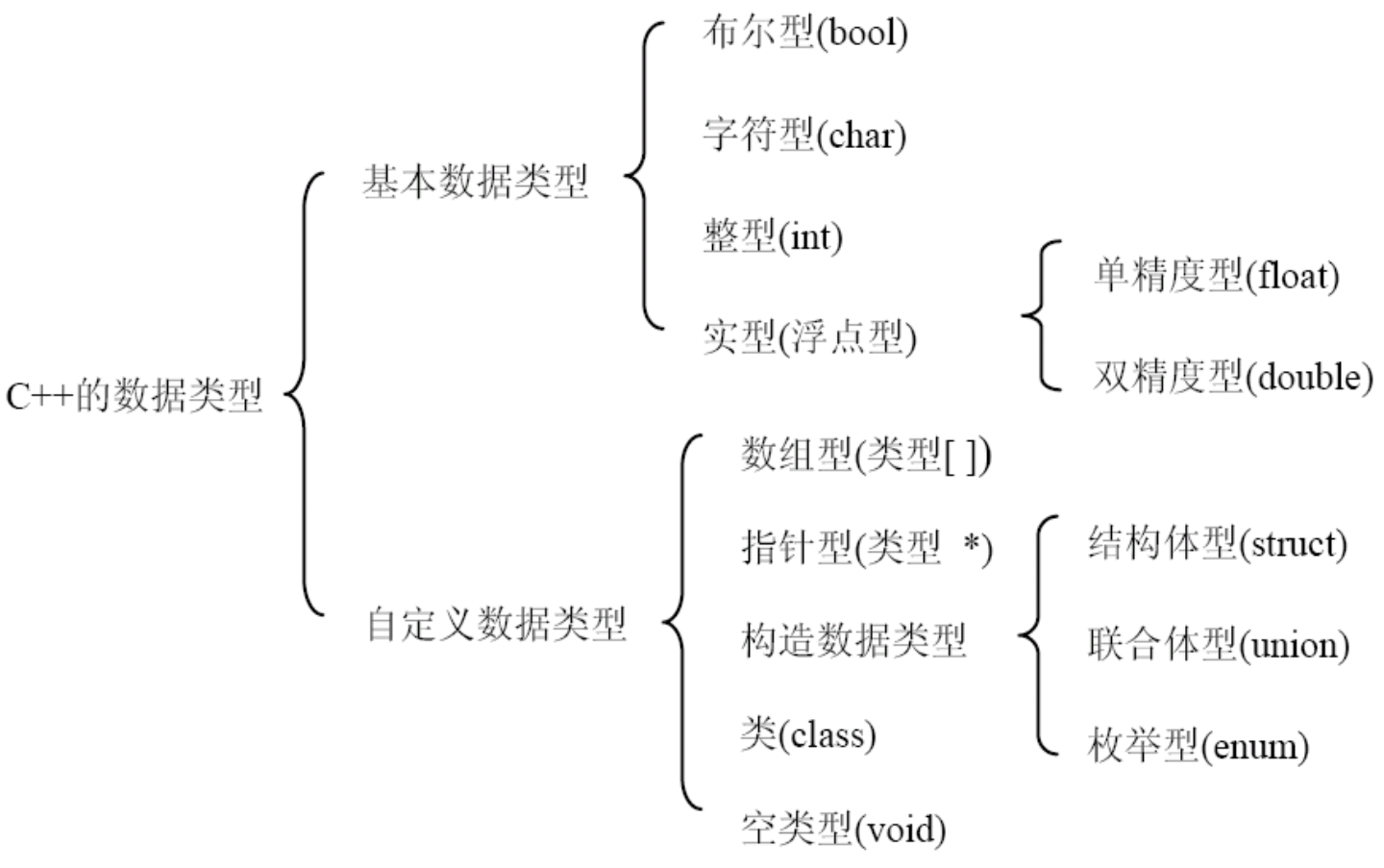
使用须知

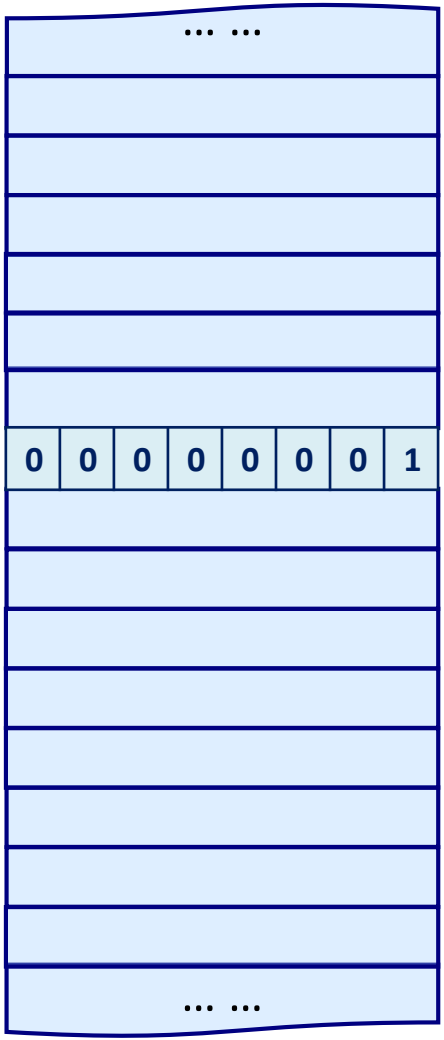
```
#include <iostream>
using namespace std;
int main()
{
        cout << "This is the first line !  \n";
        cout << '\a' << '\\' << '\n';
        return 0;
}
```



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)

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)

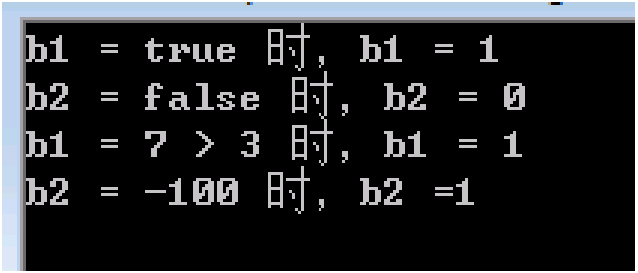


布尔型

- 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

```
#include <iostream>
using namespace std;
int main()
{
    bool b1 = true, b2 = false;
    cout << "b1 = true 时，b1 = " << b1 << endl;
    cout << "b2 = false 时，b2 = " << b2 << endl;
    b1 = 7 > 3;
    b2 = -100;
    cout << "b1 = 7 > 3 时，b1 = " << b1 << endl;
    cout << "b2 = -100 时，b2 =" << b2 << endl;
    return 0;
}
```



C/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

所有“数”都是有类型的

——常量也一样！

C/C++中的常量

- 常量：在程序运行过程中，其值保持不变的量
  - 字面常量
    - - 1，0，123，4.6，- 1.23；
  - 符号常量
    - 用一个标识符代表一个常量的，称为符号常量

```
#include <iostream>
using namespace std;
int main()
{
    const float PI = 3.14159f;
    float  r, area;
    cin >> r;
    area = r * r * PI;
    cout << "Area = " << area;
