stdio.h文件

**gets()**读取整行输入，直至遇到换行符。然后丢弃换行符，存储其余字符，并在这些字符的末尾添加一个空字符使其成为一个c字符串。

**puts()**把字符串的地址作为参数传递给他，在显示字符串时会自动在其末尾添加一个换行符，该函数在遇到空字符时就停止输出，所以必须确保有空字符

**fgets()**函数，第一个参数是输入变量。通过第二个参数兼职读入的字符数来解决溢出的问题。第三个参数是目标指针。该函数专门设计拥挤处理文件输入，所以一般情况下可能不太好用。函数返回第一个参数的地址。

和gets()的区别

1. fgets()函数的第二个参数指明了读入字符的最大数量。如果该参数的值是n，那么fgets()将读入n-1个字符，或者读到遇到的第一个换行符为止。
2. 如果fgets()读到一个换行符，会把他储存在字符串中。这点与gets()不同，gets()会丢弃换行符。
3. fgets()函数的第三个参数指明要读入的文件。如果读入从键盘输入的数据，则以stdin（标准输入）作为参数，该标识符定义在stdio.h中。

**fputs()**第一个参数是输出变量。第二个参数指明他要写入的文件。如果显示在计算机的显示器上，应该使用stdout（标准输出）作为该参数。返回指向char的指针。如果一切顺利，该函数返回的地址与传入的第一个参数相同。但是，如果读到文件的结尾，他将返回一个特殊的指针：空指针。该指针保证不会只想有效的数据，所以用于表示这种特殊情况。

空指针：可以用数字0来代替，不过在c语言中用宏NULL来代替更常见（如果在读入数据是出现某些错误，该函数也返回NULL）。

空字符是整数类型，而空指针是指针类型。两者有时容易混淆的原因是：他们都可以用数值0来表示。但是，从概念上看，两者都是不同类型的0。另外，空字符是一个字符，占1字节；而空指针是一个地址；通常占4字节。

C11新增的**gets\_s（）**函数，用一个参数限制读入的字符数。

和fgets（）的区别

1. gets（）只从标准输入中读取输入，所以不需要第三个参数。
2. 如果gets\_s（）读到换行符，会丢弃它而不是储存他。
3. 如果gets\_s（）读到最大字符数都没有读到换行符，会执行以下几部。首先把目标数组中的首字符设置为空字符，读取并丢弃随后的输入直至读到换行符或文件结尾，然后返回空指针。接着，调用依赖实现的“处理函数“（或你选择的其他函数），可能会终止或退出程序。

string.h文件

**size\_t strlen(const char \*s)**该函数返回s字符串的字符数，不包括末尾的空字符

**strcat（）**（用于拼接字符串）函数接收连个字符串作为参数。该函数把第二个字符串的备份附加在第一个字符串末尾，并把拼接后形成的新字符串作为第一个字符串，第二个字符串不变。strcat（）函数的类型是char\*（即指向char的指针）。strcat（）函数返回第一个参数，即拼接第二个字符串后的第一个字符串地址。

缺点：无法检测第一个数组是否能容纳第二个字符串。如果分配给第一个数组空间不够大，多出来的字符一处到相邻存储单元是就会出现问题。

**strncat（）**该函数第三个参数指定了最大添加字符数。

**strcmp（）**函数（用于字符串比较）。该函数通过比较运算符来比较字符串，就像比较数字一样。如果两个字符串相同，该函数就返回0，否则返回非零值。如果在字母表中的第一个字符串位于第二个字符串前面，返回负数；反之返回正数。而且返回两者ASCII码之差。比较所有的字符，不只是字母。所以，与其说该函数按字母顺序进行比较，不如说是按机器排序序列进行比较，即根据字符的数值进行比较（通常都使用ASCII值）。

**strncmp（）**函数比较第三个参数指定的字符。

**strcpy（）**（用于拷贝）函数拷贝整个字符串从临时数组拷贝至目标数组中。相当于字符串赋值运算符。第二个参数指向的字符串被拷贝至第一个参数指向的数组中。拷贝出来的字符串被称为目标字符串，最初的字符串被称为源字符串。返回类型是char\*，该函数返回的是第一个参数的值，即第一个字符的地址。第二个，第一个参数不必指向数组的开始。这个属性可用于拷贝数组的一部分。把源字符串中的空字符也拷贝在内。

