03 - 进制

C++ 程序设计进阶

SOJ 信息学竞赛教练组

2024 年 7 月 21 日

目录

变量

- 作用域
  - 局部变量：从变量的声明开始到包含它的块结束
  - 全部变量：从变量的声明开始一直到代码结束

变量

- 作用域
  - 局部变量：从变量的声明开始到包含它的块结束
  - 全部变量：从变量的声明开始一直到代码结束
- 同名变量
  - 一般在什么情况下使用同名变量?

变量

- 作用域
  - 局部变量：从变量的声明开始到包含它的块结束
  - 全部变量：从变量的声明开始一直到代码结束
- 同名变量
  - 一般在什么情况下使用同名变量?
  - 变量的作用域完全不重叠

变量

- 作用域
  - 局部变量：从变量的声明开始到包含它的块结束
  - 全部变量：从变量的声明开始一直到代码结束
- 同名变量
  - 一般在什么情况下使用同名变量?
  - 变量的作用域完全不重叠
- 引用变量
  - 引用变量是其他变量的别名，与其他变量共用同一块内存空间
  - 修改引用变量的值也会同时修改它所绑定的一般变量的值
  - 一般用于函数参数传递

- 按值传递
  - 函数中不能修改实参的值

```
// #include ...

void f(int x) {
  // 函数体
}

int main() {
  int a = 1;
  f(a);

  return 0;
}
```

函数参数传递

- 按值传递
  - 函数中不能修改实参的值
- 按引用传递
  - 函数中可以修改实参的值
  - 形参变量名前加 &，实参不用加

```
// #include ...

void f(int &x) {
  // 函数体
}

int main() {
  int a = 1;
  f(a);

  return 0;
}
```

函数参数传递

- 按值传递
  - 函数中不能修改实参的值
- 按引用传递
  - 函数中可以修改实参的值
  - 形参变量名前加 &，实参不用加
- 数组传递
  - 本质是传递数组首地址，函数中可以修改实参数组
  - 形参数组名后加 []，实参不用加

```
// #include ...

void f(int a[]) {
  // 函数体
}

int a[110];

int main() {
  f(a);

  return 0;
}
```

目录

讨论

为什么日常见到的数字都由 $0 \sim 9$ 组成?

进制

- 日常生活中数字由 $0 \sim 9$ 组成，9 的下一个数是 10，这种“逢十进一”的计数方法称为十进制。

进制

- 日常生活中数字由 $0 \sim 9$ 组成，9 的下一个数是 10，这种“逢十进一”的计数方法称为十进制。
- 进制是人为定义的带进位的计数方法

进制

- 日常生活中数字由 $0 \sim 9$ 组成，9 的下一个数是 10，这种“逢十进一”的计数方法称为十进制。
- 进制是人为定义的带进位的计数方法
- X 进制

进制

- 日常生活中数字由 $0 \sim 9$ 组成，9 的下一个数是 10，这种“逢十进一”的计数方法称为十进制。
- 进制是人为定义的带进位的计数方法
- X 进制
  - 规则：逢 X 进一

进制

- 日常生活中数字由 $0 \sim 9$ 组成，9 的下一个数是 10，这种“逢十进一”的计数方法称为十进制。
- 进制是人为定义的带进位的计数方法
- X 进制
  - 规则：逢 X 进一
  - X 进制中的 X 也称**基数**

进制

- 日常生活中数字由 $0 \sim 9$ 组成，9 的下一个数是 10，这种“逢十进一”的计数方法称为十进制。
- 进制是人为定义的带进位的计数方法
- X 进制
  - 规则：逢 X 进一
  - X 进制中的 X 也称**基数**
  - 数值的每位可由 X 个符号（也称**数码**）组成，分别代表 $0 \sim X-1$ 这 X 个数字

常见的进制

- 在计算机中，常见的有二进制、八进制、十六进制

常见的进制

- 在计算机中，常见的有二进制、八进制、十六进制
- 二进制
    - 逢二进一，由 $0 \sim 1$ 这 2 个数码组成

常见的进制

- 在计算机中，常见的有二进制、八进制、十六进制
- 二进制
  - 逢二进一，由 $0 \sim 1$ 这 2 个数码组成
- 八进制
  - 逢八进一，由 $0 \sim 7$ 这 8 个数码组成

