



计算机学院

ACAREC

先进计算机应用技术教育部工程研究中心
北京航空航天大学计算机学院

C语言指针：从入门到精通

荣文戈

w.rong@buaa.edu.cn

北京航空航天大学计算机学院



基础不牢，地动山摇



C语言一直是最流行的基础编程语言之一

Very Long Term History

To see the bigger picture, please find below the positions of the top 10 programming languages of many years back. Please note that these are *average* positions for a period of 12 months.

Programming Language	2021	2016	2011	2006	2001	1996	1991	1986
C	1	2	2	2	1	1	1	1
Python	2	5	6	8	25	25	-	-
Java	3	1	1	1	2	16	-	-
C++	4	3	3	3	3	2	2	6
C#	5	4	5	7	12	-	-	-
Visual Basic	6	13	-	-	-	-	-	-
JavaScript	7	7	10	9	9	20	-	-
PHP	8	6	4	4	10	-	-	-
Assembly language	9	11	-	-	-	-	-	-
SQL	10	-	-	-	37	-	-	-
Ada	29	28	17	16	18	7	5	2
Lisp	33	27	13	13	17	8	4	3
Pascal	268	75	15	17	15	5	3	7
(Visual) Basic	-	-	7	5	4	3	8	5



this vs. super



引用、move等



C语言代码量大概有多少？





先问一个最基本的概念

C语言中有对象（Object）吗？

有!!!

不仅有，而且是理解C语言的最核心基础概念



先看个简单的题

```
int a = 10;
```

a是什么？ a的值是多少？ a的类型是什么？ &a的类型是什么？

```
int b[10] = {10};
```

b是什么？ b的值是多少？ b的类型是什么？ &b的类型是什么？



先提几个问题

```
int a = 10;
```

```
printf("a = %d\n", a);
```

```
printf("&a = %d\n", &a);
```

&a返回值什么类型？

```
int *p = &a;
```



先提几个问题

```
int a[3] = {10};  
  
printf("a[0] = %d\n", a[0]);  
  
printf("a = %d\n", a);  
printf("&a = %d\n", &a);
```

打印a和&a的两个语句打印出来的值一样吗？

```
p = a;  
q = &a;
```

p和q该如何定义和声明？



先提几个问题

```
int a[3][4] = {10};
```

```
printf("a[0][0] = %d\n", a[0][0]);
```

```
printf("a = %d\n", a);
```

```
printf("&a = %d\n", &a);
```

```
printf("a[0] = %d\n", a[0]);
```

打印a、&a和a[0]的三个语句打印出来的值一样吗？

p = a; q = &a; r = a[0], p、q、r应该如何定义和声明？



先提几个问题

```
int a[3][4][5][6] = {10};
```

```
printf("a[0][0][0][0] = %d\n", a[0][0][0][0]);
```

```
printf("a = %d\n", a);
```

```
printf("&a = %d\n", &a);
```

```
printf("a[0] = %d\n", a[0]);
```

```
printf("a[0][0] = %d\n", a[0][0]);
```

```
printf("a[0][0][0] = %d\n", a[0][0][0]);
```

打印a、&a、a[0]、a[0][0]和a[0][0][0]的五个语句打印出来的值一样吗？

p = a; q = &a; r = a[0]; s = a[0][0]; t = a[0][0][0], p、q、r、s和t应该如何定义和声明？



先提几个问题

```
void func( int a[][4])  
{  
    // do something  
}
```

维度大小没有的原理
是什么？

```
int main()  
{  
    int a[3][4];  
  
    func(a);  
  
    return 0;  
}
```



先提几个问题

```
int a = 10;
```

```
printf("sizeof(a) = %d\n", sizeof(a));
```

```
printf("sizeof(a+1-1) = %d\n", sizeof(a+1-1));
```

sizeof(a)和sizeof(a+1-1)打印出来的值一样
背后计算大小的逻辑一样吗？



先提几个问题

```
int a[3] = {10};
```

```
printf("sizeof(a) = %d\n", sizeof(a));
```

```
printf("sizeof(a+1-1) = %d\n", sizeof(a+1-1));
```

sizeof(a)和sizeof(a+1-1)打印出来的值不一样
是什么原因呢？



先提几个问题

```
int a = 10;
```

```
printf("sizeof(a++) = %d\n", sizeof(a++));
```

现在a等于几了？



先提几个问题

`const int* p`、`int const* p`和`int* const p`有区别吗?

语义规范到底是什么样的呢?



内容简介

刚才那些题只是技巧吗？

深入了解C语言中的内存组织

深入了解C语言中的对象工作机制

深入了解C语言指针和数组原理

深入了解表达式和lvalue

。 。 。

深入了解内存（系统级）



本课程参考书

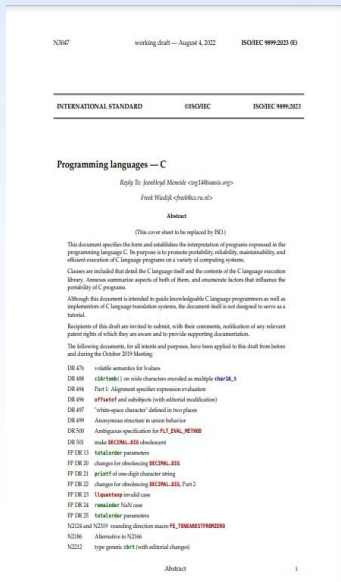
ISO/IEC JTC1/WG14 C语言标准文本

2022年8月4日发布的Working Draft

<https://www.open-std.org/jtc1/sc22/wg14/www/docs/n3047.pdf>

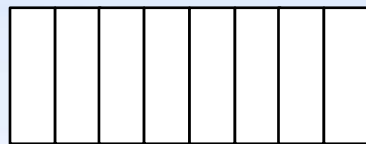
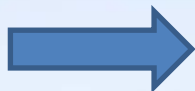
一切疑惑都可以从这个标准中找到答案