**strncpy（）**函数第三个参数指明可拷贝的最大字符数。

**fprintf（）**函数定义在stdio.h他是把数据写入字符串，而不是打印在显示器上。因此，该函数可以把多个元素组合成一个字符串。第一个参数是目标地址，接受stderr作为IO指针。其余参数和printf（）函数相同，即格式字符串和待写入项的列表。

**char \*strchr（const char \*s， int c）；**

如果s字符串包含c字符，该函数返回指向s字符串首位置的指针（末尾的空字符也是字符串的一部分，所以在查找范围内）；如果在字符串s中未找到c字符，该函数则返回空指针。

**char \*strpbrk(const char \*s1,const char \*s2);**

如果s1字符中包含s2字符串中的任意字符，该函数返回指向s1字符串首位置的指针；如果在s1字符串中未找到任何s2字符串中的字符，则返回空字符。  
**char \*strchr(const char \*s, int c);**

它表示在字符串 s 中查找字符 c，返回字符 c 第一次在字符串 s 中出现的位置，如果未找到字符 c，则返回 NULL。也就是说，strchr 函数在字符串 s 中从前到后（或者称为从左到右）查找字符 c，找到字符 c 第一次出现的位置就返回，返回值指向这个位置，如果找不到字符 c 就返回 NULL。

**char \*strrchr(const char \*s, int c)；**

该函数返回s字符串中c字符的最后一次出现的位置（末尾的空字符也是字符串的一部分，所以在查找范围内）。如果未找到c字符串，则返回空指针。

**char \*strstr(const char \*s1, const char \*s2)；**

该函数返回指向s1字符串中s2字符串出现的首位置。如果在s1中没有找到s2，则返回空指针。

**strlwr()**:函数将参数字符串中的大写转换成小写

**strupr()**:函数将参数字符串中的小写转换成大写

**void \*memcpy(void \*restrict s1,const void \*restrict s2,size\_t n)**:该函数从s2指向的位置拷贝n字节到s1指向的字节，返回s1的值。参数带关键字restrict，即假设两个内存区域之间没有重叠。如果出现重叠结果是未定义的。

**void \*memmove(void \*s1,const void \*s2,size\_t n)**:该函数从s2指向的位置拷贝n字节到s1指向的字节，返回s1的值。

stdlib.h文件

**atoi（）**函数把字符串换成double类型的值

**atol（）**函数把字符串换成long类型的值

**atof（）**函数把字符串换成float类型的值

更智能的函数

**strtol（）**函数把字符串换成long类型的值

**strtoul（）**函数把字符串换成unsigned long类型的值

**strtod（）**函数把字符串换成double类型的值

优点：识别和报告字符串中的首字符是否是数字

**内存分配函数**

void \*aligned\_alloc( size\_t alignment , size\_t size);

第一个参数代表指定的对齐，第二个参数是所需的字节数，其值应该是第一个参数的倍数。需要使用free来释放之前分配的内存

**malloc()**:该函数接受一个参数：所需的内存字节数。malloc（）函数会找到合适的空闲内存块，这样的内存是匿名的。也就是说malloc（）分配内存，但是不会为其赋名。然而，他确实返回动态内存块的首字节地址。因此，可以把该地址赋给一个指针变量，并使用指针访问这块内存。内存分配失败返回NULL。

现在创建数组的3种方法

1. 声明数组时，用常量表达式表示数组维度，用数组名访问数组的元素。可以用静态内存或自动内存创建这种数组。
2. 声明变长数组（C99新增的特性）时，用变量表达式表示数组的维度，用数组名访问数组的元素。具有这种特性的数组只能在自动内存中创建。
3. 声明一个指针，调用malloc（），将其返回值赋给指针，使用指针访问数组的元素。该指针可以是静态的或自动的

**free()**：函数的参数是之前malloc（）返回的地址，该函数释放之前malloc（）分配的内存。free()的参数应该是一个指针，指向由malloc（）分配的一块内存。不能用free（）释放通过其他方式分配的内存。

**exit()**:函数结束程序，EXIT\_FAILURE表示程序异常终止，EXIT\_SUCCESS（或者，相当于0）表示普通的程序结束。

**calloc()**:函数接受两个无符号整数作为参数。第一个是所需的存储单元数量，第二个参数是存储单元的大小。也可以用free（）释s放内存。

**atexit()**:函数通过退出时注册被调用的函数提供这种功能，接受一个函数指针作为参考。

**void qsort(void \*base, size\_t nmemb, size\_t size, int (\*compar)(const void \*,cont void \*));**

第一个参数是指针，指向待排序数组的首元素。第二个参数是待排序项的数量。第三个参数提供元素的大小。第四个参数是函数的指针。分别指向得比较两项的指针。如果第一项的值大于第二项，比较函数则返回整数；如果两项相同，返回0，如果小于，负数。