常见的进制

- 在计算机中，常见的有二进制、八进制、十六进制
- 二进制
  - 逢二进一，由 $0 \sim 1$ 这 2 个数码组成
- 八进制
  - 逢八进一，由 $0 \sim 7$ 这 8 个数码组成
- 十六进制
  - 逢十六进一，需要 16 个数码来表示 $0 \sim 15$，通常用 $A \sim F$ 或 $a \sim f$ 表示 $10 \sim 15$

权重

- 以十进制的 2157 为例

权重

- 以十进制的 2157 为例
  - $(2157)_{10} = 2 \times 10^3 + 1 \times 10^2 + 5 \times 10^1 + 7 \times 10^0$

权重

- 以十进制的 2157 为例
  - $(2157)_{10} = 2 \times 10^3 + 1 \times 10^2 + 5 \times 10^1 + 7 \times 10^0$
  - 不同位置上的数字有不同的“份量”，也称**权重**

权重

- 以十进制的 2157 为例
  - $(2157)_{10} = 2 \times 10^3 + 1 \times 10^2 + 5 \times 10^1 + 7 \times 10^0$
  - 不同位置上的数字有不同的“份量”，也称**权重**
  - 十进制个位的权重是 10^0，十位的权重是 10^1，百位的权重是 10^2……

权重

- 以十进制的 2157 为例
  - $(2157)_{10} = 2 \times 10^3 + 1 \times 10^2 + 5 \times 10^1 + 7 \times 10^0$
  - 不同位置上的数字有不同的“份量”，也称**权重**
  - 十进制个位的权重是 10^0，十位的权重是 10^1，百位的权重是 10^2……
- 八进制的 $(2157)_8 =?$

权重

- 以十进制的 2157 为例
  - $(2157)_{10} = 2 \times 10^3 + 1 \times 10^2 + 5 \times 10^1 + 7 \times 10^0$
  - 不同位置上的数字有不同的“份量”，也称**权重**
  - 十进制个位的权重是 10^0，十位的权重是 10^1，百位的权重是 10^2……
- 八进制的 $(2157)_8 = ?$
  - $(2157)_8 = 2 \times 8^3 + 1 \times 8^2 + 5 \times 8^1 + 7 \times 8^0$

权重

- 以十进制的 2157 为例
  - $(2157)_{10} = 2 \times 10^3 + 1 \times 10^2 + 5 \times 10^1 + 7 \times 10^0$
  - 不同位置上的数字有不同的“份量”，也称**权重**
  - 十进制个位的权重是 10^0，十位的权重是 10^1，百位的权重是 10^2……
- 八进制的 $(2157)_8 =?$
  - $(2157)_8 = 2 \times 8^3 + 1 \times 8^2 + 5 \times 8^1 + 7 \times 8^0$
- $(2157)_{10}$ 与 $(2157)_8$ 的**数值**大小不同

目录

计算机与二进制

- 计算机底层只使用 0 和 1 两个数字，这样做的原因有：

计算机与二进制

- 计算机底层只使用 0 和 1 两个数字，这样做的原因有：
  - 二进制只有两种状态，使用两个稳定状态的物理器件就可以表示二进制的每一位，制作成本比较低，例如用高低电平可以表示 1 和 0

计算机与二进制

- 计算机底层只使用 0 和 1 两个数字，这样做的原因有：
  - 二进制只有两种状态，使用两个稳定状态的物理器件就可以表示二进制的每一位，制作成本比较低，例如用高低电平可以表示 1 和 0
  - 二进制的 1 和 0 正好对应逻辑的真和假，为计算机实现逻辑运算提供了便利

计算机与二进制

- 计算机底层只使用 0 和 1 两个数字，这样做的原因有：
  - 二进制只有两种状态，使用两个稳定状态的物理器件就可以表示二进制的每一位，制作成本比较低，例如用高低电平可以表示 1 和 0
  - 二进制的 1 和 0 正好对应逻辑的真和假，为计算机实现逻辑运算提供了便利
- 在计算机内部，各种类型的数据（例如整数、实数和字符等）都编码为 0/1 序列