希望这是一次有趣的体验





内存基本概念简单回顾



bit (比特)
状态0/1
二进制

8个bit拼在一起, 组成一个byte (字节)
存储范围00000000~11111111



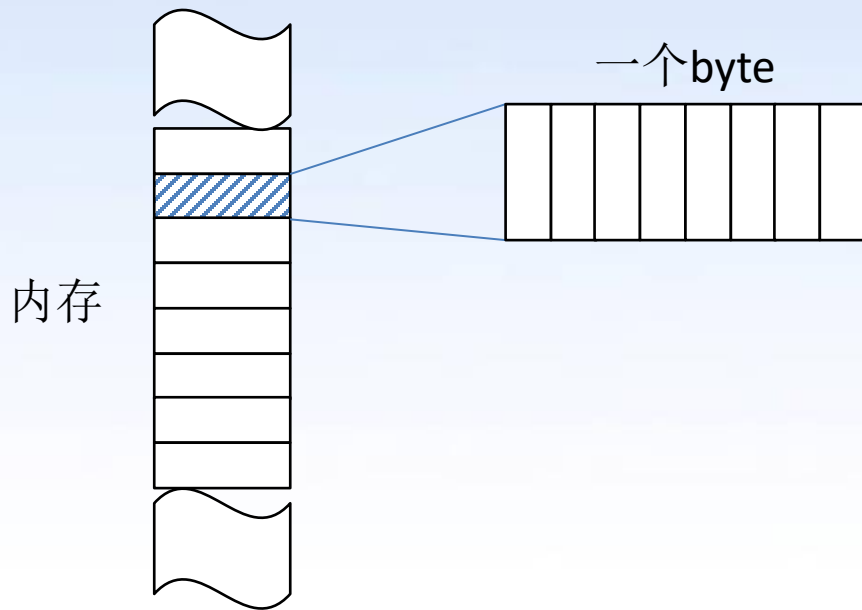
二进制转十进制

0~255

1 byte一定等于8 bit?



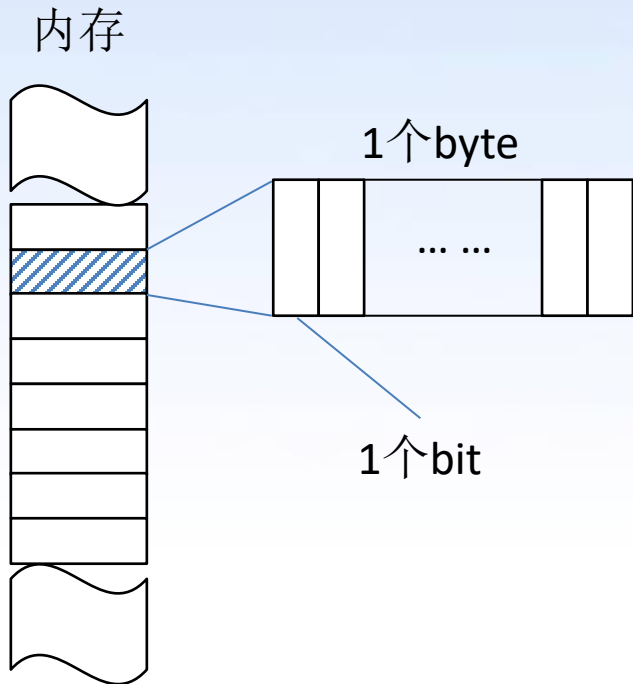
内存长什么样？



内存就是一系列byte顺序排列在一起组成的存储结构



1 byte = ? bit



byte（字节）是C语言最重要的基础存储单元

每一个字节都有一个编号，也就是地址

addressable unit of data storage...

