

第3章 顺序程序设计（1）


S.IST@XMU

复习回顾

Ø 上次课的内容：计算机中的数

- 位、字节和二进制的基本概念

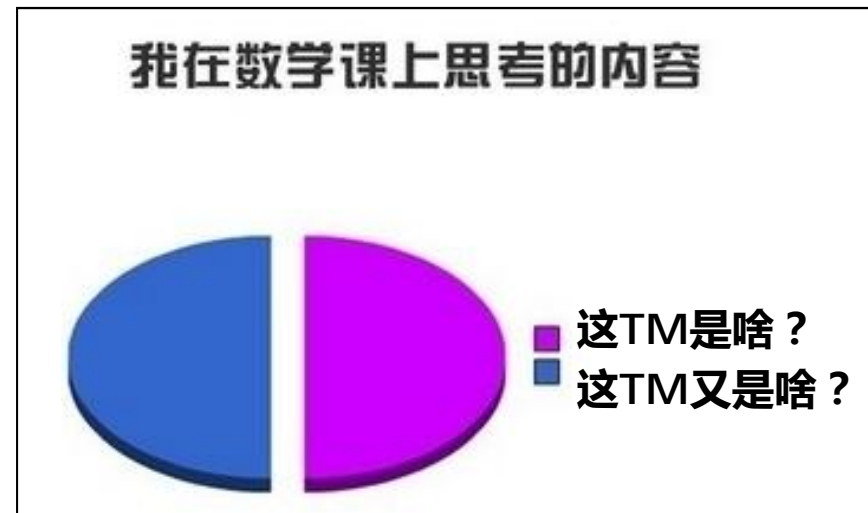
- 二、八、十、十六进制的相互转换

- 有符号数与无符号数

- 补码和原码

- ASCII码

Ø 有点像数学课，但愿
你不属于右图的情形



学习编程能带来什么

Ø 缜密的逻辑思维能力和规则感

Ø 基于检验的科学理性的精神

Ø 不盲从的批判意识

Ø 独立思考的习惯

Ø 与智者交流的机会

Ø 提升个人品味

Ø

程序员，在麻瓜的世界里挣扎，我的世界你什么时候才懂？

北京八维 2016-09-17 14:17



题记

世界迟早由比特构成，程序员就是这个世界的建筑师。所以，当我看到人们担忧机器人作乱，却从来不用担心程序员造反的时候，我都觉得人类奇妙极了。如果真有机器人统治世界的一天到来，在那一天之前，程序员应该早已经统治了世界许久。

初学C语言的忠告

1. 不能光读书，需要“读书+思考+上机练习”；
2. 学编程如习武，第一步是反复练习各种套路直到滚瓜烂熟；
3. 面对一道编程题，最重要的收获不是解决了该问题，而是学习掌握解决类似问题的常用方法（即使自己独立解决了，还是应该去搜索他人的解法，若发现别人的方法优于你的原创，那么就大方地借鉴吧）；
4. 没有思路时，不反对到网上查找参考程序，但务必看懂参考程序并熟练掌握其中的技巧；
5. 永远不要把程序错误归结为灵异事件，淡定地面对程序的错误，理智的观察分析绝对比盲目的胡乱尝试来得有效有益。

Are You Ready ?