assert.h头文件

**assert()**:接受一个整形表达式作为参数，如果表达式求知为假，assert()就在标准错误流中写入一条错误信息，并且终止程序。

**\_Static\_assert**:接受两个参数，第一个参数是整形常量表达式，第二个参数是一个字符串。如果第一个表达式求值为0，则编译器会显示字符串，而且不编译该程序。（**c11**）

stdarg.h

**va\_list**:类型代表一种用于存储形参对应的形参列表中省略号部分的数据对象。

**va\_start(va\_list name,parmN)**:把name初始化为参数列表，该宏有2个函数，va\_list类型的变量和parmN形参。

**va\_arg()**:第一个参数是va\_list类型的变量和一个类型名。第一次调用函va\_arg()时，他返回参数列表的第一项；第二次调用是返回第二项，以此类推。

**Va\_end(ap)**:参数接受va\_list类型的变量，释放动态分配用于存储参数的内存

**指针**

地址运算符：&

一般注释：后跟一个变量名时，&给出该变量的地址

示例&nurse表示变量nurse的地址

地址运算符：\*

一般注释：后跟一个指针名或者地址是，\*给出储存在地址指向地址上的值

示例：nurse=22；

ptr=&nurse；//指向nurse的指针

val=\*ptr；//把ptr指向的地址上的值赋给val

名词概念

EOF：是文件结尾

ctype.h文件

参数都是int类型

**isalnum()**字母数字

**isalpha()**字母

**isdigit()**数字

**islower()**小写字母

**isspace()**空白字符

**isupper()**大写字母

**tolower()**如果参数是大写字符，该函数返回小写字符；否则，返回原始参数

**toupper()**如果参数是小写字符，该函数返回大写字符；否则，返回原始参数

作用域：

块作用域：作用域块内

函数原型作用域：用于函数原型中的形参名，范围是从形参定义出到原型声明结束。

文件作用域：变量定义在函数的外面，具有文件作用域，从它的定义处到该定义所在的文件的末尾均可见。

链接：

外部链接：可以在多文件程序中使用

内部链接：只能在一个翻译单元中使用

存储期：

静态存储期：在程序的执行期间一直存在。

自动存储期：在进入块时，会分配内存，当退出这个块时会释放刚刚的内存

limits.h

CHAR\_BIT表示每个字节的位数

关键字

static:对于文件作用域变量，关键词static表明了其链接属性，并非存储期，static声明的文件作用域变量具有内部链接，不能在函数的形参中使用。

auto:是存储类别说明符，强调自动储存类别

register:请求形参是寄存器变量

extern:为了指出该函数使用了外部变量，可以在函数中用关键字extern再次声明。如果一个源代码文件使用的外部变量定义在另一个源代码文件中，则必须用extern在该文件中声明该变量。

typedef:声明变量名代替

限定符

volatile：限定符告知计算机，代理可以改变该变量的值。通常，他被用于硬件地址以及在其他程序或同时运行的线程中共享数据。

restrict:关键字允许编译器优化某部分代码以更好地支持计算。他只能用于指针，表明该指针是访问数据对象的唯一且初始的方式。restrict关键字有两个读者，一个是编译器，该关键字告知编译器可以自由假定一些优化方案。另一个是读者用户，该关键字告知用户要使用满足restrict要求的参数。总而言之，编译器不会检查用户是否遵循这一限制，但是无视他后果自负。

\_Atomic类型限定符（C11）：并发程序设计把程序执行分成可以同时执行多个线程。C11通过包括可选的头文件stdatomic.h和threads.h，提供了一下可选的（不是必须实现的）管理方法。值得注意的是，要通过各种宏函数来访问原子类型。当一个线程对一个原子类型的对象执行原子操作时，其他线程不能访问该对象。