1个byte由若干个bit组成

bit是最小的存储单元，取值为0或1

unit of data storage ... can have one of two values

1个byte = ? bit呢？



1个byte一定等于8个bit吗？

`<limits.h>`定义了一个宏**CHAR_BIT**

C语言中规定一个byte等于**CHAR_BIT**个bit

A byte contains **CHAR_BIT** bits

C语言的标准规定**CHAR_BIT** ≥ 8

TMS320C28x Optimizing C/C++ Compiler v15.9.0.STS

1 byte = 16 bits



测试题

- 1、32位机最多能访问4G内存的原因是什么？
- 2、对64位机器来说，最多能访问的byte数量有多少？

答案：

- 1、32位机最多能访问的内存byte总数为 2^{32} 个，因此：

$$2^{32} \div 1024 = 2^{22}K, \quad 1K=1024\text{个字节}$$

$$2^{22}K \div 1024 = 2^{12}M, \quad 1M=1024K\text{字节}$$

$$2^{12}M \div 1024 = 2^2G = 4G, \quad 1G=1024M\text{字节}$$

- 2、64位机最多能访问的byte总数为 2^{64} 个（理论上）



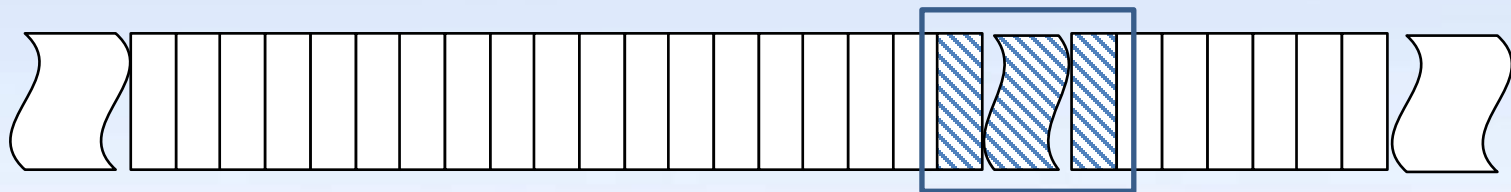
C语言中最重要的概念：对象

- 1、对象的概念
- 2、如何刻画对象的属性
- 3、理解对象类型
(非数组 vs. 数组类型, Referenced Type vs Pointer Type)
- 4、理解指针和数组也是一种普通的对象类型

本课程核心概念就是对象 (Object)



对象是什么



region of data storage

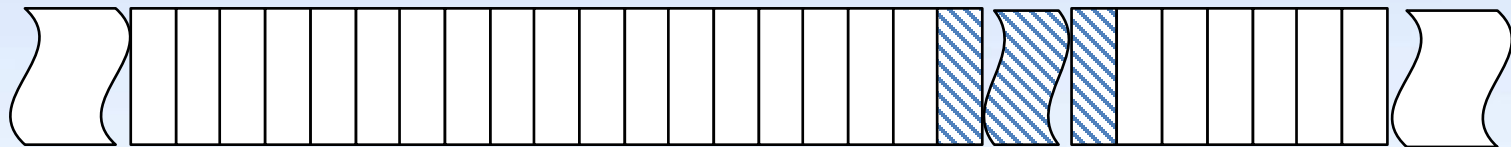
objects are composed of contiguous sequences of one or more bytes

