C++程序设计(拾陆)

徐东/计算数学

内容

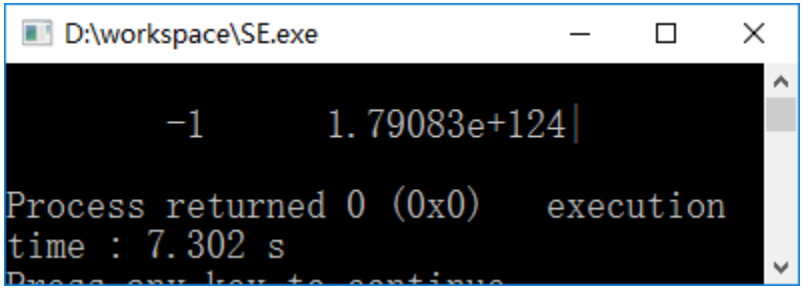
- 创建自定义的新数据类型（III）
  - 构造函数
  - 默认的构造函数
  - 析构函数
  - 访问控制级别
    - public/private
  - class与struct的区别

```
struct Student{
   string name;
   char   gender;
   int    age;
   double weight;
   double height;
   double calBmi(){
      return weight/(height*height);
   }
 };

int main(){
   const int NUMBER = 3;
   Student students[NUMBER];

   for(int i=0;i<NUMBER;++i){
      cout << students[i].name << "\t" << students[i].gender << "\t"
           << students[i].calBmi() << "|“ << endl;
   }
   return 0;
}
```

```
D:\workspace\SE.exe
        7282324 0|
        1987144546      inf|
        4310608 1.1068e+045|

Process returned 0 (0x0)   execution time : 8
```

- **原因：数组元素没有初始化**

```
struct Student{
   string name;
   char   gender;
   int    age;
   double weight;
   double height;
   double calBmi(){
      return weight/(height*height);
   }
 };

int main(){

   Student one ;
   cout << one.name << "\t" << one.age << "\t"
        << one.calBmi() << "|" <<endl;

   return 0;
}
```



- **原因：变量one没有初始化**

自定义数据类型：Student

- 问题
  1. students数组元素没有初始化
  2. 变量 one 没有初始化

- 总结：使用未初始化的Student类型变量

- 对Student类型变量进行初始化
  - 对该类型变量的组成部分（成员变量、域）进行初始化
  - 构造函数

自定义数据类型：构造函数

- 作用
  - 初始化成员变量
- 特殊的成员函数
  - 函数名与类型名称相同
  - 没有返回类型
  - 可以重载
  - 在创建变量时，由系统自动调用。

构造函数只能由系统调用

构造函数

```
struct Student{
    Student(){
        name = "mickey";
        gender = 'F';
        age = 33;
        weight = 50.0;
        height = 1.66;
    }

    string name;
    char   gender;
    int    age;
    double weight;
    double height;

    double calBmi(){
        return weight/(height*height);
    }

};
```

- **构造函数**
  - **特殊的成员函数**

类的成员可以按任意顺序在“类的定义”出现

构造函数：特殊的成员函数

```
struct Student{
    Student(){
        name = "mickey";
        gender = 'F';
        age = 33;
        weight = 50.0;
        height = 1.66;
    }

    string name;
    char   gender;
    int    age;
    double weight;
    double height;

    double calBmi(){
        return weight/(height*height);
    }

};
```

- 构造函数
  - 函数名与类型名相同

构造函数：特殊的成员函数

```
struct Student{
    Student(){
        name = "mickey";
        gender = 'F';
        age = 33;
        weight = 50.0;
        height = 1.66;
    }

    string name;
    char   gender;
    int    age;
    double weight;
    double height;

    double calBmi(){
        return weight/(height*height);
    }

};
```

- 构造函数
  - 函数名与类型名相同
  - 没有返回类型
    - 不需要 return 语句

构造函数：特殊的成员函数

```
struct Student{
    Student(){
        name = "mickey";
        gender = 'F';
        age = 33;
        weight = 50.0;
        height = 1.66;
    }

    string name;
    char   gender;
    int    age;
    double weight;
    double height;

    double calBmi(){
        return weight/(height*height);
    }

};
```