Ø学习C语言注定是痛苦并快乐的过程

Ø请做好以下储备

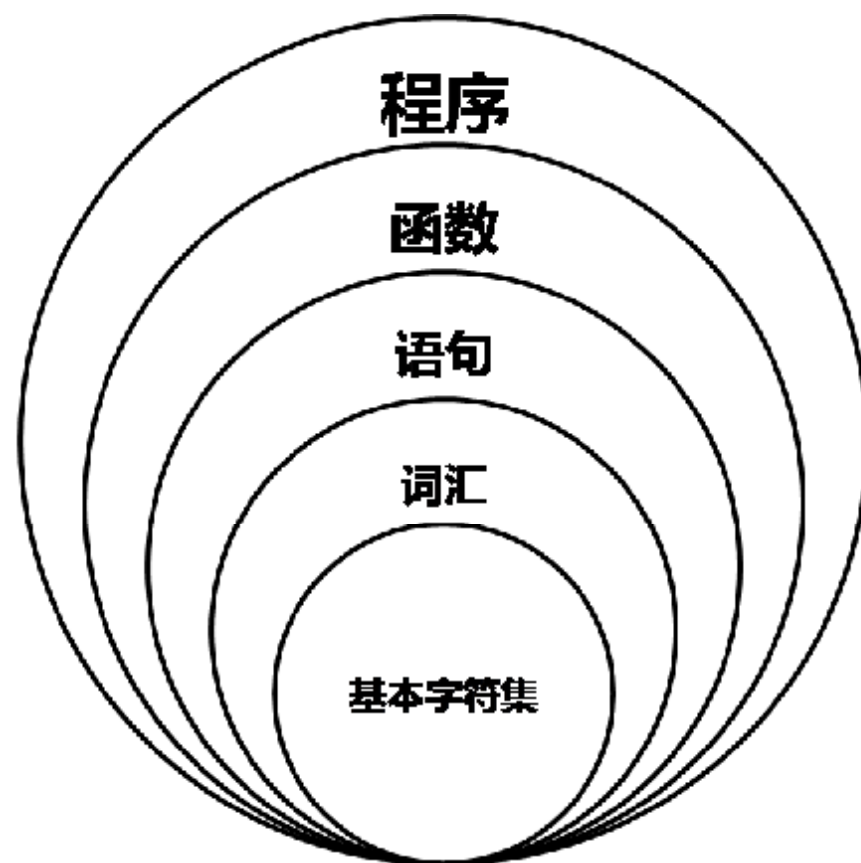
- u 耐心

- u 细心

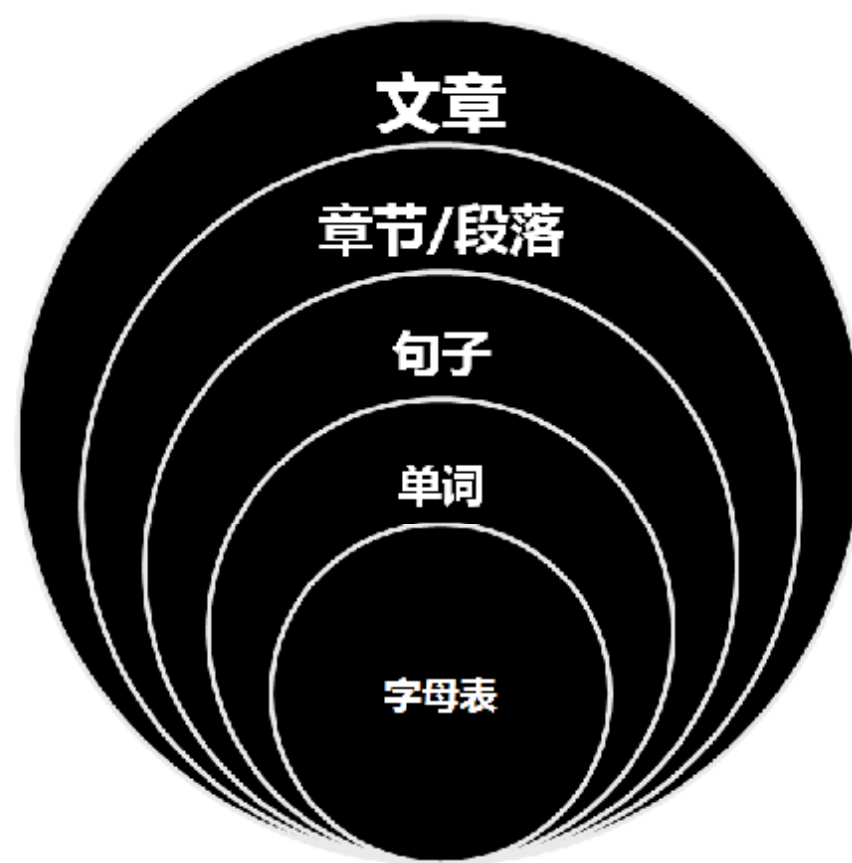
- u 不解决问题誓不罢休的意志和勇气

ØLet's go !

C语言与自然语言的对比



C语言



自然语言

基本字符集

Ø 书写C语言源文件所能使用的字符的集合

u 大小写英文字母 (52个) A-Z a-z

u 数字 (10个) 0-9

!	#	%	^	&	*	(_)	-
+	=	~	[]	'		\	;	:
"	{	}	,	.	<	>	/	?	\$

u 特殊字符 (30个)

u 格式符 (4个) 空格、水平制表符 (Tab) 等



错用非基本字符集里字符的后果

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("C语言, 我来了\n");
6      return 0;
7  }
```

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("C语言, 我来了\n");
6      return 0;
7  }
```

Build log

Output file is bin\Debug\demo2016.exe with size 28.36 KB
Process terminated with status 0 (0 minute(s), 4 second(s))
0 error(s), 0 warning(s) (0 minute(s), 4 second(s))

File	L..	Message
=== Build: Debug in demo2016 (compil		
In function 'main':		
C:\code...	5	error: stray '\243' in program
C:\code...	5	error: stray '\273' in program
C:\code...	6	error: expected ';' before 'return'



这是中文输入法的分号？！@#*&！

C语言的单词

Ø单词：由基本字符集中的符号按一定规则构成的最小语法单位

Ø六类：

u 关键字

u 运算符

u 特定字

u 分隔符

u 标识符

u 字面常量

C语言单词类型之一：关键字

【C89标准共计32个】

- 基本类型 (5个) `void int float double char`
- 类型修饰 (4个) `short long signed unsigned`
- 复杂类型 (5个) `typedef struct union enum sizeof`
- 存储级别 (6个) `const auto static register extern volatile`
- 分支结构 (5个) `if else switch case default`
- 循环结构 (3个) `while for do`
- 跳转控制 (4个) `break continue goto return`

