进制转换

10 进制和 r 进制互转

1108. 正整数 N 转换成一个二进制数

输入一个不大于 32767 的整数 n ，将它转换成一个二进制数。

输入

输入只有一行，包括一个整数 n (0 ≤ n ≤ 32767)。

输出

输出只有一行。

输入

100

输出

1100100

输入

0

输出

0

```
// 定义字符串存储 N 转换的二进制数
// 用短除法除 2 取余，将余数逆序存入字符中 s.
#include <bits/stdc++.h>//1-1108-1  正整数 N 转换成一个二进制数 宋老师视频教程
using namespace std;
string s;// 存储二进制
int n,x;
char c;
int main()
{
	cin>>n;
	while(n != 0)
	{
		x = n % 2;
		c = x + '0';

		s = c + s;// 结果逆序连接为字符串
		n = n / 2;
	}
	if(s == ""){
		cout<<0;
	}
	else{
		cout<<s;
	}
	return 0;
}
```

```cpp
#include<bits/stdc++.h>//1-1108-2  javacn 除 2 取余，结果倒过来连成字符串：
using namespace std;
int n; //10 进制转 2 进制
string r = "";// 存放转换结果
int main(){
	cin>>n;
	// 当 n!=0 循环
	while(n != 0){
		//cout<<n%2;// 取余
		// 将 n%2 的结果转换为字符 + 到 r 前面
		r = char(n%2+'0') + r;
		n=n/2;// 除 2
	}
	if(r == "") cout<<0;	// 特判输入为 0 的情况
	else cout<<r;
	return 0;
}

#include<iostream>//1-1108-3  jiangyf70 正整数 N 转换成一个二进制数
using namespace std;
string i2b(int n, string s){
    if(n)
    {
        s = char(n % 2 + '0') + s;
        return i2b(n / 2, s);
    }
    return s;
}
int main(){
    int n;
    cin >> n;
    if(n == 0) cout << 0;
    else cout << i2b(n,"");
    return 0;
}
```

1290. 二进制转换十进制

请将一个 25 位以内的 2 进制正整数转换为 10 进制！

输入

一个 25 位以内的二进制正整数。

输出

该数对应的十进制。

输入

1111111111111111111111111

输出

16777215

```
// 思路：从最低位开始(s.size()-1)，倒过来计算（按权展开）s[i] - '0'
// 准备变量 t 是 2 的 n 次方，t=1 每循环一次，t = t * 2
#include <bits/stdc++.h>//2-1290-1     二进制转十进制     宋老师视频教程
using namespace std;
string s;// 存储二进制
int r,t=1,i;//t 表示权重
int main()
{
      cin>>s;
      for( i = s.size() - 1;i >= 0;i--)
      {
            r = r + (s[i] - '0') * t;
            t = t * 2;
      }
      cout<<r;
      return 0;
}
```

二进制转换十进制

按权展开：

```
#include<bits/stdc++.h>//2-1290-2  javacn
using namespace std;
int main() {
	string s;
	cin>>s;
	int len = s.size();
	long long sum = 0,t = 1;//t 代表权重
	//11010[2 进制 ]=1X2^4+1X2^3+0X2^2+1X2^1+0X2^0=26[10 进制 ]
	// 二进制数转换成十进制数的方法是按权展开
	//2^4 表示 2 的 4 次方
	for(int i=len-1; i>=0; i--) {
		sum = sum + (s[i]-'0') * t;
		t = t * 2;
	}
	cout<<sum;
	return 0;
}
```

1289. 正整数 n 转换为 16 进制

请从键盘读入一个非负整数 n（ n 是一个不超过 18 位的非负整数），将 n 转换为 16 进制！

注意：16 进制即逢 16 进 1，每一位上可以是从小到大为 0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F 共 16 个大小不同的数，即逢 16 进 1，其中用 A，B，C，D，E，F 这六个字母来分别表示 10，11，12，13，14，15。

