今天的任务：

- 强制类型转换的顺序

1.在表达式中，char 和 short 类型的值，无论有符号还是无符号，都会自动转换成 int 或者 unsigned int（如果 short 的大小和 int 一样，unsigned short 的表示范围就大于 int，在这种情况下，unsigned short 被转换成 unsigned int）。因为它们被转换成表示范围更大的类型，故而把这种转换称为“升级（promotion）”。

2.按照从高到低的顺序给各种数据类型分等级，依次为：

long double, double, float, unsigned long long, long long, unsigned long, long,
unsigned int 和 int

这里有一个小小的例外，如果 long 和 int 大小相同，则 unsigned int 的等级应位于 long 之上。char 和 short 并没有出现于这个等级列表，是因为它们应该已经被升级成了 int 或者 unsigned int。

3. 作为参数传递给函数时，char 和 short 会被转换成 int，float 会被转换成 double。使用函数原型可以避免这种自动升级。表达式中，有符号的遇到无符号的，会自动调整成有符号的来计算。

64位/32位 就是一个时间片 cpu可以执行的字节长。
任何实型运算（浮点型） 都要先转换成double 双精度 再做运算。
char short 都要先转成int

char short-int-unsigned-long-double

**********float类型%d输出**********

float a=7.5f;

如果用printf("%d",a);输出的是0。

但float型用%d输出是否一定是0呢，答案肯定不都是0;

为什么 7.5 用%d输出的是0？分析如下：

首先来了解下printf的输出格式，int 和 long int 都是32位的，用%d输出；float 、double都是%f输出，但 float 是32位的，double 是64位的，所以在参数传递的时候C语言统一将 float 类型数值传换为 double 类型再传入 printf 函数。如果是32位整型则输出格式为%lld。

下面来讲一下 float a=7.5f ; printf("%d",a)输出为0的情况：

%d只输出低32位的数据，并将这些32位二进制以十进制数输出，编译器首先将 7.5从float类型转换为double类型，7.5在内存中的存放方式是0x40f00000，转换成double类型在内存中的数据就是这个0x401e000000000000，这个内存数据可以很明显看出低32位全是0，而%d则只能截取到低32位，所以这个以%d输出7.5的数值当然是 0了。如大家不相信可以用%lld 输出看看，这个%lld就很读到低64位数据，读出的结果就是0x401e000000000000，在屏幕上看到一个很大的十进制数。

如果我一定要输出7.5在内存中的存放方法怎么办呢？

可以用printf("%d",*(int *)&a);这里做了一下处理，不是直接把a传进来，把a所在地址里的内容处理了一下，不管a是什么类型，只对地址进行操作，利用（int *）&a，将a所在地址中的内容0x40f00000直接当成 int 类型传给printf，int 的类型数据不会再转成double类型了，所以输出正常，这个只是针对浮点型数据只占低32位，如果输出64位还得用%lld格式控制输出。

如果用printf("%d",(int)a),输出行不行，这个强制类型转换只针对a的数据类型进行转换，7.5转换 int 类型是7，而上面的*(int *)&a，是对内存中的实际存储数据进行操作，蔽开数

据类型这一层面，只将这个数据0x40f00000直接转成int类型输出。而（int）a，要先看a的类型，C语言会根据所要数据类型，对内存存储的数据进行改变，以便可以用int类型正确解析内存数据。

如果用printf("%d",(float)a)，输出什么，输出的是0，这个只是将a的float类型还转成float类型，还是自动转成doube类型，传给printf函数。

为什么float非要转成double类型呢，因为printf格式控制浮点型输出只有%f，所以统一按doube类型输出，不像整型有32位的%d或%ld，64位的有%lld，这就将32位整型和64位整型用不同的格式控制分开了，而%f则没有，所以printf输出的浮点数其实是统一遍历了64位内存，如果float传入printf没有进行转换，那么printf输出高32位数据将不可预知，printf输出结果也就不正确了，因此传入printf的浮点数都会被编译器隐含转成double类型。

**********int类型%f格式输出**********

如果定义了int a=0x40f00000;用printf("%f",a)输出的结果是多少呢？

答案是0，至少我们看的屏幕上显示的是0.000000，实际值可不是0啊，只是我们显示的精度只能有15位小数，而实际的数据可能很小很小，0.0000....000几百个0后会有几个有效数据，我们分析一下。

首先C语言把a传进printf，因为a是整型，所以不会自动转成double型数据，直接将0x40f00000传进printf，而%f寻的是64位内存，也就是把0x0000000040f00000这个内存中的数据当成浮点型输出来，那浮点型的数据是多少呢，又是怎么存储的呢？

64位浮点数的存放方式：

63位	62~52位	51~0位
1个符号位	11个阶数	52个尾数

从0x0000000040f00000来看

1）符号位是0，表示正

2）阶数是0，表示-1023 + 1023 = 0，用指数表示：1.#*2^-1023，‘#’是代表尾数。

3）尾数就是，0x0000040f00000

4）浮点二进制表示

2#1.00000000000000000000010000001111000000000000000000*2^(-1023)，-1023次方可想而知有多小。

这就是为什么我们的int型数据用%f输出是0.000000的原因。

- ## C99是啥？

C99标准是 ISO/IEC 9899:1999 - Programming languages -- C 的简称 [1] ，是C语言的官方标准第二版。1999年12月1日，国际标准化组织（ISO）和国际电工委员会（IEC）旗下的**C**语言标准委员会（ISO/IEC JTC1/SC22/WG14）正式发布了这个标准文件 [2] 。