【C99标准新增5个】 `inline restrict _bool _Complex _Imaginary`

Ø什么叫关键字？

u **普通青年**：也称**保留字**，是C语言中具有特定作用和含义的单词，在程序中**不能另作其他用途**。

u **文艺青年**：在CodeBlocks编写的程序中显示为**深蓝色**的词。

C语言单词类型之二：特定字

Ø**特定字(预定义标识符)**是一些用在C语言的**预处理命令**和**库函数名**中的单词，这些字都是由编译系统规定的，有特定含义。不是关键字，但习惯上等同看待。

└ 例如：

└ 预处理命令：`#define, #include, #ifdef, #endif`

└ 库函数：`scanf, printf`

└ 主函数名：`main`

下面有哪些关键字和特定字？

1. <code>#include</code>	<code><stdio.h></code>	//这是编译预处理指令
2. <code>int</code>	<code>main()</code>	//定义主函数main
3. <code>{</code>		//函数开始的标志
4. <code>printf</code>	<code>("void main(){}\n");</code>	//输出所指定的一行信息
5. <code>return</code>	<code>0;</code>	//函数执行完毕时返回0
6. <code>}</code>		//函数结束的标志

关键字

特定字

提示： 不要死记硬背！随着深入学习编程语法，自然就会避免将关键字定义成变量名。编程关键在于掌握基本原理、基本方法和技能，靠背诵解决不了问题。

C语言单词类型之三：标识符


Ø 标识符（用户定义标识符）是以字母（有时包括\$）或下划线打头，由字母、数字和下划线组成的字符序列，是用户根据自己需要定义的一类标识符，用于标识变量、符号常量、用户定义函数名、类型名和文件指针等。

Ø 命名标识符时应注意：

- u 必须用字母（有时候\$也可以）或下划线打头
- u 大小写有别，如：total 和 TOTAL是不同的
- u 不能用关键字和特定字命名
- u 除字母数字和下划线外一般不允许空格和其他字符
- u 长度最好不要超过31个字符
- u 最好做到见名知义，比如表示年份用 year 比 a 清楚

使用以下标识符合法吗？

Ø Double	√	但是强烈不推荐使用
Ø x<y	×	包含非法字符<
Ø _1a2	√	
Ø num0	√	
Ø if	×	不能使用关键字
Ø Mrs.Smith	×	包含非法字符.
Ø _2012	√	
Ø user_name	√	
Ø 18M	×	不能以数字打头
Ø \$100	√ or ×	某些书说不行，但CB编译正常



空口无凭,
可用程序
检验!

C语言单词类型之四：运算符

运算符也称操作符，告诉计算机如何操作数据

- (1) 算术运算符： +、-、*、/、%、++、--
- (2) 关系运算符： >、<、>=、<=、==、!=
- (3) 赋值运算符： =、+=、-=、*=、/=、%=、&=、|=、^=、<<=、>>=
- (4) 逻辑运算符： !、&&、||
- (5) 位运算符： <<、>>、~、|、^、&
- (6) 条件运算符： ? :
- (7) 逗号运算符： ,
- (8) 指针运算符： *、&
- (9) 求字节数运算符： sizeof
- (10) 分量运算符： .、->
- (11) 下标运算符： []
- (12) 强制类型转换运算符： (数据类型名)

下面有哪些标识符和运算符？

```
1. #include <stdio.h>
2. int main( )
3. {
4.     int max(int x,int y);
5.     int a,b,A;
6.     scanf("%d,%d",&a,&b);
7.     A = max(a,b);
8.     printf("max=%d\n",A);
9.     return 0;
10. }
```

```
11. int max(int x,int y)
12. {
13.     int _z;
14.     if (x>y) _z = x;
15.     else _z = y;
16.     return(_z);
17. }
```


C语言单词类型之五：分隔符

Ø **分隔符**用来界定或分割语句中的语法成分（像文章中的标点符号）

- u **分号**；表示一个语句的结束（预处理命令和{}后面不用加分号）
- u **空格 逗号**，在两个相邻的保留字或标识符之间起分割作用。连续多个空格和单个空格的作用相同，如 `int a` 和 `int a,b`
- u **单引号'与双引号"** 字符常量与字符串的开始和结束
- u **花括号{ }** 函数体的开始和结束 复合语句的开始和结束
- u **/* */与//** 多行注释的开始和结束与单行注释的开始
- u **尖括号 < >** `#include`命令中库文件名的开始和结束
- u **圆括号()** 参数列表或表达式的开始和结束
- u **运算符**也能分割单词，如`a=3` 和 `a = 3`是一样的效果