如：60 的十六进制为 3C。（字母请用大写）

输入：一个不超过 18 位的非负整数 n 。

输出：该数的十六进制值。

	// 解法二：用字符存储十六进制对应的字符，简化 16 进制转为
输入	字符的过程
100000000000	// 注意：
输出	//int 最多表达到 2^{31}-1. 10 位整数；
174876E800	//long long 最多表达到 2^{63}-1. 19 位整数；

```
#include <bits/stdc++.h>//3-1289-2    宋老师视频教程
using namespace std;
long long n,x;//n 是一个不超过 18 位的正整数
string s;
string t = "0123456789ABCDEF";
int main(){
      cin>>n;
      while(n != 0)
      {
            x = n % 16;
            // 将 x%16 转换为字符逆序存入 s
            s = t[x] + s;
            n = n / 16;
      }
      if(s == ""){            cout<<0;     }
      else{       cout<<s;     }
      return 0;
}
```

```
// 思路：除 16 取余！
// 逆序存储到字符串中要注意：
// 整数 0-9，转换为字符 "0"~"9"，x + '0'
// 整数 10-15，转换为字符 "A"~"F"，x + 'A' - 10
// 解法一：分别判断 n%16 结果在 0-9 及 10-15 的哪个范围，分别转换为对应的字符
#include <bits/stdc++.h>//3-1289-1  正整数 n 转换为 16 进制   宋老师视频教程
using namespace std;
long long n,x;//n 是一个不超过 18 位的正整数
string s;
char c;
int main()
{
	cin>>n;
	while(n != 0)
	{
		x = n % 16;
		//cout<<x<<endl;
		// 将 x 转换为字符逆序存入字符串 s
		//x0-9 -> '0'-'9'
		//x10-15 -> 'A'-'F'
		if(x < 10){	c = x + '0';		}
		else	{	c = x + 'A' - 10;	}
		s = c + s;
		n = n / 16;
	}
	if(s ==""){
		cout<<0;
	}
	else{
		cout<<s;
	}
	return 0;
}
```

正整数 n 转换为 16 进制

除 16 取余，结果倒过来连成字符串：

```
#include<bits/stdc++.h>//3-1289-3  javacn
using namespace std;

long long n;
string r = "";// 存放 16 进制
string t = "0123456789ABCDEF";// 存放 0~15 对应的字符

int main(){
	cin>>n;
	// 特判输入为 0 的情况
	if(n == 0){
		cout<<0;
		return 0;// 停止函数
	}

	// 短除法
	while(n != 0){
		//n%16：如果在 0~9 之间，转换为字符 '0'~'9'
		// 如果在 10~15 之间，转换为字符 'A'~'F'
		r = t[n%16] + r;
		n = n / 16;// 除 16
	}

	cout<<r;
	return 0;
}
```

1292. 十六进制转十进制

请将一个不超过 10 位的十六进制正整数转换为十进制整数。

输入

10 位以内的十六进制正整数，如果该十六进制中有字母，字母用大写英文字母表示。

输出

该数对应的十进制整数。

输入
2ECF
输出
11983

// 思路：逆序计算，按权展开！
// 从 s 中获取每一位 s[i] 是字符，要转换为实际的整数！
//s[i]: '0'~'9', s[i] - '0'
//s[i]: 'A'~'F', s[i] - 'A' + 10

```
#include <bits/stdc++.h>//4-1292-1  十六进制转十进制  宋老师视频教程
using namespace std;
string s;
long long r,t = 1,i;//t 表示权重，16 的 i 次方
int main(){
	cin>>s;
	// 逆序计算，按权展开
	for( i = s.size() - 1;i >= 0;i--)
	{
		if(isdigit(s[i]))// 如果 s[i]: '0'~'9'
		{
			r = r + (s[i] - '0') * t;
		}
		else
		{
			r = r + (s[i] - 'A' + 10) * t;
		}
		t = t * 16;
	}
	cout<<r;
	return 0;
}
```