对象是怎么分配出来的呢？

- 1、对象声明，例如 `int a;`
- 2、内存管理函数，例如 `malloc(4);`
- 。 。 。



刻画对象的属性：首地址



对象首地址

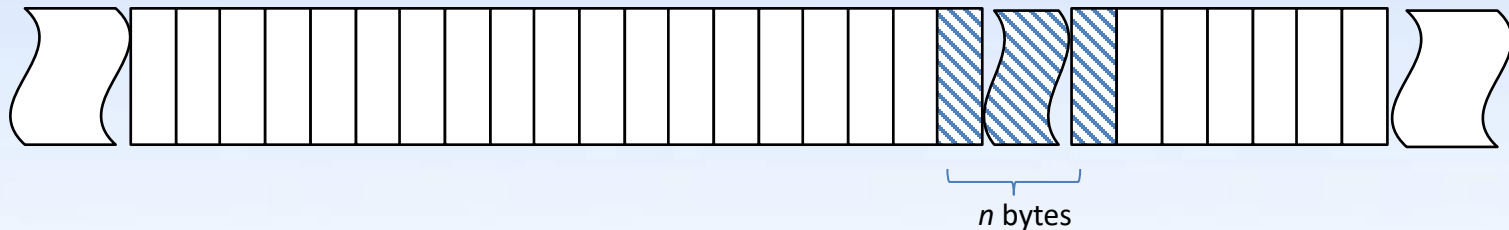
对象所占用的连续字节中第一个字节的编号

分配对象地址需要有**对齐 (Alignment)**的要求

对象的地址是系统分配的



刻画对象的属性：大小

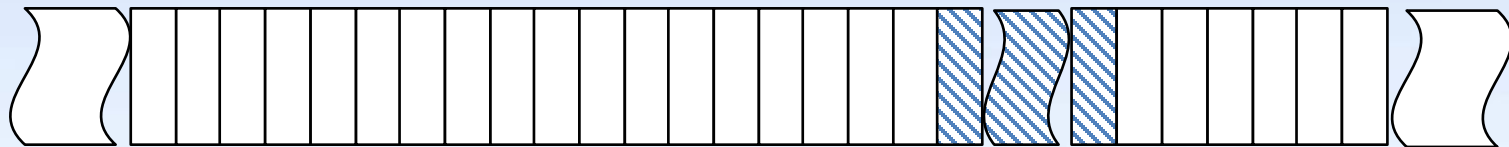


对象占用的连续字节个数

size跟什么相关呢？



刻画对象的属性：对象类型



Object Type

当我们试图理解对象，我们需要了解这个对象的对象类型

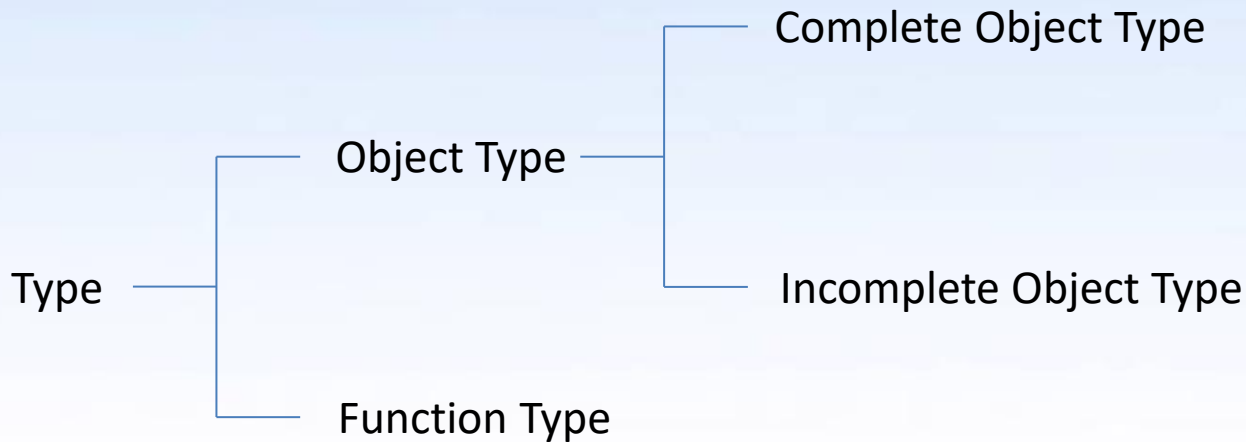
对象类型主要包括：**完全对象类型**和**不完全对象类型**

常见的完全对象类型，例如int, short, long, float, double ...



Type (类型)

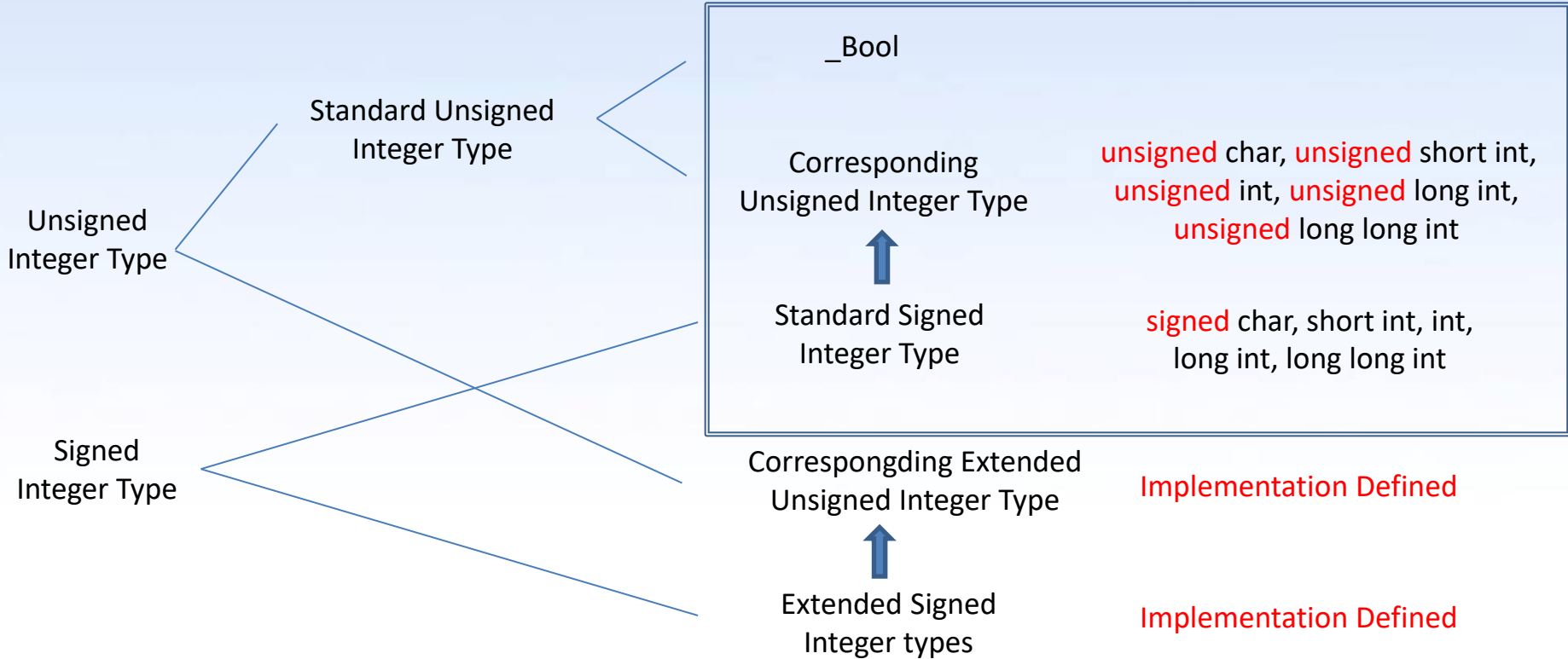
Type(类型)是用来理解一个对象或者函数返回值语义的重要依据



Object: Region of data storage in the execution environment, the contents of which can represent **values**

