高级语言程序设计

南京邮电大学计算机学院

教师：窦 轶

高级语言程序设计

第2章　初识C源程序及其数据类型

单选题 10分

设置

Nanjing University of Posts and Telecommunications

下列哪一个是C语言的发明者?

A 肯·汤普森

B 冯·诺依曼

C 丹尼斯·里奇

D 艾伦·佩利

提交

设置

华为公司自主研发的手机操作系统的名字是?

A 麒麟

B 鸿蒙

C 安卓

D iOS

提交

设置

十进制数348转换为八进制数是_____

A 534

B 435

C 345

D 354

提交

设置

二进制数1001011110010 11对应的八进制数是___

A 22703

B 22713

C 12713

D 22773

提交

投票(匿名) 最多可选2项

设置

Nanjing University of Posts and Telecommunications

你对我们慕课的初步印象？

A 辅助课前预习课后复习，对学习很有帮助

B 老师颜值高，讲课好，大爱

C 感觉一般般，不过有比没有强

D 完全没有必要存在，我觉得听课就很好

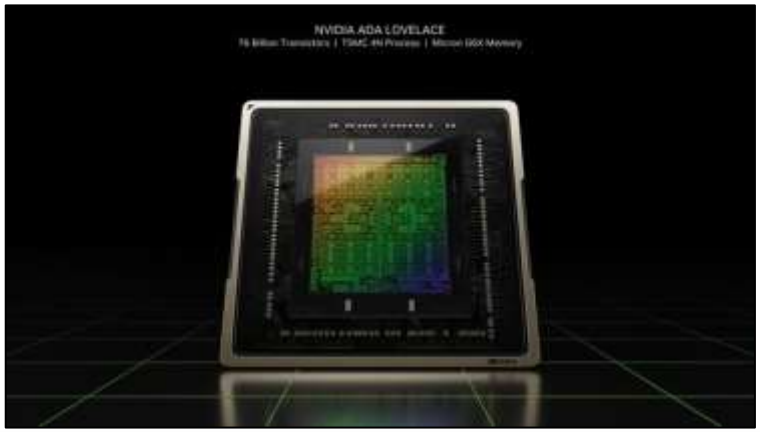
提交

世界上第一位程序员
Ada Lovelace

英国浪漫主义诗人拜伦的女儿，数学家。穿孔机程序创始人，建立了循环和子程序概念。为计算程序拟定“算法”，写作的第一份“程序设计流程图”。

为纪念ADA，1981年美国国防部指定唯一可用于军用系统开发的语言命名为“ADA语言”

世界上第一位程序员 Ada Lovelace



1842年到1843年花了9个月时间翻译了Babbage的《分析机概论》的备忘录，写了很多注记，其中给出了用计算机进行Bernoulli数求解的详细说明。由此，Ada被广泛认为是世界上第一个程序员。

2022年9月29日，NVIDIA推出Ada Lovelace架构显卡

内容提要

- C语言源程序的组成结构及6种基本符号
- C语言基本数据类型常量的表示方法
- C语言基本数据类型变量的定义、初始化和输入/输出方法

预测未来的最好办法，就是把它创造出来。
The best way to predict the future is to invent it.
——艾伦·凯（Alan Kay），图灵奖得主

2.1 C源程序及其符号

- C程序由一个或多个函数组成，有且只有一个main( )函数。

```
#include <stdio.h> //头文件
int main( ) //每个C程序有且只有一个main( )函数
{   //函数体以{开头
    printf( “Hello World.\n” ); //打印语句
    return 0;   //返回0表示程序运行正常
}   //函数体以}结尾
```

- 程序从 main( )的第一句开始执行，到它的最后一句结束。
- main()函数是C程序的执行总纲。

2.1 C源程序及其符号

- C源程序的组成（例2_1）

```
#include <stdio.h>
/*函数功能：计算两个整数的乘积
  入口参数：整型数a和b
  返回值：  整型数a和b之积
*/
int multiply( int a, int b)
{
   return (a*b);
}
```

函数首部

函数体

2.1 C源程序及其符号

函数体

```
/*主函数*/
int main( )
{  int x, y, product;
   printf("Please input two integers:");
   scanf("%d%d", &x, &y);
   /*输入两个整型数x和y*/
   product=multiply(x, y);
   /*调用函数multiply计算x和y的乘积*/
   printf("The product is %d\n", product);
     /*输出x和y的乘积*/
   return 0;
}
```

Input

Process

Output

```
Please input two
    integers:
2  3 <回车>
The product is 6
```

C语言源程序（Source Program）的基本单元 – 函数

定义

- 函数是一个过程——是指令序列执行的过程。

- 可以有输人，有输出
- 内部可以进行分支和循环
- 甚至可以嵌套调用其他函数

C语言函数内部不可定义其他函数。

例

- 做饭是一个过程
  - 输入各种食材，输出各种菜品
  - 其中可以根据条件选择合适方案
  - 其中嵌套有淘米作为子过程。

C语言源程序（Source Program）的基本组成结构

(1) 函数（Function）是C语言源程序的基本单位，即C程序是由函数构成的

- 主函数（main Function）
  - 一个C程序有且只有一个名为main的函数
- 用户自定义函数（User-Defined Function）
  - 如：multiply函数
- 库函数（Library Function）
  - 如：scanf和printf函数
  - 需要编译预处理命令（Preprocessor Directive）

    如：#include <stdio.h>

C语言源程序（Source Program）的基本组成结构

(2) 函数由函数首部（Function Header）和函数体（Function Body）两部分构成

```
/*函数首部*/
<函数返回类型>  <函数名>( <形式参数表> )
{
    <若干语句>          /*函数体*/
}
```

（在第5章中再详细介绍函数概念）

(3) 每条C语句都是以分号";"结束的

/*.....*/ 的形式表示注释（Comment）

设置

组成C程序的基本单位是_____

A 程序

B 函数

C 过程

D 调用

提交

投票 最多可选2项

设置

函数由_____和_____两部分组成

A 函数首部、函数体

B 函数头、函数体

C 函数头、函数尾

D 函数首部、函数主体

提交

C源程序中的6种基本符号

(1) 关键字（Keyword）,又称为保留字

是C语言中预先规定的具有固定功能和意义的单词。C语言提供了32个关键字，详见附录B。

(2) 标识符（Identifier）

以字母或下划线开头，后面跟字母、数字、下划线的任意字符序列。(注意：大小写敏感)

(3) 运算符（Operator）

C语言提供了34个运算符，详见附录D。

C源程序中的6种基本符号

(1) 关键字（Keyword）,又称为保留字

是C语言中预先规定的具有固定功能和意义的单词。C语言提供了32个关键字，详见附录B。

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	union
unsigned	void	volatile	while			