随堂练习

选择题

1. 在计算机内部用来传送、存贮、加工处理的数据或指令都是以什么形式进行的

   A. 二进制码

   B. 八进制码

   C. 十进制码

   D. 智能拼音码

随堂练习

选择题

1. 在计算机内部用来传送、存贮、加工处理的数据或指令都是以什么形式进行的
   - A. 二进制码
   - B. 八进制码
   - C. 十进制码
   - D. 智能拼音码

二进制数的基本运算

- 在加减乘除运算中，二进制数的运算逻辑与十进制数的运算逻辑相同

二进制数的基本运算

- 在加减乘除运算中，二进制数的运算逻辑与十进制数的运算逻辑相同
- 区别在于二进制运算是“逢二进一”，十进制运算是“逢十进一”

二进制加法

- 从低位到高位依次相加，逢二进一
  - $0 + 0 = 0$
  - $0 + 1 = 1$
  - $1 + 0 = 1$
  - $1 + 1 = 0$（进位）

二进制加法

- 从低位到高位依次相加，逢二进一
  - $0+0=0$
  - $0+1=1$
  - $1+0=1$
  - $1+1=0$（进位）
- 以 $(10111)_2+(10001)_2$ 为例

$$
\begin{array}{ccccccc}
 & & 1 & 0 & 1 & 1 & 1 \\
+ & & 1 & 0 & 0 & 0 & 1 \\
 & & {\scriptstyle 1} & & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} \\
\hline
 & 1 & 0 & 1 & 0 & 0 & 0
\end{array}
$$

二进制减法

- 从低位到高位依次相减，不够则借位，借得 2
  - $0-0=0$
  - $0-1=1$（借位）
  - $1-0=1$
  - $1-1=0$

二进制减法

- 从低位到高位依次相减，不够则借位，借得 2
  - $0 - 0 = 0$
  - $0 - 1 = 1$（借位）
  - $1 - 0 = 1$
  - $1 - 1 = 0$
- 以 $(10100)_2 - (1011)_2$ 为例

$$
\begin{array}{cccccc}
 & & 2 & & 1 & 2 \\
 & \dot{1} & 0 & \dot{1} & \dot{0} & 0 \\
- & & 1 & 0 & 1 & 1 \\
\hline
 & & 1 & 0 & 0 & 1
\end{array}
$$

二进制乘法

- 一个乘数的每一位分别乘另一个的每一位，逢二进一
  - $0 \times 0 = 0$
  - $0 \times 1 = 0$
  - $1 \times 0 = 0$
  - $1 \times 1 = 1$

二进制乘法

- 一个乘数的每一位分别乘另一个的每一位，逢二进一
  - $0 \times 0 = 0$
  - $0 \times 1 = 0$
  - $1 \times 0 = 0$
  - $1 \times 1 = 1$
- 以 $(101)_2 \times (111)_2$ 为例

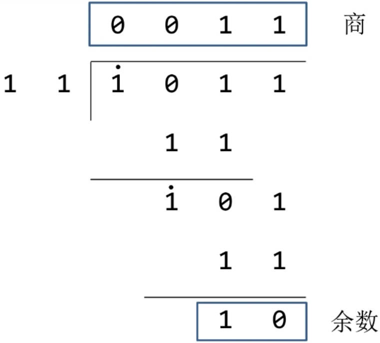
$$
\begin{array}{cccccc}
 & & & 1 & 0 & 1 \\
 & \times & & 1 & 1 & 1 \\
\hline
 & & & 1 & 0 & 1 \\
 & & 1 & 0 & 1 & \\
 & 1 & 0 & 1 & & \\
 & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & & \\
\hline
1 & 0 & 0 & 0 & 1 & 1
\end{array}
$$

二进制除法

- 从被除数的高位到低位一直除以除数，如果某一位不够除，就包含后一位继续除
  - 通过乘法和减法实现

二进制除法

- 从被除数的高位到低位一直除以除数，如果某一位不够除，就包含后一位继续除
  - 通过乘法和减法实现
- 以 $(1011)_2 \div (11)_2$ 为例



