15,051,633 名会员

第363

文章 Q&A forums stuff lounge ?

Search for articles, questions,

手表

C++ 内部——类、结构和对象

BrainlessLabs.com

2014 年 7 月 21 日 MPL

评价我: 5.00/5 (3 票)

在本章中，我们将处理类、结构和对象。所以 class 或 struct 是我们用来在 C++ 中创建类的关键字。该类可以保存静态、非静态成员变量。类可以包含静态、非静态和虚拟成员函数。

在本章中，我们将处理类、结构和对象。

所以 class 或 struct 是我们用来在 C++ 中创建类的关键字。该类可以保存静态、非静态成员变量。类可以包含静态、非静态和虚拟成员函数。要详细了解类成员的所有可能表示，请参阅Inside the C++ object model。我们不会一一介绍。我们只会讨论当前的趋势是什么（Clang 中的趋势）。

让我们举个例子：

C++ 复制代码

```
class Point{
private:
float _x;
float _y;
//static int _pointCount;
public:
Point(){_x = 0; _y = 0;}
Point(const float x, const float y):_x(x), _y(y){/*++_pointCount;*/}
float x() const{ return _x;}
float y() const{ return _y;}
float x() { return _x;}
float y() { return _y;}
//virtual float pointOps(){float ret = _x+_y; return ret;}
//static int pointCount(){return _pointCount;}
};

//int Point::_pointCount = 0;

int main(){
Point p;
return 1;
}
```

汇编生成如下。根据您的 clang 设置，您可以在 ll 文件中获得更多或更少的代码。我删除了所有与我们手头的讨论无关或对我们的讨论没有任何价值的代码。例如，我们将忽略“#0”。这基本上是属性。一组属性用“#<Number>”引用。

MC++ 复制代码

```
%class.Point = type { float, float }
```

```
define i32 @main() {
entry:
%p = alloca %class.Point, align 4
call x86_thiscallcc void @_ZN5PointC2Ev(%class.Point* %p)
ret i32 1
}

; Function Attrs: nounwind
define linkonce_odr x86_thiscallcc void @_ZN5PointC2Ev(%class.Point* %this) unnamed_addr #1
align 2 {
entry:
%this.addr = alloca %class.Point*, align 4
store %class.Point* %this, %class.Point** %this.addr, align 4
%this1 = load %class.Point** %this.addr
%_x = getelementptr inbounds %class.Point* %this1, i32 0, i32 0
store float 0.000000e+00, float* %_x, align 4
%_y = getelementptr inbounds %class.Point* %this1, i32 0, i32 1
store float 0.000000e+00, float* %_y, align 4
ret void
}
```

让我们一一剖析上面的组装。

第 1 行是 LLVM 程序集表示数据收集的方式。结构类型用于表示内存中数据成员的集合。结构的元素可以是具有大小的任何类型。

例子：

`%T1 = type { <type list> }`

同样在我们的例子中 `class A; is %class.Point = type { float, float }`

现在转到"主要"功能。正如您所猜测的那样，"align"说明符用于指定对齐方式。下面是注释的主要代码。在 llvm 中，注释以";"开头。

C++ 复制代码

```
define i32 @main() #0 {
entry:
%p = alloca %class.Point, align 4 ; Allocate memory for our class Point instance

call x86_thiscallcc void @_ZN5PointC2Ev(%class.Point* %p) ; Constructor call. The name is
mangled.

ret i32 1
}
```

在代码中，我们将构造函数视为不同的函数。那么它是如何知道对象成员的呢？

那么这是`this`传递给构造函数的指针的工作，如下所示

`@_ZN5PointC2Ev(%class.Point* %this) //The this pointer passed to the class`

现在为什么必须调用构造函数？让我们看看构造函数内部：

1. 我们分配内存来存储一个指针`%this.addr = alloca %class.Point*, align 4`。
2. "存储"指令的格式为"存储类型值，类型*目标地址"。所以`store %class.Point* %this, %class.Point** %this.addr, align 4`。
3. "`load`" 指令的参数指定了从中加载的内存地址。因此`%this1 = load %class.Point** %this.addr`。将 %this.addr 指向的内容加载到 this1 上。这基本上是`this`指针。
4. '`getelementptr`'指令用于获取聚合 数据结构的子元素的地址 。它只执行地址计算，不访问内存。所以`%_x = getelementptr inbounds %class.Point* %this1, i32 0, i32 0`获取"_x"的地址。
5. 现在`store float 0.000000e+00, float* %_x, align 4`将在此元素中存储 0 浮点值。