C源程序中的6种基本符号

(2) 标识符

以字母或下划线开头，后面跟字母、数字、下划线的任意字符序列。

系统预定义标识符：如：scanf　printf
有相对固定的含义，一般不作他用

用户自定义标识符：由用户根据编程需要自己定义的标识符

C源程序中的6种基本符号

用户自定义标识符有自己的命名规则：

- 由字母、数字、下划线构成
- 必须以字母或下划线开头
- 大小写敏感
- 不可以使用32个关键字

例如：_list, li3_1, C_programing　　合法
　　li3.1 , C++, int　　不合法

设置

下列哪一个是正确的用户自定义标识符_____

A 1A2B 必须以字母或下划线开头

B _first_end

C OK&right 由字母、数字、下划线构成

D int 不能使用关键字

提交

C源程序中的6种基本符号

(3)运算符（Operator）

C语言提供了34个运算符，详见附录D。

&: 取地址运算符，&a表示变量a的地址

(4) 分隔符（Separator）

空格、回车、逗号、分号

同一关键字、标识符中不能出现分隔符；相邻关键字、标识符之间必须使用分隔符；逗号用于分隔一条语句中的并列项

C源程序中的6种基本符号

(5) 其它符号

{ }、/* */、 //

(6) 数据（Data）

数据分为变量和常量；变量以用户自定义标识符的形式出现；

各种数据类型的字面值常量

问题：请找出例2_1中的6种符号

2.1 小结

函数是C语言的基本组成单位，一个C程序有且仅有一个main函数。

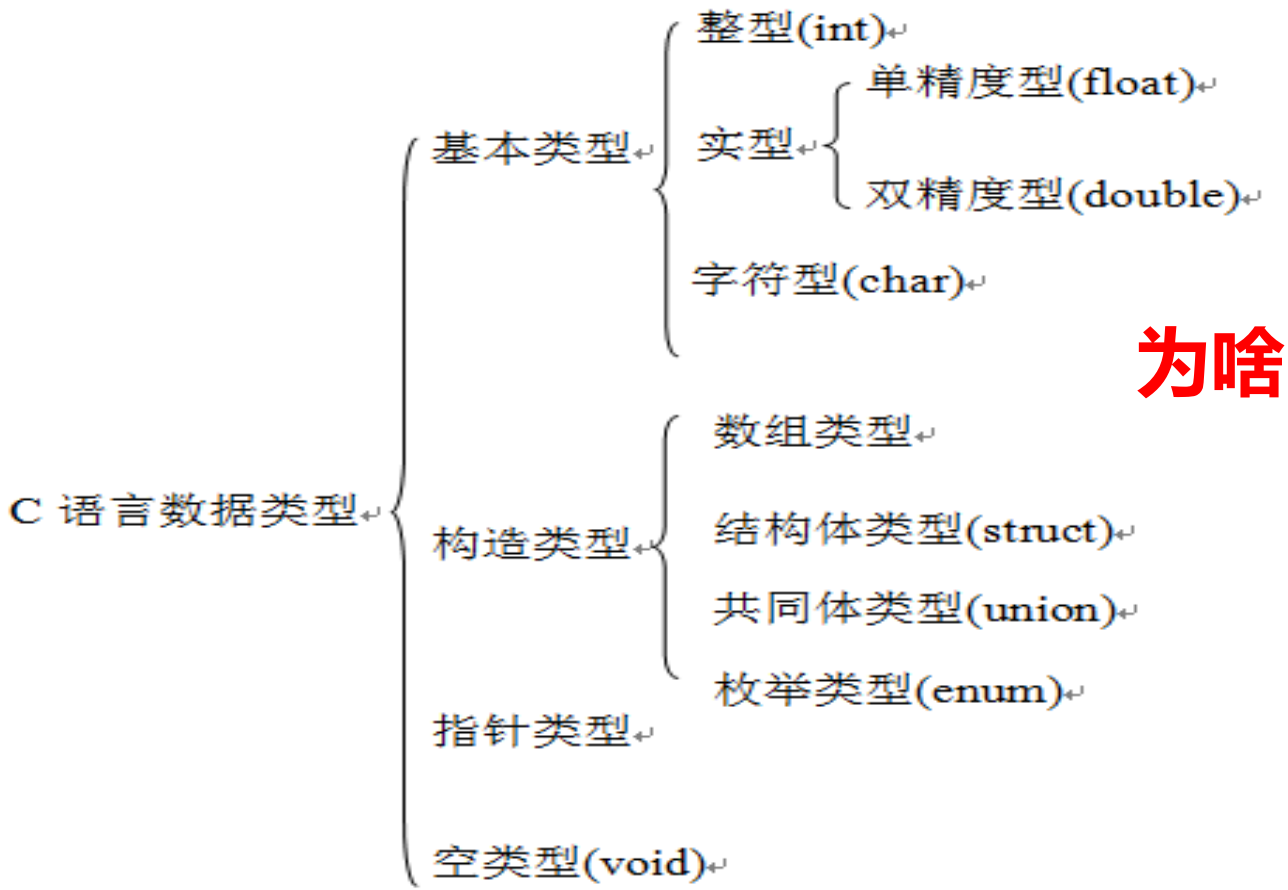
C源程序中的6中基本符号：关键字、标识符、运算符、分隔符、其他符号、数据。

用户自定义标识符的命名规则，这是编程的一项基础工作。

内容提要

- C语言源程序的组成结构及6种基本符号
- C语言基本数据类型常量的表示方法
- C语言基本数据类型变量的定义、初始化和输入/输出方法

2.2 C语言中的数据类型



为啥要有数据类型？

数据类型背后的原理

- 高等代数或者近世代数里，群、环、域的元素可以无穷多。
- 但是，计算机任何一个数据类型能表达的数都是有穷多的

计算机的硬件特点决定了它的处理方式

1. 计算机只能够处理离散而且有穷的问题
2. 人脑能够处理连续、无穷的问题
3. 数学是人脑在解决问题，编程是计算机在解决问题
4. 上述差别决定了计算机用各种范围和精度来表达数据