随堂练习

选择题

1. 二进制数 00100100 和 00010101 的和是多少

   A. 00101000

   B. 001010100

   C. 01000101

   D. 00111001

随堂练习

选择题

1. 二进制数 00100100 和 00010101 的和是多少

   A. 00101000

   B. 001010100

   C. 01000101

   D. 00111001

目录

二进制转十进制

- 一个数的数值等于各个数码与其对应的权重乘积之和
  - $(2157)_{10} = 2 \times 10^3 + 1 \times 10^2 + 5 \times 10^1 + 7 \times 10^0$

二进制转十进制

- 一个数的数值等于各个数码与其对应的权重乘积之和
  - $(2157)_{10} = 2 \times 10^3 + 1 \times 10^2 + 5 \times 10^1 + 7 \times 10^0$
- 二进制每个位的权重是 2 的幂
  - $(10011)_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
  - 计算这个展开式的十进制结果，该结果就是原二进制数对应的十进制数值
  - $(10011)_2 = 1 \times 16 + 0 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1 = 19$

二进制转十进制

- 一个数的数值等于各个数码与其对应的权重乘积之和
  - $(2157)_{10} = 2 \times 10^3 + 1 \times 10^2 + 5 \times 10^1 + 7 \times 10^0$
- 二进制每个位的权重是 2 的幂
  - $(10011)_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
  - 计算这个展开式的十进制结果，该结果就是原二进制数对应的十进制数值
  - $(10011)_2 = 1 \times 16 + 0 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1 = 19$
- 这种二进制转十进制的方法称为**按权展开求和**

例 3.1：二进制转十进制

编程题

- 输入一个正整数 n $(1 \le n \le 31)$，表示有一个 n 位的二进制数，接下来从高位到低位输入该二进制数的每一位 a_i $(0 \le a_i \le 1)$。输出该二进制数对应的十进制数值。
- 样例输入

```
5
1 0 0 1 1
```

- 样例输出

```
19
```

例 3.1：二进制转十进制

编程题

- 思考如何写代码计算以下展开式?
  - $(10011)_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

数位	1	0	0	1	1
权重	2^4	2^3	2^2	2^1	2^0
加数	1×2^4	0×2^3	0×2^2	1×2^1	1×2^0

- 实现

例 3.1：二进制转十进制

编程题

- 思考如何写代码计算以下展开式?
  - $(10011)_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

数位	1	0	0	1	1
权重	2^4	2^3	2^2	2^1	2^0
加数	1×2^4	0×2^3	0×2^2	1×2^1	1×2^0

- 实现
  - 用数组存储二进制的每位，并从低位到高位遍历（倒序遍历）

例 3.1：二进制转十进制

编程题

- 思考如何写代码计算以下展开式?
  - $(10011)_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

数位	1	0	0	1	1
权重	2^4	2^3	2^2	2^1	2^0
加数	1×2^4	0×2^3	0×2^2	1×2^1	1×2^0

- 实现
  - 用数组存储二进制的每位，并从低位到高位遍历（倒序遍历）
  - 同时，用变量 w 存储权重，初始化为 $1(2^0)$，且实现每次 $w* = 2$ 的变化

例 3.1：二进制转十进制

编程题

- 思考如何写代码计算以下展开式?
  - $(10011)_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

数位	1	0	0	1	1
权重	2^4	2^3	2^2	2^1	2^0
加数	1×2^4	0×2^3	0×2^2	1×2^1	1×2^0

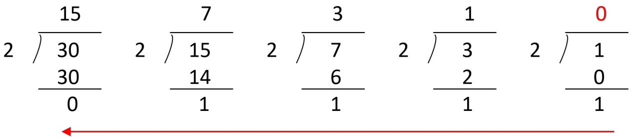
- 实现
  - 用数组存储二进制的每位，并从低位到高位遍历（倒序遍历）
  - 同时，用变量 w 存储权重，初始化为 $1(2^0)$，且实现每次 $w* = 2$ 的变化
  - 用变量 sum 记录十进制数值，每次把数位与权重的乘积累加到 sum 中