C语言单词类型之六：字面常量

Ø 字面常量：在程序中直接写出常量值的常量，通常又称直接常量。

Ø 字面常量的四种类型：

- └ 整型常量，如65，-011，0xbbc

- └ 浮点型常量，如3.14，0.314e+2，314E-2

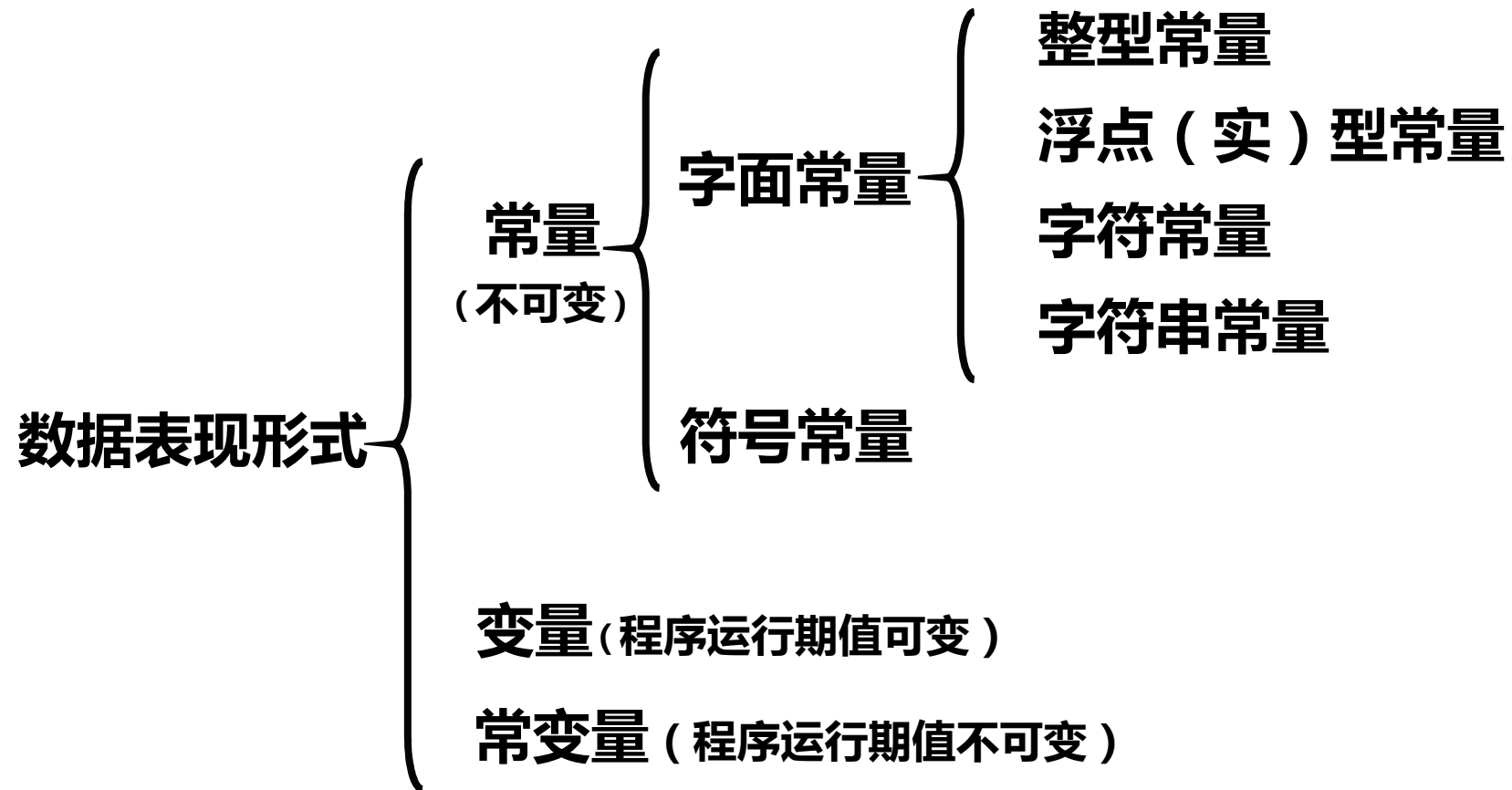
- └ 字符型常量，如'a'，'\n'，'\102'，'\x42'

- └ 字符串常量，如"I love C"

下面有哪些分隔符和字面常量

```
1. #include <stdio.h>
2. int main()
3. {
4.     int a = 65, b = -011, c = 0xbbc;
5.     float p=3.14, q=0.314e+1, r = 314E-2;
6.     char c='a', d='\n', e='\102';
7.     char s[] = "I love C";
8.     return 0;
9. }
```