☞ 计算机的草稿纸是方格纸，要进行计算就要分配一定的空间。

数据类型在哪里？

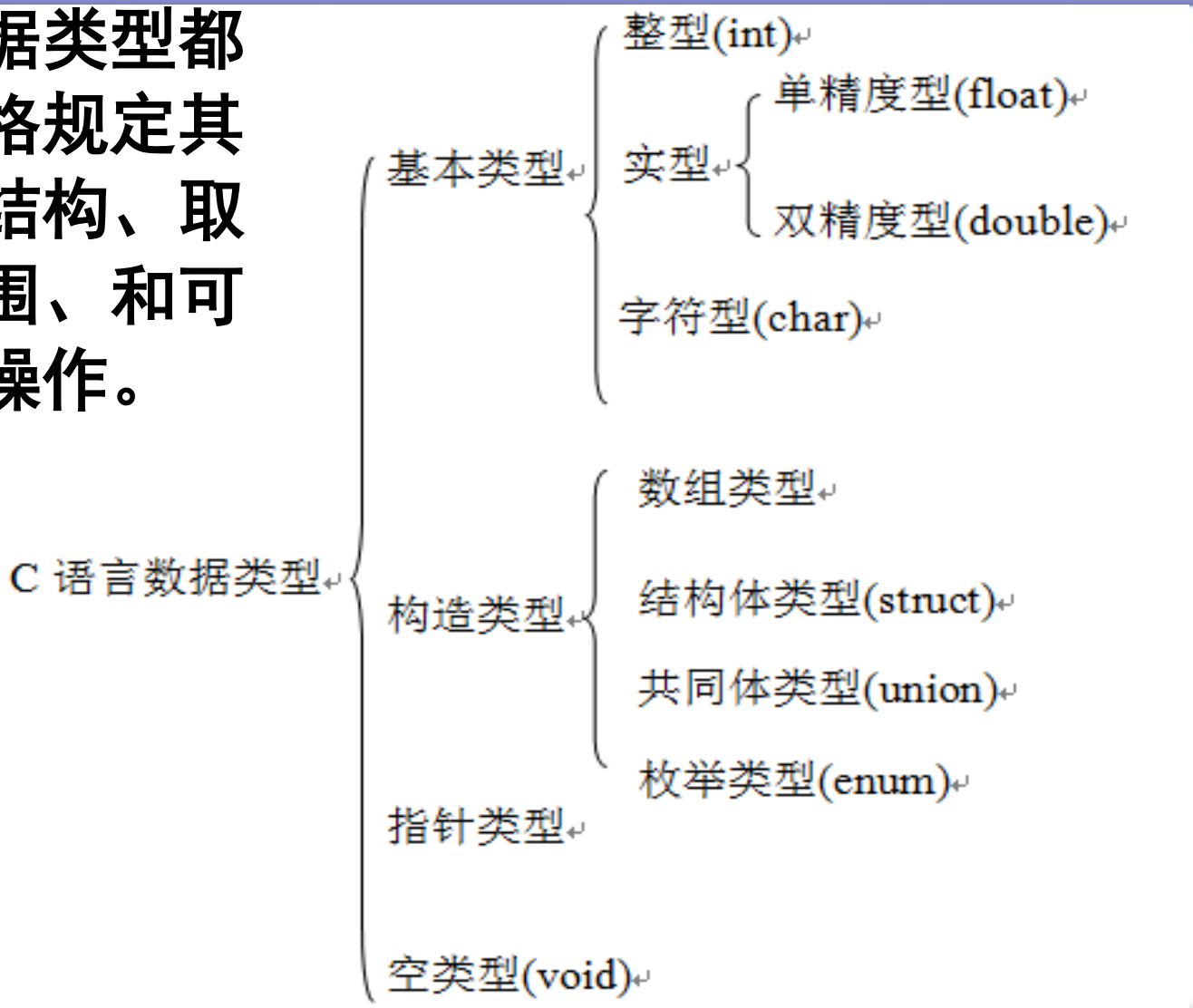
- C语言变量的数据类型是明确声明的，而且声明之后不能改变。

常见数据类型

- 整型：形如0，1，-1，2，-2，.....
- 长整型：表达的范围比整型大很多
- 浮点型（单精度）：形如0.0，-1.3，2.0（不保证精确）
- 双精度：表达的精度比浮点型大很多
- 字符型：形如‘,’，‘8’，‘j‘（字符以单引号夹住）

2.2 C语言中的数据类型

- 各数据类型都要严格规定其存储结构、取值范围、和可进行操作。



基本数据类型及其修饰符

- 基本数据类型
  - 整型（int）
  - 实型：单精度型（float）、双精度型（double）
  - 字符型（char）
- 修饰符（Modifier）
  - 短的（short）、长的（long）
  - 有符号的（signed）、无符号的（unsigned）

详见表2_1

- sizeof(数据类型)
  - 获得该数据类型所占内存的字节数

数据类型之间的比较

sizeof( ): 获得该数据类型所占内存的字节数

```
① printf(“char: %d bytes.", sizeof(char));
② printf("int : %d bytes. ", sizeof(int));
③ printf("long: %d bytes.", sizeof(long));
```

- 变量的表达能力越强，占用字节就越多。
- 不要盲目追求表达能力，因为表达能越力强，占用内存越多。

表 2_1 Visual Studio 2010 编译环境下，常用的带修饰符的基本数据类型的汇总表

简称的类型名	完整类型名	长度（字节）	取值范围
char	signed char	1	-128～127
unsigned char	unsigned char	1	0～255
short	signed short int	2	-32768～32767
unsigned short	unsigned short int	2	0～65535
int	signed int	4	-2147483648～2147483647
unsigned int	unsigned int	4	0～4294967295
long	signed long int	4	-2147483648～2147483647
unsigned long	unsigned long int	4	0～4294967295
float	float	4	绝对值：3.4×10^{-38}～3.4×10^{38}
double	double	8	绝对值：1.7×10^{-308}～1.7×10^{308}
long double	long double	8	绝对值：1.7×10^{-308}～1.7×10^{308}

数据类型之间的比较

➢ 不同数据类型，表示方式可能不同，占用字节数也可能不同。

常见数据类型占用的内存

- short型字节数不超过int型，int型字节数不超过1ong型
- char型一般占用1个字节
- float型不超过double型，double型不超过long double型

常见数据类型的表达范围和精度

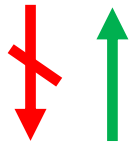
- short型的表达范围不超过int型，int型不超过1ong型
- float型的范围和精度不超过double型，double型不超过long double

数据类型之间的比较

整数类型：char型、 short型、 int型、long型。

浮点类型：float型、double型和long double型。

浮点型(float)

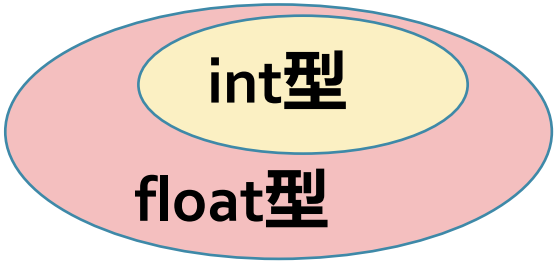


整型 (int)