例 3.1：二进制转十进制

```cpp
// #include ...

// 函数功能：返回 n 位二进制数 b[0 ~ n-1] 的十进制数值
int bin2dec(int b[], int n) {
  int sum = 0, w = 1;
  for (int i = n - 1; i >= 0; i--) {
    sum += b[i] * w;
    w *= 2;
  }
  return sum;
}

int bin[35];

int main() {
  int n;
  cin >> n;
  for (int i = 0; i < n; i++) cin >> bin[i];
  cout << bin2dec(bin, n) << endl;
  return 0;
}
```

十进制转二进制

- 除二取余法
  - 把十进制整数连续整除以 2，直到商为 0，逆序排列余数，即为该十进制对应的二进制数
  - $(30)_{10} = (11110)_2$



十进制转二进制

- $(10011)_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

数位	1	0	0	1	1
权重	2^4	2^3	2^2	2^1	2^0（值为 1）
加数	1×2^4	0×2^3	0×2^2	1×2^1	1×2^0

十进制转二进制

- $(10011)_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

数位	1	0	0	1	1
权重	2^4	2^3	2^2	2^1	2^0（值为 1）
加数	1×2^4	0×2^3	0×2^2	1×2^1	1×2^0

- %2 得到二进制数的最低位

十进制转二进制

- $(10011)_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

数位	1	0	0	1
权重	2^3	2^2	2^1	2^0（值为 1）
加数	1×2^3	0×2^2	0×2^1	1×2^0

- $\%2$ 得到二进制数的最低位
- $/2$ 去掉二进制数的最低位

十进制转二进制

- $(10011)_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

数位	1	0	0	1
权重	2^3	2^2	2^1	2^0 （值为 1）
加数	1×2^3	0×2^2	0×2^1	1×2^0

- $\%2$ 得到二进制数的最低位
- $/2$ 去掉二进制数的最低位
- 重复以上步骤不断得到二进制的每一位

十进制转二进制

- $(10011)_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

数位	1	0	0
权重	2^2	2^1	2^0（值为 1）
加数	1×2^2	0×2^1	0×2^0

- $\%2$ 得到二进制数的最低位
- $/2$ 去掉二进制数的最低位
- 重复以上步骤不断得到二进制的每一位

十进制转二进制

- $(10011)_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

数位	1	0
权重	2^1	2^0（值为 1）
加数	1×2^1	0×2^0

- %2 得到二进制数的最低位
- /2 去掉二进制数的最低位
- 重复以上步骤不断得到二进制的每一位

十进制转二进制

- $(10011)_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

数位	1
权重	2^0（值为 1）
加数	1×2^0

- %2 得到二进制数的最低位
- /2 去掉二进制数的最低位
- 重复以上步骤不断得到二进制的每一位

除二取余法

- 联系十进制数位拆分的写法
  - 对一个数值 n 在循环中重复 `n % 10, n /= 10` 的操作，可以得到这个数值十进制逆序的每一位
  - 对一个数值 n 在循环中重复 `n % 2, n /= 2` 的操作，可以得到这个数值二进制逆序的每一位
- **口诀：模二除二，逆序输出**

例 3.2：十进制转二进制

编程题

- 输入一个十进制表示的非负整数 n $(0 \leq n \leq 10^9)$，输出其二进制数值。
- 样例输入

  ```
  30
  ```

- 样例输出

  ```
  11110
  ```

例 3.2：十进制转二进制

```
// #include ...

int bin[35];
// 函数功能：输出十进制 x 的二进制表示
void dec2bin(int x) {
  int siz = 0; // 记录二进制的位数
  do {
    bin[siz] = x % 2;
    siz++;
    x /= 2;
  } while (x);
  // 逆序输出
  for (int i = siz - 1; i >= 0; i--) cout << bin[i];
  cout << endl;
}

int main() {
  int x;
  cin >> x;
  dec2bin(x);
  return 0;
}
```

