The C Language Detailed explanation

一、数据类型

1、int

（1）占用位数

int 各编译器根据硬件特性选择，要求：short int和int至少16位；long int至少32位

short不能多于int，int不能对于long

此处规定：short：16位 ；int：16位；long：32位。

（2）实际存储：以补码的形式存储。注：正数的原码、反码、补码都相同，负数的补码是在反码的基础上+1，负数的反码是在原码的基础上，符号位不变，其余位取反

int a = 8;

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

int a = -8;

原码：

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

反码：符号位不变

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

补码：实际存储，符号位不变

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

（3）范围

int：可以表示2^16=65536个数，-2^15 ~ 2^15-1,因为有个0

2、float

（1）占用位数

float：32位，和编译器位数无关

（2）实际存储

采用二进制科学计数法存储：1.xxx \* 2^n

1位，符号位(Sign) ：0代表正，1代表为负

8位，指数位（Exponent）：用于存储科学计数法中的指数数据，并且采用移位存储

移位存储：即指数+127，即 n+127

23位，尾数部分（Mantissa）：尾数部分 即xxx

float a = 8.25； 8.25 -》10进制科学计数法：8 .25 \* 10^0

二进制：1000.01 -》2进制科学计数法：1.00001 \* 2^3

符号位：0

指数位：3+127=130，转换成二进制：1000 0010

尾数部分：00001

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

注：由于尾数部分，不需要存二进制科学计数法中的整数部分，因为整数部分是确定的为1，所以原本尾数部分23位相当于变成24位，即十进制中一位数字需要二进制中的4位表示，所以float小数点后最多可以有24/4=6位

3、double

（1）占用位数

64位

（2）实际存储

采用二进制科学计数法存储：1.xxx \* 2^n

1位，符号位(Sign) ：0代表正，1代表为负

11位，指数位（Exponent）：用于存储科学计数法中的指数数据，并且采用移位存储

移位存储：即指数n+1023

52位，尾数部分（Mantissa）：尾数部分 即xxx

4、char

（1）占用位数

8位

（2）实际存储

使用ASCII码存储



char a = ‘a‘；

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

和机器字长及编译器有关系：

所以，int，long int，short int的宽度都可能随编译器而异。但有几条铁定的原则（ANSI/ISO制订的）：

1 sizeof(short int)<=sizeof(int)

2 sizeof(int)<=sizeof(long int)

3 short int至少应为16位（2字节）

4 long int至少应为32位。

unsigned 是无符号的意思。不加unsigned则都是有符号的

例如：

16位编译器

char ：1个字节

char\*(即指针变量): 2个字节

short int : 2个字节

int： 2个字节

unsigned int : 2个字节

float: 4个字节

double: 8个字节

long: 4个字节

long long: 8个字节

unsigned long: 4个字节

32位编译器

char ：1个字节

char\*（即指针变量）: 4个字节（32位的寻址空间是2^32, 即32个bit，也就是4个字节。同理64位编译器）

short int : 2个字节

int： 4个字节

unsigned int : 4个字节

float: 4个字节

double: 8个字节

long: 4个字节

long long: 8个字节

unsigned long: 4个字节

64位编译器

char ：1个字节

char\*(即指针变量): 8个字节

short int : 2个字节

int： 4个字节

unsigned int : 4个字节

float: 4个字节

double: 8个字节

long: 8个字节

long long: 8个字节

unsigned long: 8个字节

二、入门程序解析

#include <stdio.h> 包含标准库信息

main(){ 定义名为main的函数，无参数值

printf(“hello,world!\n”); main函数调用库函数printf

}

1、C语言程序都是由函数和变量组成的。函数指定要执行的计算操作；变量用来存储计算过程中使用的值；

2、每个C语言程序都必须包含一个main函数，作为程序的执行入口；

3、变量必须先声明后使用；

4、如果算术运算符的所用操作数都是整形，则执行整形运算即只保留整数部分；如果操作数中有浮点型，则在运算之前，整形数会先转换成浮点型；

5、C语言中，函数调用时传递参数传递的是值，即值传递。传递给被调用函数的参数值存放在临时变量中，而不是存放在原来的变量中。同理，被调用的函数也就不能修改调用函数的参数变量的值

void getchange(int a,int b){

a = 3;

b =4;

}

void main(){

int a = 1

int b =2;

getchange(a,b);

