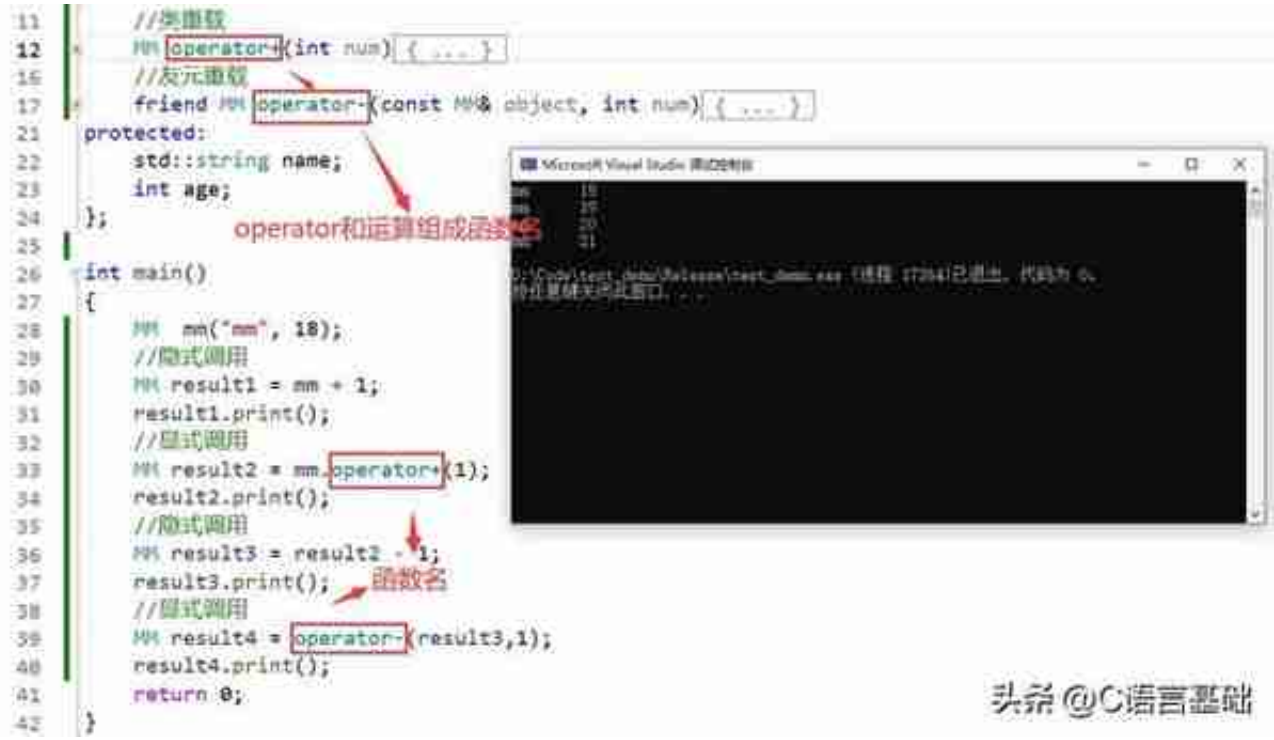
# 什么是运算符重载

在C++中，运算符重载允许开发者自定义运算符的操作和行为。这个特性允许用户将运算符用于自定义类型的对象，从而使代码更加灵活和易于读取。通常情况运算符不能直接操作自定义类型(类或者结构体),如下错误代码：




运算符重载注意项：

- . = () -> [] 只能重载为类的成员函数
- 运算符重载必须包含一个自定义类型(结构体或者类)
- . .* ?: :: 不能被重载
- 重载运算符不能无中生有
- 习惯行为: 单目用成员函数重载,双目用友元重载

# C++流运算符重载

在C++中流重载允许开发者自定义输入输出流的行为，通过流重载，我们可以以与标准类型相同的方式定制化自定义类型的输入及输出操作，方便了与标准类型交互。流重载须知两个类型

- cin： istream类的对象

- cout:： ostream类的对象

如下流重载代码：

```
#include <iostream>#include <string>class MM{public:  MM(std
::string name, int age) :name(name), age(age){}  void print(
)  {    std::cout << name << "\t" << age << std::endl;  }  f
riend std::ostream& operator<<(std::ostream& out, const MM&
object)  {    out << object.name << " " << object.age << st
d::endl;    return out;  }  friend std::istream& operator>>(
std::istream& in, MM& object)  {    in >> object.name >> obj
ect.age;    return in;  }protected:  std::string name;  int
age;};int main(){  MM  mm("mm", 18);  std::cout << mm;  std:
:cin >> mm;  mm.print();  return 0;}
```

程序运行结果如下：


# C++文本重载

1s

。为了实现这种语法，我们可以通过重载后缀运算符来实现。后缀重载函数的要点有以下两个：

- 函数名是固定的：operator”“ 后缀
- 函数参数有且只有一个函数参数，并且只能是字符类型或者unsigned long long 类型

如下文本重载代码：

```
#include <iostream>#include <string>unsigned long long opera
tor""_s(unsigned long long num) {  return num;}unsigned long
 long operator""_min(unsigned long long num){  return num*60
;}unsigned long long operator""_h(unsigned long long num){
return num * 60*60;}int main(){  int sec = 30_s + 20_min + 1
_h;  std::cout << sec << std::endl;  return 0;}
```

程序运行结果如下：


总之，重载 new 和 delete 运算符可以让我们更好地控制内存的分配和释放，可以用于实现一些高级的内存管理方案。但是，需要注意的是，重载这些运算符时需要格外小心，以确保正确的内存管理。

## C++对象的隐式转换

C++对象的隐式转换是operator的另一个用法，简单来说可以直接用对象赋值给基本数据，便捷快速的提取想要的数据类型。基本语法：operator DataType () {return data;}

如下代码：

```
#include <iostream>#include <string>class MM{public:  MM(std
::string name, int age) :name(name), age(age) {}  operator s
td::string()  {    return name;  }  operator int()  {    ret
urn age;  }  std::string getName() { return name; }  int get
Age() { return age; };protected:  std::string name;  int age
;};int main(){  MM mm("girl", 18);  //????:??????????????? 
 std::string name = mm;  int age = mm;  std::cout << name <<
 "\t" << age << std::endl;  name = mm.getName();  age = mm.g
etAge();  std::cout << name << "\t" << age << std::endl;  re
turn 0;}
```

程序运行结果如下：