- 构造函数
  - 函数名与类型名相同
  - 没有返回类型
  - 对成员变量进行初始化
    - 构造函数的作用

构造函数：特殊的成员函数

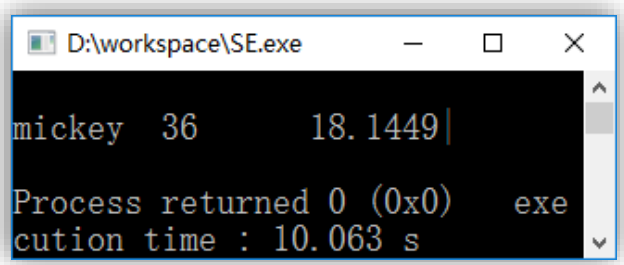
```
struct Student{
    Student(){
        name = "mickey";
        gender = 'F';
        age = 33;
        weight = 50.0;
        height = 1.66;
    }
    ……
};

int main(){
    Student one ;
    cout << one.name << "\t" << one.age << "\t"
         << one.calBmi() << "|" <<endl;

    return 0;
}
```

- 构造函数
  - 函数名与类型名相同
  - 没有返回类型
  - 对成员变量进行初始化
  - 创建变量时由系统调用

构造函数：特殊的成员函数

```
struct Student{
    Student(){
        name = "mickey";
        gender = 'F';
        age = 33;
        weight = 50.0;
        height = 1.66;
    }
    ……
};

int main(){
    Student one ;
    cout << one.name << "\t" << one.age << "\t"
         << one.calBmi() << "|" <<endl;

    return 0;
}
```

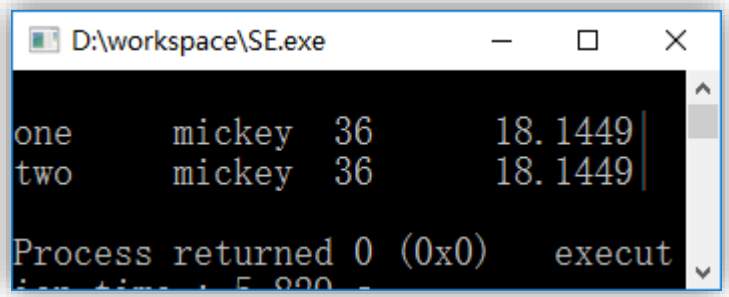
- 构造函数
  - 函数名与类型名相同
  - 没有返回类型
  - 对成员变量进行初始化
  - 创建变量时由系统调用



构造函数：特殊的成员函数

```
struct Student{
    Student(){
        name = "mickey";
        gender = 'F';
        age = 33;
        weight = 50.0;
        height = 1.66;
    }
    ……
};
```

```
Student one ;
cout << "one" <<"\t" << one.name << "\t"
     << one.age << "\t" << one.calBmi() << "|" <<endl;

Student two ;
cout << "two" <<"\t"<<two.name << "\t"
     << two.age << "\t" << two.calBmi() << "|" <<endl;
```



- 无参构造函数
  - 默认的构造函数
  - 采用统一策略对成员变量进行初始化（“克隆”）

构造函数：特殊的成员函数

```
struct Student{
    Student(){
        name = "mickey"; gender = 'F'; age = 36;
        weight = 50.0;   height = 1.66;
    }

    string name;
    char   gender;
    int    age;
    double weight;
    double height;

    Student(string name,char gender,int age,double w,double h){
        ……
    }
};
```

- **带参数的构造函数**

构造函数：特殊的成员函数

```
struct Student{
    Student(){
        name = "mickey"; gender = 'F'; age = 36;
        weight = 50.0;   height = 1.66;
    }

    string name;
    char   gender;
    int    age;
    double weight;
    double height;

    Student(string name,char gender,int age,double w,double h){
        ……
    }
};
```

- 带参数的构造函数（使用具体值对成员变量初始化）

构造函数：特殊的成员函数

```
struct Student{
    string name;
    char   gender;
    int    age;
    double weight;
    double height;
    Student(string name,char gender,int age,double w,double h){
        weight = w ;
        height = h ;

    }
};
```

构造函数：特殊的成员函数

```
struct Student{
    string name;
    char   gender;
    int    age;
    double weight;
    double height;
    Student(string name,char gender,int age,double w,double h){
        weight = w ;
        height = h ;

    }
};
```

- 使用参数 w 对成员变量 weight 进行初始化

构造函数：特殊的成员函数

```
struct Student{
    string name;
    char   gender;
    int    age;
    double weight;
    double height;
    Student(string name,char gender,int age,double w,double h){
        weight = w ;
        height = h ;

    }
};
```

- 使用参数 w 对成员变量 weight 进行初始化

构造函数：特殊的成员函数

```
struct Student{
    string name;
    char   gender;
    int    age;
    double weight;
    double height;
    Student(string name,char gender,int age,double w,double h){
        weight = w ;
        height = h ;

    }
};
```

