C语言知识基础

[文档副标题]

2023

# C程序基础

机器语言：计算机可以直接运行的语言，由0、1编写，一般为16个一组的机器指令

符号语言：即汇编语言，用英文和数字编写

高级语言：1.非结构化语言：BASIC FORTRAN

2.结构化语言：C

3.面向对象的语言：C++ Python java

高级语言需要编译程序将源程序转化为目标程序（由机器语言组成的程序）才可以被计算机运行

C语言创始人：

C源代码结构：编译预处理：#include、#define……

主函数main（）,每一个C语言程序只能有一个main函数

其他函数

程序设计语句

：

C语言是怎么从.c文件到exe文件的

1. 编写C语言文件
2. 编译C语言文件
3. 生成二进制.obj程序
4. 编译无误后再由连接器连接各个C语言文件编译生成的.obj程序
5. 连接无误，完成后即可生成exe可执行文件

数据的信息只能储存在内存中、C语言的底层是操作符

C语言想熟练运用需要学好算法与数据结构

程序：处理过程的顺序

程序设计=数据结构+算法+程序设计语句(C语言操作符的规范使用组合)

（代码函数：函数名称、函数意义、使用规则）

程序设计语言的基本成分：数据储存成分

数据运算成分

程序控制成分

数据传输成分

结构化程序设计要点：设计出的程序要可读性强，容易理解，便于维护

算法的基本结构：顺序结构、选择结构、循环结构

结构化程序设计方法：数据+流程控制+操作

结构化程序设计基本思想：1.自上而下，逐步细化

2.单入口，单出口

算法的特性:1.有穷性 2.目标确定性 3.有零个或多个输入,有一个或多个输出 4.执行有效性

C语言的代码：注释： /\*……\*/

换行： \n

打印在屏幕上： printf： 格式printf(“格式要求”，输出材料)

(同时打印时格式要 %d带符号的十进制整数

求间可不用逗号隔开） %u无符号十进制整数

%c单个字符，可能使用ASCLL表

%s字符串

%f 6位小数、%m.nf指定数据长度与小数位数、%-m.nf数据长度小于指定宽度时左对齐

%%输出百分号

%o用八进制输出

%x用十六进制输出

读取键盘输入 scanf 格式scanf(“格式要求”，&+输入材料)

同时输入时格式要求间可不用逗号隔开

输出一个字符 putchar 格式putchar()

输入一个字符 getchar 格式getchar()

scanf与printf的原理是输入或输出一个字符串，并在具有格式控制的位置进行格式输出或输入（scanf还有把数据送入到内存中）。

printf还可以指定打印宽度，m.n或-m.n

赋值运算符： = 赋值

等于： ==

C语言基础：

1.定义函数： 函数类型 函数返回值的数据类型 函数名（参数表） {函数实现}

形参：定义函数时所使用的形式参数，无具体参数值

实参：调用函数时所使用的有具体参数值的常量、变量或表达式

函数的返回值：由编写人决定

2.变量的作用域： 程序块级，函数分程序中从定义开始到语句结束

（即作用范围） 函数级，从函数内定义开始到函数结束

文件级，整个程序文件

程序级，程序中包含的所有文件

3.变量的储存类别：

静态存储变量:是指在程序运行期间分配固定的存储空间的方式。用关键词static表示

动态存储变量:是指在程序运行期间根据需要进行动态的分配 存储空间的方式,

自动型变量存储:存储空间的分配和释放工作是由编译系统自动完成的变量，被称为键字auto来定义自动变量

一般内存中供用户 使用的存储空间分为程序区、静态存储区和动态存储区三个部分

静态变量大多是全局变量，动态变量大多是局部变量

4.内部函数与外部函数： 内部函数是指可以被当前文件中其他函数调用，而不能被其他文件中的函数调用的函数，外部函数的作用域为整个程序，即可以被组成整个程序的所有源文件要在调用，只需要调用前声明一次就可以了，分别用extern和static表示

5.预处理： 使用#include指令，用于调用头文件（.h）或.c文件的变量、函数等。

宏定义代码替换：#define

条件编译：

函数调用：函数名 （“内写参数，没有可不写”）

标点符号：

{ } 在 struct 或 union 定义中划分出 结构体声明列表。

在 enum 定义中划分出 枚举列表。

划分出复合语句。复合语句可以是函数定义的一部分。

初始化时，划分出初始化器列表。

[ ]下标运算符。

声明或类型标识中，数组声明符的一部分。

初始化时，指派数组元素。(C99 起)