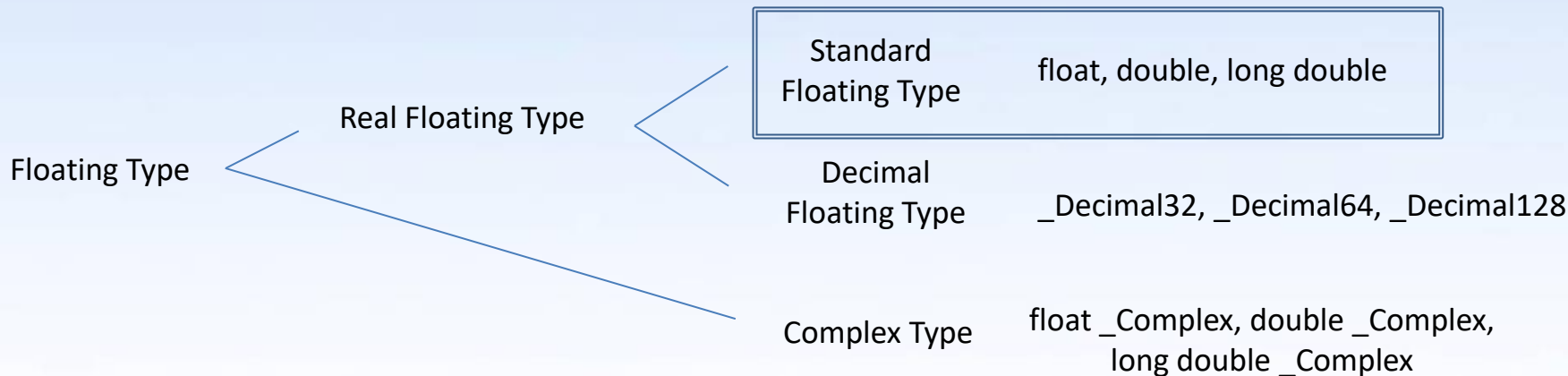


Object Type: Integer Related





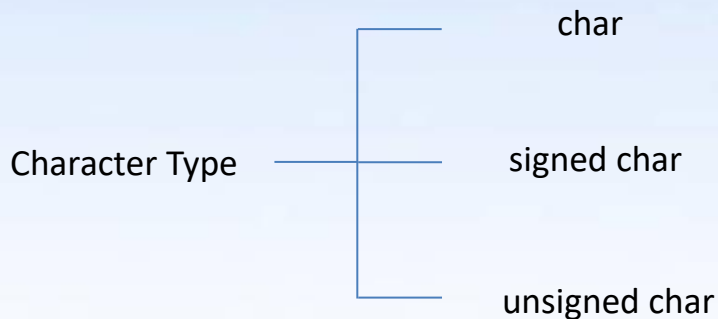
Object Type: Floating Related



Complex types are a conditional feature that implementations need **not** support



Object Type: Character Related



平时最常用的char，到底是unsigned还是signed不同平台表现不同

对于char类型来说，只用于存储ASCII字符比较安全



Object Type: Enumerated Type

```
enum WeekDay {Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday};
```

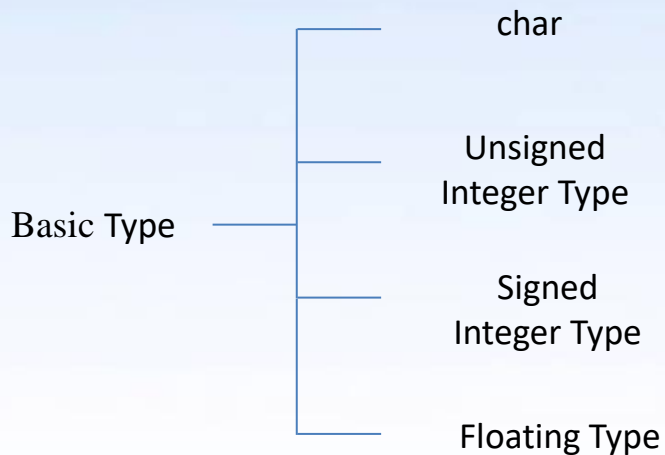
```
enum WeekDay d = Monday;
```

```
printf("%d\n", Thursday);
```

