

C语言关键字详解

原创 行走路上的少年 2019-03-09 11:32:38 1178 收藏 30 版权

分类专栏: C语言

在这里先普及一下，在我们C语言中一共有32个关键字，可能有几个在我们学习的过程中不太常用，但是大多数都是我们常用的，下来我们先来罗列一下着32个关键字都有什么，再来——说一下他们的用法：

auto	int	double	long	char	float	short	signed
unsigned	struct	union	enum	static	switch	case	default
break	register	const	valatile	typedef	extern	return	void
continue	do	while	if	else	for	goto	sizeof

说关键字之前，先声明两个概念，在后面的文章中会用到：

1. 定义：如 (int i;)
- 所谓的定义就是（编译器）创建一个对象，为这个对象分配一块内存并给它取上一个名字，这个名字就是所说的变量名或对象名。名字一但匹配就不能更改。
2. 声明：如 (extern int i;)
- 告诉编译器，这个名字已经匹配到一块内存上了；告诉编译器，这个名字我先预定了，别的地方再也不能用它来作为变量名或对象名。

1、auto

用来声明自动变量。他是存储类型标识符，表明变量（自动）具有本地范围，块范围的变量声明（如for循环体内的变量声明）默认为auto存储类型。
大多普通声明方式声明的变量都是auto变量，不需要明确指出auto关键字，编译器默认为auto的，auto变量在离开作用域时程序自动释放，不会发生溢出情况（除了包含有指针类型的类）。使用auto的优势在于不用考虑变量是否被释放，比较安全。

2.1、register

register关键字请求编译器**尽可能**（这里是尽可能，不是一定，大家在用的时候要注意）的将变量存在CPU内部寄存器中而不是通过内存寻址访问，用来提高效率。这里要注意的是register关键字不是绝对的，因为CPU的寄存器就只有那么大，如果定义的register变量太多，就会溢出。就如向一个杯子倒水，总有满的时候，到时再到就会溢出。所以在一个程序中，不要多次用到register关键字。

2.2、使用register修饰符的注意点

register变量必须是能被CPU寄存器所接受的类型。这就意味着register变量必须是一个单个的值，并且其长度应小于和等于整型的长度。而且register变量可能不存放在内存中，所以不能用取值运算符“&”来获取register变量地址。

3、static的两种用法

1. 修饰变量

被static修饰的变量称为静态局部变量，作用域仅限于变量被定义的文件中，其他文件即使使用了extern声明也没法使用他。明确的说静态局部变量的作用域是从定义开始，到文件结束。所以一般在定义变量的时候不要使用static修饰变量，如果万不得已时可以把变量定义为全局变量。
静态局部变量，在函数里面定义的，就只能在这个函数里用了，同一个文档中的其他函数也用不了。由于被static修饰的变量总是存在内存的静态区，所以即使这个函数运行结束，这个静态变量的值还是不会被销毁，函数下次使用时任然能使用到这个值。比如下面的程序：

```
1 static int j;
2 void fun1(void)
3 {
4 static int i=0;
```

```

5 | i++;
6 | }
7 | void fun2(void)
8 | {
9 | j=0;
10 | j++;
11 | }
12 | int main()
13 | {
14 | for(int k=0;k<10;k++)
15 | {
16 | fun1();
17 | fun2();
18 | }
    return 0;

```

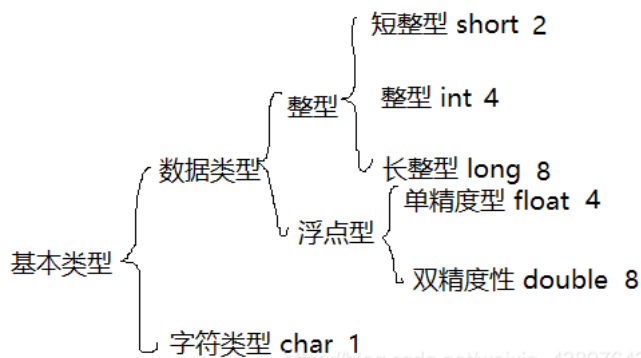
程序的运行结果为i=10; j=1;