思路：逆序计算，按权展开！

从 s 中获取每一位 s[i] 是字符，要转换为实际的整数！

s[i]：'0'~'9'，s[i] - '0'

s[i]：'A'~'F'，s[i] - 'A' + 10

```
#include<bits/stdc++.h>//4-1292-2  javacn
using namespace std;
string s;
long long r = 0,t = 1;//t 表示权重（16 的次方）
int main() {
      cin>>s;
      for(int i = s.size() - 1;i >= 0;i--)// 逆序循环字符串，从最低位开始计算
      {           // 如果当前位是数字字符 0~9
            if(isdigit(s[i])){      r = r + (s[i] - '0') * t;          }
            else  {     r = r + (s[i] - 55) * t;      } // 如果是字母 10~15

            t = t * 16;       // 权重提升
      }
      cout<<r;
      return 0;
}
#include<iostream>//4-1292-3  jiangyf70  十六进制转十进制
using namespace std;
int main()
{
    string s;
    cin >> s;
    long long n = 0, k = 1;;
    for(int i = s.size() - 1; i >= 0; i--)    {
        if(s[i] - '0' <= 9) n = n + (s[i] - '0') * k;
        else n = n + (s[i] - 55) * k;
        k *= 16;
    }
    cout << n;
    return 0;
}
```

1386 - 小丽找半个回文数

小丽同学在编程中学到了回文数的概念，如果一个数正过来读和反过来读是同一个数，那么这个数就是回文数；比如：2、5、8、66、121、686、12321 都是回文数，小丽发现，这样的数不算多。

于是小丽有个想法，如果这个数不是回文数，但这个数在 2 进制或者 16 进制下是回文数，就算这个整数是半个回文数，比如 417 并不是回文，但 417 对应的 16 进制数是 1A1 是回文数，因此 417 算半个回文数。

请你编程帮助小丽找符合条件的半个回文数。

输入

第一行是一个整数 n（5 ≤ n ≤ 100）； 第二行是 n 个整数（这些整数都是 $[0,10^8]$ 之间的整数）；

输出

所有符合条件的半个回文数，每行一个。

输入

5

121 417 27 100 21

输出

417

27

21

```
/*
如果这个数在 10 进制下不是回文数
但这个数在 2 进制或者 16 进制下是回文数
判断整数 n 在 d 进制下是否是回文
除 d 对 d 取余数，将余数存入数组，判断数组是否是回文
*/
```

```cpp
#include <bits/stdc++.h>      //1386-1  宋老师视频教程
using namespace std;
bool huiwen(int n,int d){
      bool r = true;// 假设是回文
      int a[1000] = {0};// 初始化为 0，存 n 转 d 进制后的每一位
      int k = 0;
      while(n != 0)
      {
          a[k] = n % d;
          k++;
          n = n / d;
      }
      // 判断回文：循环数组长度一半，判断对称位置是否不等
      for(int i = 0;i < k / 2;i++)
      {
            if(a[i] != a[k - i - 1])
            {
                  r = false;
                  break;
            }
      }
      return r;
}
int main(){
      int a[110],n,i;
      cin>>n;
      for(i = 0;i < n;i++)        cin>>a[i];
      for(i = 0;i < n;i++){   // 遍历每个数，判断是否是半个回文
            if(huiwen(a[i],10) == false && (huiwen(a[i],2) == true ||
huiwen(a[i],16) == true)){
                  cout<<a[i]<<endl;
          }
      }
      return 0;
}
```

```cpp
#include <bits/stdc++.h>//1386-2  解法二：每读入一个数，就判断其是否是半个回文数
using namespace std;                                    //          宋老师视频教程
bool huiwen(int n,int d){
	bool r = true;// 假设是回文
	int a[1000] = {0};// 初始化为 0，存 n 转 d 进制后的每一位
	int k = 0;
	while(n != 0)
	{
		a[k] = n % d;
		k++;
		n = n / d;
	}
	    // 判断回文：循环数组长度一半，判断对称位置是否不等
	for(int i = 0;i < k / 2;i++)
	{
		if(a[i] != a[k - i - 1])
		{
			r = false;
			break;
		}
	}
	return r;
}
int main(){
	int n,x,i;
	cin>>n;
	for(i = 0;i < n;i++)
	{
		cin>>x;
	    if(huiwen(x,10) == false &&(huiwen(x,2)==true || huiwen(x,16)== true))
	      cout<<x<<endl;
	}
	return 0;
}
```

