1.条件编译（只能与宏定义配合使用）

1》有什么用?

提高编译效率

2》用法：#if　　条件表达式

#elif　条件表达式

#endif　结束条件编译，只对一个离最近的#if起作用

#ifdef　宏名，表示定义了宏，就执行（与宏定义的相关用法）

#ifndef　宏名，表示没有定义宏，就执行（与宏定义的相关用法）

3》特点

.条件编译不能访问需要编译后才能执行的代码.

.条件编译不需要用大括号,所以必须加上结束标识,否则涵盖下面的所有代码

.条件编译一般跟宏定义配合使用,防止重复包含及交叉包含

2.typedef

一,是什么?

是一个C语言的一个关键字.

二,有什么用?

可以给一个数据类型取一个别名,方便使用.

三,怎么用?

定义: typedef　类型　别名;

如:　typedef　long int　Lint; 表示给一个long int 类型取一个别名为Lint

则:　long int a =5;与Lint　a=5;二者等效

四,什么时候用?

当希望给一个类型取一个别名时(比较类型较复杂等),可以用,方便理解调用.

五,有什么特点?

1.定义位置没有限制,函数内外都可以.

2.作用域为定义位置开始至所处作用域结束.类似于变量

3.可以用于任意数据类型.

4.别名具有数据类型的含义,这是跟宏定义最大区别.

6》与宏定义的区别

typedef功能比宏定义强大，具有类型的含义

3.static和extern

一,是什么?
是C语言的关键字.

二,有什么用?
控制一个变量或函数的作用域.

三,怎么用?
修饰函数:static 返回值 函数名(){};表示函数为内部函数只能被本文件访问.
extern 返回值 函数名(){};表示函数为外部部函数能被程序的所有文件访问.(默认)
修饰全局变量: static:表示变量只能被当前文件访问. extern:所有文件共享.(默认)
修饰局部变量: static:表示延长变量生命周期至程序结束. extern:声名一个全局变量.

四,有什么特点?
1.外部函数不同文件中也不能重名,但内部函数可以
2.同类型全局变量可以重复定义,但在内存中只有一份.
3.可以在函数内声名一个全局变量.但要使用必须在函数外定义. extern只是声名.
4.static修饰定义局部变量,则这个局部变量生命周期延长至程序结束. 但作用域不变.

5》注意：extern只能声明，不能定义

```
extern int f=5;//错误

extern int f;//正确;
```

4.auto与const

一, 是什么?
是C语言的关键字.

二,有什么用?
auto: 用于定义一个能自动回收的临时变量.这个变量在作用域内用完后会自动销毁.
canst:用于定义一个常量,这个常量在常量区,且在内存中仅有一份.且不能改变.

三,怎么用?
auto 数据类型 变量名;(默认所有局部变量都是被这个修饰,可以省略)
注意: 1. static修饰的除外. 2.全局变量除外;

canst 数据类型 变量名=值;
注意:一但定义,值就固定不能被改变.
如:

```
const int a;
a=0;//错误
```

4》补充：指针常量与常量指针

```
const int a=1;
int b=2;
int c=2;
//const int  *pa=&a;//常量的指针
//*pa=2;
  int  * constb;//指针常量
pa=&c;
```

5.goto

一, 是什么?
是C语言的关键字.

二,有什么用?
让指令能够不受条件跨越执行.简化代码

三,怎么用?
```
int a=0;
if(a==0){
        goto heima;
}
printf("itcast\n");//这行不会执行
heima:printf("heima\n");//直接跳到这里执行
```

三,什么时候用?
需要不受限制进行指令跳转时.(如:跳出多层循环嵌套)

6.register和volatile

一, 是什么?
是C语言的关键字.

二,有什么用?
register 表示让程序优先把这个数据存在寄存器中.
volatile 表示让编译器不要去优化代码,不能用缓存,每次使用都必须去内存中获取.

三,怎么用?
修饰变量:

```
register   数据类型  变量名;
volatile   数据类型  变量名;
```

四,什么时候用?
register : 如果一个变量使用相当频繁且占用内存又不是很大. 可以使用.
全局变量与加了static局部变量除外.
volatile : 如果需要让计算机严格按写的指令指执行,且能关注变量每一次值的变化,
这个时候可以使用.则编译器不会对执行指令作优化.

7.递归

一,是什么?
是指在函数的定义中使用函数自身的方法,说白了就是一个函数自己调用自己.

二,有什么用?
实现一些反复执行的操作.

三,怎么用?
//让任意一个整数倒序累加至0的值,如:为3时:3+2+1+0;

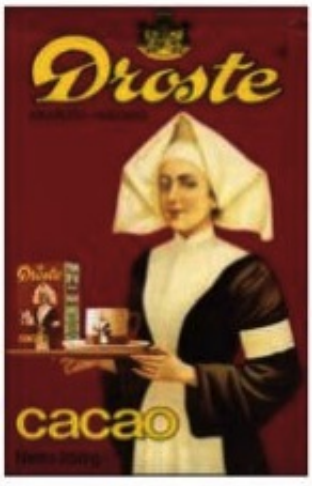
```
int add(int a){

  if(a==0)return 0;

  return a+add(a-1);
}
```

四,什么时候用?
当有一些反复的操作,但没有告知明确次数的时候.

五,有什么特点?
可以用循环实现递归的所有操作.



递归视觉形式-德罗斯特效应

8.文件操作

一,是什么?

是对文件或文件内容的输入(读)与输出(写). 文件操作的函数在<stdio.h>中声名.

二,怎么用?

1.打开一个文件并确定操作的方式:fopen("文件","打开方式");

函数会返回一个文件的指针(即结构体的指针).FILE *

2.开始操作:读或写. 常用的文件读写函数有三类:

字符读写函数:char c=fgetc(文件指针)读和fputc(字符,文件指针)写;

字符串读写函数:fgets(字符缓存,数量,文件指针)读和fputs(字符缓存,文件指针)写;

数据块(二进制)读写函数:fread(数据容器,单个数据字节数,总数,文件指针)读

fwrite(数据容器,单个数据字节数,总数,文件指针)写;

3.操作完成,关闭文件:fclose(文件的指针)

三,有什么特点?

1. 用打开文件的模式限定操用的方式.

2.打开操作文件后一定要关闭文件,否则容易引起内存泄漏.

3.输入与输出是相对于操作者而言.

4》输入和输出的理解