空类

空类或结构是没有任何成员或成员函数的结构。那么当我们拥有这些结构时会发生什么呢？是否为它们分配了内存？如果是，那为什么？让我们探索这个概念。

考虑为以下 2 个构造生成 AST

C++ 复制代码

```
// test.cpp - input

struct emptyS{};
class emptyC{};
```

C++ 复制代码

```
// AST generated by the command

// "clang.exe -S -emit-llvm  -fcolor-diagnostics -Xclang -ast-dump test.cpp"

TranslationUnitDecl 0xbe0bc0 <<invalid sloc>> <invalid sloc>
|-TypedefDecl implicit __builtin_va_list 'char *'
|-CXXRecordDecl <test.cpp:1:1, col:15> col:8 struct emptyS definition
| `-CXXRecordDecl <col:1, col:8> col:8 implicit struct emptyS
`-CXXRecordDecl <line:2:1, col:14> col:7 class emptyC definition
  `-CXXRecordDecl <col:1, col:7> col:7 implicit class emptyC
```

从上面的代码我们观察到 test.cpp 是翻译单元。

我们将类定义为“CXXRecordDecl <line:2:1, col:14> col:7 class emptyC definition”

第二个 CXXRecordDecl 是插入到 C++ 类命名空间中的隐式类名，如 C++ 标准所述。

现在让我们为此生成 IR。

C++ 复制代码

```
// Input test.cpp

struct emptyS{};
class emptyC{};

int main(){
emptyS s;
emptyC c;
return 1;
}
```

MSIL 复制代码

```
// Output test.ll

%struct.emptyS = type { i8 }  // Non zero size
%class.emptyC = type { i8 }   // Non zero size

; Function Attrs: nounwind
define i32 @main() {
entry:
%retval = alloca i32, align 4
%s = alloca %struct.emptyS, align 1 // 1 Byte aligned
%c = alloca %class.emptyC, align 1 // 1 byte aligned
store i32 0, i32* %retval
ret i32 1
}
```

现在虽然类和结构没有数据成员，但它们的大小仍然非零。

原因：确保两个不同的对象将具有不同的地址是非零的。当我们为它们分配内存时

MSIL 复制代码

```
%s = alloca 0 and %c = alloca 0 will point to the same memory location.
```

工会

union 是一个用 class-key union 定义的类；它一次只保存一个数据成员。class-key 这里的意思是关键字 class/struct/union。

现在让我们看看 aa union 的内存布局是什么样的。评论是内联的。根据您的机器，您可以获得不同的 sizeof 值。在这里我们看到 double 的大小最大。所以联合只包含 double 。

C++ 复制代码

```
// ===Input===
typedef void (*FunPtrType)(void);
union U{
int _i;
float _f;
char _c;
double _d;
void* _p;
FunPtrType _fp;
};

int main(){
int sizeInt = sizeof(int);
int sizefloat = sizeof(float);
int sizeChar = sizeof(char);
int sizeDouble = sizeof(double);
int sizeV = sizeof(void*);
int sizeFP = sizeof(FunPtrType);

U u;
return 1;
}
```

MSIL 缩小▲ 复制代码

```
// ===Output===
// Command "c:\program files\LLVM\bin\clang.exe" -S test.cpp -emit-llvm  -fcolor-diagnostics
%union.U = type { double } // From below we can see that the size of double is 8,
                           // which is the largest

; Function Attrs: nounwind
define i32 @main() #0 {
entry:
%retval = alloca i32, align 4

// Allocate size for sizeof variables
%sizeInt = alloca i32, align 4
%sizefloat = alloca i32, align 4
%sizeChar = alloca i32, align 4
%sizeDouble = alloca i32, align 4
%sizeV = alloca i32, align 4
%sizeFP = alloca i32, align 4

// Allocate size for union
%u = alloca %union.U, align 8
store i32 0, i32* %retval

// store <type> <value>, <dest address>
store i32 4, i32* %sizeInt, align 4 // size only 4
store i32 4, i32* %sizefloat, align 4 // size only 4
store i32 1, i32* %sizeChar, align 4 // size only 1
store i32 8, i32* %sizeDouble, align 4 // Largest size 8 ========
store i32 4, i32* %sizeV, align 4 // size only 4
store i32 4, i32* %sizeFP, align 4 // size only 4
```

```
ret i32 1
}
```

<上一页

参考书目：

1. Itanium 中的名称修改。

本文最初发表于http://brainlesslabs.com/inside-cpp-class-struct-and-objects

执照

本文以及任何相关的源代码和文件均根据The Mozilla Public License 1.1 (MPL 1.1)获得许可

分享

关于作者

BrainlessLabs.com

建筑师
印度

手表
该会员

我喜欢探索技术的不同方面。尝试新事物，并获得快乐。我的兴趣是编程语言和成像。但在其他事情上工作也不难。算法让我在喝咖啡休息时感到高兴。

我基本上用 C++ 编写代码，但 JAVA 对我来说并不陌生。我也知道很少的脚本语言。基本上我觉得知道一门编程语言只是一个无...

展示更多

评论和讨论

添加评论或问题 | 电子邮件提醒 | Search Comments

-- 本论坛暂无消息 --

文章 版权所有 2014 BrainlessLabs.com
其他所有内容 版权所有 © CodeProject ,
1999-2021 Web04 2.8.20210930.1