

# 可可西

首页 联系 订阅 管理

随笔- 199 文章- 0 评论- 104

## c++对象内存模型【内存布局】

### #类中的元素

0. 成员变量 1. 成员函数 2. 静态成员变量 3. 静态成员函数 4. 虚函数 5. 纯虚函数

### #影响对象大小的因素

0. 成员变量 1. 虚函数表指针 (\_vftptr) 2. 虚基类表指针 (\_vbtptr) 3. 内存对齐

\_vftptr、\_vbtptr的初始化由对象的构造函数, 赋值运算符自动完成; 对象生命周期结束后, 由对象的析构函数来销毁。

对象所关联的类型 ( type\_info ), 通常放在virtual table的第一个slot中。

**虚继承**: 在继承定义中包含了virtual关键字的继承关系;

**虚基类**: 在虚继承体系中的通过virtual继承而来的基类, 需要注意的是:

class CDerive : public virtual CBase {}; 其中CBase称之为CDerive的虚基类, 而不是说CBase就是个虚基类, 因为CBase还可以为不是虚继承体系中的基类。

虚函数被派生后, 仍然为虚函数, 即使在派生类中省去virtual关键字。

注: 【下文中\_vbptr即\_vbtptr】

### #对象内存布局分类讨论

## 公告

昵称: 可可西

园龄: 5年6个月

粉丝: 93

关注: 0

+加关注

2013年1月						
<	日	一	二	三	四	五
	30	31	1	2	3	4
	6	7	8	9	10	11
	13	14	15	16	17	18
	20	21	22	23	24	25
	27	28	29	30	31	1
	3	4	5	6	7	8

## 搜索

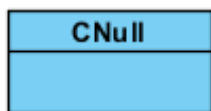
vc6变量查看器中 ( Locals, Watch1等 ) , 也可以看到部分对象布局的情况 ( **不完整, 且虚继承是错误的** ) 。

vs2005及以后版本的编译器提供了 **/d1reportSingleClassLayout[类名]** 编译选项来查看对象完整的内存布局:

```
cl classLayout.cpp /d1reportSingleClassLayoutCChildren
```

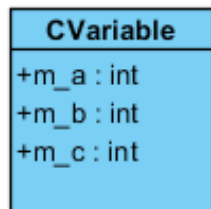
## 0. 单一类

### (1). 空类



`sizeof(CNull)=1` ( 用于标识该对象 )

### (2). 只有成员变量的类



`int nVarSize = sizeof(CVariable) = 12`

Name	Value
nVarSize	12
pVarA	0x00371048
m_a	-842150451
m_b	-842150451
m_c	-842150451

## 常用链接

[我的随笔](#)

[我的评论](#)

[我的参与](#)

[最新评论](#)

[我的标签](#)

[更多链接](#)

## 我的标签

[C++ \(68\)](#) [System \(29\)](#)

[Show \(24\)](#) [OpenGL \(18\)](#)

[mfc \(14\)](#) [algorithm \(14\)](#)

[Tools \(13\)](#) [OpenSource \(12\)](#)

[IDE \(11\)](#) [Debug \(7\)](#) [更多](#)

## 随笔分类<sup>(196)</sup>

[编程实践\(76\)](#)

[互联网技术\(5\)](#)

[计算机理论\(33\)](#)

[算法思想\(13\)](#)

[系统-软件\(42\)](#)

[作品展示\(27\)](#)

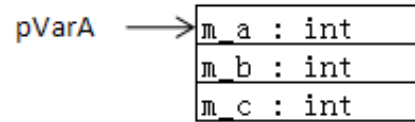
## 随笔档案<sup>(199)</sup>

[2015年1月 \(1\)](#)

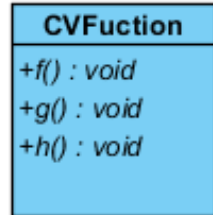
[2014年12月 \(1\)](#)

[2014年11月 \(1\)](#)

内存布局:



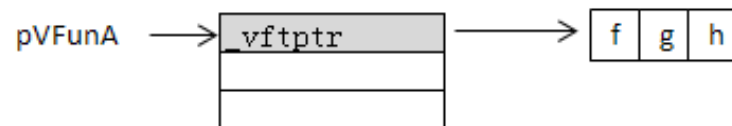
(3). 只有虚函数的类



int nVFunSize = sizeof(CVFuction) = 4 ( 虚表指针 )

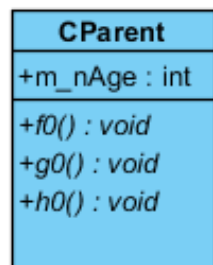
Name	Value
nVFunSize	4
pVFunA	0x00371090
_vfptr	0x0042501c const CVFuction::`vftable'
[0]	0x0040100a CVFuction::f(void)
[1]	0x00401014 CVFuction::g(void)
[2]	0x00401005 CVFuction::h(void)

内存布局:



(4). 有成员变量、虚函数的类

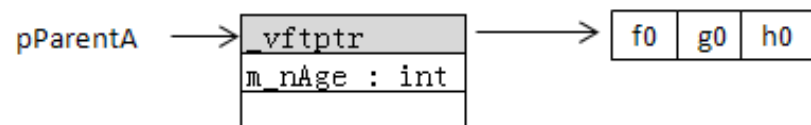
2014年9月 (2)  
 2014年8月 (3)  
 2014年7月 (4)  
 2014年6月 (1)  
 2014年4月 (2)  
 2014年3月 (1)  
 2014年2月 (2)  
 2014年1月 (2)  
 2013年12月 (1)  
 2013年11月 (3)  
 2013年10月 (1)  
 2013年9月 (2)  
 2013年8月 (2)  
 2013年7月 (4)  
 2013年6月 (1)  
 2013年5月 (4)  
 2013年4月 (4)  
 2013年3月 (1)  
 2013年2月 (1)  
 2013年1月 (5)  
 2012年12月 (1)  
 2012年11月 (4)  
 2012年10月 (4)  
 2012年9月 (1)  
 2012年8月 (2)  
 2012年6月 (2)  
 2012年5月 (3)  
 2012年4月 (5)  
 2012年3月 (4)  
 2012年2月 (6)  
 2012年1月 (6)  
 2011年12月 (7)



int nParentSize = sizeof(CParent) = 8

Name	Value
nParentSize	8
pParentA	0x003724c8
__vfp_ptr	0x00426008 const CParent::`vftable'
[0]	0x00401028 CParent::f0(void)
[1]	0x00401023 CParent::g0(void)
[2]	0x0040101e CParent::h0(void)
m_nAge	-842150451

内存布局:



## 1. 单一继承（含成员变量、虚函数、虚函数覆盖）

2011年11月 (7)  
 2011年10月 (2)  
 2011年9月 (8)  
 2011年8月 (11)  
 2011年7月 (8)  
 2011年6月 (12)  
 2011年5月 (7)  
 2011年4月 (1)  
 2011年3月 (7)  
 2011年2月 (1)  
 2011年1月 (2)  
 2010年12月 (5)  
 2010年11月 (7)  
 2010年10月 (2)  
 2010年9月 (1)  
 2010年8月 (12)  
 2010年7月 (8)  
 2010年5月 (1)  
 2009年8月 (3)