C语言的数据



整型字面常量详解

Ø 十进制整数形式：除整数0之外第一个数字不能是0，如+65

Ø 八进制整数形式：数字0开头，不能出现数字8和9，如0101

Ø 十六进制整数形式：0x或0X开头，由0~9和A~F（或a~f）组成，如0x41

Ø 字符后缀U和L

u后缀u或U表示无符号数：如635u

u后缀l或L表示长整数型：如5983672L

浮点型字面常量详解

Ø 仅采用10进制形式书写

└ **小数点表示法**：与数学表示一致，如3.14

└ 当小数点前、后的数为0时，可省略0，但小数点不能省，如23.和.23分别表示23.0和0.23

└ **指数（科学）表示法**：类似于科学记数法，如
0.314e1，314E-2

└ e（或E）前面不能没数字，后面必须整数且不能加小括号，如e5、2E(-6)都不是合法形式

字符型字面常量详解

Ø 可显示字符通常用字符本身书写，如'a'；

Ø 不可显示字符和专用字符用转义序列表示：

转义字符	转义字符的意义	十进制ASCII代码
<code>\n</code>	回车换行	10
<code>\t</code>	横向跳到下一制表位置	9
<code>\b</code>	退格	8
<code>\r</code>	回车	13
<code>\f</code>	走纸换页	12
<code>\\</code>	反斜线符" <code>\</code> "	92
<code>\'</code>	单引号符	39
<code>\"</code>	双引号符（仅在字符串中才要反斜杠）	34
<code>\a</code>	鸣铃	7
<code>\ddd</code>	3位八进制数在ascii码表对应的字符，如 <code>\100</code> 代表@	64
<code>\xhh</code>	2位十六进制数在ascii码表对应的字符，如 <code>\x21</code> 代表!	33

⚠ 注意：转义字符形式上看起来像两个或两个以上字符（如`'\'`、`'\101'`等），事实上仅表示一个字符，而`'ab'`则不合法

字符串字面常量详解

Ø 由定界符双引号 (") 括起来的字符序列，该序列可以是零个、一个或多个字符，没有字符的成为空串

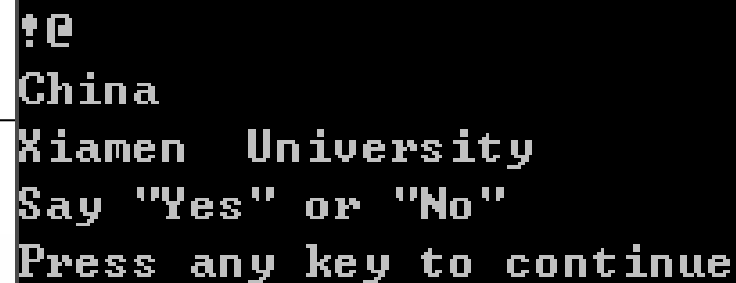
└ 例如 `"hello"` , `"\x21\100"` ,
`"China\nXiamen\tUniversity"`

└ 当双引号本身作为字符串中的字符时，应写成 `\"`，如
`"Say \"yes\" or \"No\" :"` 表示的字符串是
`Say "yes" or "No" :`

└ **注意：**不要把字符常量和字符串常量混淆，如 `'a'` 和
`"a"` 是不一样的，`"a"` 占两个字节，包括 `'a'` 和 `'\0'`

字符串常量输出举例

```
1. #include <stdio.h>
2. void main()
3. {
4.     printf( "\x21\100\n" );
5.     printf( "China\nXiamen\tUniversity\n" );
6.     printf( "Say \"Yes\" or \"No\"\n" );
7. }
```



```
!@
China
Xiamen University
Say "Yes" or "No"
Press any key to continue
```

符号常量详解

Ø 用#define指令，指定用一个符号名代表一个常量

u 如，`#define PI 3.1416`

！ 注意：行末没有分号！

u 与变量的区别：**不允许** `PI=3.1415926;`

Ø 好处

u 使该常数含义清楚

u 使用该常数的地方保持一致

u 一改全改

举例：符号常量的使用

编写程序，计算并输出半径为10的球表面积和球体的体积

```
#include <stdio.h>

#define R 10 //定义符号常量R
#define PI 3.14159 //定义符号常量PI

int main()
{
    float s, v;
    s = 4*PI*R*R; //注意：预处理后等号右边替换成4*3.14159*10*10
    v = s*R/3; //同上：预处理后等号替换成s*10*3
    printf("s=%f v=%f\n", s, v);
    return 0;
}
```

s=1256.637085 v=4188.790039

变量详解

- Ø 变量代表一个有名字的、具有特定属性的一个存储单元。
在运行期间，其值可变。
- Ø 必须**先定义，后使用**。如 `int a; a=3;`
- Ø 定义时指定名字和类型，如 `int a=3, B;`

符号表

变量类型	变量名	变量地址
...
int	a	1004
int	B	1000
...

内存状态

内存地址	...	1000	1004	...
内存数据	...	-239879	3	...
变量名称	...	B	a	...

