

云端小飞象cg

致力于图像图形的研究，并为之奋斗一辈子。眼过千遍，不如手过一遍。

- 首页
- 新随笔
- 联系
- 管理

随笔- 72 文章- 8 评论- 88

sizeof () 解析 (原)

(一) 基本概念

sizeof操作符以字节形式给出了其操作数的存储大小。操作数可以是一个表达式或括在括号内的类型名。操作数的存储大小由操作数的类型决定。

(二) 使用方法



1、用于数据类型

sizeof使用形式：sizeof (type) , 如sizeof (int)

2、用于变量

sizeof使用形式：sizeof (var_name) 或sizeof var_name

变量名可以不用括号括住。如sizeof (var_name), sizeof var_name等都是正确形式。带括号的用法更普遍，大多数程序员采用这种形式。

注意：**sizeof操作符不能用于函数类型，不完全类型或位字段。不完全类型指具有未知存储大小的数据类型，如未知存储大小的数组类型、未知内容的结构或联合类型、void类型等。**

如sizeof(max)若此时变量max定义为int max(),sizeof(char_v) 若此时char_v定义为char char_v [MAX]且MAX未知，sizeof(void)都不是正确形式。

我的联系方式： 邮箱：huangmoyue@gmail.com QQ: 350179632

昵称：云端小飞象cg
园龄：8年7个月
粉丝：71
关注：2
[+加关注](#)

< 2012年2月 >						
日	一	二	三	四	五	六
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	1	2	3
4	5	6	7	8	9	10

搜索

谷歌搜索

最新随笔

1. 重载操作符解析 (原)

(三) sizeof应用在结构上的情况

请看下面的结构:

```
struct MyStruct
{
    double doub;
    char ch;
    int i;
};
```

对结构MyStruct采用sizeof会出现什么结果呢? sizeof(MyStruct)为多少呢? 也许你会这样求:

sizeof(MyStruct)=sizeof(double)+sizeof(char)+sizeof(int)=13

以下是测试代码:

```
#include <iostream>

using namespace std;
struct MyStruct
{
    double doub;
    char ch;
    int i;
};

int main()
{
    MyStruct ms;
    cout << sizeof(ms) << endl;

    return 0;
}
```

测试结果:

2. Direct2D学习教程 (一) (原)
3. malloc, free, new, delete解析 (原)
4. 指针解析 (三) (原)
5. #ifdef, #ifndef, #define, #endif 解析 (原)
6. 指针解析 (二) (原)
7. 指针解析 (一) (原)
8. struct 结构体解析 (原)
9. void和void指针解析 (原)
10. 小数点输出精度与iomanip解析 (原)

随笔分类⁽²⁰⁾

Boost使用笔记(1)
C/C++基础知识汇编(13)
Directx2D系列(1)
Directx系列(1)
Google C++ Style 翻译
sgi_stl源码阅读与分析
STL使用系列
VC学习笔记
编程经验杂谈(2)
操作系统
计算机图形学(1)
计算视觉(1)
数据结构与算法实现
物理引擎研究



但是当在VC中测试上面结构的大小时，你会发现sizeof(ms)为16。💡 其实，这是VC对变量存储的一个特殊处理。为了提高CPU的存储速度，VC对一些变量的起始地址做了“对齐”处理。在默认情况下，VC规定各成员变量存放的起始地址相对于结构的起始地址的偏移量必须为该变量的类型所占用的字节数的倍数。

常用类型的对齐方式

类型	对齐方式（变量存放的起始地址相对于结构的起始地址的偏移量）
char	偏移量必须为sizeof(char)即1的倍数
int	偏移量必须为sizeof(int)即4的倍数
double	偏移量必须为sizeof(double)即8的倍数
short	偏移量必须为sizeof(short)即2的倍数
float	偏移量必须为sizeof(float)即4的倍数

各成员变量在存放的时候根据在结构中出现的顺序依次申请空间，同时按照上面的对齐方式调整位置，空缺的字节VC会自动填充。同时VC为了确保结构的大小为结构的字节边界数（即该结构中占用最大空间的类型所占用的字节数）的倍数，所以在为最后一个成员变量申请空间后，还会根据需要自动填充空缺的字节。

```
struct MyStruct
{
    double doub;
    char ch;
    int i;
};
```

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doub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ch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ch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i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，为了满足对齐方式对偏移量的约束问题，VC自动填充3个字节（这三个字节没有放什么东西），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12，刚好是sizeof(int)=4的倍数，所以把i存放在偏移量为12的地方，该成员变量占用sizeof(int)=4个字

3D世界

google c++ style

C++世界

Andrei
Bjarne Stroustrup
comp.lang.c++.moderated

积分与排名

积分 - 35597
排名 - 13021

阅读排行榜

- 1. VS2010 Boost编译安装（原）(18002)
- 2. 编译错误syntax error : missing ';' before 'type'原因(8471)
- 3. c/c++内存机制（一）（原）(7493)
- 4. #ifdef, #ifndef, #define, #endif 解析（原）(7201)
- 5. Direct2D学习教程（一）（原）(6640)