## 最新评论

### 1. Re:windbg调试命令

臧害

--王林森

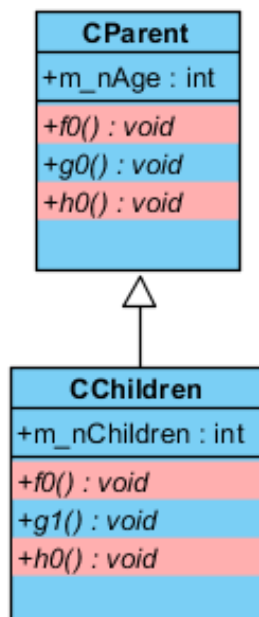
### 2. Re:Eclipse安装SVN插件

@wkl17应该是版本号，表示当前的bitmapfilters.cpp的版本号为236注：每个工程都有一个svn版本号（是一个全局的东西），只要工程中的任意文件修改，版本号就会+1...

--可可西

### 3. Re:Eclipse安装SVN插件

bitmapfilters.cpp 236 后面的



`int nChildSize = sizeof(CChildren) = 12`

vc中显示的结果（注：还有1个虚函数CChildren::g1没有被显示出来）：

Name	Value
nChildSize	12
[-] pChildrenA	0x00372510
[-] CParent	{...}
[-] _vfptr	0x0042503c const CChildren::`vftable'
[-] [0]	0x004010b4 CChildren::f0(void)
[-] [1]	0x0040102d CParent::g0(void)
[-] [2]	0x00401087 CChildren::h0(void)
m_nAge	-842150451
m_nChildren	-842150451

d1reportSingleClass查看：

236是说它已经被修改过236次,有236个版本??

(或者说,是否有相关文档可以解释这些基础知识呢?新手 不太了解,求指点)谢谢.

--wkl17

4. Re:memcpy memmove区...

在看标准库,这个函数有点费解,看了解释豁然开朗.

--罗一刀流

5. Re:memcpy memmove区...

谢谢分享!

--ROJO

## 阅读排行榜

1. Eclipse安装SVN插件(130...
2. memcpy memmove区别...
3. UE正则表达式语法(7360)
4. QT 信号和槽(4939)
5. OpenGL放大缩小实现(49...

## 评论排行榜

1. Google Map街景(红蓝立...
2. Eclipse安装SVN插件(6)
3. memcpy memmove区别...
4. c++对象内存模型【内存...
5. GDI+和GDI区别以及一些...

```

C:\WINDOWS\system32\cmd.exe

D:\program files\Microsoft Visual Studio 9.0\VC\bin>cl classLayout.cpp /dlreport
SingleClassLayoutCChildren
用于 80x86 的 Microsoft (R) 32 位 C/C++ 优化编译器 15.00.30729.01 版
版权所有(C) Microsoft Corporation。保留所有权利。

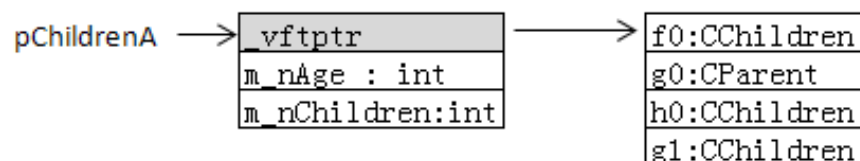
classLayout.cpp
class CChildren size(12):
    +---
    | +--- <base class CParent>
    | | <vfptr>
    | | m_nAge
    | +---
    | m_nChildren
    +---

CChildren::$vftable@:
    | &CChildren_meta
    | 0
    | &CChildren::f0
    | &CParent::g0
    | &CChildren::h0
    | &CChildren::g1

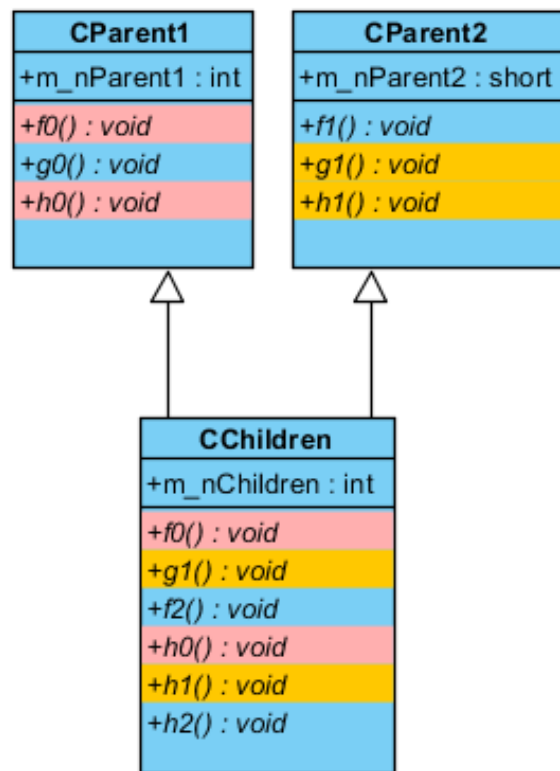
CChildren::f0 this adjustor: 0
CChildren::g1 this adjustor: 0
CChildren::h0 this adjustor: 0

```

内存布局:



## 2. 多继承（含成员变量、虚函数、虚函数覆盖）



`int nChildSize = sizeof(CChildren) = 20`

vc中显示的结果（注：还有2个虚函数CChildren::f2，CChildren::h2没有被显示出来，`this`指针的 `adjustor`[调整值]也没打印出）：

Name	Value
nChildSize	20
[-] pChildrenA	0x00372510
[-] CParent1	{...}
[-] __vfptr	0x0042504c const CChildren::`vftable' {for `CParent1'}
[-] [0]	0x004010a0 CChildren::f0(void)
[-] [1]	0x00401041 CParent1::g0(void)
[-] [2]	0x00401078 CChildren::h0(void)
m_nParent1	-842150451
[-] CParent2	{...}
[-] __vfptr	0x0042503c const CChildren::`vftable' {for `CParent2'}
[-] [0]	0x00401005 CParent2::f1(void)
[-] [1]	0x00401087 CChildren::g1(void)
[-] [2]	0x0040109b CChildren::h1(void)
m_nParent2	-12851
m_nChildren	-842150451

d1reportSingleClass查看：

```

C:\WINDOWS\system32\cmd.exe

D:\program files\Microsoft Visual Studio 9.0\VC\bin>cl classLayout.cpp /d1report
SingleClassLayoutCChildren
用于 80x86 的 Microsoft (R) 32 位 C/C++ 优化编译器 15.00.30729.01 版
版权所有(C) Microsoft Corporation。保留所有权利。

classLayout.cpp
class CChildren size(20):
    +---
    | +--- <base class CParent1>
    | | <vfptr>
    | | m_nParent1
    | +---
    | +--- <base class CParent2>
    | | <vfptr>
    | | m_nParent2
    | | <alignment member> <size=2>
    | +---
    | m_nChildren
    +---