正整数 n 转换为 8 进制

短除法：除 8 取余，注意本题的 n 要用 long long。

```
#include <bits/stdc++.h>//5-1288 javacn
using namespace std;

long long n;
string r = "";// 存储 n 对应的 8 进制

int main(){
	cin>>n;
	while(n != 0){
		// 倒过来连成字符串
		r = char(n%8+'0') + r;
		n = n / 8;
	}

	if(r == "") cout<<0;
	else cout<<r;
	return 0;
}
```

八进制转十进制

按权展开：

```
#include<bits/stdc++.h>//6-1291   javacn
using namespace std;
int main() {
	// 只能以字符串类型读入
	string s;
	cin>>s;
	int len = s.size();
	long long sum = 0,t = 1;//t 代表权重
	// 类似于 2 进制转 10 进制    8 进制转 10 进制权重为 8
	for(int i=len-1; i>=0; i--) {
		sum = sum + (s[i]-'0') * t;
		t = t * 8;
	}
	cout<<sum;
	return 0;
}
```

小丽找潜在的素数

分别定义函数：判断素数、将二进制转十进制。

```
#include <bits/stdc++.h>//7-1405    javacn
using namespace std;
// 定义函数判断素数
bool sushu(int n){
	bool r = true;// 假设是素数
	for(int i = 2; i <= sqrt(n);i++){
		if(n % i == 0){
			r = false;
			break;
		}
	}
	if(n <= 1) r = false;    // 特判特殊情况
	return r;
}
int jinzhi(string s){// 将二进制转十进制
	int r = 0,t = 1;
	for(int i = s.size()-1;i>=0;i--){    // 倒过来计算，按权展开
		r = r + (s[i] - '0') * t;
		t = t * 2;
	}
	return r;
}
int n,c = 0;
string s;
int main(){
	cin>>n;
	for(int i = 1;i <= n;i++){
		cin>>s;
		if(sushu(jinzhi(s))==true){    c++;   }
	}
	cout<<c;
	return 0;
}
```

小 X 转进制

定义函数判断整数 x 在 m 进制下是否是回文。

```
#include <bits/stdc++.h>//8-1547  javacn
using namespace std;
int n,m,c = 0;
bool huiwen(int x){
	// 将 x 转换为 m 进制
	string t = "0123456789ABCDEF";
	string r = "";
	while(x != 0){
		r = t[x%m] + r;
		x = x / m;
	}
	// 判断 r 是回文
	string r2 = r;
	reverse(r.begin(),r.end());

	if(r2 == r){	return true;	}
	else{ return false;	}
}

int main(){
	cin>>n>>m;
	// 循环 1~n，判断每个数的平方在 m 进制下是否回文
	for(int i = 1;i <= n;i++){
		if(huiwen(i*i)==true){
			c++;
		}
	}

	cout<<c;
	return 0;
}
```

```
/*
1 到 N 中有多少个整数的平方在 M 进制下是回文数呢

思路：
1. 将每个数的平方，转 M 进制
2. 判断回文
*/
```

除 D 取余法：

```
#include <bits/stdc++.h>//9-1415-1  javacn
using namespace std;
string s = "0123456789ABCDEF",r;
//短除法求解的余数从 s 中取  例如余数 3，则为 s[3]
int n,d;
int main() {
	cin>>n>>d;
	if(n == 0) {
		cout<<0;
		return 0;
	}
	while(n != 0) {
		r = s[n%d] + r;
		n = n / d;
	}
	cout<<r;
	return 0;
}
```

10 进制转 D 进制

```
#include <bits/stdc++.h>//9-1415-2  yhh133
#define ll long long
using namespace std;
string change(ll n,int x){
	string s;
	while(n){
		if(n%x>=10) s+=n%x-10+'A';
		else s+=n%x+'0';
		n/=x;
	}
	reverse(s.begin(),s.end());
	return s;
}
int main() {
	ll n;
	int k;
	cin>>n>>k;
	if(n==0) cout<<0;
	else cout<<change(n,k);
	return 0;
}
```