- 使用参数 w 对成员变量 weight 进行初始化
- 使用参数 h 对成员变量 height 进行初始化

构造函数：特殊的成员函数

```
struct Student{
    string name;
    char   gender;
    int    age;
    double weight;
    double height;
    Student(string name,char gender,int age,double w,double h){
        weight = w ;
        height = h ;
        name = name;
    }
};
```

- **使用参数 name 对成员变量 name 进行初始化**

构造函数：特殊的成员函数

```
struct Student{
    string name;
    char   gender;
    int    age;
    double weight;
    double height;
    Student(string name,char gender,int age,double w,double h){
        weight = w ;
        height = h ;
        name = name; //???
    }
};
```

- 使用参数 name 对成员变量 name 进行初始化
  - 参数名与成员变量同名

构造函数：特殊的成员函数

```
struct Student{
    string name;
    char   gender;
    int    age;
    double weight;
    double height;
    Student(string name,char gender,int age,double w,double h){
        weight = w ;
        height = h ;
        name = name; //???
    }
};
```

- 使用参数 name 对成员变量 name 进行初始化
  - 参数名与成员变量同名（成员变量被屏蔽）

构造函数：特殊的成员函数

```
struct Student{
    string name;
    char   gender;
    int    age;
    double weight;
    double height;
    Student(string name,char gender,int age,double w,double h){
        weight = w ;
        height = h ;
        name = name; // 参数name对自己进行赋值，而没有对成员变量name赋值。
    }
};
```

- **使用参数 name 对成员变量 name 进行初始化**
  - **参数名与成员变量同名（成员变量被屏蔽）**

构造函数：特殊的成员函数

```
struct Student{
    string name;
    char   gender;
    int    age;
    double weight;
    double height;
    Student(string name,char gender,int age,double w,double h){
        weight = w ;
        height = h ;
        name = name; // 参数name对自己进行赋值，而没有对成员变量name赋值。
    }
};
```

- **解决方法**
  - **改变参数的名称（确保不与成员变量同名）**

构造函数：特殊的成员函数

```
struct Student{
    string name;
    char   gender;
    int    age;
    double weight;
    double height;
    Student(string nVal,char gender,int age,double w,double h){
        weight = w ;
        height = h ;
        name = nVal;
    }
};
```

- **解决方法**
  - **改变参数的名称（确保不与成员变量同名）**

构造函数：特殊的成员函数

```
struct Student{
    string name;
    char   gender;
    int    age;
    double weight;
    double height;
    Student(string name,char gender,int age,double w,double h){
        weight = w ;
        height = h ;

    }
};
```

- **解决方法（推荐）**
  - **使用 this指针（this 代表当前对象）**

构造函数：特殊的成员函数

```
struct Student{
    string name;
    char   gender;
    int    age;
    double weight;
    double height;
    Student(string name,char gender,int age,double w,double h){
        weight = w ;
        height = h ;
        this->name = name ;
    }
};
```

- **解决方法（推荐）**
  - **使用 this指针（this 代表当前对象）**

this

- this 指针
  - 代表当前对象（变量）

```
Student(string name,char gender,
        int age,double w,double h){
    this->name = name;
    this->gender = gender;
    this->age = age;
    this->weight = w;
    height = h;
    this->calBmi();
}
```

- 用法
  - this->成员变量
  - this->成员函数([参数列表])
- this 指针只能出现在类的（成员函数）定义内

构造函数

- 默认的构造函数
  - 无参数的构造函数
  - 当设计者没有提供任何构造函数的时候，编译器自动创建一个无参数并且函数体为空的构造函数。

- 建议：至少提供两个构造函数
  - 无参数的构造函数（默认构造函数）
  - 有参数的构造函数

构造函数

```
struct Student{

  Student(){
     name = "mickey";
     gender = 'F';
     age = 36;
     weight = 50.0;
     height = 1.66;
  }

  Student(string name,char gender,
          int age,double w,double h){
     this->name = name;
     this->gender = gender;
     this->age = age;
     this->weight = w;
     height = h;
  }
  ......
};
```

- 如何在构造函数中调用同类的其他构造函数?