}

执行：getchange(a,b) 时，实际传过去的只是1，2这两个值，void getchage(int a, int b)中的a，b属于临时变量。执行完之后main函数中的a，b的值还是1，2不会变。

6、局部变量（自动变量）和全局变量的作用域

（1）函数内部的变量，是局部变量，只能在函数内部使用，其它函数不能访问。当函数执行完毕后，属于函数的局部变量都会消失即删除或者说是内存回收。

（2）定义在所有函数外部的变量叫做外部变量。或者说是全局变量。也可以在函数内部使用关键字extern来显示说明该变量是来自外部即外部变量。可以不写。

在函数中，可以改变外部变量的值。

7、什么是定义？什么是声明？它们之间的区别是什么？

**所谓定义就是（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，这个名字就是就是我们经常所说的变量名或对象名。**

**声明有2重含义：**

**（1） 告诉编译器，这个名字已经匹配到一块内存上，下面的代码用到变量或者对象是在别的地方定义的。声明可以出现多次。**

**（2） 告诉编译器，这个名字已经被预定了，别的地方再也不能用它来作为变量名或对象名。**

**定义和声明的最重要区别就是：**

**定义创建对象并为这个对象分配了内存，声明没有分配内存。**

如：  extern int a；声明一个变量

        extern  Int a=56;定义一个变量，因为初始化了。类似于这种有extern也有初始化的语句，只能出现在全局作用域，如果出现在函数内部，这是错误的。

        int a=56； int a；均为定义一个变量

变量的声明有两种情况：

1、一种是需要建立存储空间的。例如：int a 在声明的时候就已经建立了存储空间。

2、另一种是不需要建立存储空间的。 例如：extern int a 其中变量a是在别的文件中定义的。

总之就是：把建立空间的声明称为“定义”，把不需要建立存储空间的称为“声明”。

三、常量

1、常量的表示

int型常量：1234

如：int a = 4; a是一个整形变量，4是变量的值

a = a +12; 12是一个整形常量

long型常量：23998478190L或213214885l ，结尾加：L或l

无符号的常量：结尾加U或u，如：1236287578UL，指unsigned long类型

float常量：结尾：F或f，如：13.4F或者1e-2即1\*10^-2。

double常量：结尾没后缀的小数就是double型。

八进制：前缀：0

十六进制：前缀：0X

字符常量：一个字符常量是一个整数，用单引号括起来，如：'a'。

字符串常量：“adhah”，双引号括起来的0个或多个字符组成。字符串常量就是字符数组，并且会在结尾处加一个'\0'（即空字符，值为0），所以字符串常量的长度比看到的多1.

注：”a”和'a'是不一样的，‘a’是一个整数，是字符常量；”a”是一个包含一个字符和一个结束符‘\0’的字符数组。

枚举常量：枚举是一个常量整型值的列表。如：enum char {a,b,c}。没有显示说明的情况下，enum类型中的第一个枚举名的值为0，第二个为2，第三个为3，以此类推。如果只指定了部分枚举名的值，则未指定的值根据最后一个指定值向后递增。

2、常量表达式

仅仅只包含常量的表达式叫做常量表达式，这种表达式在编译的时候求值，而不是在运行的时候求值。

3、类型转换

当一个运算符的几个操作数类型不同时，需要通过一些规则把它们转换成相同的类型

（1）自动转化

（2）强制转化

四、变量

变量名：编译后，名字都变为了地址，没有名字信息

全局变量（外部变量）：出现在代码块{}之外的变量就是全局变量。

局部变量（自动变量）：一般情况下，代码块{}内部定义的变量就是自动变量，也可 使用auto显示定义。

静态变量：是指内存位置在程序执行期间一直不改变的变量，用关键字static修饰。代码块内部的静态变量只能被这个代码块内部访问，代码块外部的静态变量只能被定义这个变量的文件访问。

注意：C语言中函数默认都是全局的，可以使用static关键字将函数声明为静态函数（只能被定义这个函数的文件访问的函数）。

[静态变量](https://www.baidu.com/s?wd=静态变量&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWrjf1PHuhnv79uWmYujwB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHTkPj61PjD1" \t "_blank)，就是在定义的时候，有static 修饰的变量，形式为

static TYPE var\_name = init\_value;

而动态变量，形式为

TYPE var\_name = init\_value;

