

# C语言

## 基础知识

### 总体

程序结构是三种：**顺序结构**、**选择结构(分支结构)**、**循环结构**。

C语言基本数据类型分为**整型**、**实型**、**字符型**。

**内部变量**：在C语言中,在**函数体内部的变量**,叫内部变量,也叫**局部变量**。

**外部变量**：是在函数外部定义的**全局变量**，它的作用域是从变量的定义处开始，到本程序文件的结尾。在此作用域内，全局变量可为各个函数所引用。编译时将外部变量分配在**静态存储区**。

C源文件的基本单位是**函数**

### 标准输入输出文件

键盘是标准输入文件 显示器是标准输出文件。

### ++

注意，++运算符会在比较完继续运行。

#### 一定注意前后关系

```
int n = 3;
int m;
m = -n++;
```

注意++在变量后，所以先进行赋值。

又因为-的优先级大于赋值

所以有

```
m = -n;
n = n + 1;
m = -3;
n = 4;
```

### 标识符

标识符分为**关键字**、**预定义标识符**、**用户标识符**。

**关键字**：不可以作为用户标识符号。**main define scanf printf 都不是关键字**。迷惑你的地方**if**是可以做为用户标识符。**因为if中的第一个字母大写了，所以不是关键字**。

**预定义标识符**：背诵define scanf printf include。记住预定义标识符可以做为用户标识符。

**用户标识符**：自定义。

# 整数与实数

C语言提供的**实型**变量有两种类型：**单精度（float）**和**双精度（double）**

## 基本数据类型

C语言包含的数据类型**short、int、long、char、float、double**的六种基本数据类型

## 存储类别

C语言中存储类别又分为四类：**自动（auto）**、**静态（static）**、**寄存器的（register）**和**外部的（extern）**。

## 表达式

int x=y=10: 错啦，**定义时，不可以连续赋值。**

%符号两边要求是整数。

**逗号表达式**优先级别**最低**。**表达式的数值逗号最右边的那个表达式的数值。**

z=(2, 3, 4)的z的数值就是4。

z=2,3,4 z的值是2。

注释不是C语言

## 赋值

赋值的结合性是自右向左

所以如果在while里面赋值，计算机选择左值进行判断。

## 强转

一定是（int）a，注意类型上一定有括号的。

## 字符

大写字母和小写字母转换的方法： 'A'+32='a' 相互之间一般是相差32。

'1' 是字符占一个字节，"1"是字符串占两个字节(含有一个结束符号)。

## 位运算

转换成二进制计算

<<左移一位表示乘以2；>>右移一位表示除以2。

&表示按位与

|表示按位或

^表示按位异或 两个位相同则为0，不同则为1

~表示取反

## 标准输入输出

格式说明	表示内容	格式说明	表示内容
%d	整型 int	%c	字符 char
%ld	长整型 long int	%s	字符串
%f	浮点型 float	%o	八进制
%lf	double	%#o	带前导的八进制
%%	输出一个百分号	%x	十六进制
%5d		%#x	带前导的十六进制

注：带前导的是指会补上0x（十六进制） 0（八进制）

**注意注意注意！： %f会补足六位小数！！！！** printf ("%10f", 1.25) ; 小数要求补足6位的，没有六位的补0,。结果为 1.250000

这个有推广的意义，注意 x = (int) x 这样是把小数部分去掉。

## break continue

break：退出一层循环

continue：跳过本次循环

## 关系表达式

< > <= >= 这些比较符是**自左向右**

1<0<2 最后的返回值是1

因为要1<0为假得到0，表达式就变成了0<2那么运算结果就是1

!> && > || 优先的级别

## 条件表达式 条件运算符

表达式1？表达式2：表达式3

真前假后

结合方向：自右向左

## 循环结构

do-while()循环的最后一个while();的分号一定不能够丢。do - while循环是至少执行一次循环。

## while ( (c=getchar()) !='\n') 和 while (c=getchar() !='\n') 的差别

先看a = 3 != 2 和 (a=3) !=2 的区别： (! =号的级别高于=号 所以第一个先计算 3 !=2)

第一个a的数值是得到的1；第二个a的数值是3。

# 函数

如何求阶层：n!

```
int fun(int n)
{
    int p=1;
    for(i=1;i<=n;i++) p=p*i;
    return p;
}
```

函数的参数可以是常量，变量，表达式，甚至是函数调用（类似递归）。

## 地址

数组名：表示第一个元素的地址。数组名不可以自加，他是地址常量名。（考了很多次）

函数名：表示该函数的入口地址。

字符串常量名：表示第一个字符的地址。

## 数组

### 一维数组

数组的下标的下限是0

a表示数组名，是第一个元素的地址，也就是元素a[0]的地址。（等价于&a）

### 二维数组

二维数组初始化不能缺省列的数量

**行指针：指的是一整行，不指向具体元素。**

**列指针：指的是一行中某个具体元素**

**行指针**

初始化：int (\*p)[5]= &a[0]; 指向每行有5列的指针

**列指针**

初始化：int \*p= a[0];

## 枚举

关键字 enum

```
enum week{ Mon, Tues, wed, Thurs, Fri, Sat, Sun };
```

Mon的值是0 依次加一

也可以直接赋值并依次加一

## 注意事项

1. 枚举列表中的 Mon、Tues、Wed 这些标识符的作用范围是全局的（严格来说是 main() 函数内部），不能再定义与它们名字相同的变量。
2. Mon、Tues、Wed 等都是**常量**，不能对它们赋值，只能将它们的值赋给其他的变量。