2 进制和 8、16 进制互转

二进制转换八进制

```
#include<bits/stdc++.h>//1-1293  javacn
using namespace std;
int main(){
	string s;
	cin>>s;
	// 使用循环在字符串 s 前面补 0 使得字符串 s 的长度是 3 的倍数
	while(s.size()%3!=0){
		s = '0' + s;
	}
	int len = s.size(),i;
	//3 个 2 进制数转成 1 个 8 进制数 例如：100 14+02+01=4
	for(i=0;i<len-2;i=i+3){
		cout<<(s[i]-'0')4 + (s[i+1]-'0')2 + (s[i+2]-'0')1 ;
	}
	return 0;
}
```

1294. 二进制转十六进制

请将一个不超过 100 位的二进制数转换为十六进制数！

输入：一个不超过 100 位的二进制整数。

输出：该数对应的十六进制数。

输入：1100100111101111100000100010011

输出：C9EF8213

```
#include <bits/stdc++.h>//2-1294-1  宋老师视频教程
using namespace std;// 将 4 位的二进制转换为 1 位的 16 进制
char num(string s){     //1101：从最低位开始按权展开，转换为 10 进制
      int r = 0,i,t = 1;       // 再转换为 16 进制的字符
      for(int i = s.size() - 1;i >= 0;i--)
      {
            r = r + (s[i] - '0') * t;
            t = t * 2;
      }
      char c;// 存储 1 位的 16 进制字符
      if(r < 10)       c = r + '0';
      else       c = r + 'A' - 10;
      return c;
}
int main(){
      string s,t;// 存放二进制
      cin>>s;
      if(s.size() % 4 == 1)      {       s = "000" + s;       }      // 补 0
      else if(s.size() % 4 == 2){       s = "00" + s;       }
      else if(s.size() % 4 == 3){       s = "0" + s;       }
      // 每 4 位一格，将 4 位的二进制转换为对应的 16 进制
      for(int i = 0;i < s.size();i = i + 4){
          t = s.substr(i,4);
          cout<<num(t);
      }
      return 0;
}
```

二进制转十六进制

思路：

第一步：判断字符串的长度是否是 4 的倍数，如果不是，则补 0。

第二步：每 4 位 2 进制转换为 1 位的 16 进制输出。

```
#include <bits/stdc++.h>//2-1294 -2 javacn
using namespace std;

/*
1. 判断字符串长度是否是 4 的倍数，如果不是补 0
2. 将每 4 位的 2 进制转换为 1 位的 16 进制，输出
*/
string s;// 存放 2 进制
string t = "0123456789ABCDEF";
int main(){
	cin>>s;
	// 如果长度不是 4 的倍数，补 0
	//11010
	if(s.size()%4==1) s = "000" + s;
	else if(s.size()%4==2) s = "00" + s;
	else if(s.size()%4==3) s = "0" + s;

	//cout<<s<<endl;
	// 计算
	//0101,1010
	for(int i = 0;i < s.size();i = i + 4){
		// 将 4 位的 2 进制：s[i] s[i+1] s[i+2] s[i+3]
		// 转 16 进制
		int x = (s[i+3]-'0')*1+(s[i+2]-'0')*2+(s[i+1]-'0')*4+(s[i]-'0')*8;
		cout<<t[x];
	}
	return 0;
}
```

/*1359 - 【基础】八进制转换二进制

题目描述　请将一个 100 位以内的 8 进制整数转换为 2 进制整数！

输入　100 位以内的 8 进制整数　　输出　该数对应的 2 进制整数

样例

输入 12376532347173217361

输出 1010011111110101011010011100111001111011010001111011110001

*/

将每位的 8 进制转换为 3 位的 2 进制：

```
#include<bits/stdc++.h>//3-1359-1  javacn
using namespace std;
int main() {
	string s,r;
	int x;
	// 定义字符串数组，枚举所有情况
	string t[8]= {"000","001","010","011","100","101","110","111"};
	cin>>s;
	// 调用函数，求每位 8 进制对应的二进制
	for(int i=0; i<s.size(); i++) {
		x=s[i]-'0';
		r=r+t[x];
	}
	// 删除字符串前面的 0
	while(r[0]=='0') {
		r.erase(0,1);
	}

	if(r == "") cout<<0;
	else cout<<r;
	return 0;
}
```