例 3.2：十进制转二进制

```cpp
// #include ...

int bin[35];
// 函数功能：输出十进制 x 的二进制表示
void dec2bin(int x) {
  int siz = 0; // 记录二进制的位数
  do {
    bin[siz] = x % 2;
    siz++;
    x /= 2;
  } while (x);
  // 逆序输出
  for (int i = siz - 1; i >= 0; i--) cout << bin[i];
  cout << endl;
}

int main() {
  int x;
  cin >> x;
  dec2bin(x);
  return 0;
}
```

例 3.2：十进制转二进制

```cpp
// #include ...

int bin[35];
// 函数功能：输出十进制 x 的二进制表示
void dec2bin(int x) {
  int siz = 0; // 记录二进制的位数
  do {
    bin[siz++] = x % 2;
    x /= 2;
  } while (x);
  // 逆序输出
  for (int i = siz - 1; i >= 0; i--) cout << bin[i];
  cout << endl;
}

int main() {
  int x;
  cin >> x;
  dec2bin(x);
  return 0;
}
```

二进制与十进制的转换

- 二进制整数转十进制整数
  - 按权展开求和
- 十进制整数转二进制整数
  - 模二除二，逆序输出

目录

八进制转十进制

- $(107)_8$ 按权展开是怎样的?
  - $1 \times 2^2 + 0 \times 2^1 + 7 \times 2^0$
  - $1 \times 8^0 + 0 \times 8^1 + 7 \times 8^2$
  - $1 \times 8^2 + 0 \times 8^1 + 7 \times 8^0$

八进制转十进制

- $(107)_8$ 按权展开是怎样的?
  - $\cancel{1 \times 2^2 + 0 \times 2^1 + 7 \times 2^0}$
  - $\cancel{1 \times 8^0 + 0 \times 8^1 + 7 \times 8^2}$
  - $1 \times 8^2 + 0 \times 8^1 + 7 \times 8^0$

八进制转十进制

- $(107)_8$ 按权展开是怎样的?
  - $\cancel{1 \times 2^2 + 0 \times 2^1 + 7 \times 2^0}$
  - $\cancel{1 \times 8^0 + 0 \times 8^1 + 7 \times 8^2}$
  - $1 \times 8^2 + 0 \times 8^1 + 7 \times 8^0$
- 八进制各个数位的权重是 8 的幂，高位权重大于低位权重

八进制转十进制

- $(107)_8$ 按权展开是怎样的?
  - $\cancel{1 \times 2^2 + 0 \times 2^1 + 7 \times 2^0}$
  - $\cancel{1 \times 8^0 + 0 \times 8^1 + 7 \times 8^2}$
  - $1 \times 8^2 + 0 \times 8^1 + 7 \times 8^0$
- 八进制各个数位的权重是 8 的幂，高位权重大于低位权重
- 八进制转十进制的方法同样是按权展开求和
  - 列出展开式，计算其的十进制数值

十六进制转十进制

- X 进制转十进制的方法都是按权展开求和

十六进制转十进制

- X 进制转十进制的方法都是按权展开求和
- 十六进制各个数位的权重是 16 的幂

十六进制转十进制

- X 进制转十进制的方法都是按权展开求和
- 十六进制各个数位的权重是 16 的幂
- 十六进制通常用 $A \sim F$ 或 $a \sim f$ 表示 $10 \sim 15$，展开时要将字母转换成对应的数字

十六进制转十进制

- X 进制转十进制的方法都是按权展开求和
- 十六进制各个数位的权重是 16 的幂
- 十六进制通常用 $A \sim F$ 或 $a \sim f$ 表示 $10 \sim 15$，展开时要将字母转换成对应的数字
- $(A2)_{16} = ?$

十六进制转十进制

- X 进制转十进制的方法都是按权展开求和
- 十六进制各个数位的权重是 16 的幂
- 十六进制通常用 $A \sim F$ 或 $a \sim f$ 表示 $10 \sim 15$，展开时要将字母转换成对应的数字
- $(A2)_{16} = ?$
  - $(A2)_{16} = 10 \times 16^1 + 2 \times 16^0 = 162$