1. 修饰函数

在函数前面加static使得函数成为静态函数。此处“static”的含义不是指储存方式，而是指对函数的作用域仅局限于本文件。使用内部函数的好处是：不同的人编写的函数，不担心自己的函数名和别人的函数名重复。

4、基本数据类型——short、int、long、char、float、double

下面说的字节数是在64位编译器中的：



在定义变量的时候根据变量要进行的功能和类型选择关键字的，一般情况下要计算有小数的选择double，因为double类型比float类型精度高。

5、sizeof

在学习的过程中sizeof经常会被误认为函数，其实它是一个关键字。sizeof是用来计算一个变量在内存中所占空间大小的关键字，一般情况下我们喜欢用它来计算字符数组的长度。

6、signed、unsigned

signed为带符号整型，数据在计算机中，存储的时候用0、1表示，一般的类型如int、short等系统会默认为带符号整型，就好比一个数前面不用加“+”号，我们会默认他为正的。计算机在表示一个数的时候会自动的把第一位表示为符号位的。

unsigned为无符号整型，它只能表示大于或等于0的数，要使用unsigned类型是，要在基本类型前面加上unsigned，这个不能省略。

7、if、else

这个组合大家见得太多了，在开始学C的时候就会接触，这里就不多说了。但是在写if后面的条件是，尽可能的简介清晰，一目了然。

7.1下面说几个与零值比较是，if后面正确的写法：

1. bool变量与零值比较

比如bool Show=FALSE;

一般比较为：if (Show) 、if (! Show) ;

2. float或double与零值比较

因为float和double类型的数据都是有精度限制的，所以不能用来直接和零比较，所以在一般实际应用中我们一般用无限接近零的小数和变量比较大小（这个无限接近零

的小数的大小可以根据你的实际所需要的精度大小自己调剂) 我们一般用 $e-n$ 表示, 这里 n 可以根据实际情况自己的定义大小 ($e-n$ 表示 $1 \times 10^{(-n)}$);

3. 指针变量与零值比较

我们一般有`if (NULL==p) ; if (NULL!=p)` 来对指针变量和零比较;

7.2、else和if匹配的问题

C语言规定: else始终与同一括号内最近的未匹配的if语句结合。这样就要求我们在书写代码的时候, 尽量把每一个if和else的括号都带上, 最好用缩进距离控制匹配情况。

8、switch、case组合

很多人在学switch语句是都不理解, 既然有了if、else语句我们为什么还要学习switch语言。

这里解释一下switch语句, 在一般可能的情况比较少, 如2, 3 中情况下我们用if语句比较方便, 但是如果超过3个或者更多的情况时用if语句比较起来比较麻烦, 而且写出来的程序不利于观赏, 比较烦琐。

在对switch书写时每个case语句后面不要忘记加break, 否则会产生多分支重叠(除非代码中需要多分支重叠) 在switch语句最后一定要使用default 分支, 即使不需要default处理, 也可避免让人误以为你忘了default处理。

这里需要了解的一点是case语句的排序, 尽量是有顺序的, 以便方便人们阅读和修改。

9、break和continue的区别

break用来终止本层循环。

continue表示终止本次循环。

他们的区别就好像你今天早上要跑10圈, 在跑到第n圈的时候 ($n \leq 10$)遇到break, 你就需要终止今天的晨跑, 而continue就是终止第n圈, 从起点第n+1圈开始继续, 直到10圈完, 或者遇到break。

10、do、while、for

C语言里面三种循环语句: do-while循环、while循环、for循环。

do-while语句: 通常用于代码执行一次在判断是否执行循环的情况。

while语句: 用来执行不知道循环次数的循环语句。

for语句: 用来执行事先直到循环次数的循环语句。

11、goto

goto称为无指向跳转的语句, 这就表明他可以向前跳转也可以像后跳转, 他也可以和if语句组成一个简单的循环。但是在大多数书上不建议用goto语句, 因为它太自由了。goto语句的使用代码如下:

```
1 goto loop;
2 ...
3 ...
   lp:loop;
```

12、void

12.1、void的字面意思是“空类型”, void则为“空类型指针”, void可以指向任何类型的数据。void几乎只有“注释”和“限制程序”的作用。

void的作用在于:

- (1) 对函数返回的限定;
- (2) 对函数参数的限定;

