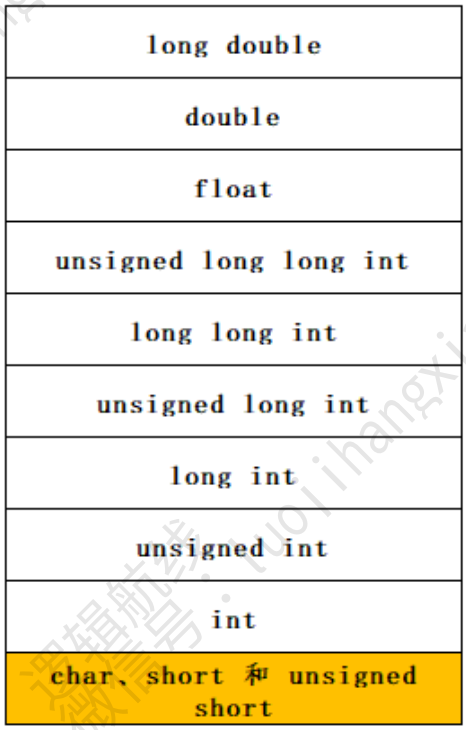
类型转换

当运算符的操作数具有不同的数据类型时，C++ 会自动将它们转换为相同的数据类型。当它这样做时，遵循一组规则。

升格

当数据从低容量类型向高容量类型转化时，它们遵照以下规则。

就像军队的军官有军阶一样，数据类型也可以按等级排名。如果一个数字数据类型可以容纳的数字大于另一个数据类型，那么它的排名就高于后者。例如，float 类型就超越了 int 类型，而 double 类型又超越了 float 类型。下图列出了从高到低排列的数据类型。



升格规则 1：无论何时在数学表达式中使用这些数据类型的值，char、short、unsigned short 都将自动升级为 int 类型。

升格规则 2：当运算符使用不同数据类型的两个值时，较低排名的值将被升级为较高排名值的类型。在下面的表达式中，假设 years 是一个 int 变量，而 interestRate 是一个 double 变量，在乘法发生之前，years 中的值将升级为 double 类型：

```
years * interestRate
```

升格规则 3：当表达式的最终值分配给变量时，它将被转换为该变量的数据类型。在下面的语句中，假设 area 是一个 long int 长整型变量，而 length 和 width 都是 int 整型变量：

```
area = length * width;
```

因为存储在 length 和 width 中的值是相同的数据类型，所以它们都不会被转换为任何其他数据类型。但是，乘法的结果将被升级为 long int 类型，这样才可以存储到 area 中。

小提示：

1、将整型数据赋值给浮点型变量时，数值不变，但是以指数形式存储。

a) double a= 4; //a=4.000000

2、字符型数据可以赋值给整型变量，此时存入的是字符的 ASCII 码。

降格

降格规则 1：将一个 int，short 或 long 型数据赋值给一个 char 型变量，只将低 8 位原封不动的送到 char 型变量中。

降格规则 2：将有符号型数据赋值给长度相同的无符号型变量，连同原来的符号位一起传送。

降格规则 3：将 double 型数据赋值给 float 型变量时，注意数值范围溢出。参见上文浮点型在内存中的存储。双精度尾数位有 52 位，单精度的尾数位有 23 位。转换时，按照从高到低的位数依次写入，截取最低位。

降格规则 4：float,double 传入 int 时，直接截断小数部分。例：

```
int a = 4.6 + 3;    //a = 7;
```

基本运算

算术运算符包括以下符号：

运算符	术语	示例	结果
+	正号	+3	3
-	负号	-3	-3
+	加	10 + 5	15
-	减	10 - 5	5
*	乘	10 * 5	50
/	除	10 / 5	2
%	取模(取余)	10 % 3	1
++	前置递增	a=2; b=++a;	a=3; b=3;
++	后置递增	a=2; b=a++;	a=3; b=2;
--	前置递减	a=2; b=--a;	a=1; b=1;
--	后置递减	a=2; b=a--;	a=1; b=2;

示例代码

```
#include <bits/stdc++.h>

using namespace std;

int main(int argc, char **argv) {
        cout << 3 + 3 << endl;  //加法计算，输出 6
        cout << 3 - 3 << endl;  //减法计算，输出 0
        cout << 3 * 3 << endl;  //乘法计算，输出 9
        cout << 3 / 3 << endl;  //除法计算，输出 1
        cout << 3 % 3 << endl;  //取余计算，输出 0
        return 0;
}
```

小提示：在除法中整数和整数相除，结果必然为整数。如果想获得正确的浮点数，必须把其中的一个数值变成浮点数。例如：

```
#include <bits/stdc++.h>

using namespace std;

int main(int argc, char **argv) {
        cout << 3 / 2 << endl;  //输出 1
        cout << 3.0 / 2 << endl;  //输出 1.5
        cout << 3 / 2.0 << endl;  //输出 1.5
        cout << 3.0 / 2.0 << endl;  //输出 1.5
        return 0;
}
```

变量空间大小

我们使用 size 函数来获取一个变量所占的空间大小，如下图所示：

```
#include <iostream>
using namespace std;
int main(int argc, char** argv) {
        int a;
        cout<<sizeof(a);  //输出 4，即整形变量占 4 个字节
        return 0;
}
```

课后练习

1、【2019 年 CSP-S1 提高级初赛 1】若有定义 int a = 7, float x = 2.5, y = 4.7;，则表达式 x + a % 3 * (int) (x + y) % 2 的值是（ ）
A 0.000000　　B 2.750000　　C 2.500000　　D 3.500000

2、【2019 年 CSP-S1 提高级初赛 5】设变量 x 为 float 型且已赋值，下列哪条语句能将 x 中的数值保留到小数点后两位，并将第三位四舍五入（ ）。
A x=（x*100+0.5）/100.0
B x= int（x*100+0.5）/100.0
C x=（x/100+0.5）/100.0
D x= x*100+0.5/100.0

3、【2014 年提高组初赛】若有变量 int a, float x,y 且 a=7, x=2.5, y=4.7，则表达式 x+a%3*(int)(x+y)%2/4 的值大约是（ ）。
A 2.50000　　B 2.750000　　C 3.500000　　D 0.000000

4、【2013 年提高组初赛 13】把 64 位非零浮点数强制转换成 32 位浮点数后，不可能（ ）
A 大于原数　　B 小于原数　　C 等于原数　　D 与原数符号相反

参考答案

1、D　解析：注意计算顺序。浮点型默认输出 6 位小数。
2、B　解析：乘以 100 实现了保留 2 位小数,加 0.5 是对第 3 位四舍五入，强制转成 int 型后,会保留整数。
3、A
4、D　解析：因为把 64 位非零浮点数强制转换成 32 位浮点数后，只是丢弃了更高的数据存储精度（有效数字位数少了），不会影响数值的符号及前 7 位有效数字的。
大于的情况-1.1111111111 变成-1。
等于的情况-1.0000000000 变成-1。
小于的情况 1.23456789 变成 1。