

# C语言概述

参考解析：【解析】算法的特征：

- ①有穷性：一个算法(对任何合法的输入)在执行有穷步后能够结束,并且在有限的时间内完成。
- ②确定性：算法中的每一步都有确切的含。
- ③可行性：算法中的操作能够用已经实现的基本运算执行有限次来实现。
- ④输入：一个算法有零个或者多个输入,零个输入就是算法本身确定了初始条件。
- ⑤输出：一个算法有一个或者多个输出,以反映出数据加工的结果。

## 浮点常量

- 小数形式：必须有小数点，小数点前面可以没有数字
- 指数形式：e或E的前后必须有数字，前面可以是小数，后面必须为整数

## C语言标识符开头必须是字母或下划线

- `\xhh` TODO
- `\ddd`

预定义标识符可以作为用户标识符，原来的预定义标识符将会被用户标识符覆盖，预定义标识符的原意失效

# 运算符与表达式

`%` 的左右两边必须都是整数，不能应用于float或double类型，不会自动强制类型转换

```
int a = b = 0;
```

问题：b尚未定义，不能直接用于给a赋值

## 涉及到负数的模运算

C99 规定，如果 `%` 左边的操作数是正数，则模除的结果为正数或零；如果 `%` 左边的操作数是负数，则模除的结果为负数或零

## 逗号表达式的值为其中最后一个表达式的值

```
// a = 0, b = 0, c = 0
```

```
c = (a += ++b, b += 4);
```

```
1. a += ++b ==> a = 1, b = 1
```

2. `b += 4`  $\implies$  `b = 5`
3. 把5赋给c  $\implies$  `c = 5`

`printf`输出隐含右对齐 右对齐?

## 选择 && 循环

ASCII码

```
'\0': 0
'0': 48
'a': 97
'A': 65
```

! 的优先级高于 ==

```
// int x = 10, y = 0;
if (x = y)的表达式值为0, if语句不执行
```

## 函数

全局变量的作用域由具体定义位置和extern说明来决定范围

## 指针

## 字符串

- 不能将一个整数直接赋给指针变量作为地址
- 字符串初始化的方法

1. 使用指针初始化

```
char *s; s = "string";
```

2. 使用字符数组初始化

使用字符数组方式的话必须在定义的时候初始化

```
char s[] = "string"; char s[] = {"string"}; 【括号是初始化数组的方式】
```

字符数组不能相互赋值

- ```
char *c;  
scanf("%s", c);
```

上述代码并无语法错误，但是由于字符指针并未被赋值，指向一个不确定的区域，因此在这个不确定的区域存放字符串的话可能有无法预知的错误

数组名不能指向别的地方，只能指向数组

## 函数指针

---

```
void (*函数名)(参数列表);
```

## 结构体

（结构体中的）数组不能直接赋值

## I/O

`getchar()` //TODO

`getchar()`可以识别换行符