```cpp
#include <iostream>
#include <iomanip>
#include <cstdio>
#include <cstdlib>
#include <math.h>
using namespace std;

void precision();
void common_fun();
void ops();
void ascii();

int main()
{
  // ascii();
  // precision();
  // common_fun();
  // ops();
  return 0;
}

void ascii()
{
  /*
    整数
      short
      int
      long long int
      unsigned int
    小数（浮点）
      float
      double
    字符
      char
    他们之间可以互相转化，但是注意转化是要留意数据的大小是否会溢出(大小超出合理范围)
  */
  int x = 5;
  short y = x;
  cout << y << endl;

  x = 65;
  char c = x; // 'A' char 的转化遵循ascii表格

  x = -100;
  unsigned int u = x; // unsigned int 不能存储负数 所以虽然不报错 但是答案不会正确
  cout << u << endl;
```

```
  x = 2147483647;
  short p = x; // short 的范围是 -32768 ~ 32767 所以这里也会溢出
  cout << p << endl;
  // bool 类型转化
  cout << bool(0) << endl;

  float g;
  g = 3 / 2;
  cout << "g: " << g << endl;

  /*
    强制类型转化
    (数据类型)(表达式)
  */
  // 0.2 + 0.3 = 0.5 但是 类型转换时是保留整数 所以两个都是0
  x = (int) (0.2 + 0.3) + (int) (0.4 + 0.5);
  cout << x << endl; // 0
  x = 0.2 + 0.3 + 0.4 + 0.5;
  cout << x << endl; // 1
};

void precision()
{
  /*
    double 15位有效数字
    float 7位有效数字
  */
  float f = 0.1284427;
  double x = 12.23452356455749;
  /*
    c语言保留精度 %.nlf 123.43645745 7 123.436
    m是总长度 (包括整数,小数点和小数部分)
    n是小数点后的位数
    lf 是double类型特有的输入输出符号标志 告诉程序这里按照double类型进行解析
    float - f
    double - lf 15个有效数字
    int - d
    char - c
    string - s
    short - h
    long long int - ld
  */
  printf("%.12lf\n", x);
  // 注意如果这里如果保留30位小数 并不会成功 从第16位开始就不准确了，因为double类型只能保留15位有效数字
  printf("%.30lf\n", x);
  // c++ 默认输出保留6位有效数字
  cout << x << endl;
  // c++ 保留精度 所有有效数字
```

```
  cout << setprecision(10) << x << endl; // setprecision 可以设置输出的精度
  // c++ 保留精度 小数部分
  cout << fixed << setprecision(10) << x << endl; // fixed 表示保留小数部分10位

  printf("%10.7lf\n", x);
}

void common_fun(){
  printf("%d\n", abs(-10));
  cout << abs(-10) << "\n"; // abs 取绝对值
  cout << pow(2, 3) << endl; // pow 求幂 次方 2^3 = 8
  cout << sqrt(4) << endl; // sqrt 求平方根 2^2 = 4 4的平方分跟就是2
  cout << ceil(4.1) << endl; // ceil 向上取整
  cout << floor(4.9) << endl; // floor 向下取整
  cout << round(4.5) << endl; // round 四舍五入
  cout << max(1, 2) << endl; // max 取最大值
  cout << rand() << endl; // rand 生成随机数 试一下加上这个头文件 #include <cstdlib>
}

void ops(){
  /*
  算术运算符 + - * / %取余数 ++自增 --自减
  = 赋值运算符
  关系运算符 == != > < >= <=
  逻辑运算符 && || !
  3 == 4 => false => 0
  !(3 == 4) && (3 > 2) false
  (3 == 4) || (3 > 2) true=1
  位运算符  & | ^ ~ << >>  3 & 4
  优先级 () > ! > ~ > 算术运算符>关系运算符>逻辑运算符>位运算符
  */
  // 位运算符 &
  int a = 3, b = 4;
  cout << (a & b) << endl; // 3 & 4 = 0
  /* & 两边都为1时才为1，  其他时候都是0
   3 -> 011
        &&&
   4 -> 100
        000 -> 0
  */
  cout << (a | b) << endl; // 3 | 4 = 7
  /* | 两边有一个为1就为1
   3 -> 011
        |||
   4 -> 100
        111 -> 7
  */
  cout << (a << 2) << endl; // 3 << 2 = 12
```

```
  /* << 左移运算符
   3 -> 011
        <<2 指把整体向左移动两位 空的用0补充
       01100 -> 12
  */
  cout << (b >> 2) << endl; // 4 >> 2 = 1
  cout << (5 >> 1) << endl; // 5 >> 1 = 2
  /* >> 右移运算符
   4 -> 100
        >>2 指把整体向右移动两位 空的用0补充
        001 -> 1
   5 -> 0101
        >>1 指把整体向右移动一位 空的用0补充
        00010 -> 2
  */
  return;
}
```