十进制转八进制

- 十进制转二进制用“除二取余法”，十进制转八进制应该用什么方法?
- 除八取余法
- $(207)_{10} = ?$
  - $207 \div 8 = 25 \dots 7$
  - $25 \div 8 = 3 \dots 1$
  - $3 \div 8 = 0 \dots 3$
  - 余数逆序，可得其八进制 $(317)_8$

十进制转十六进制

- 十进制转 X 进制的方法都是除 X 取余法
- 算出来的余数是大于等于 10 的数字要转换成对应的字母
- $(719)_{10} =?$
  - $719 \div 16 = 44 \dots 15(F)$
  - $44 \div 16 = 2 \dots 12(C)$
  - $2 \div 16 = 0 \dots 2$
  - 余数逆序，可得其八进制 $(2CF)_{16}$

目录

二进制实数转十进制

- $(10.101)_2$ 按权展开是怎样的?

$$\begin{aligned}(10.101)_2 &= 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} \\ &= 1 \times 2 + 0 \times 1 + 1 \times 0.5 + 0 \times 0.25 + 1 \times 0.125 \\ &= (2.625)_{10}\end{aligned}$$

十进制实数转二进制

- 整数部分用**除二取余法**，小数部分用**乘二取整法**

十进制实数转二进制

- 整数部分用**除二取余法**，小数部分用**乘二取整法**
- 以 $(2.375)_{10}$ 为例

十进制实数转二进制

- 整数部分用**除二取余法**，小数部分用**乘二取整法**
- 以 $(2.375)_{10}$ 为例
  - 整数部分 $(2)_2$ 转十进制为 $(10)_2$

十进制实数转二进制

- 整数部分用**除二取余法**，小数部分用**乘二取整法**
- 以 $(2.375)_{10}$ 为例
  - 整数部分 $(2)_2$ 转十进制为 $(10)_2$
  - 小数部分 $(0.375)_{10}$ 使用乘二取整法：

十进制实数转二进制

- 整数部分用**除二取余法**，小数部分用**乘二取整法**
- 以 $(2.375)_{10}$ 为例
  - 整数部分 $(2)_2$ 转十进制为 $(10)_2$
  - 小数部分 $(0.375)_{10}$ 使用乘二取整法：
    - $0.375 \times 2 = 0.75$

十进制实数转二进制

- 整数部分用**除二取余法**，小数部分用**乘二取整法**
- 以 $(2.375)_{10}$ 为例
  - 整数部分 $(2)_2$ 转十进制为 $(10)_2$
  - 小数部分 $(0.375)_{10}$ 使用乘二取整法：
    - $0.375 \times 2 = 0.75$
    - $0.75 \times 2 = 1.5$

十进制实数转二进制

- 整数部分用**除二取余法**，小数部分用**乘二取整法**
- 以 $(2.375)_{10}$ 为例
  - 整数部分 $(2)_2$ 转十进制为 $(10)_2$
  - 小数部分 $(0.375)_{10}$ 使用乘二取整法：
    - $0.375 \times 2 = 0.75$
    - $0.75 \times 2 = 1.5$
    - $0.5 \times 2 = 1.0$

十进制实数转二进制

- 整数部分用**除二取余法**，小数部分用**乘二取整法**
- 以 $(2.375)_{10}$ 为例
  - 整数部分 $(2)_2$ 转十进制为 $(10)_2$
  - 小数部分 $(0.375)_{10}$ 使用乘二取整法：
    - $0.375 \times 2 = 0.75$
    - $0.75 \times 2 = 1.5$
    - $0.5 \times 2 = 1.0$
    - 直到小数部分为 $.0$ 时停止计算

十进制实数转二进制

- 整数部分用**除二取余法**，小数部分用**乘二取整法**
- 以 $(2.375)_{10}$ 为例
  - 整数部分 $(2)_2$ 转十进制为 $(10)_2$
  - 小数部分 $(0.375)_{10}$ 使用乘二取整法：
    - $0.375 \times 2 = 0.75$
    - $0.75 \times 2 = 1.5$
    - $0.5 \times 2 = 1.0$
    - 直到小数部分为 $.0$ 时停止计算
    - 小数部分转二进制的结果：顺序记下商的整数部分 $(0.011)_2$