安全的类型转换

整数类型中的小类型转大类型

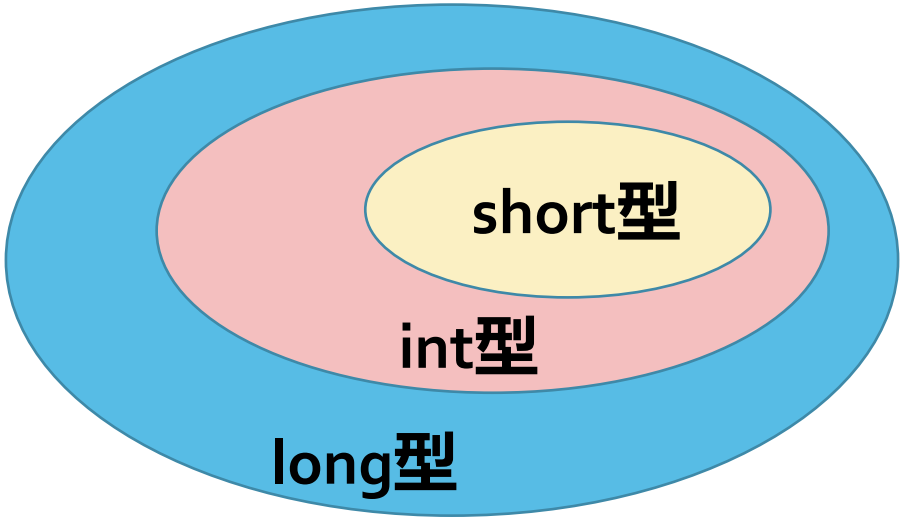
浮点类型中的小类型转大类型



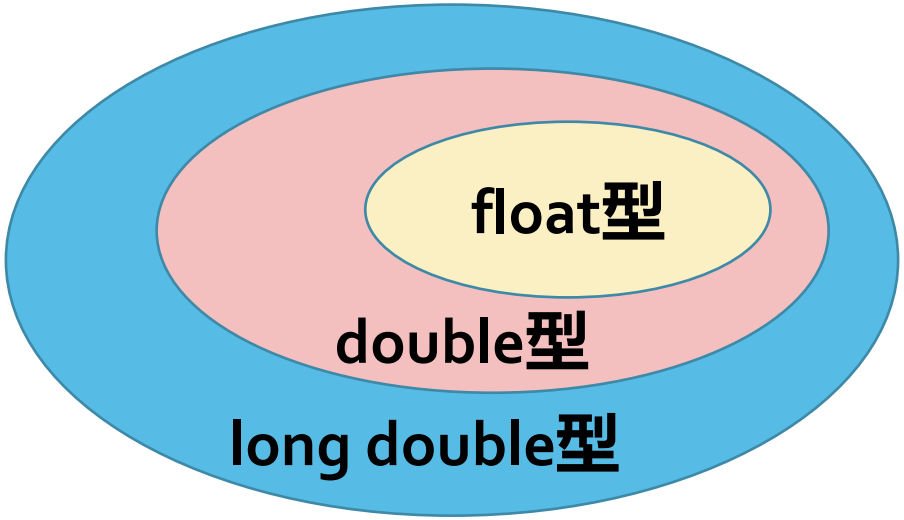
瓶里的水倒进桶里是安全的，桶里的水倒进瓶里不安全。

- 整型转浮点型:(float)10——转为10.0，不损失精度
- 浮点型转整型:(int)1.5—转为1，损失精度。

数据类型之间的比较



数据类型之间的比较



unsigned 使用前后的对比

表 2_1 Visual Studio 2010 编译环境下，常用的带修饰符的基本数据类型的汇总表

简称的类型名	完整类型名	长度（字节）	取值范围
char	signed char	1	-128～127
unsigned char	unsigned char	1	0～255
short	signed short int	2	-32768～32767
unsigned short	unsigned short int	2	0～65535
int	signed int	4	-2147483648～2147483647
unsigned int	unsigned int	4	0～4294967295
long	signed long int	4	-2147483648～2147483647
unsigned long	unsigned long int	4	0～4294967295
float	float	4	绝对值：3.4×10^{-38}～3.4×10^{38}
double	double	8	绝对值：1.7×10^{-308}～1.7×10^{308}
long double	long double	8	绝对值：1.7×10^{-308}～1.7×10^{308}

unsigned 使用前后的对比


```
#include <stdio.h>
/* 演示了有符号整数和无符号整数之间的差别*/
int main()
{
    short int i;                    // 有符号短整数
    short unsigned int   j;         // 无符号短整数
    j = 50000;
    i = j;
    printf( "i=%d,    j=%d\n" , i   ,   j   );
    return 0;
}
```

输出结果：
i=-15536,　　j=50000

50000超出了 有符号短整型 32767 的最大值范围，所以造成数据溢出

简称的类型名	完整类型名	长度（字节）	取值范围
short	signed short int	2	-32768～32767
unsigned short	unsigned short int	2	0～65535

有/无符号整数的表达范围与占用字节数的关系

有符号的整数类型(即可以表示负数的整数类型)

char, short, int, long, long long

整数类型表示范围和占用字节数的关系

type型变量的范围是:$-2^{8\times sizeof(type)-1}$~$2^{8\times sizeof(type)-1}-1$

无符号整数的表达范围与占用字节数的关系

unsigned type型变量的范围是: 0~$2^{8\times sizeof(type)}-1$

有/无符号整数的表达范围与占用字节数的关系

有穷性

任何数据类型能表达的数字的个数都是有限多的

例

1. 如果 `sizeof(int)`=4，那么，`int`型一共有 4×8 个位，
2. 形成的“0-1 排列”一共有 $2^{4\times 8}$ 个，
3. 于是`int`型最多只能表示 2^{32} 个数字。

unsigned int型范围

			0	1	……	$2^{31}-1$	……	$2^{32}-1$

unsigned 使用前后的对比

例：在一台机器上，可能是这样的：

① int型的范围是：-2^{31}~$2^{31}-1$

② unsigned int型的范图是：0～$2^{32}-1$

2的31次方：2147483648
2的32次方：4294967296

int型范围	-2^{31}	……	-1	0	1	……	$2^{31}-1$		
unsigned int型范围				0	1	……	$2^{31}-1$	……	$2^{32}-1$

表: int 与unsigned int的表达范围对比

两者的范围并无包含关系。

int型与unsigned int的转换在0～$2^{31}-1$范围内是安全的。

简称的类型名	完整类型名	长度（字节）	取值范围
int	signed int	4	-2147483648～2147483647
unsigned int	unsigned int	4	0～4294967295

若要表示出50！的结果，根据基本数据类型的取值范围你会选择哪种数据类型呢？

- A. 无符号长整型
- B. 双精度实型
- C. 字符型
- D. 都可以

小结

在学习C语言之初，首先要明白不同数据类型的含义，即了解各自的取值范围、存储结构和能对其进行的操作。

2.3 常量

- 常量（Constant）是计算机程序运行过程中其值不能发生改变的量。
  - 整型常量
  - 实型常量
  - 字符常量
  - 字符串常量
  - 符号常量

2.3 常量

常量
const

所谓的常量就是指在计算机程序运行过程中其值不会也不能发生改变的量。

比如：3就表示一个整型常量，在任何运算过程中这个数字本身是不会发生变化的。

例：圆周率，水的密度，身份证号码

2.3 常量的使用

常量

为了可读性和可维护性，常量有特殊的声明方式。

好处

① 增强可读性："440023199705062643" 非常难读

② 增强可维护性：如果换了一个人，身份证号改了，那么，只需要修改一次就可以

③ 避免数据破坏，一旦对常量进行修改，编译器会报错

2.3 常量的定义

在 C 中，有两种简单的定义常量的方式：

- 使用 #define 预处理器： #define 可以在程序中定义一个常量，它在预处理阶段会被替换为其对应的值。

  例：`#define PI 3.14`

- 使用 const 关键字：const 关键字用于声明一个只读变量，即该变量的值不能在程序运行时修改。

  例：`int const PEOPLE_NUM = 112`

符号常量

- 定义形式：#define <标识符> <字符串>
  - 如：#define PI 3.14159
- 注意：#define不是C语言语句，后面没有分号
  - #define是一条预处理命令，称为宏定义
  - 不会对<字符串>的信息做操作！！
  - 若将上句写成 #define PI 3.14159
    则PI将被替换成 3.14159