结果为3

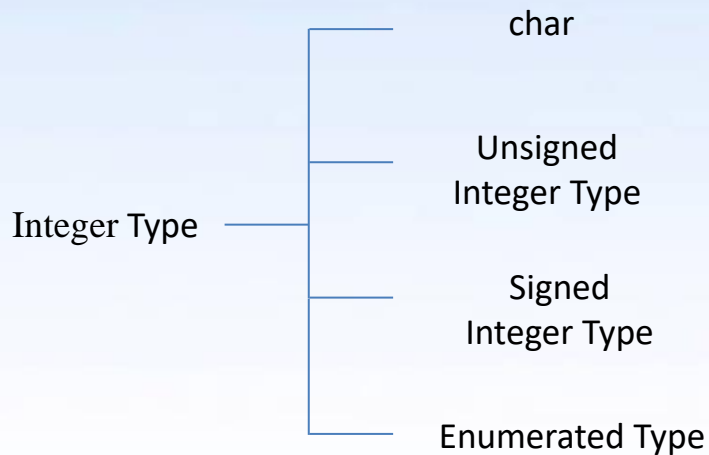


Object Type: Basic Type



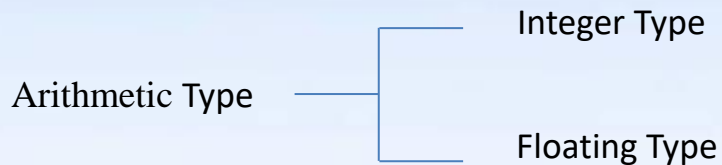


Object Type: Integer Type



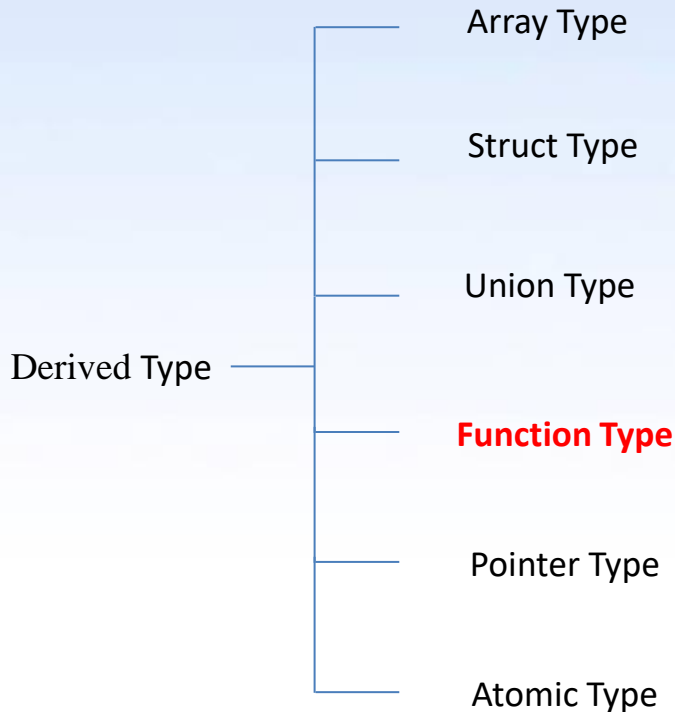


Object Type: Arithmetic Type





Derived Type



Derived Type可以递归地构造

These methods of constructing derived types can be applied recursively.



Array Type

- 1、Element Type (**T**): 需要是Object Type
- 2、Number of Elements (**N**)

Array of T, 例如: 假设**T**是int类型, **N**是5, 则该类型为int[5]

递归地构造

Array of T, 例如: 假设**T**是int[5]类型, **N**是3, 则该类型为int[3][5]

数组类型要从一维的角度去理解



int[5]真的是对象类型吗？

sizeof(5) vs. sizeof 5

sizeof(int) vs. sizeof int

sizeof(int[5]) vs. sizeof int[5]

sizeof用括号引导的只能是对象类型



Function Type

自己
总结

函数类型说明了：1、返回值类型；2、参数数量和类型

Type-name(argument1-type, argument2-type..., argumentn-type)

```
int func(int a, int b)
{
    // do something
    return 0;
}
```

func函数的函数类型是int(int, int)



Pointer Type

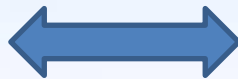
给定任何一个Type (T)，都有对应的一个指针类型 **Pointer to T**

1、Object Type

2、Function Type



Referenced Type



Pointer Type

int

int[5]

int*

int*

int(*)[5]

int**

任何指针类型也是一个**Object Type**，把指针类型当作普通**Object Type**看待

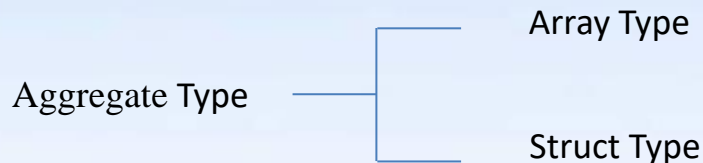


Scalar Type





Aggregate Type

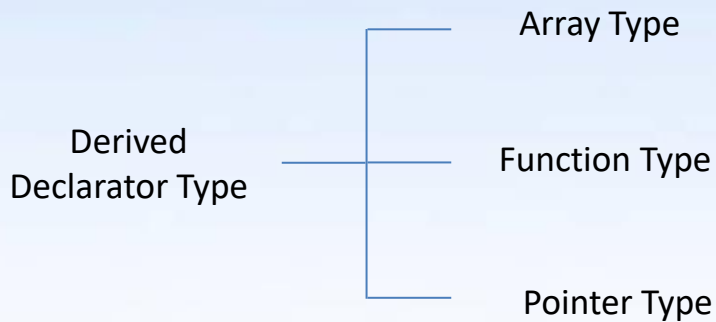


聚合类型不包括联合体，因为联合体类型任一时刻只能包括一个成员类型

Note that aggregate type does not include union type because an object with union type can only contain one member at a time



Derived Declarator Type





Incomplete Object Type

缺少必要信息确定Object大小的Type

- 1、缺少元素个数的数组，如extern int a[];
- 2、Variable Length Array（Length还未确定的时候）
- 3、包含Incomplete Object的struct/union
- 4、void



Type的限定符 (qualifier)