变量的命名

Ø 变量名必须是一个合法的用户定义标识符

Ø 注意事项：

- ⌞ 不能用C语言保留字或特定字作为变量名。

- ⌞ 不建议使用以下划线开头的变量名。

- ⌞ 命名时应注意区分大小写，并**尽量避免只是大小写上有区别的变量名**。例如同时使用sum, Sum, SUM三个变量就容易混淆。

- ⌞ **避免使用类似的变量名**。例如使用 entry_total 和 all_total 要好于 使用 total 和 totals。

变量的定义

Ø 变量定义的一般形式：

变量类型 **变量名**; **//注释**

└ 例如：`int sum; //存储求和结果的变量`

Ø 目的：

└ 定义变量的名称

└ 定义变量的数据类型

└ 提供关于此变量的描述信息

Ø 定义同一类型的多个变量时，允许写在同一行

└ 例如：`int weight,height; //存储身高、体重的两个变量`

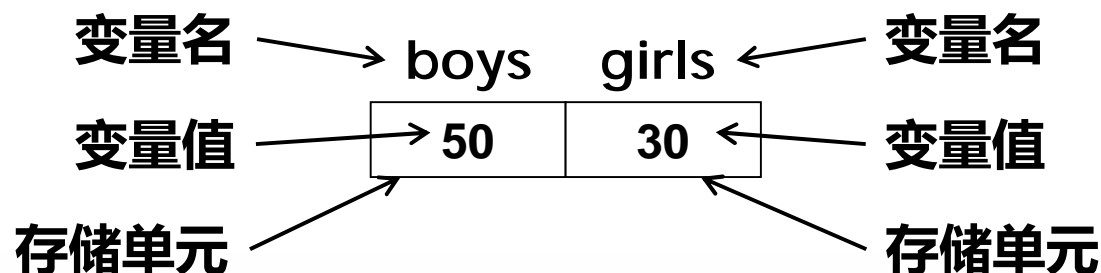
变量名与变量的值

Ø 一个变量总是拥有一个**变量名**，变量名总是对应一个内存位置。

Ø **变量的值**是该内存位置可以存放的一个值，且该值可以被改变。

Ø 举例：

```
int boys=50,girls=30; //当前教室中男生、女生的数量
```



变量的初始化

Ø 变量初始化的一般形式：

变量类型 变量名 = 初始值; // 注释

└ 例如：`int sum = 0; // 存储求和结果的变量`

Ø 初始值可以是以下形式：

└ 常量，如1、3.14、'a'