## 字符串

---

### 相关函数

#### strcat函数——字符串连接函数

一般形式：strcat(字符数组1, 字符数组2);

#### strcpy/strncpy函数——字符串复制函数

一般形式：strcpy(字符数组1, 字符串2);

作用：将字符串2复制到字符数组1中去。

#### strcmp函数——字符串比较函数

一般形式：strcmp(字符串1, 字符串2);

作用：用来比较两个字符串的差异,从第一个字符依次比较

#### strlen函数——测字符串长度的函数

注意 strlen函数遇到第一个\0就会停下。

#### strlwr函数——转换为小写的函数

一般形式：strlwr(字符串)

#### strupr函数——转换为大写的函数

一般形式：strupr(字符串)

## 常见错误

```
char a[5], *p=a;
p="abcd";
/*
此时双引号创建了一个新内存，并返回了地址给p
*/
printf("%s", p);
for(int i=0; i<=5; i++){
    printf("%c", a[i]);
}
/*
数组a的内容并没有改变
*/
```

如果将第二行改为

```
*p='a';
```

此时修改了a数组的第一个元素、

## 指针

指针运算只能进行加减运算，其他运算是不允许的。

## 函数指针

返回值类型 (\* 指针变量名) ([形参列表]);

1、int func(int x); /\* 声明一个函数 \*/

2、int (\*f)(int x); /\* 声明一个函数指针 \*/

3、f=func; /\* 将func函数的首地址赋给指针f \*/

## 结构体

结构体指针取值的两种表达方式

```
struct temp {  
    int a;  
};  
struct temp *p;  
//两种取值表达方式  
p->a;  
*(p).a;
```

## 宏定义

### 含义

用一个指定的标识符来代表一个字符串，#define 标识符 字符串

### 易错知识

看这样一个例子

```
#define f(x)  x*x
```

宏代换是直接进行代换的。

定义	实际为
int a = f(3);	int a = 3*3;
int a = f(3+1);	int a = 3+1*3+1;
int a = f((3+1));	int a = (3+1)*(3+1);

实际上应该是

```
#define f(X) ((X)*(X))
```

所以一定要用括号括住所有地方

## 文件操作

### fopen

```
FILE * fopen(char *filename, char *mode);
```

第一个参数：文件名字

第二个参数：mode打开文件方式

mode	意义	备注
r	只读	文件必须存在，否则打开失败
r+	读写	文件必须存在。在只读 r 的基础上加 '+' 表示增加可写的功能。下同
rb	二进制读	功能同模式“r”，区别：b表示以二进制模式打开。下同
rb+	二进制读写	功能同模式“r+”。二进制模式
rt	只读	在Windows机器上，当读入文本时，回车字符\r被删除。
rt+	读写	读写打开一个文本文件，允许读和写
a	追加只写	若文件存在，则位置指针移到文件末尾，在文件尾部追加写入，故该方式不删除原文件数据；若文件不存在，则打开失败
a+	读写	在“a”模式的基础上，增加可读功能
ab	二进制追加	功能同模式“a”。二进制模式
ab+	二进制读写	功能同模式“a+”。二进制模式
w	只写	若文件存在，则清除原文件内容后写入；否则，新建文件后写入
w+	读写	新建一个文件，先向该文件中写入数据，然后可从该文件中读取数据
wb	二进制写	功能同模式“w”。二进制模式
wb+	二进制读写	功能同模式“w+”。二进制模式

### fclose

```
int fclose(FILE * stream);
```

简述：打开文件流记得关闭哦~

第一个参数：文件流

返回值：success: 0, failure: EOF

### fwrite

```
size_t fwrite(void *buffer, size_t size, size_t count, FILE * stream);
```

简述：将buffer指向的内存数据块写入fp所指的文件

第二个参数：写入元素的大小

第三个参数：写入元素的数量

第四个参数：文件流

返回值：返回值比较有意思是size\_t,元素的数量，你可以认为是unsigned int/long，具体还是根据操作系统。

## fread

```
size_t fread ( void *buffer, size_t size, size_t count, FILE * stream );
```

简述：函数从文件 fp 中读出“size\*count”个字节保存到 buffer 中~  
buffer是缓冲区 所以read的意思是查找并存入缓冲区

第一个参数：保存元素的指针

第二个参数：提取元素的大小

第三个参数：提取元素的数量

第四个参数：文件流

返回值：返回值比较有意思是size\_t,元素的数量，你可以认为是unsigned int/long，具体还是根据操作系统。

## fgetc fputc

```
int fgetc(FILE *fp);  
int fputc(int c, FILE *fp);
```

fgetc的作用是读取一个字符并把指针指向下一个字符。

若读取成功则返回该字符，若到文件末尾则返回EOF。

fputc是将字符c写入fp所指的文件中。

若成功则返回字符c，失败则返回EOF。

## fgets fputs

```
char *fgets(char *s, int n, FILE *fp);  
int fputs(const char *s, FILE *fp);
```

fgets是将fp所指的文件中读取字符串并在字符串末尾添加'\0'，然后存入s中。（最多读取n-1个字符）。

fputs是将s中的字符串输入到fp所指的文件中。

## fscanf fprintf

```
fscanf(fp, "%d", &stu[i].ID);  
fprintf(fp, "%d", stu[i].ID);
```

第一个参数是文件指针，后面两个和之前的一样。

## feof

```
int feof(FILE *fp);
```

当fp指向末尾时**返回非0值**，否则返回0。



-----by Jerry