```



```

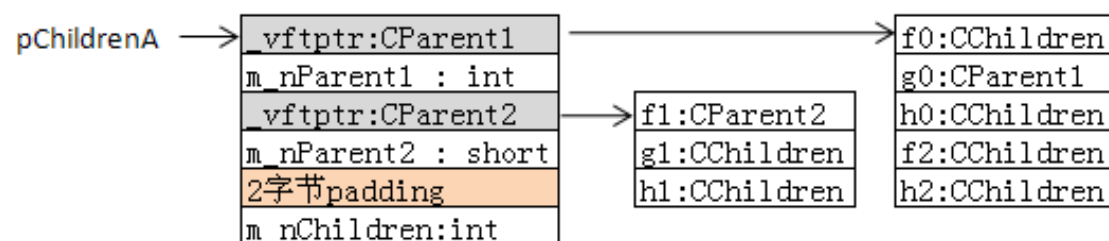
CChildren::$vftable@CParent1@:
    | &CChildren_meta
    | 0
0    | &CChildren::f0
1    | &CParent1::g0
2    | &CChildren::h0
3    | &CChildren::f2
4    | &CChildren::h2

CChildren::$vftable@CParent2@:
    | -8
0    | &CParent2::f1
1    | &CChildren::g1
2    | &CChildren::h1

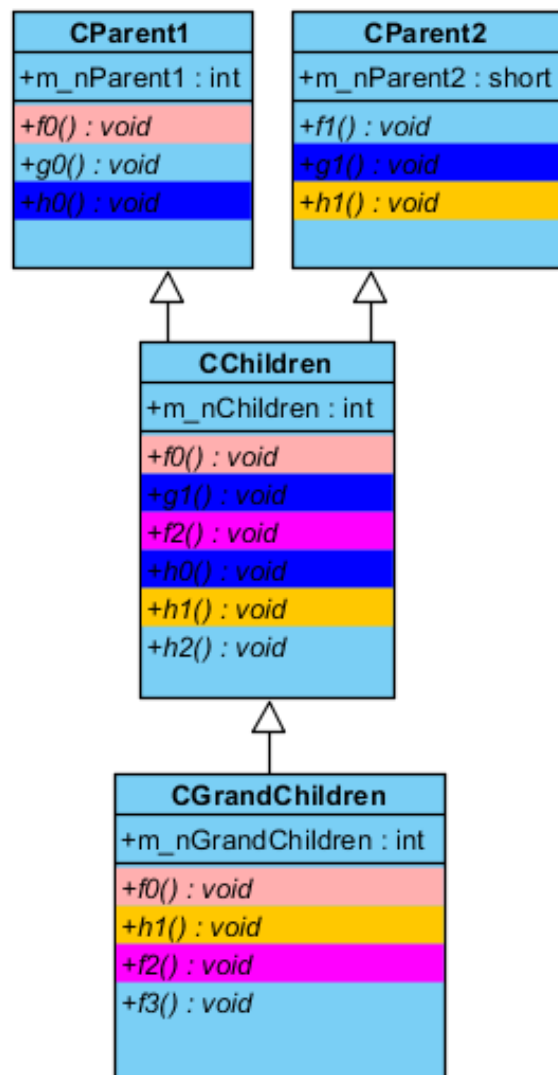
CChildren::f0 this adjustor: 0
CChildren::g1 this adjustor: 8
CChildren::f2 this adjustor: 0
CChildren::h0 this adjustor: 0
CChildren::h1 this adjustor: 8
CChildren::h2 this adjustor: 0

```

内存布局:



### 3. 深度为2的继承（含成员变量、虚函数、虚函数覆盖）



`int nGrandSize = sizeof(CGrandChildren) = 24`

vc中显示的结果（注：还有3个虚函数CGrandChildren::f2，CChildren::h2，CGrandChildren::f3没有显示出来，`this`指针的`adjustor`[调整值]也没打印出）：

d1reportSingleClass查看:

<http://www.cnblogs.com/kekec/archive/2013/01/27/2822872.html>

```

12      | | | m_nParent2
      | | | <alignment member> <size=2>
      | | +---+
16      | | m_nChildren
      | | +---+
20      | m_nGrandChildren
      +---+

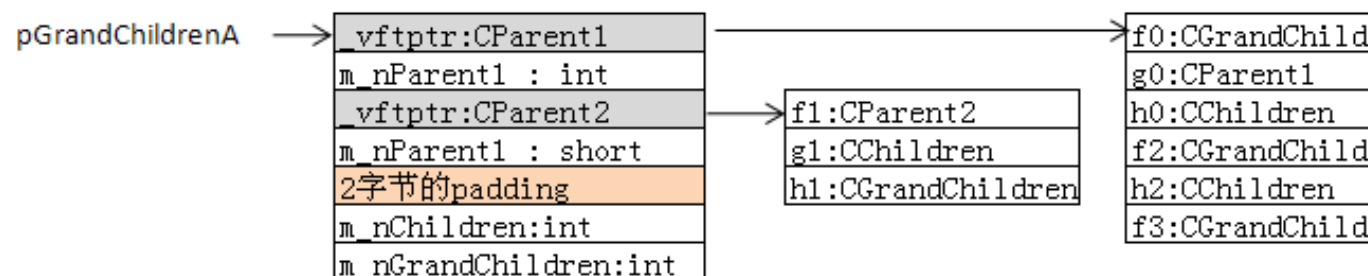
CGrandChildren::$vftable@CParent1@:
      | &CGrandChildren_meta
      | 0
0      | &CGrandChildren::f0
1      | &CParent1::g0
2      | &CChildren::h0
3      | &CGrandChildren::f2
4      | &CChildren::h2
5      | &CGrandChildren::f3

CGrandChildren::$vftable@CParent2@:
      | -8
0      | &CParent2::f1
1      | &CChildren::g1
2      | &CGrandChildren::h1

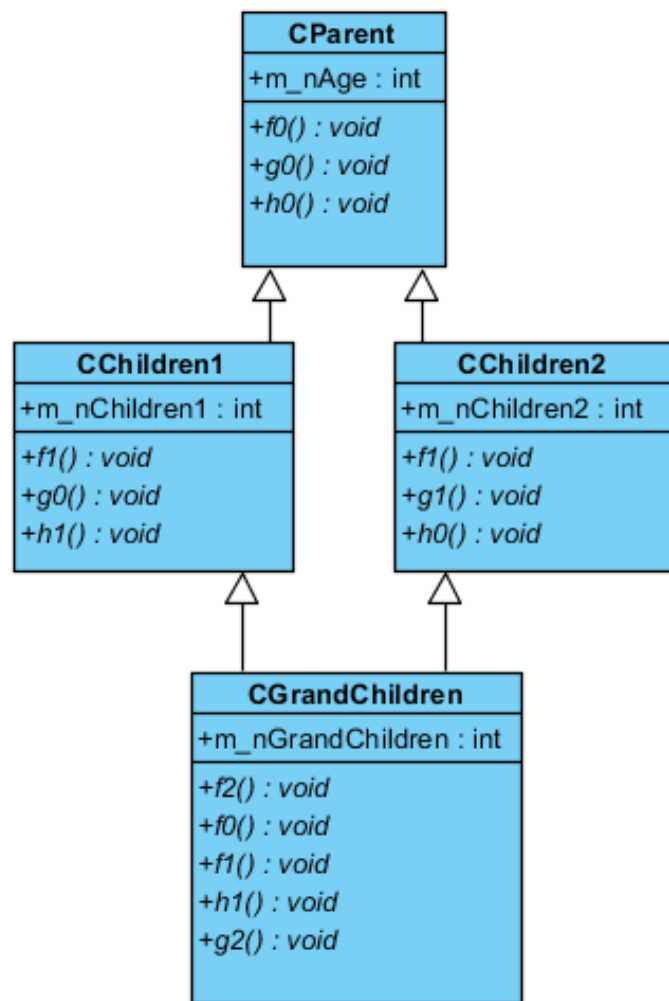
CGrandChildren::f0 this adjustor: 0
CGrandChildren::h1 this adjustor: 8
CGrandChildren::f2 this adjustor: 0
CGrandChildren::f3 this adjustor: 0