（符号常量的具体用法在第9章介绍）

符号常量

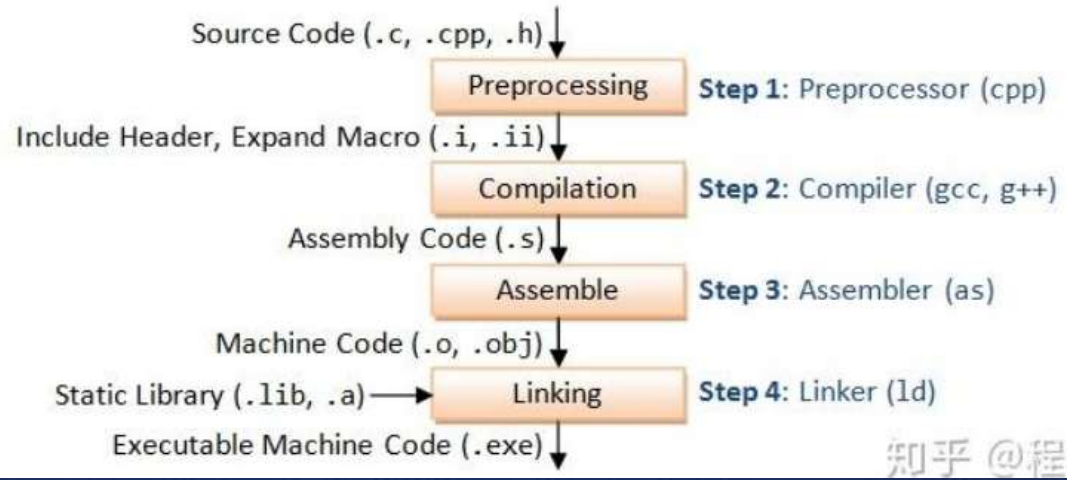
使用 #define 预处理器： `#define PI 3.14 //预处理阶段把程序中所有的 PI 替换为 3.14`

以后每次在程序里遇见 PI，就自动转为 3.14，然后再编译。

替换过程中不做任何检查。

从 C 语言文件到二进制文件的详细过程

预处理-> 编译-> 汇编 ->链接



C 语言的只读变量修饰符——const

基本类型只读变量

```
int const PEOPLE_NUM = 112;
```

或者`const int PEOPLE_NUM = 112;`

相比 define 的优缺点

① 进行类型检查，使程序更容易避免错误

② 占用内存空间，程序效率较低

③ 不可以用作静态数组的长度

整型常量

- 整型常量：十进制、八进制、十六进制数三种形式来表示

进制数	表示方式	举例
八进制整型	由数字 0 开头	034, 065, 057
十进制整型	如同数学中的数字	123, -78, 90
十六进制整型	由 0X 或 0x 开头	0x23, 0Xff, 0xac

（1）长整型常量：后缀L或l，如：125L，56l等

（2）无符号整型常量：后缀U或u，如：60U，256u等

（3）无符号长整型常量：后缀LU，Lu，lU或lu，

如：360LU

设置

下列最大的整型常量是_____

A 017 8+7 =15

B 0x17 16 +7 = 23

C 17

D 170

提交

实型常量

- 实型常量只采用十进制表示：小数形式、指数形式两种

形式	表示方式	举例
小数形式	数字 0~9 和小数点组成，数字前面可带正负号	3.14，–0.123，10.，.98
指数形式	尾数、e 或 E 和指数三部分组成，即科学计数法，其中，尾数可表示成整数或小数形式，且不能省略；指数必须是整数	3.0e8，6.8E-5，9.9e+20 等，但 e2，3e2.0 不合法 6.8E-5//表示数值 6.8 ×10^-5

3. ≠3　　.15≡0.15

尾数通常要进行归一化处理,即尾数必须大于0.1,小于1。例如:123.78,表示为浮点型应是:0.12378 × 10^3

设置

下列哪一个是正确的实型常量_____

A 234E3.1

B E3

C 234.

D 234

形式	表示方式
小数形式	数字 0~9 和小数点组成，数字前面可带正负号
指数形式	尾数、e 或 E 和指数三部分组成，即科学计数法，其中，尾数可表示成整数或小数形式，且不能省略；指数必须是整数

提交

实型常量

- 实型常量只采用十进制表示：小数形式、指数形式两种

（1）系统默认表示实型常量double类型(写3.0 或者3.)，刻意表达float型，需加后缀F或f，如：3.14f，6.8e-5F等

（2）单精度浮点型小数点后面有效数字为7位和双精度浮点型小数点后面有效数字为16位。

（3）长实型 long double类型，添加后缀为L或l

（4）在实际使用中，由于整型数的取值范围有限定，所以当取值范围不能满足要求时，一般采用实型数来表示，比如计算阶乘时，用整型就不合适了。

字符常量

- 字符常量用一对单引号将一个字符括起来表示
  - 如：‘A’，‘b’，‘5’，‘#’等
- ASCII表（参见附录A）
  - 为每个字符定义唯一的整数编码——ASCII码
  - 如：‘A’的ASCII码是65，‘5’的ASCII码是53等
- 转义字符（Escape Character）
  - 单引号括起的以反斜杠\开头的字符序列来表示
  - 如：‘\n’表示换行符，‘\a’表示响铃符，‘\t’表示制表符等（参见表2_4）

ASCII 码表

☞ 每一个字符都对应到一个正整数，然后被计算机存储和操作。

ASCII 码表

把每个字符映射为整数，如'a' 映射为 97，'A' 映射为 65。

字符	ASCII 码	字符	ASCII 码	字符	ASCII 码
0	48	A	65	d	100
1	49	B	66	!	33
2	50	C	67	”	34
3	51	D	68	#	35
4	52	a	97	$	36
5	53	b	98	+	75
6	54	c	99	/	79

表: ASCII 码表的片段

字符常量

表2.4 常用的转义字符

字符	含　义	字符	含　义
'\n'	换行（Newline）	'\a'	响铃报警（Alert or Bell）
'\r'	回车（不换行）（Carriage Return）	'\"'	一个双引号（Double Quotation Mark）
'\0'	空字符（Null）	'\''	单引号（Single Quotation Mark）
'\t'	水平制表（Horizontal Tabulation）	'\\'	一个反斜线（Backslash）
'\v'	垂直制表（Vertical Tabulation）	'\?'	问号（Question Mark）
'\b'	退格（Backspace）	'\ddd'	1到3位八进制ASCII码值所代表的字符
'\f'	走纸换页（Form Feed）	'\xhh'	1到2位十六进制ASCII码值所代表的字符

字符常量

- 字符常量‘5’与整型常量5之间的区别与联系
  - 区别：5表示一个整数，而‘5’表示一个字符
  - 联系：‘5’的ASCII码为整数53，即：字符类型本质上也是整型！
- 数字与数字字符之间的转换
  - 如：’5’-48 => 5，5+48 =>’5’
- 大小写字母之间的转换
  - 如：‘A’+32=>‘a’，‘a’-32=>‘A’

设置

下列哪一个字符与其他3个字符不相等_____

'\ddd'	1到3位八进制ASCII码值所代表的字符
'\xhh'	1到2位十六进制ASCII码值所代表的字符

A ‘a’

B ‘A’

C ‘\x41’ ‘\x41=4*16 + 1 =65

D ‘\101’ ‘\101’ = 8^2 + 1 =65

提交

设置

下列哪一个不是正确的字符常量____

'\ddd'	1 到 3 位八进制 ASCII 码值所代表的字符
'\xhh'	1 到 2 位十六进制 ASCII 码值所代表的字符

A ‘\n’

B ‘x’

C ‘\108’

D ‘\x43’

提交

字符串常量

- 用一对双引号将零个或多个字符序列括起来
  - 如：“hello”，“This is a program”，“A”等
- 每一个用双引号括起来的字符串常量的末尾都添加一个空字符‘\0’作为结束标记