构造函数

- 如何在构造函数中调用同类的其他构造函数?
  - 在构造函数的初始化列表中调用另一个构造函数
  - C++ 11 支持

```
Student(string name,char gender,int age,double w,double h){
    this->name = name;
    this->gender = gender;
    this->age = age;
    this->weight = w;
    height = h;
}

Student(): Student("mickey",'F',36,50.0,1.66) {

}
```

构造函数

```
Student(string name,char gender,int age,double w,double h){
      this->name = name;
      this->gender = gender;
      this->age = age;
      this->weight = w;
      height = h;
}
Student() : Student("mickey",'F',36,50.0,1.66) {
}
```

- **初始化列表**
  - **冒号（：）与左大括号（{）之间的范围**

构造函数

```
Student(string name,char gender,int age,double w,double h){
        this->name = name;
        this->gender = gender;
        this->age = age;
        this->weight = w;
        height = h;
}
Student() : Student("mickey",'F',36,50.0,1.66) {
}
```

- **在初始化列表中，调用**

  Student(string,char,int,double,double)。

任务肆拾玖

- 处理学生的个人信息
  - 姓名、性别、年龄、身高、体重、bmi
  - 课程成绩
  - 平均成绩
- 平均成绩由课程成绩计算获得（算术平均）
- 课程成绩
  - 每位学生的课程数量不同

任务肆拾玖

```
struct Student{
  //课程的成绩



  string name;
  char   gender;
  int    age;
  double weight;
  double height;

  double calBmi(){
    return weight/(height*height);
  }

};
```

每位学生的课程数量不同

解决方法：

任务肆拾玖

```
struct Student{
  //课程的成绩
  double *courseGrades;


  string name;
  char   gender;
  int    age;
  double weight;
  double height;

  double calBmi(){
    return weight/(height*height);
  }

};
```

每位学生的课程数量不同

解决方法：动态数组

任务肆拾玖

```
struct Student{
  //课程的成绩
  double *courseGrades;
  int    courseNumber;  //每位学生选修的课程数量

  string name;
  char   gender;
  int    age;
  double weight;
  double height;
  double calBmi(){
    return weight/(height*height);
  }

};
```

任务肆拾玖

```
struct Student{
    double *courseGrades; //保存课程成绩
    int    courseNumber;  //每位学生选修的课程数量

    Student(string name,char gender,int age,
            double w,double h,int courseNumber){
        this->name = name;
        this->gender = gender;
        this->age = age;
        this->weight = w;
        height = h;



    }
};
```

任务肆拾玖

```
struct Student{
    double *courseGrades; //保存课程成绩
    int    courseNumber;  //每位学生选修的课程数量

    Student(string name,char gender,int age,
            double w,double h,int courseNumber){
        this->name = name;
        this->gender = gender;
        this->age = age;
        this->weight = w;
        height = h;



    }
};
```

任务肆拾玖

```
struct Student{
    double *courseGrades; //保存课程成绩
    int    courseNumber;  //每位学生选修的课程数量

    Student(string name,char gender,int age,
            double w,double h,int courseNumber){
        this->name = name;
        this->gender = gender;
        this->age = age;
        this->weight = w;
        height = h;
        this->courseNumber = courseNumber;
        this->courseGrades = new double[this->courseNumber];
    }
};
```

任务肆拾玖

- 如何处理“课程成绩”的输入 ?
  - 成员函数
    - 函数名 `inputCourseGrades`
    - 参数列表 无
    - 返回类型 `void`（没有返回值）

任务肆拾玖：课程成绩的输入

```
struct Student{

    double *courseGrades;
    int     courseNumber;   //每位学生选修的课程数量

    void inputCourseGrades(){
        cout << "开始输入课程成绩 ： " << endl;

        for(int i = 0; i < courseNumber; ++i){
          cout << "第 " << i+1 << " 门课程的成绩 ： ";
          cin >> courseGrades[i];
        }

        cout << "成绩输入完毕。" << endl;
    }
};
```

- **Student 的成员函数 inputCourseGrades()**

任务肆拾玖：课程成绩的输入

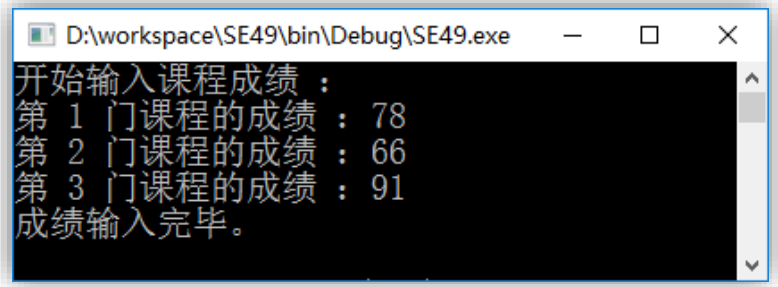
```
struct Student{
   void inputCourseGrades(){
      cout << "开始输入课程成绩 :" << endl;
      for(int i = 0; i < courseNumber; ++i){
        cout << "第 " << i+1 << " 门课程的成绩 :";
        cin >> courseGrades[i];
      }
      cout << "成绩输入完毕。" << endl;
   }

   ……
 };

int main(){
   Student one("mickey", 'F', 36, 50.0, 1.66, 3 );
                                    //输入成绩

   return 0;
}
```