```

内存布局:



## 4 重复继承（含成员变量、虚函数、虚函数覆盖）



`int nGrandSize = sizeof(CGrandChildren) = 28`

vc中显示的结果（注：还有大量的虚函数没有显示出来，`this`指针的`adjustor`[调整值]也没打印出）：

`thunk`函数：一种形实转换辅助函数；主要做`this`指针调整，函数调用重定向。

Name	Value
nGrandSize	28
[-] pGrandChildrenA	0x00372510
[-] CChildren1	{...}
[-] CParent	{...}
[-] __vfptr	0x00425078 const CGrandChildren::`vftable'{for `CChildren1'}
[-] [0]	0x00401078 CGrandChildren::f0(void)
[-] [1]	0x00401023 CChildren1::g0(void)
[-] [2]	0x0040100f CParent::h0(void)
[-] m_nAge	-842150451
[-] m_nChildren1	-842150451
[-] CChildren2	{...}
[-] CParent	{...}
[-] __vfptr	0x00425064 const CGrandChildren::`vftable'{for `CChildren2'}
[-] [0]	0x00401019 [thunk]:CGrandChildren::f0`adjustor{12}' (void)
[-] [1]	0x00401037 CParent::g0(void)
[-] [2]	0x004010ff CChildren2::h0(void)
[-] m_nAge	-842150451
[-] m_nChildren2	-842150451
[-] m_nGrandChildren	-842150451

d1reportSingleClass查看:

```

C:\WINDOWS\system32\cmd.exe

classLayout.cpp
class CGrandChildren  size(28):
    +---
    | +--- <base class CChildren1>
    | | +--- <base class CParent>
    | | | {vfptr}
    | | | m_nAge
    | | +---
    | | m_nChildren1
    | +---
    | +--- <base class CChildren2>
    | | +--- <base class CParent>

```

```

12      | | | <vfptr>
16      | | | m_nAge
      | | +---
20      | | m_nChildren2
      | +---
24      | m_nGrandChildren
      +---

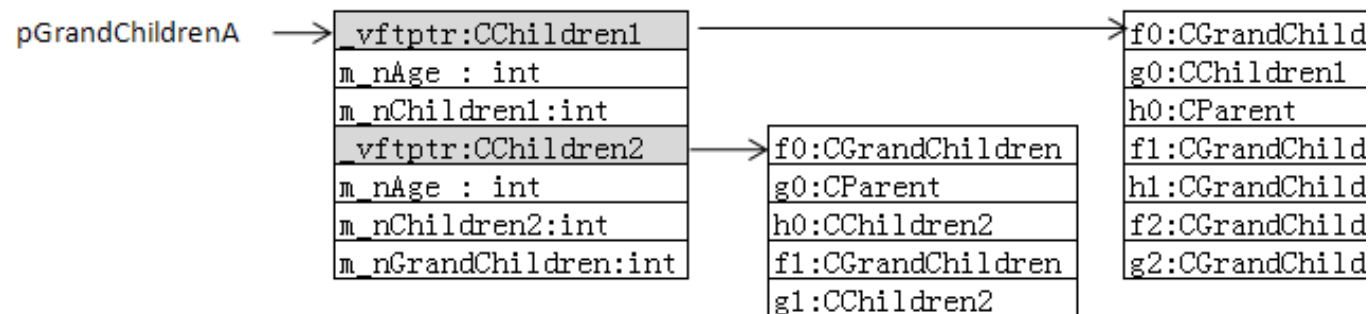
CGrandChildren::$vftable@CChildren1@:
      | &CGrandChildren_meta
      | 0
0      | &CGrandChildren::f0
1      | &CChildren1::g0
2      | &CParent::h0
3      | &CGrandChildren::f1
4      | &CGrandChildren::h1
5      | &CGrandChildren::f2
6      | &CGrandChildren::g2

CGrandChildren::$vftable@CChildren2@:
      | -12
0      | &thunk: this-=12; goto CGrandChildren::f0
1      | &CParent::g0
2      | &CChildren2::h0
3      | &thunk: this-=12; goto CGrandChildren::f1
4      | &CChildren2::g1

CGrandChildren::f2 this adjustor: 0
CGrandChildren::f0 this adjustor: 0
CGrandChildren::f1 this adjustor: 0
CGrandChildren::h1 this adjustor: 0
CGrandChildren::g2 this adjustor: 0

```

内存布局:



由于m\_nAge在内容中存在两个拷贝，因此我们不能直接通过pGrandChildrenA->m\_nAge来访问该变量，

这样会存在二义性，编译器无法知道应该访问CChildren1中的m\_nAge，还是CChildren2中的m\_nAge。

为了标识唯一的m\_nAge，就需要带上其所在范围的类名了。如下：

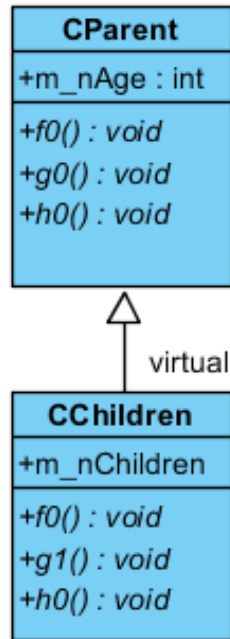
```

1 pGrandChildrenA->CChildren1::m_nAge = 1;
2 pGrandChildrenA->CChildren2::m_nAge = 2;