我们知道如果两个指针到的类型相同, 我们可以直接在他们之间相互赋值, 但是如果类型不同我们就需要对赋值的指针强制类型转换成被赋值的指针类型, 而void则不同, 任何类型的指针都可以直接赋值给它, 无需进行强制转换。但是不能理解为void可以无需转换的赋值给其它类型的函数。

12.2、void修饰函数返回值和参数

C语言规定如果函数没有返回值, 那么应声明为void类型。如果函数无参数, 那么应声明其参数为void。

12.3、void指针和变量

按照ANSI标准, 不能对void指针进行算法操作。

持: 算法操作的指针必须确定知道其指向数据类型

点赞 5

评论 1

分享

收藏 30

手机看

打赏

...

关注

如果函数的参数可以是任意类型指针，那么应声明其参数为void*。

void不能代表一个真实的变量。因为定义变量时必须分配空间，定义void类型变量，编译器到底分配多大的内存？

13、return

return用来种植一个函数并返回其后面跟着的值。

return语句可返回指向“栈内存”的“指针”，因为该内存存在函数体结束时被自动销毁。

14、const

const是constant的缩写，是恒定不变的意思。const修饰的是只读变量，其值在编译时不能被使用。

14.1、const修饰变量、数组、指针的一般写法

```
1 //修饰变量 (以整形为例, 其他同下)
2 int const i=2;
3 const int i=2;
4 //修饰数组
5 int const a[5]={1,2,3,4,5};
6 const int a[5]={1,2,3,4,5};
7 //修饰指针
8 const int* p;//p可变, p指向的对象不可变
9 int const* p;//p可变, p指向的对象不可变
10 int* const p;//p不可变, p指向的对象可变
    const int* const p;//p不可变, p指向的对象不可变
```

15、volatile

volatile是易变得、不稳定的意思。volatile关键字和const一样是一种类型修饰符，用它修饰的变量表示可以被某些编译器未知的因素更改，比如操作系统、硬件或者其他线程等。volatile关键字是一种类型修饰符，用它表示的类型变量表示可以被某些编译器未知因素更改。遇到这个关键字声明的变量，编译器对访问该变量的代码就不再行优化，从而可以提供对特殊地址的稳定访问。

16、extern

extern是计算机语言中的一个关键字，可至于变量或者函数前，亦表示变量或者函数的定义在别的文件中。提示编译器遇到

此变量或函数时，在其它模块中寻找其定义，另外，extern也可用来进行链接指定。

17、struct

struct是计算机语言学习过程中比较重要的关键字，它的作用是将一些相关联的数据打包成一个整体，方便使用。

就好比要表示一个学生，我们要从多个方面表示，如学号、姓名、班级等方面表示它，此时我们就可以用struct关键字，定义一个结构体类型表示这个学生。

结构体的一般定义形式为：

```
1 struct 结构名
2 {
3 //成员变量
4 };
```

17.1、结构体的大小

我们用下面的结构体来计算一下结构体的内存大小：

```
1 struct MyStruct
2 {
3 double dda1;
4 char dda;
5 int type;
6 };
```

为上面的结构分配空间的时候，VC根据成员变量出现的顺序和对齐方式，先为第一个成员dda1分配空间，其起始地址跟结构的起始地址相同（刚好偏移量0刚好为sizeof(double)的倍数），该成员变量占用sizeof(double)=8个字节；接下来为第二个成员dda分配空间，这时下一个可以分配的地址对于结构的起始地址的偏移量为8，是sizeof(char)的倍数，所以把dda存放在偏移量为8的地方满足对齐方式，该成员变量占用sizeof(char)=1个字节；接下来为第三个成员type分配空间，这时下一个可以分配的地址对于结构的起始地址的偏移量为9，不是sizeof(int)=4的倍数，为了

自动填充3个字节（这三个字节没有放什么东西）

👍 点赞 5

💬 评论 1

🔗 分享

★ 收藏 30

📱 手机看

💰 打赏

...

