# C语言学习笔记

# 记录你学习过程中的所见所思！酸甜苦辣！

# 看什么看！ 赶紧填坑啊！

1. C运算符优先级

|  |  |
| --- | --- |
| 运算符 （优先级从高到低） | 结合律 |
| ++,--（后缀）,()（函数调用）,[]{}  （复合字面量）, . -> | 从左往右 |
| ++,--（前缀）,- + ~ ! \*（解引用）  ,&（取址）,sizeof \_Alignof（类型名）  （本栏都是一元运算符） | 从右往左 |
| （类型名） | 从右往左 |
| \* / % | 从左往右 |
| + -（二元运算符） | 从左往右 |
| << >> | 从左往右 |
| < > <= >= | 从左往右 |
| == != | 从左往右 |
| & | 从左往右 |
| ^（抑或） | 从左往右 |
| | | 从左往右 |
| && | 从左往右 |
| || | 从左往右 |
| ?= | 从右往左 |
| = \*= /= += -= <<= >>= &= != ^= | 从右往左 |
| ,（逗号运算符） | 从左往右 |

2. 逻辑表达式的求值顺序:从左往右，当发现可以使整个表达式为假的条件时立即停止求值。

3. 复浮点数

#include<complex,h>

double \_Complex z =3.0;

double \_Complex w =4.0 \* I;

double \_complex=6.0 - 8.0 \* I;

4. 存储类别

关键字: auto,extern,static,register,\_thread\_local (C11)

一般注释:变量的存储类别取决于它的作用域、链接和存储期。存储类别由声明变量的位置和与之关联的关键字决定。定义在所有函数外部的变量具有文件作用域、外部链接、静态存储期。声明在函数中的变量是自动变量，除非该变量前面使用了其它关键字，它们具有块作用域、无链接、自行存储期。以static关键字声明在函数外部的变量具有块作用域、无链接、静态存储期。以static关键字声明在函数外部的变量具有文件作用域、内部链接、静态存储期。以static关键字声明在函数外部的变量具有温恩键作用域、内部链接、静态存储期。

C11新增了一个存储类别说明符:\_Thread\_local。以该关键字声明的对象具有线程存储期，看意思是在线程中声明的对象在该线程运行期间一直存在，且在线程开始时被初始化。因此，这种对象属于线程私有。

属性：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 存储类别 | 存储期 | 作用域 | 链接 | 如何声明 |
| 自动 | 自动 | 块 | 无 | 在块中 |
| 存储器 | 自动 | 块 | 无 | 在块中，使用关键字register |
| 静态、外部链接 | 静态 | 文件 | 外部 | 在所有函数外部 |
| 静态、内部链接 | 静态 | 文件 | 内部 | 在所有函数外部，使用关键字static |
| 静态、无链接 | 静态 | 块 | 无 | 在块中，使用关键字static |
| 线程、外部链接 | 线程 | 文件 | 外部 | 在所有块的外部，使用关键字\_Thread\_local |
| 线程、内部链接 | 线程 | 文件 | 内部 | 在所有块的外部，使用关键字static和\_Thread\_local |
| 线程、无链接 | 线程 | 块 | 无 | 在块中，使用关键字static和\_Thread\_local |

注意：关键字extern只能用来再次声明在别处已定义过的变量。在函数外部定义变量，该变量具有外部链接属性。

另外，C还提供了动态分配内存。（malloc()函数系列）

5. 限定符

关键字

使用下面关键字限定变量：const、volatile、restrict

一般注释

限定符用于限制变量的使用方式。不能改变初始化以后的const变量。编译器不会假设volatile变量不被某些外部代理改变，restrict限定的指针是访问它所指向内存的唯一方式（在特定作用域中）。

const int\* ptr = &joy; 声明创建了指针ptr，该指针不能用来改变量joy的值，但是它可以指向别的位置。

int\* const ptr = &joy; 声明创建了指针ptr，不能改变该指针的值，即ptr 只能指向joy，但是可以用它来改变joy的值。

void simple(const char\* s);声明表明形参s被传递给simple()的值初始化后，simple()不能改变s指向的值。

void simple(int\* const pi);……

void interleave(int\* restrict pi,int\* restrict p2, int n);声明表明p1和p2是访问它们所指向内存的唯一方法，这意味着这两个块不能重叠。

6. goto语句

Example:

#include<stdio.h>

int main()

{

goto label;

printf("222\n");

printf("333\n");

label : printf("111\n");

return 0;

}

Console: 111\n

7. ctype.h

|  |  |
| --- | --- |
| isalnum | 字母或数字 |
| isalpha | 字母 |
| isblank | 空格或制表符 |
| iscntrl | 控制字符（如Ctrl+B） |
| isdigit | 数字 |
| isgraph | 非空格打印字符 |
| islower | 小写字符 |
| isprint | 打印字符 |
| ispunct | 标点字符（除空格、字母、数字） |
| isspace | 空白字符（空格、换行符、换页符、回车符、垂直或水平制表符、或其他实现定义的字符） |
| isupper | 大写字符 |
| isxdigit | 十六进制数字字符 |
| tolower | 返回小写，否自不动 |
| toupper | 大写 |

8. math.h

|  |  |
| --- | --- |
| double ceil(double x); | 向上取整 |
| double fabs(double x); | 绝对值 |
| double floor(double x); | 向下取整 |
| double round(double x); | 四舍五入 |
|  |  |

9. stdlib.h

|  |  |
| --- | --- |
| double atof(const char\* nptr); | 返回把字符串nptr开始部分的数字（和符号）字符转换为double类型的值，跳过开始的空白，遇到第一个非数字字符时结束转换；未发现return 0; |
| int atoi | int |
| int atol | long |
| double strtod(const char\* restrict npt,char\*\* restrict ept); | 类似atof，转换成功把数字后第一个私服的地址赋给ept指向的位置；转换失败npt赋值给ept指向的位置。 |
|  |  |