任务肆拾玖：课程成绩的输入

```
struct Student{
   void inputCourseGrades(){
      cout << "开始输入课程成绩 :" << endl;
      for(int i = 0; i < courseNumber; ++i){
        cout << "第 " << i+1 << " 门课程的成绩 :";
        cin >> courseGrades[i];
      }
      cout << "成绩输入完毕。" << endl;
   }

   ……
 };

int main(){
   Student one("mickey", 'F', 36, 50.0, 1.66, 3 );
   one.inputCourseGrades(); //输入成绩

   return 0;
}
```

任务肆拾玖：课程成绩的输入

```
struct Student{
    void inputCourseGrades(){
        cout << "开始输入课程成绩 :" << endl;
        for(int i = 0; i < courseNumber; ++i){
          cout << "第 " << i+1 << " 门课程的成绩 :";
          cin >> courseGrades[i];
        }
        cout << "成绩输入完毕。" << endl;
    }
    ……
 };

int main(){
    Student one("mickey", 'F', 36, 50.0, 1.66, 3 );
    one.inputCourseGrades(); //输入成绩（调用成员函数）

    return 0;
}
```



任务肆拾玖

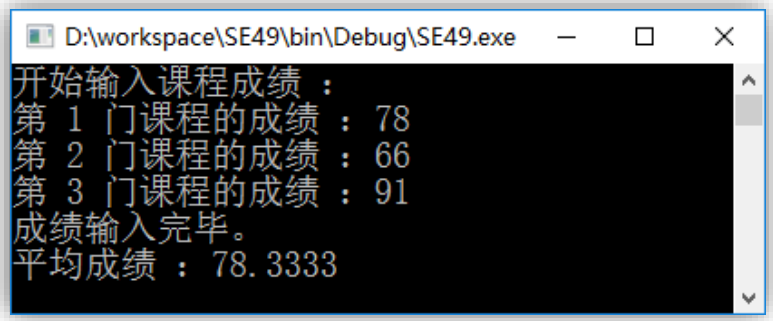
- 如何处理“平均成绩”？
  - 成员函数
    - 函数名　　gpa (grade point average)
    - 参数列表　　无
    - 返回类型　　double

任务肆拾玖：平均成绩

```
struct Student{
    double *courseGrades;
    int    courseNumber;

    double gpa(){
        double sum = 0.0;
        for(int i = 0; i < courseNumber; ++i){
            sum += courseGrades[i];
        }
        return sum/courseNumber;
    }
    ……
}
```

任务肆拾玖：平均成绩

```
struct Student{
    double *courseGrades;
    int     courseNumber;
    double gpa(){
        double sum = 0.0;
        for(int i = 0; i < courseNumber; ++i){
            sum += courseGrades[i];
        }
        return sum/courseNumber;
    }
    ……
}
int main(){
    Student one("mickey",'F',36,50.0,1.66,3);
    one.inputCourseGrades();
    cout << "平均成绩 : " << one.gpa() << endl;
    return 0;
}
```



任务肆拾玖

```
Student(string name,char gender,int age,double w,double h,
        int courseNumber){
      this->name = name;
      this->gender = gender;
      this->age = age;
      this->weight = w;
      height = h;
      this->courseNumber = courseNumber;
      this->courseGrades = new double[this->courseNumber];
}
```

- 在构造函数中，创建动态数组。
- 何时释放动态数组?

任务肆拾玖

```
Student(string name,char gender,int age,double w,double h,
        int courseNumber){
      this->name = name;
      this->gender = gender;
      this->age = age;
      this->weight = w;
      height = h;
      this->courseNumber = courseNumber;
      this->courseGrades = new double[this->courseNumber];
}
```

- 在构造函数中，创建动态数组。
- 何时释放动态数组？在变量的生命周期结束的时候

任务肆拾玖

```
Student(string name,char gender,int age,double w,double h,
        int courseNumber){
      this->name = name;
      this->gender = gender;
      this->age = age;
      this->weight = w;
      height = h;
      this->courseNumber = courseNumber;
      this->courseGrades = new double[this->courseNumber];
}
```

- 在构造函数中，创建动态数组。
- 在析构函数中，释放动态数组。

析构函数

- 特殊的成员函数
  - 函数名　　~ 类型的名称
  - 参数列表　　无
  - 返回类型　　无
  - 不可重载　　形式固定

- 在变量消失的时候，由系统自行调用，用于释放变量所占用的相关资源（内存、文件等等）。

析构函数

```
struct Student{

    ~Student(){
        cout << "只能被系统调用的析构函数" << endl;
        delete[] this->courseGrades;
        this->courseGrades = nullptr;
    }
    ……
} ;
```