```

## 5. 单一虚继承（含成员变量、虚函数、虚函数覆盖）





`int nChildSize = sizeof(CChildren) = 20`

d1reportSingleClass查看：

```

C:\WINDOWS\system32\cmd.exe

D:\program files\Microsoft Visual Studio 9.0\VC\bin>cl classLayout.cpp /dlreport
SingleClassLayoutCChildren
用于 80x86 的 Microsoft (R) 32 位 C/C++ 优化编译器 15.00.30729.01 版
版权所有(C) Microsoft Corporation。保留所有权利。

classLayout.cpp
class CChildren size(20):
    +---
    0      | {vfptr}
    4      | {vbptr}
    8      | m_nChildren
    +---
    +--- <virtual base CParent>
    12     | {vfptr}
    16     | m_nAge
    +---

CChildren::$vftable@CChildren@:
    | &CChildren_meta
    | 0
    0      | &CChildren::g1

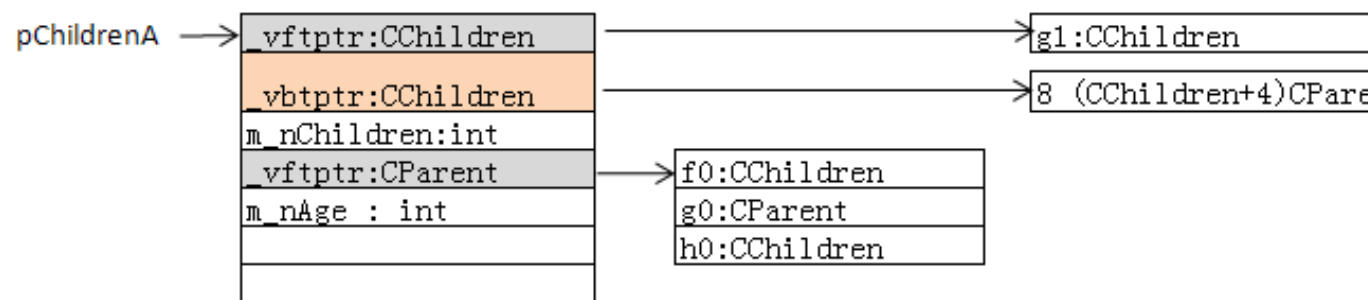
CChildren::$vbtable@:
    0      | -4
    1      | 8 <CChildrend(CChildren+4)CParent>

CChildren::$vftable@CParent@:
    | -12
    0      | &CChildren::f0
    1      | &CParent::g0
    2      | &CChildren::h0

CChildren::f0 this adjustor: 12
CChildren::g1 this adjustor: 0
CChildren::h0 this adjustor: 12

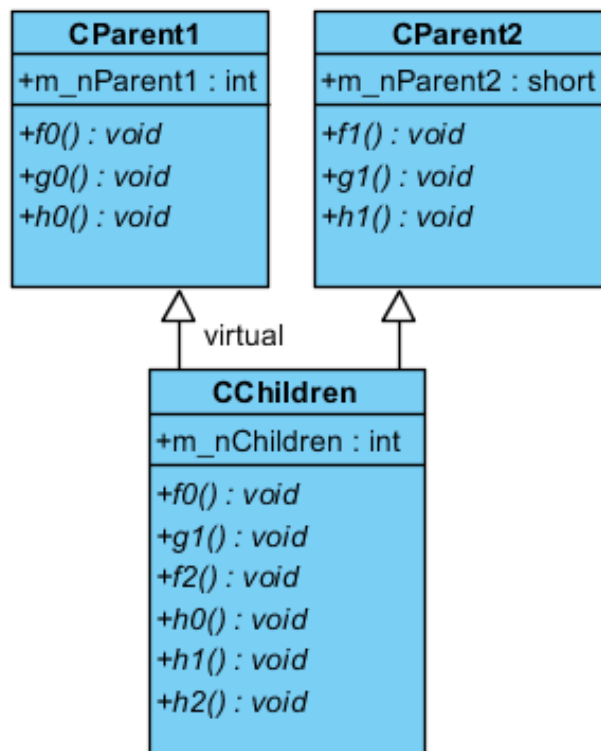
```

内存布局:



## 6. 多虚继承（含成员变量、虚函数、虚函数覆盖）

(1) virtual CParent1, CParent2



`int nChildSize = sizeof(CChildren) = 24`

d1reportSingleClass查看：

```

C:\WINDOWS\system32\cmd.exe

classLayout.cpp
class CChildren size(24):
    +---
    | +--- (base class CParent2)
    | | {vfptr}
    | | m_nParent2
    | | <alignment member> <size=2>
    | +---
    | 8 {vbptr}
    | 12 m_nChildren
    +---
    +--- (virtual base CParent1)
    | {vfptr}
    | 20 m_nParent1
    +---

CChildren::$vftable@CParent2@:
    | &CChildren_meta
    | 0
    | 0 | &CParent2::f1
    | 1 | &CChildren::g1
    | 2 | &CChildren::h1
    | 3 | &CChildren::f2
    | 4 | &CChildren::h2

CChildren::$vbtable@:
    | 0 | -8
    | 1 | 8 <CChildrend<CChildren+8>CParent1>

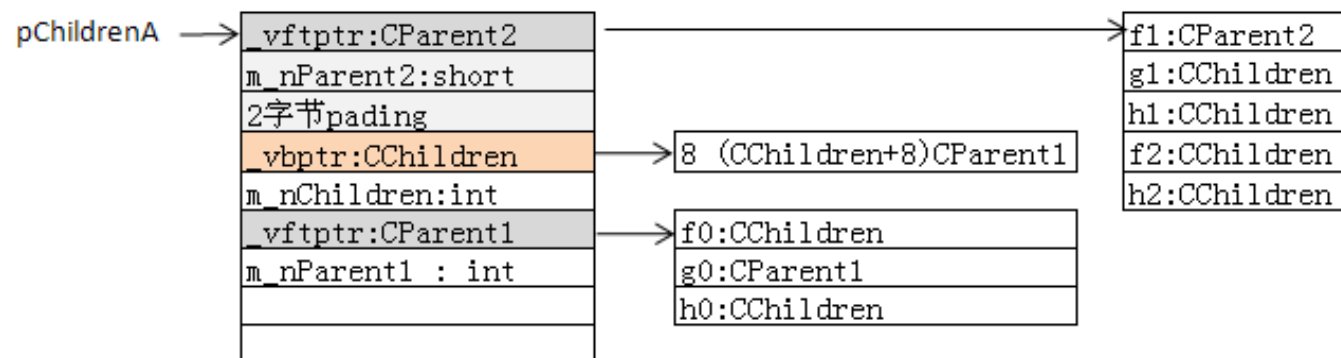
CChildren::$vftable@CParent1@:
    | -16
    | 0 | &CChildren::f0
    | 1 | &CParent1::g0
    | 2 | &CChildren::h0

CChildren::f0 this adjustor: 16
CChildren::g1 this adjustor: 0
CChildren::f2 this adjustor: 0
CChildren::h0 this adjustor: 16
CChildren::h1 this adjustor: 0

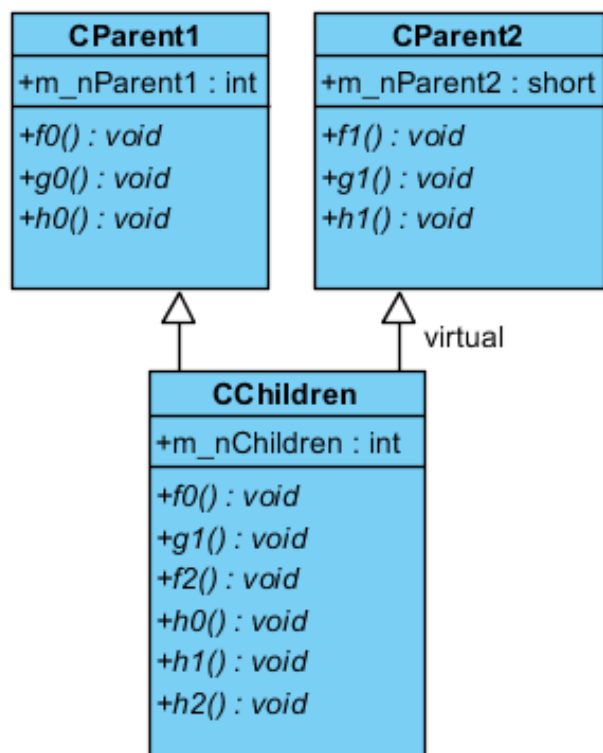
```

```
CChildren::h2 this adjustor: 0
```