"hello"	h	e	l	l	o	\0
"A"	A	\0				

- 字符串常量的实际所占字节数总是比其双引号中的字符数多1

单选题 10分

设置

Nanjing University of Posts and Telecommunications

sizeof(“hello!”)的值是_____

A 6

B 7

C 8

D 9

提交

小结

本节介绍C语言中常用的常量表示方法，正确使用常量是代码的基本要求，注意和以前数学、物理学科中表达方式的区别。

- 特别是字符型常量，并不需要背下整张ASCII编码表，但建议各位同学记住表中三个标记位置：

内容提要

- C语言源程序的组成结构及6种基本符号
- C语言基本数据类型常量的表示方法
- C语言基本数据类型变量的定义、初始化和输入/输出方法

2.4 变量

- 变量（Variable）是计算机程序运行过程中其值可以发生改变的量，用于存放程序运行时输入、处理、输出过程中所涉及的各种数据
- 数据类型、变量名两部分
  - 变量的定义及初始化
  - 变量的输入、输出
  - 用const限定变量

变量的定义及初始化

- “先定义、后使用”原则
- 变量定义（Variable Definition）的格式
  - <数据类型> <变量名1> [, <变量名2> , <变量名3> , ...] ;
- 变量定义语句的例子：
  - int a; /*表示定义了一个整型变量a*/
  - double x, y, z; /*表示定义了三个双精度实型变量，分别为x, y, z*/
  - char c; /*表示定义了一个字符型变量c*/
  - long total; /*表示定义了一个长整型变量total，等价于long int total; */

推荐的命名法之一——驼峰命名法

小驼峰法

当变量名或函数名是由一个或多个单词连结在一起，而构成的唯一识别字时，第一个单词以小写字母开始；从第二个单词开始以后的每个单词的首字母都采用大写字母。

常用于变量、函数和方法的命名：myFirstName, myLastName,studentID

大驼峰法

相比小驼峰法，大驼峰法把第一个单词的首字母也大写了。

常用于于类名：DataBaseUser

推荐的命名法之二——下划线命名法

下划线命名法

用于命名变量、函数和方法，与小驼峰法的区别在于，单词之间全部用下划线连接，不使用大写字母。

例如：my_first_name, my_last_name, student_id

在多数情况下，可读性比程序效率更重要。

变量的定义及初始化

在变量名定义之后，系统根据其类型为变量分配了一定大小的内存空间，该内存空间中是随机数，要使变量有确切值，可以通过下列3种方式之一：

初始化　　赋值　　读入

变量的定义及初始化

- 在定义变量的同时给出初值，称为变量的初始化
- 格式如下：
  - <数据类型> <变量名1>=<初值1> [, <变量名2>=<初值2> , …] ;
  - 如：int a=1; /*定义整型变量a，初值为1*/
  - double x=1.23, y=2.236068; /*定义双精度实型变量x,y，x的初值为1.23, y的初值为2.236068 */
  - char d='T'+32; /*定义字符型变量d，初值为小写字母't'*/
  - double z=x; /*定义变量z，初值为x的值，这里是1.23*/
- 在变量定义及初始化后，可用赋值运算修改变量的值
  - 如：z=3.6; /*将变量z的值改为3.6*/

设置

下列变量定义及初始化正确的是_____

A int x=2,y=x;

B int x,y=x*2+5;

C int 3=x;

D int x,y; double x;

提交

变量的定义及初始化

下列程序段正确的是_____。

A.
```
double a,h,s;
s=1/2.0*a*h;
a=3.2;
h=1.5;
printf("三角形面积s=%f", s);
```

B.
```
s=1/2.0*a*h;
double a=3.2,h=1.5;
printf("三角形面积s=%f", s);
```

C.
```
double a=3.2, s, h=1.5;
s=1/2.0*a*h;
printf("三角形面积s=%f", s);
```

D.
```
double a=3.2, s, h=1.5;
s=1/2*a*h;
printf("三角形面积s=%f", s);
```

声明变量

在计算机中

① 上述第一步，就是声明变量

② 第二步，是给变量赋予初始值

```
int a;
a =12222;
```

例

当我们要在家里存储一些水、酒、油的时候，我们要

① 准备一些容器，比如水杯、水桶、水池、酒瓶、油罐

② 往里面装入酒、水或者油

注意到不同容器占用空间的大小各不相同

容器可以不装满，但一定不能溢出。

声明变量

南京邮电大学 Nanjing University of Posts and Telecommunications

例

```
int a;  // 为变量 a 分配了一片内存空间，通常为 4 个字节
a =12222;
```

① 只要 a 这个变量还生效，这片存储空间就会一直被占用。

② 这片空间的最低单元编号，也叫做 a 的地址。

③ 以后每次提及 a，就是提及这片内存。

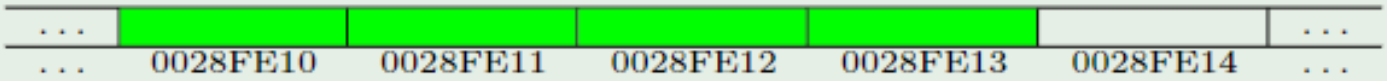


表: 变量 a 占据的内存

如图，变量 a 占据 4 个字节，地址为 0028FE10。

变量在内存中的物理形式：地址

每一个变量都占据一片固定的连续内存空间。一旦声明，立刻分配，并且所分配的空间不能再用于存储其他变量。

每个变量都占用计算机草稿纸上一定数量的方格。

定义

如果一个变量占据多个单元，则取各个单元中的最低编号作为该变量的地址值。

变量的地址，从生效之后到失效之前，不能改变。(内存硬盘换页和cache交换的情况除外。)

变量相当于容器，地址相当于容器所在的位置

例

- 变量相当于一个水杯，水杯要放在桌面上(地址)。

获取变量的地址

形式

& 变量名

例

```
int a = 0;
printf("%p", &a); //地址的输出形式为%p
```

☞ 地址值以十六进制数表示

例

```
int a = 2; float s = 1.5; char ch = 'F';
// 以下输出的地址各不相同
printf("address of a: %p\n", &a);
printf("address of s: %p\n", &s);
printf("address of ch: %p\n", &ch);
```

```
address of ch: 0x7ffd65a5e8b3, sizeof(ch): 1
address of s: 0x7ffd65a5e8b4, sizeof(s): 4
address of a: 0x7ffd65a5e8b8, sizeof(a): 4
```

变量的输入和输出——scanf

- 按照指定格式，从键盘读入若干数据给变量
- scanf(<格式控制字符串>, <变量地址列表>);

<格式控制字符串>：格式转换说明符、输入分隔符

<变量地址列表>：若干输入变量的地址

- 例如：
  - scanf("%d%d", &x, &y);

默认情况下，使用间隔符

从键盘输入：3 ␣ 4 <回车>

  - scanf("%d,%d", &x, &y);

从键盘输入：3，4 <回车>

设置

scanf(“x=%d,y=%d”,&x,&y); 如果欲从键盘上读入1和2分别给x和y变量，则正确的输入是__

A 1 2<回车>

B 1，2<回车>

C x=1,y=2 <回车>

D x=1 , y=2 <回车>

提交

变量的输入和输出——scanf

- 函数scanf的格式转换说明符

格式转换说明符	用 法
%d 或%i	输入十进制整数
%o	输入八进制整数
%x	输入十六进制整数
%c	输出一个字符，空白字符（包括空格、回车、制表符）也作为有效字符输入
%s	输入字符串，遇到第一个空白字符时结束
%f 或%e	输入一个 float 型的实数，以小数形式或指数形式输入均可
%lf	输入一个 double 型的实数，以小数形式或指数形式输入均可
%%	输入一个百分号%