🔔 关注

的起始地址的偏移量为12，刚好是sizeof(int)=4的倍数，所以把type存放在偏移量为12的地方，该成员变量占用sizeof(int)=4个字节；这时整个结构的成员变量已经都分配了空间，总的占用的空间大小为：8+1+3+4=16，刚好为结构的字节边界数（即结构中占用最大空间的类型所占用的字节数sizeof(double)=8）的倍数，所以没有空缺的字节需要填充。所以整个结构的大小为：sizeof(MyStruct)=8+1+3+4=16，其中有3个字节是VC自动填充的，没有放任何有意义的东西。

这里要注意，在一个结构体中，相同的数据元素，不同的排列方式可能导致它所占的存储空间大小不同。

就比如下面的结构体，和上面的数据元素相同，但是排列顺序不同。

```
1 struct MyStruct
2 {
3     char dda;
4     double dda1;
5     int type;
6 };
```

这个结构，占用24字节的空间。

18、union

共用体是一种特殊形式的变量，使用关键字union来定义，共用体声明和共用体变量定义与结构体十分相似。其形式为：

```
1 union 公共体名{
2     数据类型 成员名;
3     数据类型 成员名;
4     ...
5 }变量名;
```

共用体表示几个变量共用一个内存位置，在不同的时间，保存不同的数据类型和不同长度的变量。在union中，所用共用体成员公用一个空间，并且同一时间只能储存其中一个成员变量的值。

下列表示声明一个共用体foo；

```
1 union foo{//共用类型foo
2     int i;//整数类型i
3     char c;//字符类型c
4     double k;//双精度类型k
5 };
```

再用已声明的共用体可定义共用体变量。

例如用上面说明的共用体定义一个名为bar的共用体变量，可写成：

```
union foo bar;
```

在共用体变量bar中，整型变量i和字符变量c共用同一内存位置。

当一个共用体被声明时，编译程序自动地产生一个变量，其长度为联合中类型字节数最多的变量的类型长度的整数倍。以上例而言，最大长度是double数据类型，所以foo的内存空间就是double型的长度。

```
1 union foo{//共用类型foo
2     char s[10]; /*“字符”类型的数组“s”下面有“10”个元素*/
3     int i;      /*“整数”类型*/
4 };
```

在这个union中，foo的内存空间的长度为12，是int型的3倍，而并不是数组的长度10。若把int改为double，则foo的内存空间为16，是double型的两倍。

它的内存大小与struct的类似，可以参考一下。

共用体的存储模式为：大端模式和小端模式。

大端模式：字数据的高字节存储在低地址中，而字数据的低字节则存放在高地址中。

小端模式：字数据的高字节存储在高地址中，而字数据的低字节则存放在低地址中。

共用体和结构体有下列区别：

1. 共用体和结构体都是由多个不同的数据类型成员组成，但在任何同一时刻，共用体只存放了一个被选中的成员，而结构体的所有成员

点赞 5

评论 1

分享

收藏 30

手机看

打赏

...

关注

2. 对于共用体的不同成员赋值, 将会对其它成员重写, 原来成员的值就不存在了, 而对于结构体的不同成员赋值是互不影响的。

19、enum

enum为枚举类型的关键字

1. 枚举类型定义的一般形式为:

enum 枚举名{ 枚举值表 };

在枚举值表中应罗列出所有可用值。这些值也称为枚举元素。

例如:

该枚举名为weekday, 枚举值共有7个, 即一周中的七天。凡被说明为weekday类型变量的取值只能是七天中的某一天。

2. 枚举变量的说明

如同结构体 (struct) 和共用体 (union) 一样, 枚举变量也可用不同的方式说明, 即先定义后说明, 同时定义说明或直接说明。

设有变量a,b,c被说明为上述的weekday, 可采用下述任一种方式:

```
1 enum weekday{sun,mon,tue,wed,thu,fri,sat};
2 enum weekday a,b,c;
3 //或者为:
4 enum weekday{sun,mon,tue,wed,thu,fri,sat}a,b,c;
5 //或者为:
   enum{sun,mon,tue,wed,thu,fri,sat}a,b,c;
```

枚举类型在使用中有以下规定:

1. 枚举值是常量, 不是变量。不能在程序中用赋值语句再对它赋值。
2. 枚举元素本身由系统定义了一个表示序号的数值, 从0开始顺序定义为0, 1, 2...。如在weekday中, sun值为0, mon值为1, sat值为6。

枚举与#define区别:

1. #define宏常量是在预编译阶段进行简单替换。枚举常量则是在编译的时候确定其值。
2. 一般在编译器里, 可以调试枚举常量, 但不是能调试宏常量。
3. 枚举可以一次定义大量相关常量, 而#define宏一次只能定义一个。

20、typedef

typedef 的真正意思是给一个已经存在的数据类型取一个别名, 而非定义一个新的数据类型。

编程中使用typedef目的一般有两个, 一个是给变量一个易记且意义明确的新名字, 另一个是简化一些比较复杂的类型声明。

typedef & 结构的问题:

当用下面的代码定义一个结构时, 编译器报了一个错误, 代码如下:

```
1 typedef struct tagNode
2 {
3     char* pItem;
4     pNode* pNext;
   }pNode;
```

下面分析一下错误:

C语言当然允许在结构中包含指向它自己的指针, 我们可以在建立链表等数据结构的实现上看到无数这样的例子, 上述代码的根本问题在于typedef的应用。

根据我们上面的阐述可以知道: 新结构建立的过程中遇到了pNext域的声明, 类型是pNode, 要知道pNode表示的是类型的新名字, 那么在类型本身还没有建立完成的时候, 这个类型的新名字也不存在, 也就是说这个时候编译器根本不认识pNode。

解决方案:

```
1 //1、
2 typedef struct tagNode
3 {
4     char* pItem;
5     struct tagNode* pNext;
6 }*pNode;
7 //2、
8 typedef struct tagNode* pNode;
9 struct tagNode
10 {
11     char* pItem;
12     pNode pNext;//这边不用pNode*,pNode已经表示了struct tagNode*
13 };

```

typedef和#define的区别

1. 原理不同
- #define是C语言中定义的语法，是预处理指令，在预处理时进行简单而机械的字符串替换，不作正确性检查，只有在编译已被展开的源程序时才会发现可能的错误并报错。
- typedef是关键字，在编译时处理，有类型检查功能。它在自己的作用域内给一个已经存在的类型一个别名，但不能在一个函数定义里面使用typedef。用typedef定义数组、指针、结构等类型会带来很大的方便，不仅使程序书写简单，也使意义明确，增强可读性。
2. 功能不同
- typedef用来定义类型的别名，起到类型易于记忆的功能。
- #define不只是可以为类型取别名，还可以定义常量、变量、编译开关等。
3. 作用域不同
- #define没有作用域的限制，只要是之前预定义过的宏，在以后的程序中都可以使用，而typedef有自己的作用域。

可能用一点瑕疵，会不断改进呢，希望大家可以指点不足，谢谢！

2020 IBM 行业创新季 | 速应时变 智达新态

09-21

点击获取IBM行业解决方案

C语言中32个关键字详解

10-25

C语言中32个关键字详解

C

优质评论可以帮助作者获得更高权重

评论

EndOrBegin:

很详细，赞一个！

9月前

回复

...

👍

求C语言中的32个关键字及其意思?

xiachong27的博客

676

一、C语言的关键字共有32个，根据关键字的作用，可分其为数据类型关键字、控制语句关键字、存储类型关键字...

架构-如何画架构图

AdrianAndroid的专栏

5666

资料 如何画出一张合格的技术架构图？ 如何画架构图？ 产品经理的高阶能力：架构图的设计与画法 设计图都不...

C语言关键字解析_编程鸟-CSDN博客

9-11

在C语言中有32个关键字,如下表所示:释:(1)声明:1)告诉编译器,这个名字已经匹配到一块内存上;2)告诉编译器,这个...

C语言的关键字_详解_张亚楠的博客-CSDN博客

9-11

C语言的修行之旅(一)一、C语言的关键字C语言中的32个关键字及其意思如下:由 ANSI 标准定义的C 语言关键字共...

C语言 结构体、共用体

Genven_Liang的博客

7462

C语言 结构体、共用体 一、简述 对结构体、共用体的认识。 结构体是一种自定义的复合数据类型。 ...

c语言关键字

01-08

C语言常见关键字解析，便于初级学者和中级学者的学习和查阅

C语言关键字详解 - CSDN博客

8-16

在百度知道看到这个回答,不错,贴过来学习。 一、C语言的关键字:

👍 点赞5

💬 评论1

🔗 分享

★ 收藏30

📱 手机看

💰 打赏

...

关注