八进制转换二进制

```
#include <bits/stdc++.h>//3-1359-2    xingdian119
using namespace std;
long long n2shi(string s, int jin) {  // 其他进制数转为 10 进制
  long long r = 0, t = 1;             // t: 表示权重
  for (int i = s.size() - 1; i >= 0; i--) {
    r += (s[i] - '0') * t;
    t *= jin;
  }
  return r;
}
string shi2n(long long n, int jin) {  // 10 进制转为其他进制数
  string t = "0123456789ABCDEF", r = "";
  if (n == 0)
    r = '0';
  else {
    while (n != 0) {
      r = t[n % jin] + r;
      n /= jin;
    }
  }
  return r;
}
int main() {
  string s;
  cin >> s;
  cout << shi2n(n2shi(s, 8), 2);
}
```

1306 . 十六进制转二进制

请将一个不超过 100 位的十六进制数转换为二进制数!

输入：一个不超过 100 位的十六进制整数。

输出：该数对应的二进制数。

输入 :123456789ABCDEF

输出 :100100011010001010110011110001001101010111100110111101111

输入：0

输出：0

```
// 思路：将每一位的 16 进制数，转换为 4 位的二进制数!
// 第一步：将每位 16 进制转换为 4 位的 2 进制，连接到字符串上!
// 第二步：删除前导 0，也就是要从第一个非 0 开始输出!
#include <bits/stdc++.h>//4-1306-1 宋老师视频教程
using namespace std;
string t[16] = {"0000","0001","0010","0011","0100","0101","0110","0111","1000","
1001","1010","1011","1100","1101","1110","1111"};
string s,r; //s：存放 16 进制，r：存放 2 进制
int main(){
	cin>>s;
	int x;	//s[1] 转换为 0-15 之间的整数，然后再对应的 4 位 2 进制
	for(int i = 0;i < s.size();i++)
	{
		if(isdigit(s[i])){	x = s[i] - '0';	}	// 如果是 0~9
		else	{	x = s[i] - 'A' + 10;	}
		r = r + t[x];
	}
	// 删除前导 0	//00010101111	// 当 r[0] 是 '0' 则删除
	while(r[0] == '0'){
		r.erase(0,1);
	}
	if(r == "")	cout<<0;
	else	cout<<r;
	return 0;
}
```

思路：将每一位的 16 进制数，转换为 4 位的二进制数！

第一步：将每位 16 进制转换为 4 位的 2 进制，连接到字符串上！

第二步：删除前导 0，也就是要从第一个非 0 开始输出！

```
#include <bits/stdc++.h>//4-1306-2    javacn
using namespace std;
/*
1. 将每一位的 16 进制（0~15）转 2 进制（不足 4 位补 0）
2. 删除前导 0
1A
0001,1010
*/
string t[16] = {"0000","0001","0010","0011","0100","0101","0110","0111","1000","
1001","1010","1011","1100","1101","1110","1111"};
string s;
string r = "";// 存储 2 进制的结果

int main(){
	cin>>s;
	// 循环将每一位的 16 进制转换为 4 位的 2 进制
	for(int i = 0;i < s.size();i++){
		// 如果是字符 '0'~'9'
		if(isdigit(s[i])){	r = r + t[s[i]-'0'];		}
		else{ r = r + t[s[i]-55]; } // 说明是 'A'（65）~'F'，换成 10~15
	}
	// 删除前导 0
	// 当 r 的第 1 位是 0，删除
	while(r[0]=='0'){
		r.erase(0,1);
	}
	// 如果输入为 0，那么结果是 0000，会被上面的 while 删前导 0，都删掉
	if(r == "") cout<<0;
	else cout<<r;
}
```