变量的输入和输出——scanf

- 函数scanf的格式修饰符

格式修饰符	用 法
英文字母 l	加在格式符 d，i，o，x，u 之前用于输入 long 型数据 加在格式符 f，e 之前用于输入 double 型数据
英文字母 L	加在格式符 f，e 之前用于输入 long double 型数据
英文字母 h	加在格式符 d，i，o，x 之前用于输入 short 型数据
域宽 m	指定输入数据的宽度（列数），系统自动按此宽度截取输入数据
忽略输入*	表示对应的输入项在读入后将不传送给相应的变量

- scanf("%2d%3d%4d",&a,&b,&c);

若从键盘输入：1234567890<回车>
则输入结果：a=12，b=345，c=6789

变量的输入和输出——printf

- 格式输出函数printf的用法
  - printf(<格式控制字符串>, <输出参数表>);

<格式控制字符串>：格式转换说明符、普通字符

<输出参数表>：需要输出的数据项的列表

- 例如：
  - printf("The product is %d\n", product);

若此时product的值为6，则输出结果为：
The product is 6 <换行>

变量的输入和输出——printf

- 函数printf的格式转换说明符

格式转换说明符	用 法
%d 或%i	输出带符号的十进制整数，正数的符号省略
%u	以无符号的十进制整数形式输出
%o	以无符号的八进制整数形式输出，不输出前导符 0
%x	以无符号的十六进制整数（小写）形式输出，不输出前导符 0x
%c	输出一个字符
%s	输出字符串
%f	以十进制小数形式输出实数（包括单、双精度），隐含输出 6 位小数，
	输出的数字并非全部是有效数字，单精度实数的有效位数一般为 7 位，双精度实数的有效位数一般为 16 位
%e	以指数形式（小写 e 表示指数部分）输出实数，要求小数点前必须有且仅有 1 位非零数字
%g	自动选取 f 或 e 格式中输出宽度较小的一种使用，且不输出无意义的 0
%%	显示百分号%

变量的输入和输出——printf

- 例如：
  - printf("%f, %f", 3.14, 3.14159265);

    输出结果为：3.140000, 3.141593
  - int a=127;

    printf("%d, %o, %x", a, a, a);

    输出结果为：127, 177, 7f
  - char ch='A';

    printf("%c, %d", ch, ch);

    输出结果为：A, 65

变量的输入和输出——printf

- 函数printf的格式修饰符

格式修饰符	用　法
英文字母 l	修饰格式符 d、i、o、x、u 时，用于输出 long 型数据
英文字母 L	修饰格式符 f、e、g 时，用于输出 long double 型数据
最小域宽 m（整数）	指定输出项输出时所占的总列数。若 m 为正整数，当输出数据的实际宽度小于 m 时，在域内向右靠齐，左边多余位补空格；当输出数据的实际宽度大于 m 时，按实际宽度全部输出；若 m 有前导符 0，则左边多余位补 0。若 m 为负整数，在域内向左靠齐，右边多余位补空格
显示精度.n（大于等于0的整数）	精度修饰符位于最小域宽修饰符之后，则一个圆点及其后的整数构成。对于浮点数，用于指定输出的浮点数的小数位数；对于字符串，用于指定从字符串左侧开始截取的子串字符个数
-	有-表示左对齐输出，如省略表示右对齐输出。

变量的输入和输出——printf

- 例如：
  - printf("%.10f,%.10f",3.14159265358979 3,
    3.14159265358979**f**);

输出结果为： 3.1415926536, 3.1415927410

单精度浮点型小数点后面有效数字为7位。

- 例2_2 日期格式转换

日期格式分类	日期格式	示　例
标准	YYYY-MM-DD	2018-9-10
中国	YYYY年MM月DD日	2018年9月10日
美国	MM/DD/YYYY	9/10/2018
英国	DD/MM/YYYY	10/9/2018

- 在VS2010环境下运行该程序并分析运行结果

变量的输入和输出

- 例2_2日期格式转换

思考题1：若日期输入格式为YYYYMMDD，且严格按4位年、2位月和2位日的宽度进行输入，不足宽度的需在前面补0。程序应如何修改？

思考题2：若输入scanf语句中的变量前忘写了取地址符&，程序运行结果会怎样？

变量的输入和输出——getchar、putchar

- 字符型变量的输入和输出函数
  - <变量>= getchar( );
  - putchar(<参数>);
- 例2_3 作业等级的输入和输出
  - 在VS2010环境下运行该程序并分析运行结果

思考题：若用户从键盘输入ABC〈回车〉，上例的运行结果会怎样？

用const修饰符限定变量

- 只读变量（Read-Only Variable）
  - const <数据类型> <只读变量名1>=<值1>
    [,<只读变量名2>=<值2>,...];

定义时必须给定初始值，在后面的程序中将不能被修改

- 例2_4 计算圆的面积和周长

```
const double pi=3.14159;      /*定义只读常量pi*/
```

思考题：在程序中增加一条修改语句：
pi=3.14 ;并再次编译程序，观察编译器会给出什么错误提示信息？

小结

变量的定义方法

变量获得值的三种途径：初始化、输入、赋值

介绍了两组输入输出函数的使用，格式化输入输出和字符的输入输出

*2.5 基本数据类型在计算机内部的表示

基本数据类型在计算机内部的表示

- 整型数据在内存中的存储形式
- 字符型数据在内存中的存储
- 实型数据在内存中的存储

整型数据在内存中的存储形式

- 整型数据可分为基本整型（int）、短整型（short）和长整型（long）

区别在于占用内存空间的多少

在VS2010环境下，	
int、long	4个字节，
short	2个字节。

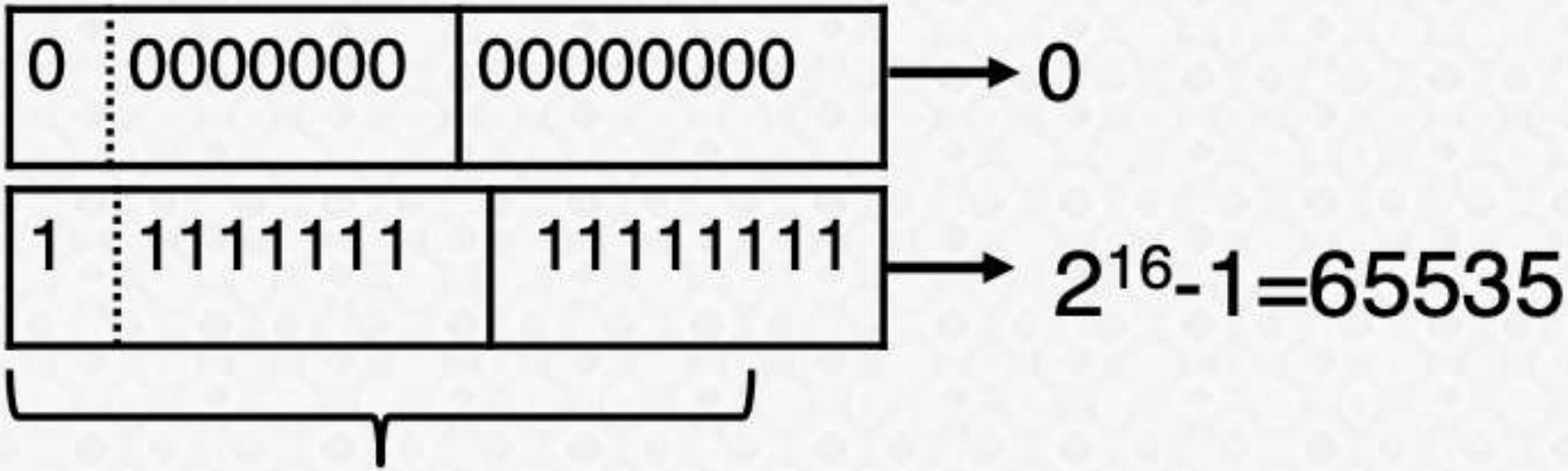
- 字符型数据的长度为1个字节。
- 实型数分为单精度（float）和双精度（double）

