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`

```
#include <stdio.h>

struct A
{
    // default to public
    int i;

    int getI()
    {
        return i;
    }
};

class B
{
    // default to private
    int i;

    int getI()
    {
        return i;
    }
};

int main(void)
{
    A a;
    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`

```cpp
#include <stdio.h>
#include "lesson_16-2.h"

void Operator::setOperator(char c)
{
    m_operateType = c;
}

void Operator::setParameter(int a, int b)
{
    m_op1 = a;
    m_op2 = b;
}

int Operator::result(int& result)
{
    if (m_operateType == '*')
        result = m_op1 * m_op2;
    else if (m_operateType == '/')
        result = m_op1 / m_op2;
    else if (m_operateType == '+')
        result = m_op1 + m_op2;
    else if (m_operateType == '-')
        result = m_op1 - m_op2;
    else
        return -1;
    return 0;
};

int main(void)
{
    Operator op;
    int number = 0;
    op.setOperator('-');
    op.setParameter(5, 2);
    op.result(number);

    printf("number= %d\n", number);
}
```

改进

`lesson_16-3.h`

```cpp
#ifndef _lesson_16_3_H
#define _lesson_16_3_H
class Operator
{
private:
    char m_operateType;
```

```
    double m_op1;
    double m_op2;

public:
    void setOperator(char);
    void setParameter(double, double);
    bool result(int&);
};
#endif
```

lesson_16-3.cpp

```
#include "lesson_16-3.h"

void Operator::setOperator(char c)
{
    m_operateType = c;
}

void Operator::setParameter(double a, double b)
{
    m_op1 = a;
    m_op2 = b;
}

bool Operator::result(int& result)
{
    /*
    if (m_operateType == '*')
        result = m_op1 * m_op2;
    else if (m_operateType == '/')
        result = m_op1 / m_op2;
    else if (m_operateType == '+')
        result = m_op1 + m_op2;
    else if (m_operateType == '-')
        result = m_op1 - m_op2;
    else
        return -1;
    return 0;

    */

    bool status = true;

    switch(m_operateType)
    {
        case '+':
            result = m_op1 + m_op2;
            break;
        case '-':
            result = m_op1 - m_op2;
```

```
            break;
        case '*':
            result = m_op1 * m_op2;
            break;
        case '/':
            if ((-0.00000001 < m_op2) && (m_op2 < 0.00000001))
            {
                status = false;
            }
            else
            {
                result = m_op1 / m_op2;
            }
            break;
        default:
            status = false;
            break;
    }
    return status;
}
```

lesson_16-3_main.cpp

```
#include "stdio.h"
#include "lesson_16-3.h"

int main(void)
{
    Operator op;
    int number = 0;
    op.setOperator('-');
    op.setParameter(5, 2);
    op.result(number);

    printf("number= %d\n", number);
}
```

> 每个头文件不允许在同一个翻译单元中重复引用？？？

小结

- C++引进新的关键字 `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