属性说明符中，划分出属性。(C23 起)

# 预处理指令起始标志。

字符串化的预处理操作。

## 记号粘贴的预处理操作

( )表达式分组，改变结合顺序。

函数调用运算符。

sizeof 或 \_Alignof (C11 起)表达式中划分出运算对象。

显式类型转换中划分出类型标识。

复合字面量中划分出标识。(C99 起)

声明或类型标识中分组，改变结合顺序。

函数声明（声明或类型标识）中划分出参数列表。

if、switch、while、do-while 或 for 语句中，划分出控制块。

仿函数宏定义中，划分出宏参数。

defined 、 \_\_has\_include 或 \_\_has\_c\_attribute (C23 起)预处理运算符的一部分。

泛型选择表达式的一部分。 (C11 起)

\_Atomic 类型说明符中,划分出类型标识。(C11 起)

静态断言声明中，划分出操作对象。(C11 起)

\_Alignas 说明符中，划分出操作对象。(C11 起)

属性中, 划分出操作对象。(C23 起)

; 如下语法的结束：

语句 (含 for 循环的初始化子句)

声明或 结构体声明列表

分隔 for 循环中的第二和第三子句。

: 条件运算符的一部分。

标签声明的一部分。

标志位域成员声明的 宽度 的起始。

泛型关联中，分隔类型标识或 default 和待选表达式。(C11 起)

... 函数声明的参数列表中, 标志变长参数。

宏定义中, 标志变参数宏。 (C99 起)

? 条件运算符的一部分。

:: 在属性中指示属性作用域。(C23 起)

. 成员访问运算符。

在初始化时， 标志 struct/union 成员指派符的起始 (C99 起)

-> 成员访问运算符。

~ 一元补运算符（又称逐位非运算符）。

! 逻辑非运算符。

+ 正号。

加法运算符

* 负号。

减法运算符。

\* 间接引用运算符（也称解引用运算符）。

乘法运算符。

声明符或类型标识中的指针运算符或成员指针运算符。

在函数声明的变长数组声明中的数组长度占位符。(C99 起)

/ 除法运算符。

% 取余运算符（取模运算符）。

^ 逐位异或运算符。

& 取地址运算符。

逐位与运算符。

| 逐位或运算符。

= 简单赋值运算符。

在初始化中，分隔对象和初始化列表。

在枚举定义中引入枚举常量的值。

+= 复合赋值运算符。

-= 复合赋值运算符。

\*= 复合赋值运算符。

/= 复合赋值运算符。

%= 复合赋值运算符。

^= 复合赋值运算符。

&= 复合赋值运算符。

|= 复合赋值运算符。

== 等于运算符。

!= 不等于运算符。

< 小于运算符。

#include 指令中标志头文件名起始。

> 大于运算符。

<= 小于等于运算符。

>= 大于等于运算符。

&& 逻辑与运算符。

|| 逻辑或运算符。

<< 移位运算符。

>> 移位运算符。

<<= 复合赋值运算符。

>>= 复合赋值运算符。

++ 自增运算符。

-- 自减运算符。

, 逗号运算符。

以下列表的列表分隔符：

声明中声明符列表

初始化，包括复合字面量 (C99 起)中的初始化器列表

函数调用中实参列表

枚举声明中的枚举项列表

仿函数宏定义中宏形参列表

泛型选择表达式中泛型关联列表(C11 起)

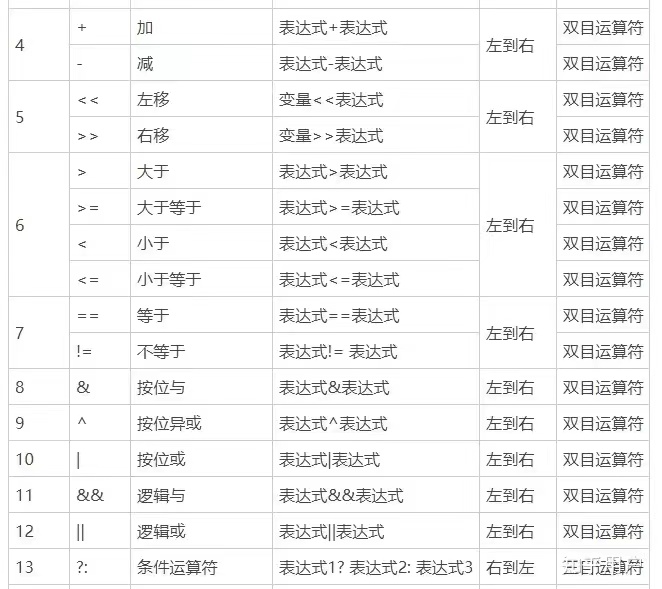
属性列表(C23 起)