析构函数

```
struct Student{

    ~Student(){
        cout << "只能被系统调用的析构函数" << endl;
        delete[] this->courseGrades;
        this->courseGrades = nullptr;
    }
    ……
} ;
```

- **函数名称　~Student**
  - **波浪线　~**

析构函数

```
struct Student{

    ~Student(){
        cout << "只能被系统调用的析构函数" << endl;
        delete[] this->courseGrades;
        this->courseGrades = nullptr;
    }
    ……
} ;
```

- **形式固定（不可重载）**
  - **没有参数、没有返回类型**

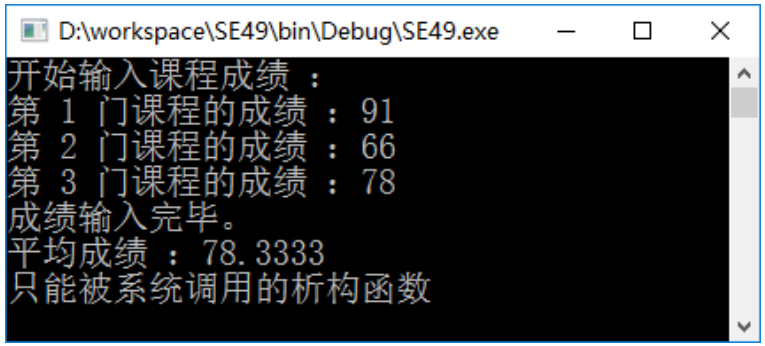
析构函数

```
struct Student{

    ~Student(){
        cout << "只能被系统调用的析构函数" << endl;
        delete[] this->courseGrades;
        this->courseGrades = nullptr;
    }
    ……
} ;
```

- **作用**
  - **（在变量消失时）释放变量（对象）所占用的相关资源**

析构函数

```
struct Student{
    ~Student(){
        cout << "只能被系统调用的析构函数" << endl;
        delete[] this->courseGrades;
        this->courseGrades = nullptr;
    }
    ……
} ;
int main(){
    Student one("mickey",'F',36,50.0,1.66,3);
    one.inputCourseGrades();
    cout << "平均成绩 : " << one.gpa() << endl;
    return 0;
}
```



- **在变量消失的时候，系统自行调用析构函数，释放资源。**

访问控制级别

```
struct Student{
    ……
    double weight;
    ……
    double calBmi(){
      return weight/(height*height);
    }
};

int main(){
    Student one("mickey",'F',36,50.0,1.66,3);
    cout << one.calBmi() << endl;




    return 0;
}
```

访问控制级别

```
struct Student{
    ……
    double weight;
    ……
    double calBmi(){
      return weight/(height*height);
    }
};

int main(){
    Student one("mickey",'F',36,50.0,1.66,3);
    cout << one.calBmi() << endl;  // 输出 mickey 的 bmi 指数

    one.weight = -45.5;
    cout << one.calBmi() << endl;

    return 0;
}
```

访问控制级别

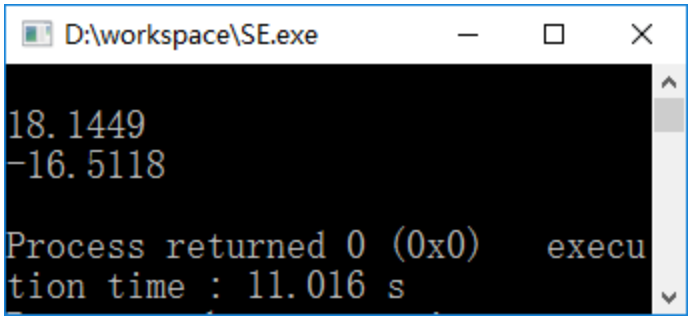
```
struct Student{
    ……
    double weight;
    ……
    double calBmi(){
      return weight/(height*height);
    }
};

int main(){
    Student one("mickey",'F',36,50.0,1.66,3);
    cout << one.calBmi() << endl;  // 输出 mickey 的 bmi 指数

    one.weight = -45.5;
    cout << one.calBmi() << endl;

    return 0;
}
```

编译器会发现问题吗?