原子操作：所谓原子操作是指不会被线程调度机制打断的操作；这种操作一旦开始，就一直运行到结束。

**rand()**：生成随机数在0-RAND\_MAX之间，RAND\_MAX定义在stdlib.h中，其值通常是INT\_MAX

文件函数

**fopen()**:函数声明在stdio.h。他的第一个参数是待打开文件的名称，更确切地说是一个包含改文件名的字符串地址。第二个参数是一个字符串，指定待打开文件的模式。成功打开文件后，fopen（）返回文件指针，失败返回NULL。

|  |  |
| --- | --- |
| 模式字符串 | 含义 |
| “r” | 以读模式打开文件 |
| “w” | 以写模式打开文件，把现有文件的长度截成0，如果文件不存在，创建一个文件 |
| “a” | 以写模式打开文件，在现在有文件末尾添加内容，如果文件不存在，则创建一个文件 |
| “r+” | 以更新模式打开文件（即可以读写文件） |
| “w+” | 以更新模式打开文件（即，读和写），如果文件存在，则将其长度截为0；如果文件不存在，则创建一个新文件 |
| “a+” | 以更新模式打开文件（即，读和写），在现有文件的末尾添加内容，如果文件不存在则创建一个新文件；可以读整个文件，但是只能从末尾添加内容 |
| “rb”,”wb”,”ab”,”ab+”,”a+b”,”wb+”,”w+b”,”ab+”,”a+b” | 与上一个模式类似，但是以二进制模式而不是文本模式打开文件 |
| “wx”,”wbx”,”w+x”,”wb+x”或“w+bx” | （c11）类似非x模式，但是如果文件已存在或以独占模式打开文件，则打开文件失败 |

文件指针的类型是指向FILE的指针，定义在stdio.h中

**getc()**:从文件中拿出一个字符，ch=getchar()与ch=getc(fp)相同不过需要一个指针来指向哪个文件

**putc()**:将字符放入文件指针指定的文件，第一个参数是待写入的字符，第二个参数是文件指针

**fclose()**:函数关闭指定的文件，必要时刷新缓冲区。关闭成功返回0，否则返回EOF

**fscanf()**:函数第一个参数是文件指针，剩下的和scanf()一样

**rewind()**：函数让程序回到文件开始处，接受一个文件指针作为参数。

**fseek()**:在fopen()打开的文件中直接移动到任意字节处。有三个参数，返回int类型的值。第一个参数是FILE指针，指向待查找的文件，第二个参数是偏移量。该阐述表示从起始点开始要移动的距离。参数必须是一个long类型的值，一个为正（前移）、负（后移）或0（保持不动），第三个参数是模式，该参数确定起始点。正常返回0，出现错误返回值为-1。

|  |  |
| --- | --- |
| 模式 | 偏移量的起始点 |
| SEEK\_SET | 文件开始处 |
| SEEK\_CUR | 当前位置 |
| SEEK\_END | 文件末尾 |

**ftell()**:函数返回一个long类型的值，表示文件中的当前位置

**int fgetpos(FILE \* restrict stream, fops\_t \* restrict pos)**:函数返回0，如果失败返回非0

**int fsetpos(FILE \* stream, const fpos\_t \*pos)**:函数返回0，如果失败返回非0

**fpos\_t**代表file position type,文件定位类型，他的变量或者数据对象可以在文件中指定一个位置，它不能是数组类型。

**int ungetc(int c, FILE \*fp)**:函数把c指定的字符放回输入流中。如果把一个字符放回输入流，下次调用标准输入函数是将读取该字符。

**int fflush(FILE \*fp)**：调用该函数引起输出缓冲区所有的未写入数据被发送到fp指定的文件。这个过程被称为刷新缓冲区。如果fp是空指针，所有输出缓冲区都被刷新。

**int setvbuf(FILE \* restrict fp, char \* restrict buf, int mode , size\_t size)**:函数创建了一个标准I/O函数替换使用的缓冲区，在打开文件后未对流进行其他操作之前，调用该函数。指针fp识别待处理的流，buf指向待使用的存储区。如果buf的值不是NULL，则必须创建一个缓冲区。size告诉setvbuf（）数组的大小。mode的选择如下：\_IOFBF表示完全缓冲；\_IOLBF表示行缓冲；\_IONBF表示无缓冲。如果操作成功，函数返回0，否则返回一个非零值。

**size\_t fwrite(const void \* restrict ptr, size\_t size , size\_t nmemb, FILE \* restrict fp)**:函数把二进制数据写入文件，ptr是待写入数据块的地址，size表示待写入数据块的大小，nmemb表示待写入数据块的数量，fp指定待写入的文件。成功返回写入项的数量，就是nmemb，但是出现写入错误，返回值会比nmemb小。

**size\_t fread(void \* restrict ptr, size\_t size, size\_t nmemb, FILE \* restrict fp)**:接受的参数和fwrite相同。ptr是待读取文件数据在内存中的地址，fp指定待读取的文件。函数返回成功读取项的数量。正常情况下，该返回值就是nmemb，但如果出现读取错误或读到文件结尾，该返回值就会比nmemb小。