float	4字节
double	8字节

➢ 位数越多，可表示的数的范围越大，精度越高。

整型数据在内存中的存储形式

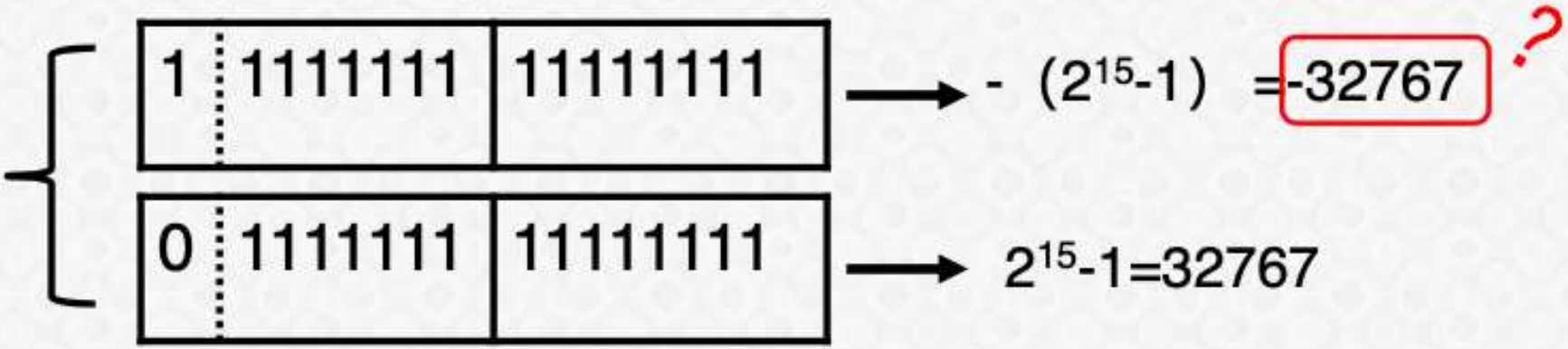
- 无符号：全部比特位都用于表示数据。



short型，2字节存储，计16个比特位

整型数据在内存中的存储形式

- 有符号：最高位0表示整数，1表示负数。



short型，2字节存储，最高位为符号位，15个比特位表示数据

整型数据在内存中的存储形式

- 整型数据在内存中的存储形式
  - 原码（True Code）
  - 反码（Ones-Complement Code）
  - 补码（Complement Code）
- 整数都是以二进制补码的方式存储的
  - 正数的补码就是其原码
  - 负数的补码是其反码+1

整型数据在内存中的存储形式

- 原码：最高位0表示正数，1表示负数，其余位该数绝对值对应的二进制编码。

例如：$(-43)_{原} = 10101011$
$(43)_{原} = 00101011$

- 反码：保持符号位不变，其余各位对原码按位求反。

例如：$(-43)_{反} = 11010100$

- 补码：在反码的基础上加1。

例如：$(-43)_{补} = 11010101$

整型数据在内存中的存储形式

- **short型整数-32645和-123的补码**

-32645 的原码

1	111 1111	1000 0101

-123 的原码

1	000 0000	0111 1011

-32645 的反码

1	000 0000	0111 1010

-123 的反码

1	111 1111	1000 0100

-32645 的补码

1	000 0000	0111 1011

-123 的补码

1	111 1111	1000 0101

字符型数据在内存中的存储形式

- 字符型数据的长度为1个字节，在内存中以对应的ASCII码存放。

01000001

- 从数据的计算机内部表示形式看，char型与int型本质上说相同的。所以字符‘A’也可以看作是整数65.

> 大写字母A加32就是小写字母a，可以用加减法来转换字母的大小写

- char型数据的长度只有一个字节，所以取值范围为0～127，不能越界

实型数据在内存中的存储形式

- 实型数据无论是小数形式还是指数表示形式，在计算机内部都是用二进制的浮点方式将实数分为阶码和尾数两个部分分别存储。

$R=S\times 2^{j}$　　R：实数

S：尾数，有符号的纯小数　　j：阶码，有符号的整数

12.625——→ 1100.101

0.1100101×2^{100}

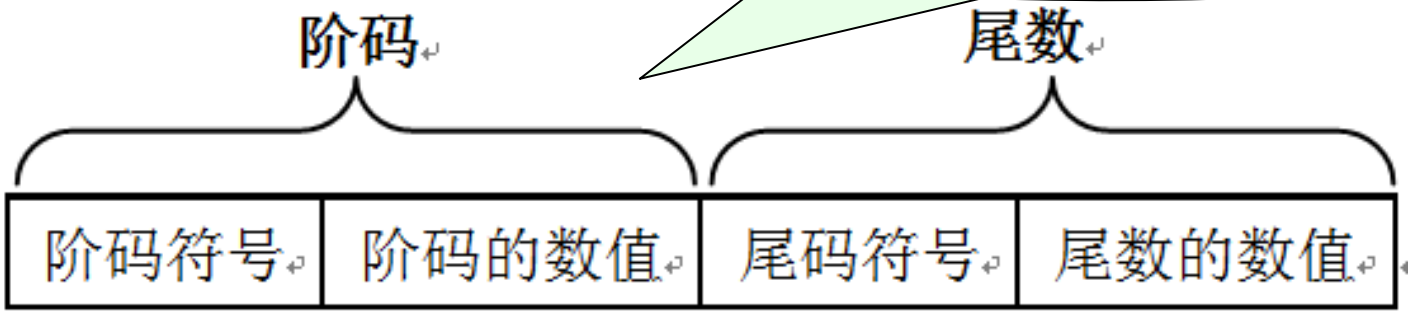
阶码100是二进制的4

尾数0.1100101

实型数据在内存中的存储形式

- 二进制的浮点方式将实数分为阶码和尾数两部分进行存贮

尾码部分的位数决定了实数的精度。
阶码的位数决定了实数的取值范围



- 例如：
  - 十进制数12.625对应的二进制数1100.101，则
  - 1100.101 = 0. 1100101 × 2 ^{100}
  - 尾数S=0. 1100101，阶码j=100

本章小结

- C语言源程序的组成结构及6种基本符号
- C语言基本数据类型
- 常量的表示方法
- 变量的定义、输入和输出方法

输入理想的程序

输出快乐的人生