十进制实数转二进制

- 整数部分用**除二取余法**，小数部分用**乘二取整法**
- 以 $(2.375)_{10}$ 为例
  - 整数部分 $(2)_2$ 转十进制为 $(10)_2$
  - 小数部分 $(0.375)_{10}$ 使用乘二取整法：
    - $0.375 \times 2 = 0.75$
    - $0.75 \times 2 = 1.5$
    - $0.5 \times 2 = 1.0$
    - 直到小数部分为 $.0$ 时停止计算
    - 小数部分转二进制的结果：顺序记下商的整数部分 $(0.011)_2$
  - $(2.375)_{10} = (10.011)_2$

实数储存的精度问题

- 计算机储存数字都是以二进制形式储存

实数储存的精度问题

- 计算机储存数字都是以二进制形式储存
- 以 $(0.6)_{10}$ 转换为二进制为例

实数储存的精度问题

- 计算机储存数字都是以二进制形式储存
- 以 $(0.6)_{10}$ 转换为二进制为例
  - $0.6 \times 2 = 1.2$

实数储存的精度问题

- 计算机储存数字都是以二进制形式储存
- 以 $(0.6)_{10}$ 转换为二进制为例
  - $0.6 \times 2 = 1.2$
  - $0.2 \times 2 = 0.4$

实数储存的精度问题

- 计算机储存数字都是以二进制形式储存
- 以 $(0.6)_{10}$ 转换为二进制为例
  - $0.6 \times 2 = 1.2$
  - $0.2 \times 2 = 0.4$
  - $0.4 \times 2 = 0.8$

实数储存的精度问题

- 计算机储存数字都是以二进制形式储存
- 以 $(0.6)_{10}$ 转换为二进制为例
  - $0.6 \times 2 = 1.2$
  - $0.2 \times 2 = 0.4$
  - $0.4 \times 2 = 0.8$
  - $0.8 \times 2 = 1.6$

实数储存的精度问题

- 计算机储存数字都是以二进制形式储存
- 以 $(0.6)_{10}$ 转换为二进制为例
  - $0.6 \times 2 = 1.2$
  - $0.2 \times 2 = 0.4$
  - $0.4 \times 2 = 0.8$
  - $0.8 \times 2 = 1.6$
  - 计算过程进入无限循环，无法用有限位二进制准确表示

实数储存的精度问题

- `double` 类型只保证 15 位有效数字的精度

实数储存的精度问题

- `double` 类型只保证 15 位有效数字的精度
- 平时尽量避免浮点数计算

实数储存的精度问题

- `double` 类型只保证 15 位有效数字的精度
- 平时尽量避免浮点数计算
  - 可用 $i * i <= n$ 替代 $i <= sqrt(n)$

实数储存的精度问题

- `double` 类型只保证 15 位有效数字的精度
- 平时尽量避免浮点数计算
  - 可用 $i * i <= n$ 替代 $i <= sqrt(n)$
- 若需判断浮点数是否相等，需要允许微小误差

实数储存的精度问题

- `double` 类型只保证 15 位有效数字的精度
- 平时尽量避免浮点数计算
  - 可用 $i * i <= n$ 替代 $i <= sqrt(n)$
- 若需判断浮点数是否相等，需要允许微小误差
  - 需用 `if (abs(a - b) < 1e-10)` 替代 `if (a == b)`

目录

进制

- 进制
  - 基数、数码、权重、数值、二进制的加减乘除

进制

- 进制
  - 基数、数码、权重、数值、二进制的加减乘除
- 进制转换
  - 二进制整数转十进制（按权展开求和）
  - 十进制整数转二进制（除二取余法）
  - X 进制整数转十进制（按权展开求和）
  - 十进制整数转 X 进制（除 X 取余法）

进制

- 进制
  - 基数、数码、权重、数值、二进制的加减乘除
- 进制转换
  - 二进制整数转十进制（按权展开求和）
  - 十进制整数转二进制（除二取余法）
  - X 进制整数转十进制（按权展开求和）
  - 十进制整数转 X 进制（除 X 取余法）
- 实数储存有精度问题，应避免进行浮点数运算

Thank you!