

# 16\_类的真正形态

- `struct` 在C语言中已经有了自己的含义，所以C++**必须继续兼容**
- 在C++中提供了新的关键字 `class` 用于类定义
- `class` 和 `struct` 的用法完全相同

## class 和 struct 本质区别

这里的 `struct` 是C++中的 `struct`，不是C语言中的 `struct`

- 在用 `struct` 定义类时，所有的成员**默认访问级别**为 `public`
- 在用 `class` 定义类时，所有的成员**默认访问级别**为 `private`

## 实验

lesson\_16-1.cpp

```
1  #include <stdio.h>
2
3  struct A
4  {
5      // default to public
6      int i;
7
8      int getI()
9      {
10         return i;
11     }
12 };
13
14 class B
15 {
16     // default to private
17     int i;
18
19     int getI()
20     {
21         return i;
22     }
23 };
24
25 int main(void)
26 {
27     A a;
28     a.i = 1;
```

```

29
30     printf("a.getI() = %d\n", a.getI());
31
32     B b;
33     //b.i = 2; // error: 'int B::i' is private within this context
34     //printf("b.getI() = %d\n", b.getI()); //error: 'int B::getI()' is private within
this context
35
36     return 0;
37 }

```

## 小实例

开发一个四则运算的类

- 提供 `setOperator` 函数设置运算类型
- 提供 `setParameter` 函数设置运算参数
- 提供 `result` 函数进行运算
  - 其返回值表示运算的合法性
  - 通过引用参数返回结果
- 将类的实现与定义分开
  - `.h` 头文件只有类的声明
    - 成员变量和成员函数的声明
  - `.cpp` 源文件中完成类的其它实现
    - 成员函数的具体实现

lesson\_16-2.h

```

1  class Operator
2  {
3  private:
4      char m_operateType;
5      int m_op1;
6      int m_op2;
7      int m_op3;
8      int number;
9
10 public:
11     void setOperator(char);
12     void setParameter(int, int);
13     int result(int&);
14 };

```

lesson\_16-2.cpp

```

1  #include <stdio.h>
2  #include "lesson_16-2.h"
3
4  void Operator::setOperator(char c)
5  {
6      m_operateType = c;
7  }
8
9  void Operator::setParameter(int a, int b)
10 {
11     m_op1 = a;
12     m_op2 = b;
13 }
14
15 int Operator::result(int& result)
16 {
17     if (m_operateType == '*')
18         result = m_op1 * m_op2;
19     else if (m_operateType == '/')
20         result = m_op1 / m_op2;
21     else if (m_operateType == '+')
22         result = m_op1 + m_op2;
23     else if (m_operateType == '-')
24         result = m_op1 - m_op2;
25     else
26         return -1;
27     return 0;
28 };
29
30 int main(void)
31 {
32     Operator op;
33     int number = 0;
34     op.setOperator('-');
35     op.setParameter(5, 2);
36     op.result(number);
37
38     printf("number= %d\n", number);
39 }

```

## 改进

lesson\_16-3.h

```

1  #ifndef _lesson_16_3_H
2  #define _lesson_16_3_H
3  class Operator
4  {
5  private:
6      char m_operateType;

```

```
7     double m_op1;
8     double m_op2;
9
10    public:
11        void setOperator(char);
12        void setParameter(double, double);
13        bool result(int&);
14    };
15 #endif
```

lesson\_16-3.cpp

```
1 #include "lesson_16-3.h"
2
3 void Operator::setOperator(char c)
4 {
5     m_operateType = c;
6 }
7
8 void Operator::setParameter(double a, double b)
9 {
10    m_op1 = a;
11    m_op2 = b;
12 }
13
14 bool Operator::result(int& result)
15 {
16    /*
17     * if (m_operateType == '*')
18     *     result = m_op1 * m_op2;
19     * else if (m_operateType == '/')
20     *     result = m_op1 / m_op2;
21     * else if (m_operateType == '+')
22     *     result = m_op1 + m_op2;
23     * else if (m_operateType == '-')
24     *     result = m_op1 - m_op2;
25     * else
26     *     return -1;
27     * return 0;
28     */
29
30
31    bool status = true;
32
33    switch(m_operateType)
34    {
35        case '+':
36            result = m_op1 + m_op2;
37            break;
38        case '-':
39            result = m_op1 - m_op2;
```

```

40         break;
41     case '*':
42         result = m_op1 * m_op2;
43         break;
44     case '/':
45         if ((-0.00000001 < m_op2) && (m_op2 < 0.00000001))
46         {
47             status = false;
48         }
49         else
50         {
51             result = m_op1 / m_op2;
52         }
53         break;
54     default:
55         status = false;
56         break;
57 }
58 return status;
59 }

```

lesson\_16-3\_main.cpp

```

1  #include "stdio.h"
2  #include "lesson_16-3.h"
3
4  int main(void)
5  {
6      Operator op;
7      int number = 0;
8      op.setOperator('-');
9      op.setParameter(5, 2);
10     op.result(number);
11
12     printf("number= %d\n", number);
13 }

```

每个头文件不允许在同一个翻译单元中重复引用???

## 小结

- C++引入新的关键字 `class` 用于定义类
- `struct` 和 `class` 的区别在于 默认访问级别的不同
- C++中的类支持 声明 和 实现 的分离
  - 在头文件中声明类
  - 在源文件中实现类