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 ==> b = 5
3. 把5赋给c ==> 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 && 数组

不能用变量定义数组长度

```
// 下面定义数组的方式是错误的，因为N是变量
int N = 10;
int x[N];
```

指针是变量，数组名不是变量

当把数组名传递给一个函数时，实际上传递的是该数组第一个元素的地址

在函数定义中，形式参数 `char s[];` 和 `char *s` 是等价的

常量0可以赋值给指针，等价于NULL

字符串

不能将一个整数直接赋给指针变量作为地址

数组名不能指向别的地方，只能指向数组

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；
   >
   > 数组定义后不可对数组整体赋值；

3. 使用字符数组初始化，数组中的单个字符可以修改，但数组名不可变（始终指向同一个存储位置）

   使用指针初始化，指针可以被修改指向其他地址，但无法修改字符串的内容，指针定义的字符串存储在静态常量区，可读但不可写

   下图做了一个验证，指针方式定义的字符串为常量，a和b指向的都是常量区那个，而字符数组定义的字符串是在堆区新开辟的

```
int main() {
    char  *a = "Hello world";
    char  *b = "Hello world";
    char  c[] = "Hello world";
    printf("a's address is: %p\n", a);
    printf("b's address is: %p\n", b);
    printf("c's address is: %p\n", c);
    return 0;
}
```

```
learn_cpp
C:\Users\Administrator\CLionProjects\le
a's address is: 00405047
b's address is: 00405047
c's address is: 0061FF0C
```

> 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

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