内存布局:

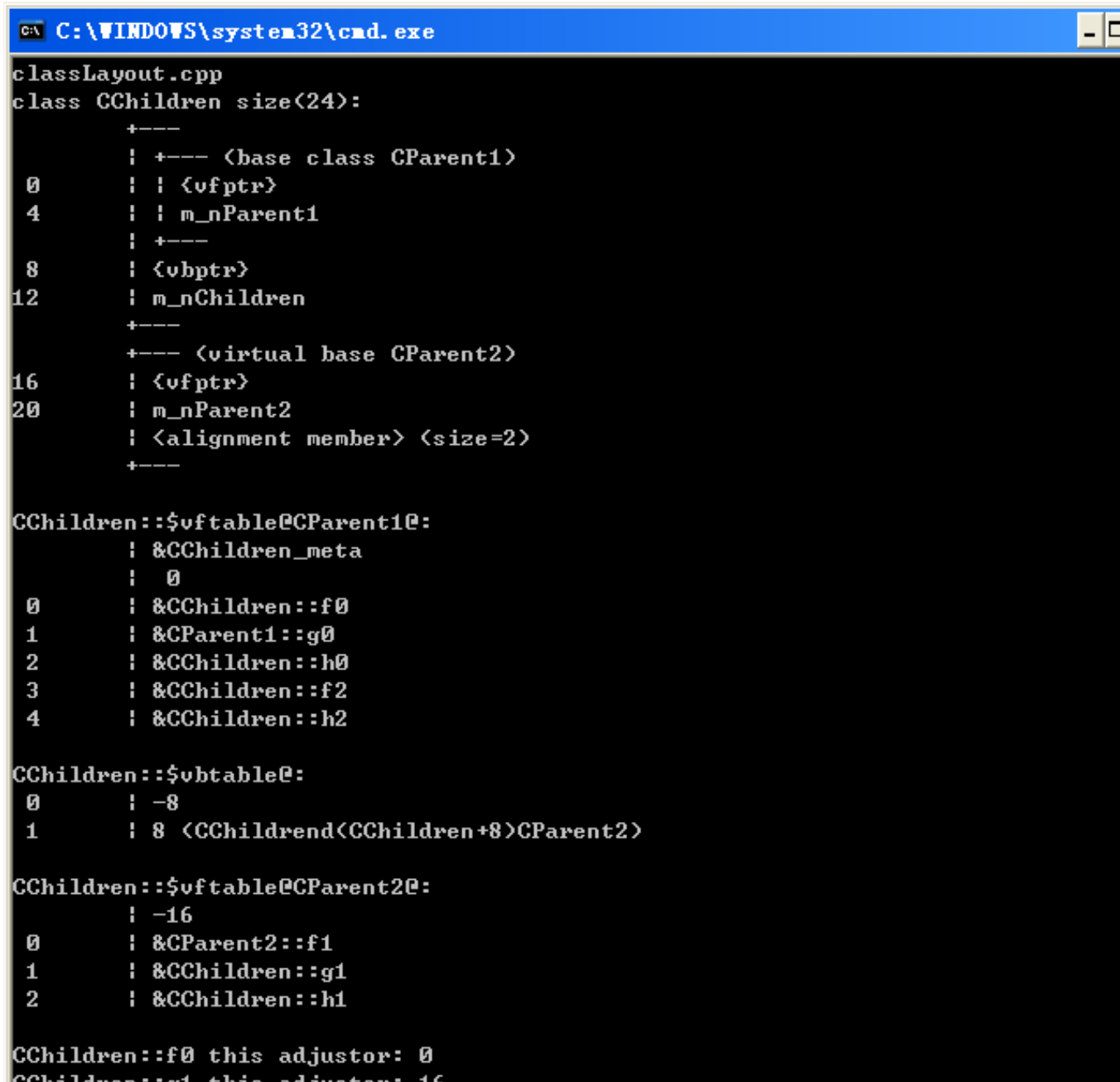


(2) CParent1, virtual CParent2



```
int nChildSize = sizeof(CChildren) = 24
```