每位转成 4 位二进制，拼接后删除前导零

```cpp
#include<bits/stdc++.h>//4-1306-3  jiangyf70
using namespace std;
int main()
{
    string s;
    cin >> s;
    if(s == "0") //特判 0
    {
        cout << 0;
        return 0;
    }
    int a;
    string h, x;
    for(int i = 0; i < s.size(); i++)
    {
        h = "";
        if(s[i] <= '9') a = s[i] -'0';
        else a = s[i] - 'A' + 10;
        for(int j = 0; j < 4; j++) //转成 4 位二进制
        {
            h = char(a % 2 + '0') + h;
            a /= 2;
        }
        x += h;
    }
    while(x[0] == '0') x.erase(0, 1);
    cout << x;
}
```

麻烦的做法，一个一个转换

1、输入字符串，遍历每个字符 2、将每个字符转换成二进制字符串 3、拼接到新的字符串中输出（先删除前导 0）

```
#include <bits/stdc++.h>//4-1306-4    1989444607
using namespace std;
// 将字符转换为字符串
string cTos(char c){
	string res ="";
	int num=0;
	if(isdigit(c)){   num = c-'0';      }
	else{ num = c-'A'+10;   }
	while(num){
		int x = num%2;
		res=char(x+'0')+res;
		num/=2;
	}
	// 字符转换为二进制字符串，特别注意字符 0 的情况
	if(res.size()%4==1){    res="000"+res;    }
	else if(res.size()%4==2){     res = "00"+res;   }
		else if(res.size()%4==3){     res = "0"+res;    }
			else if(res.size()==0){ res = "0000";     }// 字符为 0 时
	return res;
}
int main(){
	string s,s2;
	cin>>s;
	for(int i=0;i<s.size();i++){  s2+=cTos(s[i]);   }
	while(s2[0]=='0'){      s2.erase(0,1);    }     // 删除前导 0
	// 输出，特别注意 0 的情况
	if(s2==""){ cout<<0;    }
	else{ cout<<s2;   }
	return 0;
}
```

十六进制转二进制

```
#include <bits/stdc++.h>//4-1306-5 yhh133
#define ll long long
using namespace std;
ll change1(string &s,int x){
	ll sum=0,t=1;
	for(int i=s.size()-1;i>=0;i--){
		if(s[i]>='A'&&s[i]<='F'){
			sum+=(s[i]-'A'+10)*t;
		}
		else sum+=(s[i]-'0')*t;
		t*=x;
	}
	return sum;
}
string change2(ll n,int t){
	string s;
	while(n){
		if(n%t>=10) s+=(n%t-10+'A');
		else s+=n%t+'0';
		n/=t;
	}
	reverse(s.begin(),s.end());
	return s;
}
int main() {
	string s;
	cin>>s;
	ll num=change1(s,16);
	if(num==0) cout<<0;
	else cout<<change2(num,2);
	return 0;
}
```

十六进制转换

思路：借助二进制为中间转换值，先将 16 进制转换成 2 进制，再将 2 进制转换成 8 进制
因为 1 位 16 进制数对应 4 位 2 进制数，3 位 2 进制数对应 1 位 8 进制数。

```
#include<bits/stdc++.h>//5-1295  javacn
using namespace std;
//a 数组存储：1 位 16 进制数对应 4 位 2 进制数
string a[]= {"0000","0001","0010","0011","0100","0101","0110","0111","1000","100
1","1010","1011","1100","1101","1110","1111"};
int main() {
	string s,r="",x;
	cin>>s;
	// 十六进制 -> 二进制
	int i,t;
	for(i=0; i<s.size(); i++) {
		if(s[i]>='A' && s[i]<='F') {  t = s[i]-'A'+10;  }
		else {	t = s[i]-'0';	}
		//t 是对应的下标
		r = r + a[t];
	}
	// 将前导 0 去掉
	while(r[0]=='0') r.erase(0,1);
	// 二进制数 r -> 八进制
	// 前导补 0  3 位二进制数为 1 组
	while(r.size()%3!=0) {
		r =  '0' + r;
	}
	//100 -> 14+02+01=4
	for(i=0; i<r.size(); i=i+3) {
		cout<<(r[i]-'0')*4 + (r[i+1]-'0')*2 + (r[i+2]-'0')*1;
	}
	return 0;
}
```