标准输入函数返回EOF，则通常表明函数已达到文件结尾。然而出现读取错误是，函数也会返回EOF

**int feof(FILE \* fp)**:当上一次调用检测到文件结尾是，函数会返回一个非零值，否则返回0

**int ferror(FILE \* fp)**:当读或写出现错误，函数返回一个非零值，否则返回0

声明符号优先级

1. 数组名后面的[]和函数名后面的()具有相同的优先级。他们比\*（解引用运算符）的优先级高。
2. []和()的优先级相同，由于都是从左往右结合，所以下面的声明中，在引用方括号之前，\*先于rusks结合。因此rusks是一个指向数组的指针，该函数内含10个int类型的元素

int (\* rusks) [10];

**结构体**

联合

在联合中，一次只储存一个值。即使有足够的空间，可不能同时存储一个char类型和一个int类型。

**位操作**

~：二进制反码

把1变成0，0变成1

&：与

当两个数对应的位都为1时计算结果为1

|：或

两个数对应的位有一个为1时计算结果为1

^：异或

两个数对应的位不一样时计算结果为1

**掩码**

MASK中对flag取&

0对位掩盖，1对位透明

打开位

flags=flags | MASK;

关闭位

flags=flags &~ MASK;

切换位

flags=flags ^ MASK;

检查值

if(flags == MASK )

puts(“Wow!”);

**移位运算符**

左移:<<

将其左侧运算对象每一位的值向左移动，丢失位用0补上

右移:>>

将其左侧运算对象每一位的值想有侧移动，丢失位用0补上

位移运算符

移位运算符针对2的幂提供快速有效的乘法和触发

number << n 乘以2的n次幂

number >> n 除以2的n次幂

**对齐特性（c11）**

对齐指的是如何安排对象在内存中的位置

\_Alignof 运算符给出一个类型的对齐要求，在关键字\_Alignof后面的圆括号中写上类型名即可： size\_t d\_align = \_Alignof(float);

一般而言，对齐值都应该是2的非负整数次幂。较大的对齐值被称为stricter或stronger，较小的对齐值被称为weaker。

非实现函数

**预处理器**

宏参数创建字符串：#运算符

#define PSQR(x) printf(“The square of “#x” is %d.\n”,((x)\*(x)));

Printf(“The square of “ “变量名“ “ is %d.\n”,((变量)\*(变量)));

预处理器粘合剂：##运算符

#define XNAME(n) x##n

意思是 xn（n为变量）

变参宏：…和\_ \_VA\_ARGS\_ \_

#undef

用于取消已经定义的#define指令

条件编译

#ifdef MAVIS

#include”hourse.h”//如果已经用#define定义了MAVIS，则执行下面的指令

#define STABLES 5

#else

#include”cow.h”//如果没有用#define定义MAVIS，则执行下面的指令

#define STABLES 15

#endif

#ifndef SIZE

#define SIZE 100//如果SIZE没有被定义则执行下面的语句

#endif

#if SYS==1（后面跟整形表达式，如果表达式为非零，则表达式为真，执行下面的语句）

#include “ibm.h”

#elif SYS==2

#include “cao.h”

#endif

其中#if define(SYS)等同于 #ifdef SYS

|  |  |
| --- | --- |
| 宏 | 含义 |
| \_ \_DATE\_ \_ | 与处理的日期 |
| \_ \_FILE\_ \_ | 表示当前源代码文件名的字符串字面量 |
| \_ \_LINE\_ \_ | 表示当前源代码文件中行号的整型常量 |
| \_ \_STDC\_ \_ | 设置为1时，表明实现遵循c标准 |
| \_ \_STDC\_HOSTED\_ \_ | 本机环境设置为1，否则设置为0 |
| \_ \_STDC\_VERSION\_ \_ | 支持C99标准，设置为199901L；支持C11标准，设置为201112L |
| \_ \_TIME\_ \_ | 翻译代码的时间，格式为”hh:mm:ss” |
| \_ \_func\_ \_ | 读取当前函数位置 |

#line 1000//把当前行号重置为1000

#line 10 “cool.c”//把当前行号重置为10，把文件名重置为cool.c

#error 让预处理器发出一条错误消息

#pragma把编译器指令放入源代码中

\_Pragma预处理运算符，把字符串转换成拨通的编译指令（c99）

C11新增的泛型选择表达式

\_Generic(x, int :0,float :1,double:2,default:3)

C99

内联函数

Inline static void eatline()内联函数

C11

\_Noreturn函数

表明调用完成后函数不返回主调函数