c++ 中使用 new 创建的对象和不使用 new 创建对象的区别

原创 Mr wu52 已于 2024-10-18 17:43:39 修改

文章标签： c++

在 C++ 中，使用 `new` 关键字和不使用 `new` 创建对象的方式有几个重要的区别。这些区别主要体现在内存管理、生命周期和作用域方面。以下是详细的比较：

1. 内存分配

- **使用 `new` 创建对象**：
  - 当使用 `new` 创建对象时，内存是在堆（heap）上分配的。堆是一个可动态分配内存的区域，适用于需要在程序运行时确定大小的对象。
  - 例如：

```
MyClass* obj = new MyClass(); // 在堆上分配
```

- **不使用 `new` 创建对象**：
  - 直接创建对象时，内存是在栈（stack）上分配的。栈是用于存储局部变量和函数调用的区域，其大小通常在编译时确定。
  - 例如：

```
MyClass obj; // 在栈上分配
```

2. 对象生命周期

- **使用 `new` 创建的对象**：
  - 对象的生命周期由程序员控制，直到调用 `delete` 释放它。使用 `new` 分配的内存不会自动释放，程序员需要手动管理内存，以防内存泄漏。

- 例如：

```
delete obj; // 需要手动释放
```

- **不使用 `new` 创建的对象**：
  - 对象的生命周期由作用域决定。当对象超出其作用域时，自动调用 析构函数，释放内存。
  - 例如：

```
{
    MyClass obj; // obj 的作用域结束时自动释放
} // 此处 obj 被销毁
```

3. 作用域和访问

- **使用 `new` 创建对象**：
  - 通常会返回一个指向对象的指针，因此需要使用指针来访问对象的成员。
  - 对象的作用域不受限制，可以在整个程序中访问（直到显式删除）。
- **不使用 `new` 创建对象**：
  - 对象的作用域限制在其定义的代码块内。超出这个作用域后，无法再访问该对象。
  - 可以直接使用对象名来访问其成员。

4. 性能和开销

- **使用 `new` 创建对象**：
  - 在堆上分配和释放内存通常比在栈上慢，因为堆管理需要更复杂的机制。
  - 需要考虑内存碎片化的问题。
- **不使用 `new` 创建对象**：

- 栈上分配和释放内存速度快，因为只需调整栈指针。
- 更容易进行内存管理，通常也不需要担心内存泄漏。

5. 使用场景

- **使用 `new` 创建对象**:
  - 适用于需要动态大小的对象（如数组），或者在对象的生命周期需要超出其定义作用域时。
  - 例如，数据结构（如链表、树）通常在堆上动态分配。
- **不使用 `new` 创建对象**:
  - 适用于确定大小且生命周期相对较短的对象（如局部变量）。
  - 更简单的类和临时对象通常在栈上创建。

示例代码

```cpp
#include <iostream>

class MyClass {
public:
    MyClass() { std::cout << "Constructor called\n"; }
    ~MyClass() { std::cout << "Destructor called\n"; }
};

int main() {
    // 使用 new 创建对象
    MyClass* heapObj = new MyClass(); // 在堆上创建
    delete heapObj; // 需要手动删除

    // 不使用 new 创建对象
    {
        MyClass stackObj; // 在栈上创建
    } // stackObj 的作用域结束，自动调用析构函数


```

```
    return 0;
}
```

总结

在 C++ 中，使用 `new` 和不使用 `new` 创建对象的主要区别在于内存分配位置、生命周期管理、作用域以及性能开销。选择使用哪种方式取决于具体的应用场景和需求。在需要动态内存管理时使用 `new`，而在可预见的短生命周期和固定大小的对象时，使用栈上分配会更简单和安全。