

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 :如果需要对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

黑马程序员 —为莘莘学子改变命运而讲课
itheima.com 为千万人少走弯路而著书!

文件操作

操作者

输入

输出

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? **a**

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. 打开文件的模式限定操作的方式.

```
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000
```