错误的赋值

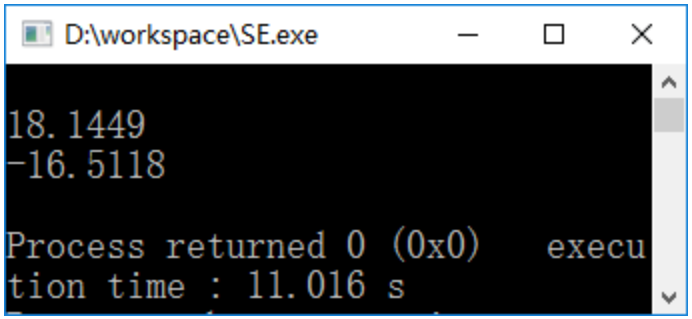
访问控制级别

```
struct Student{
    ……
    double weight;
    ……
    double calBmi(){
      return weight/(height*height);
    }
};

int main(){
    Student one("mickey",'F',36,50.0,1.66,3);
    cout << one.calBmi() << endl;  // 输出 mickey 的 bmi 指数

    one.weight = -45.5;
    cout << one.calBmi() << endl;

    return 0;
}
```



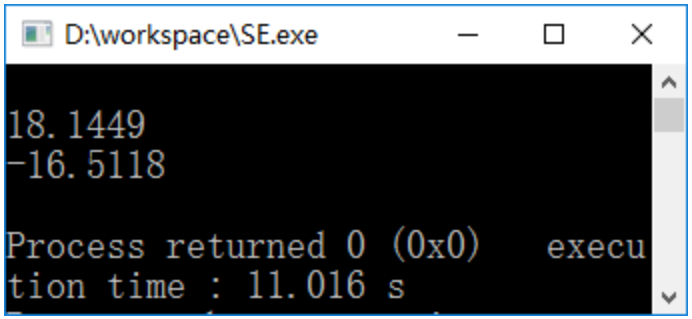
访问控制级别

```
struct Student{
    ……
    double weight;
    ……
    double calBmi(){
      return weight/(height*height);
    }
};

int main(){
    Student one("mickey",'F',36,50.0,1.66,3);
    cout << one.calBmi() << endl;  // 输出 mickey 的 bmi 指数

    one.weight = -45.5;   // 合法的赋值语句
    cout << one.calBmi() << endl;

    return 0;
}
```



访问控制级别

```
struct Student{
    ……
    double weight;
    ……
    double calBmi(){
      return weight/(height*height);
    }
};

int main(){
    Student one("mickey",'F',36,50.0,1.66,3);
    cout << one.calBmi() << endl;  // 输出 mickey 的 bmi 指数

    one.weight = -45.5;  // 合法的赋值语句（应该禁止这种行为）
    cout << one.calBmi() << endl;

    return 0;
}
```



访问控制级别

- 访问控制
  - 防止对任何资源进行未授权的访问，从而使（数据类型的）成员在合法的范围内使用。

- 设置成员的访问级别
  - `private`
  - `public`

访问控制级别

- private
  - 私有的
  - 只能被自身所在类的其他成员访问
- public
  - 公共的
  - 可以被外界不受限制地访问

访问控制级别

```
struct Student{
    ……
    double weight;
    ……
    double calBmi(){
      return weight/(height*height);
    }
};

int main(){
    Student one("mickey",'F',36,50.0,1.66,3);
    cout << one.calBmi() << endl;  // 输出 mickey 的 bmi 指数

    one.weight = -45.5;   // 应该禁止这种行为
    cout << one.calBmi() << endl;

    return 0;
}
```

把 weight 设置为私有成员

设置weight为类型Student的私有成员

```
struct Student{
     string name;
     char   gender;
     int    age;

  private:
     double weight;
     double height;

     double calBmi(){
       return weight/(height*height);
     }

     ……
};
```

- private　关键字　　私有访问控制级别

访问控制级别

SE.cpp - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

Student::

Student(string name, char gender, int age, double w, double h, int courseNumber)

Management

Projects Symbols

Workspace

Start here SE.cpp

```
6   struct Student{
7       string name;
8       char   gender;
9       int    age;
10
11  private:
12      double weight;
13      double height;
14
15      double calBmi(){
16        return weight/(height*heig
17      }
18
19
20      Student(string name,char gen
21         this->name = name;
22         this->gender = gender;
23         this->age = age;
24         this->weight = w;
25         height = h;
26
27      }
28  };
29
30  int main(){
31
32      Student one("mickey",'F',36,50.0,1.66,3);
```

```
error: 'Student::Student(std::string, char, int, double, double, int)' is private
error: within this context
error: 'double Student::calBmi()' is private
error: within this context
error: 'double Student::weight' is private
error: within this context
error: 'double Student::calBmi()' is private
error: within this context
```

Logs & others

Code::Blocks | Search results | Cccc | Build log | Build messages | CppCheck | CppCheck messages | Cscope | Debugger