     return 0;
}
```

常量有类型吗？

- 整型常量的后缀
  - n = 10000L; //长整型常量
  - m = -0x88abL; //长整型十六进制常量
  - k = 10000U; //无符号整型常量
  - i = 07777LU; //无符号长整型八进制常量
- 浮点型常量的后缀
  - x = 3.1415F //单精度浮点型常量
  - y = 3.1415L //长双精度浮点型常量
- 说明：
  - 浮点型常量默认为double 型；
  - U, L, F均可以小写；

变量定义“须知”

关于C/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

C++ 语言的 保留字

- C++环境下的 63个 保留字：

asm	do	if	return	typedef
auto	double	inline	short	typeid
bool	dynamic_cast	int	signed	typename
break	else	long	sizeof	union
case	enum	mutable	static	unsigned
catch	explicit	namespace	static_cast	using
char	export	new	struct	virtual
class	extern	operator	switch	void
const	false	private	template	volatile
const_cast	float	protected	this	wchar_t
continue	for	public	throw	while
default	friend	register	true	
delete	goto	reinterpret_cast	try	

关于变量的命名

- 匈牙利命名法

  由Microsoft著名开发人员Excel主要设计者Charles Simonyi在其博士论文中提出。

1. 以一个或者多个小写字母开头，来指定数据类型。
2. 其后是一个或者多个第一个字母大写的单词，指出变量的用途。

如：chGrade; nLength; bOnOff; strStudentName;

关于变量的命名

- 驼峰命名法（骆驼式命名法）
  - 由一个或多个单字连结在一起；
  - 第一个单词以小写字母开始；
  - 第二个单词的首字母大写或每一个单词的首字母都采用大写字母；

myFirstName, myLastName, nextStudentName,

intCount, printEmployeePaychecks()