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 $\text{sizeof}(\text{double})=8$) 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: $\text{sizeof}(\text{MyStruct})=8+1+3+4=16$, 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。

下面再举个例子, 交换一下上面的MyStruct的成员变量的位置, 使它变成下面的情况:


```
struct MyStruct
{
    char ch;
    double doub;
    int i;
};
```

在VC环境下, 可以得到 $\text{sizeof}(\text{MyStruct})$ 为24。结合上面提到的分配空间的一些原则, 分析下VC怎么样为上面 的结构分配空间:

```
struct MyStruct
{
    char ch;           // 偏移量为0, 满足对齐方式, ch占用1个字节;
    double doub;       // 下一个可用的地址的偏移量为1, 不是sizeof(double)=8
                        // 的倍数, 需要补足7个字节才能使偏移量变为8 (满足对齐
                        // 方式), 因此VC自动填充7个字节, doub存放在偏移量为8
                        // 的地址上, 它占用8个字节。
    int i;             // 下一个可用的地址的偏移量为16, 是sizeof(int)=4的倍
                        // 数, 满足int的对齐方式, 所以不需要VC自动填充, i存
                        // 放在偏移量为16的地址上, 它占用4个字节。
};                    // 所有成员变量都分配了空间, 空间总的大小为1+7+8+4=20, 不是结构
                        // 的节边界数 (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, 所以需要填充4个字节, 以满足结构的大小为
                        // sizeof(double)=8的倍数。
```

所以该结构总的大小为: $\text{sizeof}(\text{MyStruct})$ 为 $1+7+8+4+4=24$ 。其中总的有 $7+4=11$ 个字节是VC自动填充的, 没有放任何有意义的东西。

VC对结构的存储的特殊处理确实提高CPU存储变量的速度, 但是有时候也带来了一些麻烦, 我们也屏蔽掉变量默认的对齐方式, 自己可以设定变量的对齐方式。

 VC中提供了`#pragma pack(n)`来设定变量以n字节对齐方式。n字节对齐就是说变量存放的起始地址的偏移量有两种情况: 第一, 如果n大于等于该变量所占用的字节数, 那么偏移量必须满足默认的对齐方式, 第二, 如果n小于该变量

的类型所占用的字节数，那么偏移量为n的倍数，不用满足默认的对齐方式。结构的总大小也有个约束条件，分下面两种情况：如果n大于所有成员变量类型所占用的字节数，那么结构的总大小必须为占用空间最大的变量占用的空间数的倍数；否则必须为n的倍数。

下面举例说明其用法：

```
#pragma pack(push)    //保存对齐状态
#pragma pack(4)        //设定为4字节对齐
struct MyStruct
{
    char ch;
    double doub;
    int i;
};
#pragma pack(pop)     //恢复对齐状态
```

测试结果：



(四) sizeof用法总结

1. 参数为数据类型或者为一般变量。

例如sizeof(int),sizeof(long)等等。这种情况要注意的是不同系统或者不同编译器得到的结果可能是不同的。例如int类型在16位系统中占2个字节，在32位系统中占4个字节。

2. 参数为数组或指针。

下面举例说明。

```
int a[50];           //sizeof(a)=4*50=200; 求数组所占的空间大小
int *a = new int[50]; // sizeof(a)=4; a为一个指针，sizeof(a)是求指针的大小,在32位系统中，当然是占4个字节。
```

3. 参数为其他。

```
int func(char s[5])
{
    return 1;           //函数的参数在传递的时候系统处理为一个指针，所以sizeof(s)实际上为求指针的大小。
}
```

sizeof(func("1234")); //因为func的返回类型为int，所以相当于求sizeof(int)，其值为4.

分类: [C/C++基础知识汇编](#)

好文要顶

关注我

收藏该文







云端小飞象cg

关注 - 2

粉丝 - 71

+加关注

1

0

« 上一篇: [c/c++内存机制（一）（原）](#)
» 下一篇: [国外大牛的编程经验](#)

posted @ 2012-02-02 12:43 云端小飞象cg 阅读(812) 评论(7) 编辑 收藏

发表评论

- #1楼 2012-02-05 12:48 | [beings](#)

字节对齐部分讲的挺清楚的,我转载了字节对齐部分<http://www.cppexample.com/standard/alignment.html>,文章底部注明了本文连接

回复 引用
- #2楼 2012-02-10 11:59 | [getgoing](#)

还有sizeof(expr)结果是编译时常量

支持(0) 反对(0)
- #3楼[楼主] 2012-02-11 09:09 | [云端小飞象cg](#)

@ getgoing
谢谢提醒，这个很浅显的问题没有包含进去。

支持(0) 反对(0)
- #4楼 2012-02-15 22:36 | [☆A希亿](#)

楼主的文章简明易懂，清晰透彻。好文章。。我转载了。。

支持(0) 反对(0)
- #5楼 2012-02-23 13:12 | [molixiaoge](#)

回复 引用