即没有static 修饰。其中的=init\_value均可省略。

区分定义在函数外的全局变量，和函数内的[局部变量](https://www.baidu.com/s?wd=局部变量&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWrjf1PHuhnv79uWmYujwB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHTkPj61PjD1" \t "_blank)，作用域，生命周期，及无显式[初始化](https://www.baidu.com/s?wd=初始化&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWrjf1PHuhnv79uWmYujwB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHTkPj61PjD1" \t "_blank)时的初始值，均有区别。

1 动态全局变量：

作用域为整个项目，即最终编译成可执行文件的所有文件中均可以使用动态全局变量。

生命周期为从程序运行到程序退出，即贯穿整个运行时间。

无显式[初始化](https://www.baidu.com/s?wd=初始化&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWrjf1PHuhnv79uWmYujwB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHTkPj61PjD1" \t "_blank)时默认[初始化](https://www.baidu.com/s?wd=初始化&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWrjf1PHuhnv79uWmYujwB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHTkPj61PjD1" \t "_blank)值为0。

2 [静态全局变量](https://www.baidu.com/s?wd=静态全局变量&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWrjf1PHuhnv79uWmYujwB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHTkPj61PjD1" \t "_blank)：

作用域为当前文件，从定义/声明位置到文件结尾。

生命周期为从程序运行到程序退出，即贯穿整个运行时间。

无显式初始化时默认初始化值为0。

3 动态[局部变量](https://www.baidu.com/s?wd=局部变量&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWrjf1PHuhnv79uWmYujwB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHTkPj61PjD1" \t "_blank)：

作用域为当前函数，从定义位置，到其所在的{}的结束位置。

生命周期为从[函数调用](https://www.baidu.com/s?wd=函数调用&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWrjf1PHuhnv79uWmYujwB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHTkPj61PjD1" \t "_blank)到函数退出。

无显式初始化时默认初始化值为随机值。

4 静态[局部变量](https://www.baidu.com/s?wd=局部变量&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWrjf1PHuhnv79uWmYujwB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHTkPj61PjD1" \t "_blank)：

作用域为当前函数，从定义位置，到其所在的{}的结束位置。

生命周期为从程序运行到程序退出，即贯穿整个运行时间，当下次[函数调用](https://www.baidu.com/s?wd=函数调用&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWrjf1PHuhnv79uWmYujwB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHTkPj61PjD1" \t "_blank)时，静态局部变量不会被再次初始化，而是沿用上次函数退出时的值。

无显式初始化时默认初始化值为0

三、内存管理机制

1、内存四区

计算机中内存是分区管理的，程序和程序之间的内存是独立的，不能互相访问。每个程序的内存也是分区管理的，一个程序所占用的内存分为很多区域，我们主要了解四个区域

|  |
| --- |
| Code Area |
| Static Area |
| Heap栈 |
| Stack堆 |

Data Area :Static Area\Heap\Stack

Dynamic Area(动态区):Heap\Stack

2、Code Area：代码区

程序被操作系统加载到内存时，所有可执行代码（指令、常量字符串等）都加载到代码区，程序运行期间不能改变。

3、Static Area：静态区

存放所有的全局变量和静态变量

4、Stack：栈区

栈是一种先进后出的内存结构，所有的自动变量（局部变量）、函数形参都存储在栈中。栈区在程序运行时都可以动态修改。

当一个自动变量超出其作用域时，自动从栈中弹出。

重：（1）每个线程都有自己的专属栈；

（2）栈的最大尺寸固定，超出会引起栈溢出；

（3）变量离开作用域时，自动从栈中弹出（栈内存回收）。

（4）栈的地址不是连续的。

5、Heap：堆区

堆也是一种可以在程序运行过程中随时修改的内存区域，但没有栈先进后出的顺序。堆更像一个大容器。堆内存空间的申请和释放都需要手动通过代码实现。

重：（1）堆的地址是连续的，申请堆内存地址时，系统会找出没有使用的堆内存，然后按照申请的大小，将这些空间连成一片，类似链表；

（2）使用完堆内存之后，一定要手动释放，因为堆内存不会没有自动回收机制；6、stack和heap区别

（1）申请和回收方式不同

堆和栈的第一个区别就是申请方式不同：栈（英文名称是stack）是系统自动分配空间的，例如我们定义一个 char a；系统会自动在栈上为其开辟空间。而堆（英文名称是heap）则是程序员根据需要自己申请的空间，例如malloc（10）；开辟十个字节的空间。