└ 常量组成的表达式，如1 + 2 + 3、3.14 * 5

└ 常量和已被初始化的变量组成的表达式，如x + 1

Ø 可在同一定义对任意变量进行初始化

└ 注意：`int weight, height = 170; /* 表示height被初始化为170，而weight未被初始化。 */`

常变量详解

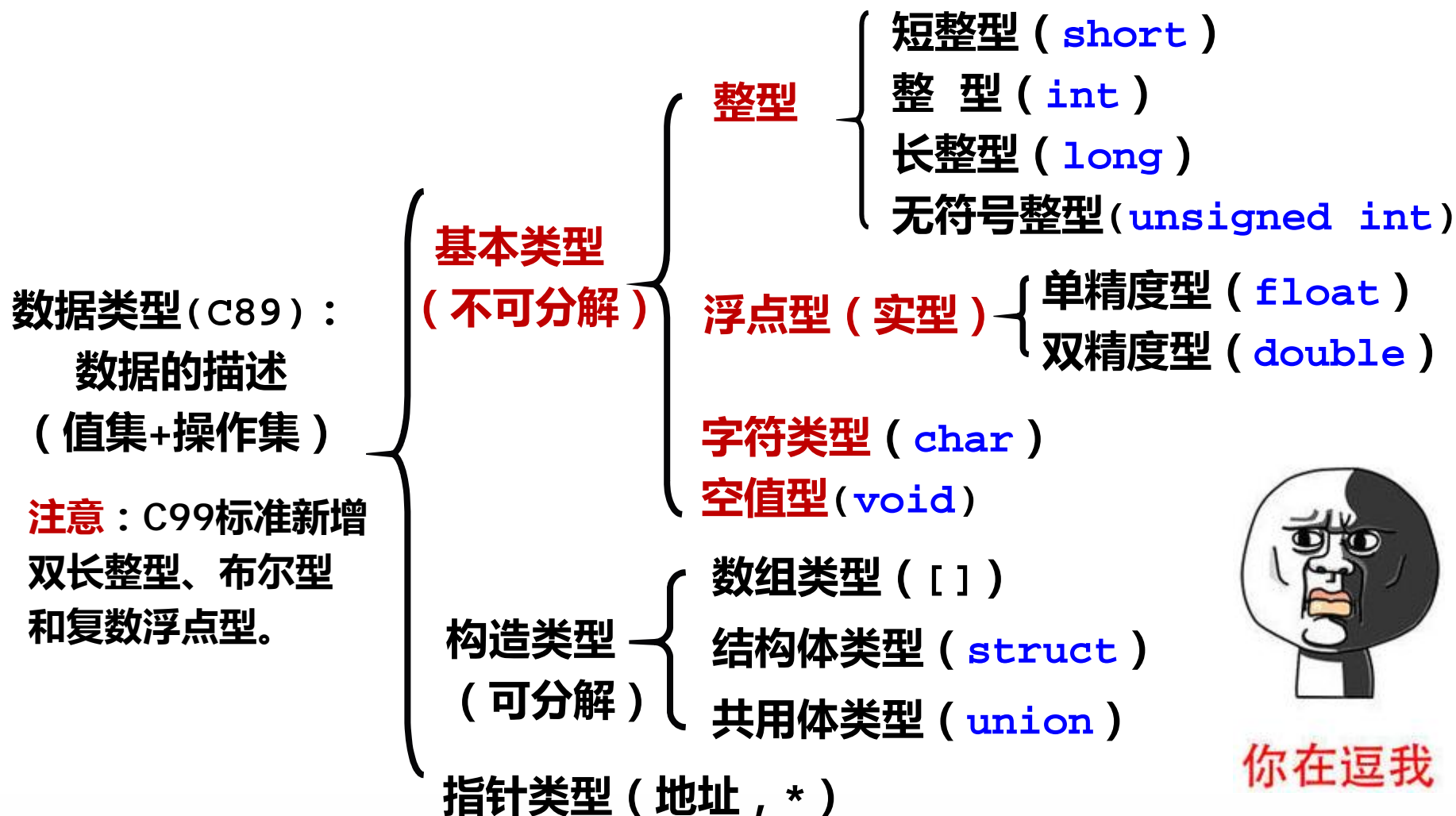
Ø C99标准新定义

Ø 常变量具有变量的基本属性：有类型，占内存单元，只是程序运行时**不允许改变其值**。

Ø **定义方式**，在普通变量定义前加const关键字，
如：`const int a=3;`

Ø **与符号常量的区别**：符号常量的符号不对应内存地址，而常变量名称对应内存地址

C语言中任何数据都有数据类型



为什么需要不同的数据类型

Ø 为何数学里的数值不需要数据类型

- u 数学不同于计算机：抽象 vs 具体，精确 vs 近似

Ø 计算机科学是人造学科

- u 追求便利：不同数据类型编码方式不同，允许的操作也不同，划分数据类型，既有利于人类的理解，也便于计算机的存储和处理

- u 追求优化：使用多种数据类型可以节省内存开销

基本数据类型之一：整型

整型：用于描述整数,在C 语言中,整型用int 来说明,分为**基本型、短整型、长整型、无符号型**。

类型	类型标识符	在内存所占的字节数	数值范围
基本型	int	4 (曾经是2)	-2147483648~2147483647 即 $-2^{31} \sim (2^{31}-1)$
短整型	short int	2	-32768~32767 即 $-2^{15} \sim (2^{15}-1)$
长整型	long int	4	-2147483648~2147483647 即 $-2^{31} \sim (2^{31}-1)$
无符号整型	unsigned int	4 (曾经是2)	0~4294967295 即 $0 \sim (2^{32}-1)$
无符号短整型	unsigned short	2	0~65535 即 $0 \sim (2^{16}-1)$
无符号长整型	unsigned long	4	0~4294967295 即 $0 \sim (2^{32}-1)$

整型的“溢出”

```
1. #include <stdio.h>
2. int main()
3. {
4.     short int a,b;    //定义短整型变量a和b
5.     long int c;       //定义长整型变量c
6.     a = 32767;
7.     b = a+1;
8.     c = 32768;
9.     printf("a=%d,b=%d,c=%ld\n",a,b,c);
10.    return 0;
11. }
```

a=32767,b=-32768,c=32768

类型	类型标识符	在内存所占的字节数	数值范围
长整型	long int	4	-2147483648~2147483647
短整型	short int	2	-32768~32767

无符号整型与负值

```
1. #include <stdio.h>

2. int main()
3. {
4.     unsigned short a = 50;
5.     unsigned short b = -1;

6.     printf("a = %u, b = %u\n", a, b);
7.     return 0;
8. }
```

a = 50, b = 65535

为什么？ 因为系统先把-1以补码方式，存入变量b，变量b的二进制按无符号数解释，即得结果。

基本数据类型之二：浮点型

单精度 (float)

4字节, 范围 $\pm(3.4 \times 10^{-38} \sim 3.4 \times 10^{38})$

有效精度: 6~7位

更高精度 (long double)

16字节, $\pm(1.1 \times 10^{-4932} \sim 1.1 \times 10^{4932})$

有效精度: 18~19位

双精度 (double)