- ## c中有那些数据类型

C 数据类型

在 C 语言中，数据类型指的是用于声明不同类型的变量或函数的一个广泛的系统。变量的类型决定了变量存储占用的空间，以及如何解释存储的位模式。

C 中的类型可分为以下几种：

序号	类型与描述
1	基本类型： 它们是算术类型，包括两种类型：整数类型和浮点类型。
2	枚举类型： 它们也是算术类型，被用来定义在程序中只能赋予其一定的离散整数值的变量。
3	void 类型： 类型说明符 void 表明没有可用的值。
4	派生类型： 它们包括：指针类型、数组类型、结构类型、共用体类型和函数类型。

数组类型和结构类型统称为聚合类型。函数的类型指的是函数返回值的类型。在本章节接下来的部分我们将介绍基本类型，其他几种类型会在后边几个章节中进行讲解。

整数类型

下表列出了关于标准整数类型的存储大小和值范围的细节：

类型	存储大小	值范围
char	1 字节	-128 到 127 或 0 到 255
unsigned char	1 字节	0 到 255
signed char	1 字节	-128 到 127
int	2 或 4 字节	-32,768 到 32,767 或 -2,147,483,648 到 2,147,483,647
unsigned int	2 或 4 字节	0 到 65,535 或 0 到 4,294,967,295
short	2 字节	-32,768 到 32,767
unsigned short	2 字节	0 到 65,535
long	4 字节	-2,147,483,648 到 2,147,483,647
unsigned long	4 字节	0 到 4,294,967,295

int型数据，在内存中占4字节，4*8=32位，有一位是符号位，包含正负的2的32次个方数据
short int 型变量 在内存中占2字节， 也就是 8位。最多可以有2的4次方位 也就是16位的二进制，
但是第十六位是符号位 所以就只有15位存数据 2^15=32768
但是计算机没法录入32768 因为有正的就一定有负的。16位全是0代表0，若符号位是1，其余是0 代表负值的最大位，
即正值能代表的最大位+1.【所以左边界能取到负的2的位数-1的平方，右边只能取到2的位数-1的平方再-1了】】
输入的数值溢出时，计算机以0-32767-（-32768）-0这样的循环中的数字来带入计算。

注意，各种类型的存储大小与系统位数有关，但目前通用的以*64*位系统为主。以下列出了*32*位系统与*64*位系统的存储大小的差别（*windows* 相同）：

Windows vc12		Linux gcc-5.3.1		Compiler
win32	x64	i686	x86_64	Target
1		1	1	char
1		1	1	unsigned char
2		2	2	short
2		2	2	unsigned short
4		4	4	int
4		4	4	unsigned int
4		4	8	long
4		4	8	unsigned long
4		4	4	float
8		8	8	double
4		4	8	long int
8		8	8	long long
8		12	16	long double

为了得到某个类型或某个变量在特定平台上的准确大小，您可以使用 **sizeof** 运算符。表达式 *sizeof(type)* 得到对象或类型的存储字节大小。下面的实例演示了获取 int 类型的大小：

实例

```
#include <stdio.h> #include <limits.h> int main() { printf("int 存储大小 : %lu \n", sizeof(int)); r
eturn 0; }
```

当您在 Linux 上编译并执行上面的程序时，它会产生下列结果：

```
int 存储大小 : 4
```

浮点类型

下表列出了关于标准浮点类型的存储大小、值范围和精度的细节：

类型	存储大小	值范围	精度
float	4 字节	1.2E-38 到 3.4E+38	6 位小数
double	8 字节	2.3E-308 到 1.7E+308	15 位小数
long double	16 字节	3.4E-4932 到 1.1E+4932	19 位小数

头文件 float.h 定义了宏，在程序中可以使用这些值和其他有关实数二进制表示的细节。下面的实例将输出浮点类型占用的存储空间以及它的范围值：

实例

```
#include <stdio.h> #include <float.h> int main() { printf("float 存储最大字节数 : %lu \n", sizeof(f
loat)); printf("float 最小值: %E\n", FLT_MIN ); printf("float 最大值: %E\n", FLT_MAX ); printf("精
度值: %d\n", FLT_DIG ); return 0; }
```

当您在 Linux 上编译并执行上面的程序时，它会产生下列结果：

```
float 存储最大字节数 : 4
float 最小值: 1.175494E-38
float 最大值: 3.402823E+38
精度值: 6
```

---

void 类型

void 类型指定没有可用的值。它通常用于以下三种情况下：

序号	类型与描述
1	**函数返回为空** C 中有各种函数都不返回值，或者您可以说它们返回空。不返回值的函数的返回类型为空。例如 void exit (int status);
2	**函数参数为空** C 中有各种函数不接受任何参数。不带参数的函数可以接受一个 void。例如 int rand(void);
3	**指针指向 void** 类型为 void * 的指针代表对象的地址，而不是类型。例如，内存分配函数 void *malloc(size_t size); 返回指向 void 的指针，可以转换为任何数据类型。

C 基本语法　　C 变量

```
常用基本数据类型占用空间（64位机器为例）
 char :  1个字节
 int : 4个字节
```

笔记列表

```
 float: 4个字节
 double: 8个字节
```

基本类型书写

整数

```
 a, 默认为10进制 , 10 , 20。
 b, 以0开头为8进制, 045, 021。
 c., 以0b开头为2进制, 0b11101101。
 d, 以0x开头为16进制, 0x21458adf。
```

小数

单精度常量: 2.3f 。

双精度常量: 2.3, 默认为双精度。

字符型常量

用英文单引号括起来, 只保存一个字符'a'、'b' 、'*' , 还有转义字符 '\n' 、'\t'。

字符串常量

用英文的双引号引起来 可以保存多个字符: "abc"。