d1reportSingleClass查看：



```

C:\WINDOWS\system32\cmd.exe

classLayout.cpp
class CChildren size(24):
    +---
    | +--- <base class CParent1>
    | | {vfptr}
    | | m_nParent1
    | +---
    | {vbptr}
    | m_nChildren
    +---
    +--- <virtual base CParent2>
    | {vfptr}
    | m_nParent2
    | <alignment member> <size=2>
    +---

CChildren::$vftable@CParent1@:
    | &CChildren_meta
    | 0
    | &CChildren::f0
    | &CParent1::g0
    | &CChildren::h0
    | &CChildren::f2
    | &CChildren::h2

CChildren::$vbtable@:
    | -8
    | 8 <CChildren@CChildren+8>CParent2>

CChildren::$vftable@CParent2@:
    | -16
    | &CParent2::f1
    | &CChildren::g1
    | &CChildren::h1

CChildren::f0 this adjustor: 0
CChildren::f1 this adjustor: 16

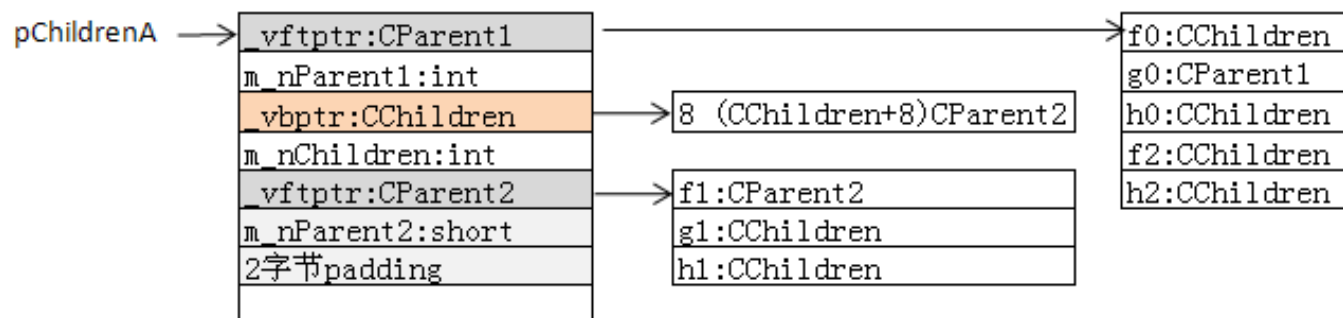
```

```

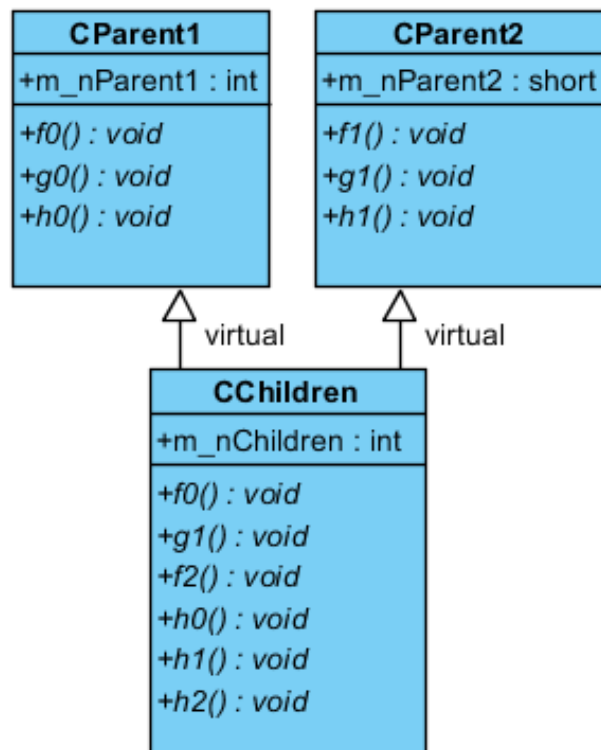
CChildren::g1 this adjustor: 16
CChildren::f2 this adjustor: 0
CChildren::h0 this adjustor: 0
CChildren::h1 this adjustor: 16
CChildren::h2 this adjustor: 0

```

内存布局:



(3) virtual CParent1, virtual CParent2



`int nChildSize = sizeof(CChildren) = 28`

d1reportSingleClass查看：

```

C:\WINDOWS\system32\cmd.exe
classLayout.cpp
class CChildren size(28):
    +---
    0    | {vfptr}
    4    | {vbptr}
    8    | m_nChildren
    +---
    +--- <virtual base CParent1>
    12   | {vfptr}
    16   | m_nParent1
    +---
    +--- <virtual base CParent2>
    20   | {vfptr}
  
```



```

24      : m_nParent2
      : <alignment member> <size=2>
      +---

CChildren::$vftable@CChildren@:
      : &CChildren_meta
      : 0
0      : &CChildren::f2
1      : &CChildren::h2

CChildren::$vtable@:
0      : -4
1      : 8 <CChildren@CChildren+4>CParent1>
2      : 16 <CChildren@CChildren+4>CParent2>

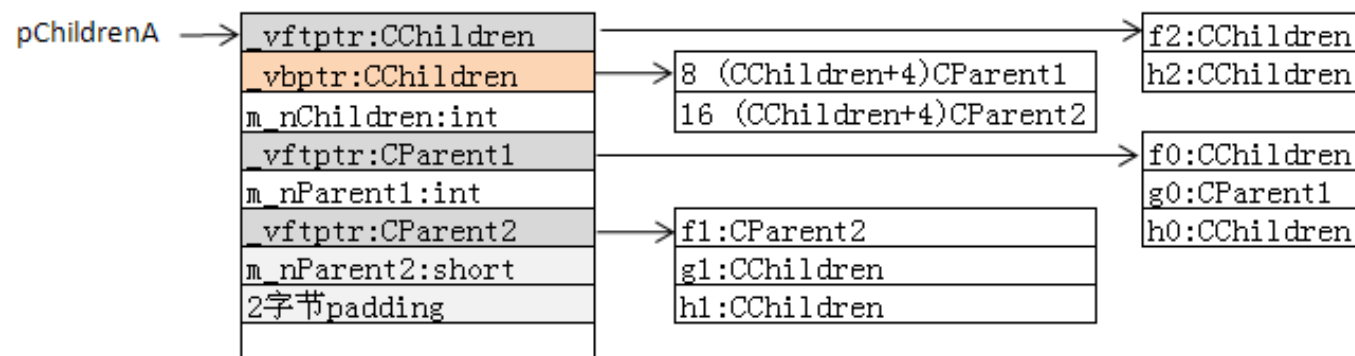
CChildren::$vftable@CParent1@:
      : -12
0      : &CChildren::f0
1      : &CParent1::g0
2      : &CChildren::h0

CChildren::$vftable@CParent2@:
      : -20
0      : &CParent2::f1
1      : &CChildren::g1
2      : &CChildren::h1

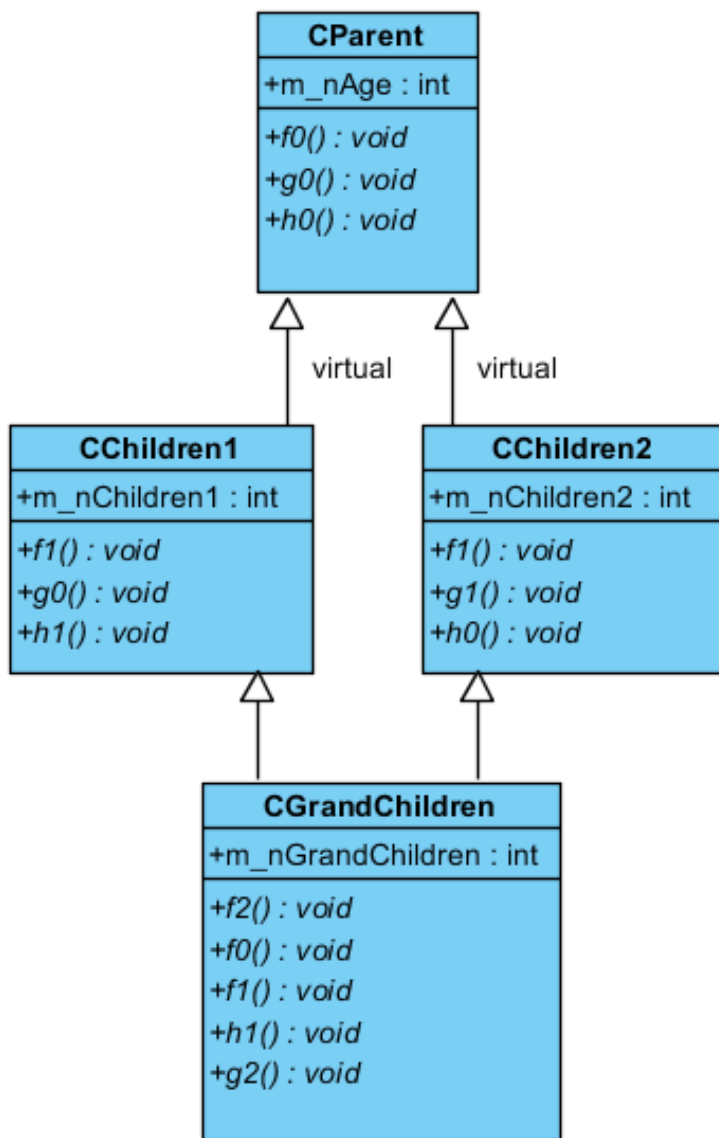
CChildren::f0 this adjustor: 12
CChildren::g1 this adjustor: 20
CChildren::f2 this adjustor: 0
CChildren::h0 this adjustor: 12
CChildren::h1 this adjustor: 20
CChildren::h2 this adjustor: 0