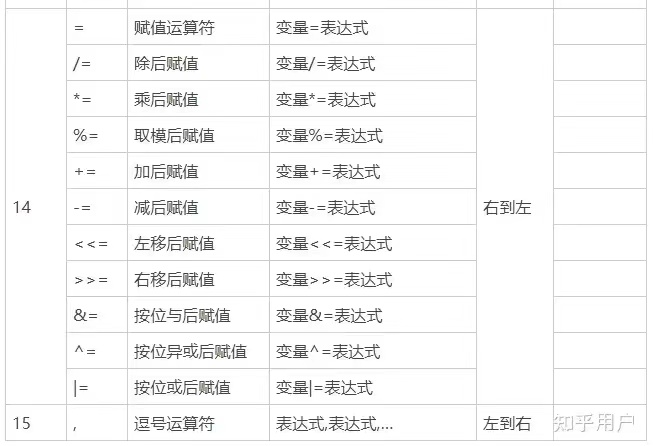
静态断言声明中分隔参数。(C11 起)

泛型选择表达式中，分隔控制表达式和泛型关联列表。(C11 起)

运算符优先级与括号







标识符：用于标记程序中代码函数及数据

标识符命名规则：1.只能以字母或下划线‘\_’开始（字母有大小写之分）

2.所有标识符都只由数字、字母、下划线组成

标识符分类：1.特定词

2.关键词

3.一般标识符

变量：程序运行中发生改变的数据

常量：程序运行中不发生改变的数据

声明变量的要求：声明变量时要声明变量的数据类型、变量名称、储存长度、

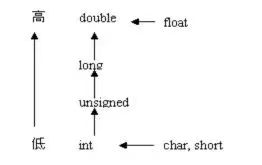
变量之所以具有数据类型视为了利用一个模板来方便开辟内存空间

原码与补码：数据在储存单元中储存方式为2进制，并且以0、1储存于内存中的字节中的各个位中，一个字节有8个位。字节中的第一位储存正负性，0为正，1为负。正数补码与原码相同，负数补码是将其正数原码的除第一位以外的其余位取相反（0变1,1变0）。

一个字节，储存数字为5

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

数据类型自动转换：



强制型数据转换：（数据类型名）表达式

sizeof运算符： 使用方法sizeof(数据类型或者变量标识符)

整型（分short、常型、long、long long四种类型）

字符型

基本类型 单精度型

浮点型 复合浮点型

双精度型

数组类型

枚举类型

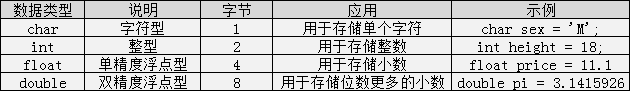
基本数据类型 构造类型 函数类型

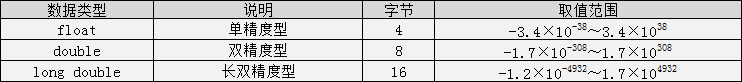
结构体类型

共用体类型

指针类型

空类型





初始化变量结构： 数据类型 变量名=初始值

整型常量表示方法 1.十进制正常书写

2.十六进制开头写0X，后面用1~9,A~F（代表10~15）表示

3.八进制开头写0，后接八进制数

4.长整形表示，后面加L

浮点型常量表示方法 1.十进制正常书写

2.指数型写为a\*E,E后加整数次方

字符常量 使用单引号括起来

字符串常量 使用双引号括起来

符号常量 使用define定义 格式：#define 符号名 数值

常变量 在数据类型前加const

算数运算符：

加法运算符： +

减法运算符： -

乘法运算符： \*

除法运算符 /

取余数运算： % 如10%3 的指为1

自加1、自减1： i++或++i i--或--i (i在前是i使用后加1，在后相反)

复合运算符 (前加初级运算符)= 用与自身替换运算

关系运算符： 大于 >

小于 <

大于等于 >=

小于等于 <=

不等于 !=

逻辑运算符： 与（即且） &&

或 ||

非 !