8字节, 范围 $\pm(1.7 \times 10^{-308} \sim 1.7 \times 10^{308})$

有效精度: 15~16位

注意：浮点型存储数据有误差，例如：

```
1. #include <stdio.h>
2. int main()
3. {
4.     float a = 12.3, b;
5.     double c;
6.     b = c = 12345.678;
7.     printf("a=%f,b=%f,c=%lf\n",a,b,c);
8.     return 0;
9. }
```

a=12.300000,b=12345.677734,c=12345.678000

浮点型数据的存储方式

Ø 规范化指数形式的基本思想（具体实现可自行查阅资料）

- └ 在指数形式中，小数部分中小数点前的数字为0，小数点后第一位数字不为0的表示形式，如 $0.314159e001$
- └ 在二进制表示中，用1个位表示正负符号，用若干位表示小数部分的值，用其余位表示指数部分（包括符号）
 - ┆ 小数部分占位越多，有效数字越多，精度越高
 - ┆ 指数部分占位越多，表示的范围越大



Ø 注意：浮点型数据能表示的只是实数轴上的离散点！

浮点数舍入误差带来的局限

Ø 应避免两个量级差别很大的数进行加法或减法

```
1. #include <stdio.h>
2. int main()
3. {
4.     float a = 1000000+0.01;
5.     float b = 1000000-0.01;
6.
7.     printf("a=%f, b=%f\n", a, b);
8.
9.     return 0;
10. }
```

a=1000000.000000,b=1000000.000000

基本数据类型之三：字符型

字符型 (char)：一个字符按其对应的ASCII码的整数形式存储。因此字符型数据类似于小范围的整型数据。

类型	类型标识符	在内存所占的字节数	数值范围
基本字符型	char	1	依编译系统而定
有符号字符型	signed char	1	-128 ~ 127 即 $-2^7 \sim (2^7-1)$
无符号字符型	unsigned char	1	0 ~ 255 即 $0 \sim (2^8-1)$

如何测定你的系统的char是signed还是unsigned？

```
1. #include <stdio.h>
2. int main()
3. {
4.     char c = 255; //把一个128~255之间的整数赋值给字符变量c
5.     printf("c=%d\n", c); //若输出负数，则为signed，否则为unsigned
6.     return 0;
7. }
```

字符型变量与整数的关系举例

Ø 字符型变量的值可以赋给一个整型变量；

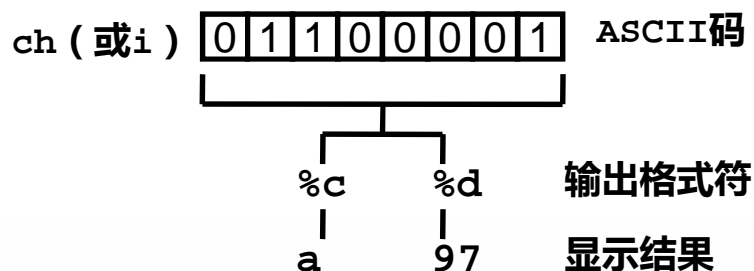
└ 如右例第8行

Ø 字符型变量可以和整数混合运算；

└ 如右例第7行

Ø 整数值可以以字符的形式输出, 反之亦然。

└ 如右例第9、10行



```
1. #include <stdio.h>
2. int main()
3. {
4.     char ch;
5.     int i;
6.     ch = 'A';
7.     ch = ch+32;
8.     i = ch;
9.     printf("%d is %c\n",ch,i);
10.    printf("%c is %d\n",ch,ch);
11.    return 0;
12. }
```

97 is a
a is 97

基本数据类型之四：空型

Ø 空值型：又称无值型

└ 用关键字**void**表示, **void**型的值集为空集,可出现在函数定义的头部,表示该函数没有返回值

└ 例如

```
1. void main()  
2. {  
3.     printf("An example of void type.\n");  
4. }
```

怎样确定常量的类型

Ø在Visual C++中

└ 整型常量

┆ 凡在 $-2^{31} \sim 2^{31}-1$ 之间的不带小数点的数都作为int型，分配4个字节

└ 浮点型常量

┆ 凡以小数形式或指数形式出现的实数，都按双精度(double型)处理，分配8个字节

作业 2016/09/23

1. printf 与 转义字符

- a) 仅用一条printf语句，打印 $1 + 2$ 和 $3 + 4$ 的值，用两个空行隔开；
- b) 试着把%d中的两个字符（百分号和小写字母d）输出到屏幕；
- c) 试着把\n中的两个字符（反斜线和小写字母n）输出到屏幕；
- d) 像b)、c)那样也需要“特殊方法”才能输出的字符有哪些？

注意事项：

- (1) 作业写在纸上，作业纸抬头写上学号和姓名；
- (2) 觉得课上没讲，可以问Google问百度，最好在电脑验证一下是否正确；
- (3) 下周二（9月27日课间休息时间交给助教）

上机练习（不用交）：用编程工具编译运行本讲义所有例子程序，注意代码格式与讲义保持一致。