1、const

2、volatile

3、restrict

unqualified/qualified version是两种不同的Type，但大小，表示值，对齐方式一样

4、_Atomic

_Atomic(type-name)和type-name大小，表示值，对齐方式不一样

Atomic types are a conditional feature that implementations need **not** support



分配对象

在C语言中，分配一段内存（对象）的方法主要有两种（**还有其他的方法**）：

- 1、**变量声明**（例如：**局部变量、全局变量等**）
- 2、**使用malloc分配**

无论那种方法，内存分配的过程都是一样的，区别只是释放内存的不同需求

我们先来看变量声明的方法



基本概念：变量声明

在C语言中，声明一个变量具有两个基本要素：

- 1、对象类型 Object Type
- 2、变量名称（标示符） Identifier

对象类型约定了这段内存的类型和大小

变量名称则是这块内存的别名，用于识别定位这块内存

对象类型进一步分为：

- 1、非数组对象类型
- 2、数组对象类型

变量声明可以形式化成 $T O$



基本概念：非数组对象类型及声明

包括int, short, float, double, char, 结构体/联合体等等

声明方式如下：

对象类型为int, 变量名称为a

```
int a;
```

对象类型为float, 变量名称为b

```
float b;
```

对象类型为double, 变量名称为c

```
double c;
```

任何对象类型分配的内存大小可以通过sizeof这个操作符来获取，例如：

`sizeof(int), sizeof(float), sizeof(double)`



基本概念：数组对象类型

数组对象类型是多个同样的对象类型组成的一种**一维构造类型**
包含**两要素**：1、元素个数，2、元素的对象类型

1、数组对象类型的元素可以是非数组对象类型

`int[2]`包括2个元素，每个元素对象类型是`int`

`char[3]`包括3个元素，每个元素对象类型是`char`

任何数组对象类型分配的内存也可以通过`sizeof`这个操作符来获取

`sizeof(int[2]), sizeof(char[3])`



基本概念：数组对象类型

数组对象类型是多个同样的对象类型组成的一种**一维派生类型**

包含**两要素**：1、元素个数，2、元素的对象类型

2、数组对象类型的元素也可以是其他数组对象类型

`float[2][3]`包括2个元素，每个元素对象类型是`float[3]`

`double[2][3][4]`包括2个元素，每个元素对象类型是`double[3][4]`

仍然可以通过`sizeof`这个操作符来获取对象类型的大小

`sizeof(float[2][3]), sizeof(double[2][3][4])`

注意：C语言数组对象类型其实都是一维的



基本概念：数组对象类型的声明

数组对象类型申明方式如下：

对象类型为`int[2]`, 变量名称为`a`

```
int a[2];
```

对象类型为`float[2][3]`, 变量名称为`b`

```
float b[2][3];
```

对象类型为`double[2][3][4]`, 变量名称为`c`

```
double c[2][3][4];
```

需要能正确的识别出数组变量声明中的对象类型和变量名称



思考题

- 1、`char[3][5]`对象类型包括几个元素，每个元素是什么对象类型？
- 2、`sizeof(int[7][8])=?`

答案：

- 1、`char[3][5]`对象类型是一个一维数组对象类型，包括3个元素，其中每个元素的对象类型是`char[5]`
- 2、`sizeof(int[7][8])=7*sizeof(int[8])=7*8*sizeof(int)=7*8*4=224`字节



基本概念：指针对象类型

- 1、指针对象类型是一个非数组对象类型
- 2、任何一种对象类型，都有指向其的一个指针对象类型，反之亦然



int对象类型（Referenced Type）对应的指针对象类型是int*（Pointer Type）

int[2]对象类型（Referenced Type）对应的指针对象类型是int(*)[2]（Pointer Type）



基本概念：指针对象类型

指针对象类型也有相应的指向其的另一个指针对象类型

int*、float*、double*



int**、float**、double**

int(*)[2]、float(*)[2][3]、double(*)[2][3][4]

int(**)[2]、float(**)[2][3]、double(**)[2][3][4]

指向int*对象类型的指针对象类型是int**

指向int(*)[2]对象类型的指针对象类型是int(**)[2]



基本概念：指针对象类型

指针对象类型也可以用来构造数组类型，作为数组对象中元素的对象类型

`int*[2]`、`float*[2][3]`、`double*[2][3][4]`

`int*[2]`包括2个元素，每个元素对象类型是`int*`

`float*[2][3]`包括2个元素，每个元素对象类型是`float*[3]`

`double*[2][3][4]`包括2个元素，每个元素对象类型是`double*[3][4]`

提示：把`int*`、`float*`、`double*`当作一个整体来看待

把`int*`当作一个整体，比如看作PINT

`int*[2]`  `PINT[2]`

`PINT[2]`包括2个元素，每个元素类型是PINT



基本概念：指针对象类型

指针对象类型构成的数组对象也有其对应的指针对象

`int*[2]`、`float*[2][3]`、`double*[2][3][4]`  `int*(*)[2]`、`float*(*)[2][3]`、`double*(*)[2][3][4]`

提示：把 `int*`、`float*`、`double*` 当作一个整体来看待

把 `int*` 当作一个整体，比如看作 `PINT`

`int*[2]`  `PINT[2]`

`PINT[2]`  `PINT*(*)[2]`



基本概念：指针变量申明

对象类型是int*，变量名是a

```
int* a
```

对象类型是float*，变量名是b

```
float* b
```

对象类型是double*，变量名是c

```
double* c
```

对象类型是int(*)[2]，变量名是d

```
int (*d)[2]
```

对象类型是float(*)[2][3]，变量名是e

```
float (*e)[2][3]
```

对象类型是double(*)[2][3][4]，变量名是f

```
double (*f)[2][3][4]
```

对象类型是int*[2]，变量名是g

```
int* g[2]
```

对象类型是float*[2][3]，变量名是h

```
float* h[2][3]
```

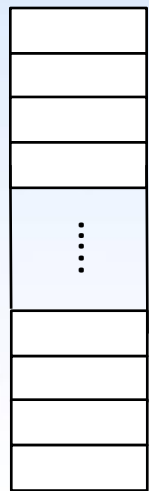
对象类型是double*[2][3][4]，变量名是i

```
double* i[2][3][4]
```

理解sizeof(int*), sizeof(int**), sizeof(int(*)[2])



指针对象类型的取值范围



第0个byte

第1个byte

第2个byte

第*i*个byte

编号

第 $2^{32}-1$ 个byte

32位机内存

任何指针对象的值就是内存中一个字节**编号**

取值范围就是[0 ~ 最大可寻址的字节编号]

对**32**位机器而言，可以寻址的字节编号范围为
[0 ~ $2^{32}-1$]