```

内存布局:



## 7. 钻石型的虚拟多重继承（含成员变量、虚函数、虚函数覆盖）



`int nGrandChildSize = sizeof(CGrandChildren) = 36`

d1reportSingleClass查看：

thunk函数：一种形实转换辅助函数；主要做this指针调整，函数调用重定向。



```

classLayout.cpp
class CGrandChildren    size(36):
    +---
    | +--- <base class CChildren1>
    | | <vfptr>
    | | <vbptr>
    | | m_nChildren1
    | +---
    | +--- <base class CChildren2>
    | | <vfptr>
    | | <vbptr>
    | | m_nChildren2
    | +---
    | m_nGrandChildren
    +---
    +--- <virtual base CParent>
    | <vfptr>
    | m_nAge
    +---

CGrandChildren::$vftable@CChildren1@:
    | &CGrandChildren_meta
    | 0
    | &CGrandChildren::f1
    | &CGrandChildren::h1
    | &CGrandChildren::f2
    | &CGrandChildren::g2

CGrandChildren::$vftable@CChildren2@:
    | -12
    | &thunk: this-=12; goto CGrandChildren::f1
    | &CChildren2::g1

CGrandChildren::$vftable@CChildren1@:
    | -4
    | 24 <CGrandChildren(CChildren1+4)CParent>

CGrandChildren::$vftable@CChildren2@:
    | -4
    | 12 <CGrandChildren(CChildren2+4)CParent>

CGrandChildren::$vftable@CParent@:

```

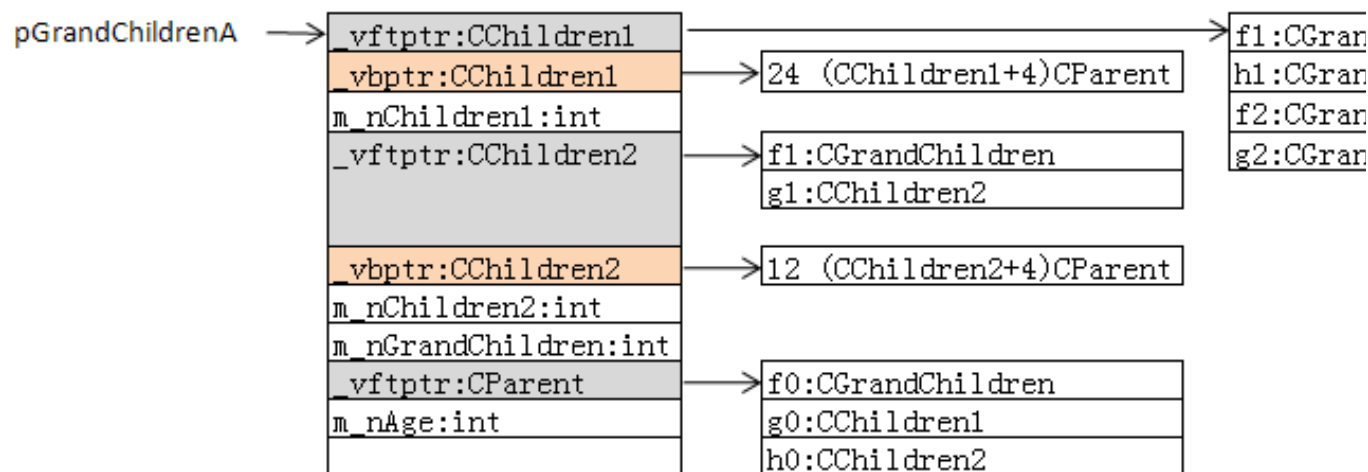
```

    : -28
0    : &CGrandChildren::f0
1    : &thunk: this-=16; goto CChildren1::g0
2    : &thunk: this-=4; goto CChildren2::h0

CGrandChildren::f2 this adjustor: 0
CGrandChildren::f0 this adjustor: 28
CGrandChildren::f1 this adjustor: 0
CGrandChildren::h1 this adjustor: 0
CGrandChildren::g2 this adjustor: 0

```

内存布局:



## #外部参考

C++类对应的内存结构

陈皓- C++ 虚函数表解析

陈皓 - C++ 对象的内存布局（上）

陈皓 - C++ 对象的内存布局（下）

钻石型虚拟多重继承的C++对象内存模型

分类: [计算机理论](#)

标签: C++

绿色通道:

[好文要顶](#)

[关注我](#)

[收藏该文](#)

[与我联系](#)



可可西

关注 - 0

粉丝 - 93

[+加关注](#)

4

0

(请您对文章做出评价)

« 上一篇: [英雄杀-斗魂活动](#)

» 下一篇: [函数调用-汇编分析](#)

posted @ 2013-01-27 19:15 可可西 阅读(2506) 评论(5) 编辑 收藏

## 评论列表

#1楼

2013-01-27 19:28 山是水的故事

不错，我这几天也正在看深入理解C++对象模型/。

支持(0) 反对(0)

#2楼

[楼主] 2013-01-27 19:47 可可西

大家可以一起深入探讨一下！欢迎留言。

支持(0) 反对(0)

#3楼

2013-04-11 19:51 lewvan00

写得真不错！

mark！

支持(0) 反对(0)

#4楼

2013-08-22 11:02 dyc0113

Very Good

支持(0) 反对(0)

#5楼

2014-02-20 00:11 Learning hard

这可是非常好，我都不得不推荐下了

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

【免费课程】案例：慕课网2048私人订制

【推荐】50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库

融云，免费为你的App加入IM功能——让你的App“聊”起来！！

【活动】百度开放云限量500台，抓紧时间申请啦！



### 最新IT新闻:

- 苹果音乐流媒体夏天发布 还要亲自做唱片公司
  - Windows 10 for Phone初探
  - 苹果CEO库克呼吁加强隐私保护
  - 阿里巴巴: 支持和认可快的与滴滴合并
  - 创业者必看: 你需要的不是钱, 而是商业模式
- » 更多新闻...



### 最新知识库文章:

- Bitmap的秘密
  - 我该如何向非技术人解释SQL注入?
  - 使用2-3法则设计分布式数据访问层
  - 数据清洗经验
  - 设计中的变与不变
- » 更多知识库文章...

---

Copyright ©2015 可可西