逻辑运算符优先级与结合性：

选择结构：

if单项选择型： if （条件） 函数代码语句

if else双项选择型： if （条件）

函数代码语句1

else

函数代码语句2

If else多层嵌套 if (表达式1) (语句1)

else if (表达式2) (语句2)

……………

else if (表达式n) (语句n)

else (语句n+1)

Switch多项多分支选择型： switch (表达式)

{ case 常量表达式

语句；break；

………重复至n }

循环结构：

while条件-循环结构: while(条件)

{

循环体

}

do-while循环-条件结构： do

{

循环体

}while(条件)；

for循环结构 for(判断式，判断为真的流程，判断为假的流程) 语句

for(循环变量初始值，循环条件，循环变量改变式) 语句

高级流程控制语句

goto语句 使用方法：goto 标号

break语句

continue语句

数组：由合法的同一数据类型构成的数据序列集合

数组中的元素占据连续的内存储存空间（即内存地址）

一维数组声明格式：

数据类型 数组名【内写元素个数（不说明个数就不用填写）】 = {………………}

二维数组声明格式：

数据类型 数组名【内写元素个数】【内写元素个数】 = {………………}

比如：

int shuju 【2】【4】={{1,2,3,4},{5,6,7,8}}

该数组表现

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |

字符数组声明格式：

Char 数组名[元素个数] = {………………………}

字符数组在内存中储存长度比元素个数多一个，多了一个结束符‘\0’

字符串输出函数：puts(数组名)

字符串输入函数：gets(数组名)

字符串连接函数：strcat

字符串复制粘贴函数：strcpy

字符串ASCLL码大小比较函数：strcmp

字符串长度比较函数：strlen

字符串小写转换函数：strlwr

字符串大写转换函数：strupr

定义数组时应对所有元素进行赋值，且元素个数不能超过定义时设置的值，未有赋值的元素系统会自动赋值为0。（字符型初始化为’\0’,指针型初始化为NULL）

数组元素引用（引用序列号所标志的数据）：

一维引用： 数组名 {元素序列号}

二维引用： 数组名 【元素序列号】【元素序列号】

其他类比

数组序列号：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G | H | I | J |

序列号 0 1 2 3 4 5 6 7 8 9

内存地址：内存中各个字节所对应的内存编号

指针： 数据的内存地址（指针本身只是一个地址，除了这个特性以外没有任何特性）

指针之所以运用得如此广泛，是凭借\*与&两个操作符

指针变量：用于储存其他变量的内存地址的工具

声明并初始化指针： 数据类型 \*指针变量名 = &变量名

空指针：将&变量名部分替换为NULL

\*是一种数据类型 指针类型

int \*是指指向整型的指针数据类型，其他类似

指针的用处：

（1）可以提高程序的编译效率和执行速度，使程序更加简洁。

（2）通过指针被调用函数可以向调用函数处返回除正常的返回值之外的其他数据，从而实现两者间的双向通信。

（3）利用指针可以实现动态内存分配。

（4）指针还用于表示和实现各种复杂的数据结构，从而为编写出更加高质量的程序奠定基础。

（5）利用指针可以直接操纵内存地址，从而可以完成和汇编语言类似的工作。

（6）更容易实现函数的编写和调用

指针的算数运算：可减不可加

结构体：

转意字符：

\’ 输出单引号字符’

\” 输出双引号字符”

\? 输出一个问号

\\ 输出反斜线字符\

\a 警告(alert) 产生声音或视觉信号

\b 退格(backspace) 将光标当前位置后退一个字符

\f 换页 将光标当前位置移到下一页的开头

\n 换行 将光标当前位置移到本行的开头

\r 回车 将光标当前位置移到下一行的开头

\t 水平制表符 将光标当前位置移到下一个Tab位置

\v 垂直制表符 将光标当前位置移到下一个垂直制表对齐点

\o、\oo或\ooo 输出与该八进制码ASCLL对应的字符

o代表一个八进制数字

\xh【h……】 输出与该十六进制码对应的与该十六

其中b代表一个十六进制数 进制码对应的字符