由于栈上的空间是自动分配自动回收的，所以栈上的数据的生存周期只是在函数的运行过程中，运行后就释放掉，不可以再访问。而堆上的数据只要程序员不释放空间，就一直可以访问到，不过缺点是一旦忘记释放会造成内存泄露

（2）申请后系统的响应

栈：只要栈的剩余空间大于所申请空间，系统将为程序提供内存，否则将报异常提示栈溢出。

堆：首先应该知道操作系统有一个记录空闲内存地址的链表，当系统收到程序的申请时，会遍历该链表，寻找第一个空间大于所申请空间的堆结点，然后将该结点从空闲结点链表中删除，并将该结点的空间分配给程序，另外，对于大多数系统，会在这块内存空间中的首地址处记录本次分配的大小，这样，代码中的 delete语句才能正确的释放本内存空间。另外，由于找到的堆结点的大小不一定正好等于申请的大小，系统会自动的将多余的那部分重新放入空闲链表中。

（3）申请效率

栈：由系统自动分配，速度较快。但程序员是无法控制的。

堆：是由new分配的内存，一般速度比较慢，而且容易产生内存碎片,不过用起来最方便。

（4）申请大小限制

栈：在Windows下,栈是向低地址扩展的数据结构，是一块连续的内存的区域。这句话的意思是栈顶的地址和栈的最大容量是系统预先规定好的，在 WINDOWS下，栈的大小是2M（也有的说是1M，总之是一个编译时就确定的常数），如果申请的空间超过栈的剩余空间时，将提示overflow。因此，能从栈获得的空间较小。

堆：堆是向高地址扩展的数据结构，是不连续的内存区域。这是由于系统是用链表来存储的空闲内存地址的，自然是不连续的，而链表的遍历方向是由低地址向高地址。堆的大小受限于计算机系统中有效的虚拟内存。由此可见，堆获得的空间比较灵活，也比较大。

（5）存储内容

由于栈的大小有限，所以用子函数还是有物理意义的，而不仅仅是逻辑意义。

栈： 在函数调用时，第一个进栈的是主函数中函数调用后的下一条指令（函数调用语句的下一条可执行语句）的地址，然后是函数的各个参数，在大多数的C编译器中，参数是由右往左入栈的，然后是函数中的局部变量。注意静态变量是不入栈的。   
当本次函数调用结束后，局部变量先出栈，然后是参数，最后栈顶指针指向最开始存的地址，也就是主函数中的下一条指令，程序由该点继续运行。

堆：一般是在堆的头部用一个字节存放堆的大小。堆中的具体内容有程序员安排。

（6）存取效率

char s1[] = "aaaaaaaaaaaaaaa";

char \*s2 = "bbbbbbbbbbbbbbbbb";

aaaaaaaaaaa是在运行时刻赋值的；放在栈中。   
而bbbbbbbbbbb是在编译时就确定的；放在堆中。   
但是，在以后的存取中，在栈上的数组比指针所指向的字符串(例如堆)快。

参考：

https://[www.cnblogs.com/lln7777/archive/2012/03/14/2396164.html](http://www.cnblogs.com/lln7777/archive/2012/03/14/2396164.html)

https://www.cnblogs.com/yif1991/p/5049638.html

四、关键字

1、extern：表示变量、常量或函数，定义在外部；

如：两个.c文件1.c 、2.c

1.c

int a=3;

2.c

main(){

extern int a;

printf(“%d\n”,a);

}

执行结果：3

因为：由于全局变量本身具有外部链接的特性，extern int a让编译器在编译阶段用一个临时的标示符表示a，等到链接阶段的时候，再去其它链接文件找到a正确的定义并替换掉2.c中a使用的位置。

但是，如果1.c中使用了静态的全局变量

即：static int a = 3;

则2.c中不能通过extern声明去使用1.c中的全局变量，因为静态全局变量作用域是当前文件

但是，如果在2.c中使用宏定义

如：#include “1.c” 则可以使用静态全局变量，因为include是将代码直接加/复制在宏定义的位置，使得a是在当前页面

1. const：指定变量的值不能修改；
2. 控制流
3. if。。。。else

五、C程序编译详解