转换成**16**进制表示

[0x00000000 ~ 0xFFFFFFFF]

0x00000000=**NULL**



int* p vs. int *p: 空格放在哪更好?

```
int *p;  
int* p;
```

1、int *p和int* p是一样的
都定义了一个int*类型的变量，变量名为p

```
int *p, q;  
int* p, q;
```

2、int *p, q和int* p, q是一样的
定义了一个int*类型的变量，变量名为p，同时定义了一个int类型的变量，变量名为q

```
int *p, *q;
```

3、int *p, *q才是定义了两个int*类型的变量，变量名分别为p和q

```
int* p;  
int* q;
```

4、建议int*，因为int*是一个对象类型，因此多个int*对象的变量申明，建议一行一个



基本概念：对象类型总结

假设对象类型Object_Type (Obj_T)，变量名Name (N)

- 1、变量声明可以形式化定义为Obj_T N;
- 2、对象类型Obj_T对应的指针类型可以形式化记为Obj_T*

注1：具体变量声明语法和对象类型有关，例如：

int a; vs. int b[10];

注2：Obj_T*里面*的位置跟语法有关，例如：

int*; vs. int(*)[10];



基本概念：对象类型总结

Variable Category	Variable Type	Variable Name	Variable Declaration	Element Type	Corresponding Pointer Type	Size
Non-Array Variables	int	a	int a;	N/A	int*	4
	char	b	char b;	N/A	char*	1
	float	c	float c;	N/A	float*	4
	int*	d	int* d;	N/A	int**	4
	char*	e	char* e;	N/A	char**	4
	float*	f	float* f;	N/A	float**	4
Array Variables	int[2]	g	int g[2];	int	int(*)[2]	8
	char[2][3]	h	char h[2][3];	char[3]	char(*)[2][3]	6
	float[2][3][4]	i	float i[2][3][4];	float[3][4]	float(*)[2][3][4]	96
	int*[2]	j	int* j[2];	int*	int**(2)[2]	8
	char*[2][3]	k	char* k[2][3];	char*[3]	char**(2)[3]	24
	float*[2][3][4]	l	float* l[2][3][4];	float*[3][4]	float**(2)[3][4]	96



思考题

- 1、`int**[3]` 对象类型包括几个元素，每个元素是什么对象类型？
- 2、指向`int**[3]`对象类型的指针对象类型是什么？
- 3、如何声明一个变量，变量名为`a`，对象类型为`int**[3]`

答案（提示：`int**`也是一个对象类型，将`int**`当作一个整体来看）：

1、`int**[3]`对象类型是一个一维数组类型，包括3个元素，其中每个元素的对象类型是`int**`

2、`int**(*)[3]`

3、`int** a[3]`（建议）或`int **a[3]`



思考题

- 1、32位机里面，`int*`对象类型的取值需要几个字节来存储？
- 2、64位机里面，`int*`的取值范围是多大？需要几个字节来存储？

答案

- 1、32位机里面，`int*`的取值范围是`[0x00000000, 0xFFFFFFFF]`，因此4个字节来存储就够了，所以`sizeof(int*)=4`
- 2、64位机里面，理论上可以访问的字节有 2^{64} 个，因此字节编号范围为`[0x0000000000000000, 0xFFFFFFFFFFFFFFFF]`，需要8个字节来存储，因此`sizeof(int*)=8`

`sizeof(int*)`可以判断你的编译器是8/16/32/64位



了解一下typedef

C语言提供了一个typedef的方法来为对象类型提供别名

Original Type	Target Type	Typedef Declaration
char	INT1	typedef char INT1;
short	INT2	typedef short INT2;
int	INT4	typedef int INT4;
int*	PINT	typedef int* PINT;
char*	PCHAR	typedef char* PCHAR;
int(*)[2]	PAINT	typedef int (*PAINT)[2];
int**	PPINT	typedef int** PPINT;
int[2]	AINT	typedef int AINT[2];
int[2][3]	AAINT	typedef int AAINT[2][3];

1、typedef char INT1;
将char定义一个别名INT1
char c; vs. INT1 c;

2、typedef int AINT[2];
注意定义数组别名的语法
int a[2]; vs. AINT a;



int [3]真的是对象类型吗？

Original: int[3], Target: VINT

```
typedef int VINT[3];
```

通过typedef新定义出来的VINT是不是对象类型？

```
VINT e;
```

```
VINT* p = &e;
```

```
int (*q)[3] = &e;
```

思考：VINT是不是一种对象类型？

是



如何将 `int [2] [3]` 定义为 `VVINT` 类型?

方法1

Original : `int[2][3]`, Target: `VVINT`

```
typedef int VVINT[2][3];
```

```
VVINT g;  
VVINT* p = &g;  
int (*q)[2][3] = &g;
```

方法2

利用刚才定义的 `VINT` 的方法

- 1、`typedef int VINT[3];`
- 2、`typedef VINT VVINT[2];`

```
VVINT g;  
VVINT* p = &g;  
int (*q)[2][3] = &g;
```



int* vs. PINT

```
typedef int* PINT;
```

1、 int* p, q;

2、 PINT r, s;

1、 p是int*类型， q是int类型

2、 r和s都是PINT类型， 也就是int*类型



思考题

```
typedef int* PINT;  
PINT p;
```

```
PINT* q = &p;
```

标示符`q`对应那块内存，实际是什么对象类型？

答案

```
int** q = &p;
```