File	Line	Message
D:\workspace\S...	20	error: 'Student::Student(std::string, char, int, double, double, int)' is private
D:\workspace\S...	32	error: within this context
D:\workspace\S...	15	error: 'double Student::calBmi()' is private

D:\workspace\SE.cpp | Windows (CR+LF) | WINDOWS-936 | Line 20, Column 1 | Insert | Read/Write | default

- 访问控制级别的范围

访问控制级别

- 范围
  - 从访问控制级别设置（冒号后）开始，直到出现新的访问控制级别或达到类定义的尾部为止。

访问控制级别的范围

```
struct Student{
      string name;
      char   gender;
      int    age;

  private:
      double weight;
      double height;


      double calBmi(){
        return weight/(height*height);
      }


      ……
};
```

私有成员

访问控制级别的范围

```
struct Student{
     string name;
     char   gender;
     int    age;

  private:
     double weight;
     double height;

     double calBmi(){
       return weight/(height*height);
     }

     ……
};
```

- 修改 calBmi() 及其后续成员的访问级别为 public 。

访问控制级别的范围

```
struct Student{
      string name;
      char   gender;
      int    age;

   private:
      double weight;
      double height;
   public:
      double calBmi(){
        return weight/(height*height);
      }

      ……
};
```

???

私有成员

公共成员

访问控制级别

- `struct`
  - **成员的默认访问级别** `public`

访问控制级别的范围

```
struct Student{
      string name;
      char   gender;
      int    age;

   private:
      double weight;
      double height;
   public:
      double calBmi(){
        return weight/(height*height);
      }

      ……
};
```

默认访问级别

私有成员

公共成员

访问控制级别的范围

```
struct Student{
      string name;
      char   gender;
      int    age;

   private:
      double weight;
      double height;

   public:
      double calBmi(){
        return weight/(height*height);
      }

      ……
};
```

公共成员

私有成员

公共成员

访问控制级别

- private
  - 成员变量
  - 内部成员函数（仅供类内部调用）

- public
  - 提供外界使用的接口（函数）
    - 函数的形式固定不变

自定义数据类型：struct

```
struct Student{
   double calBmi(){
     return weight/(height*height);
   }
   Student(string name,char gender,int age,double w,double h){
      this->name = name;
      this->gender = gender;
      this->age = age;
      this->weight = w;
      height = h;
   }
   private:
     string name;
     char   gender;
     int    age;
     double weight;
     double height;
};
```

自定义数据类型：struct

- 修改私有成员的方法
  - 提供 public 成员函数（公共接口）

```
struct Student{




    ……

    private:
      double weight;
      ……
};
```

自定义数据类型：struct

- 修改私有成员的方法
  - 提供 public 成员函数（公共接口）

```
struct Student{
    void setWeight(double value){
        this->weight = value;
    }
    ……

    private:
     double weight;  // 体重可以被修改
      ……
};
```

通过公共接口修改成员变量的值

```
struct Student{
  void setWeight(double value){
     this->weight = value;
  }
  ……
  private:
    double weight;  // 体重可以被修改
    ……
};
int main(){
   Student one("mickey",'F',36,50.0,1.66,3);
   cout << one.calBmi() << endl;

                                // 修改体重
   cout << one.calBmi() << endl;

   return 0;
}
```

通过公共接口修改成员变量的值

```
struct Student{
  void setWeight(double value){
     this->weight = value;
  }
  ……
  private:
    double weight;  // 体重可以被修改
    ……
};
int main(){
   Student one("mickey",'F',36,50.0,1.66,3);
   cout << one.calBmi() << endl;

   one.setWeight(45.6);    // 修改体重
   cout << one.calBmi() << endl;

   return 0;
}
```

使用访问控制实现数据的“封装”

```
struct Student{
  void setWeight(double value){
     this->weight = value;
  }
  ……
  private:
    double weight;
    ……
};
```

```
struct Student{
   void setWeight(double value){
      if(value > 0 && value < 200)
          this->weight = value;
   }
   ……
   private:
     double weight;
     ……
};
```

```
int main(){
   Student one("mickey",'F',36,50.0,1.66,3);
   cout << one.calBmi() << endl;

   one.setWeight(45.6);    // 修改体重
   cout << one.calBmi() << endl;

   return 0;
}
```

使用访问控制实现数据的“封装”

```
struct Student{
  void setWeight(double value){
     this->weight = value;
  }
  ……
  private:
    double weight;
    ……
};
```

```
struct Student{
   void setWeight(double value){
      if(value > 0 && value < 200)
          this->weight = value;
   }
   ……
   private:
     double weight;
     ……
};
```

- 公共成员函数
  - 函数声明不变
  - 具体实现代码的变动不影响外部（调用）代码

使用访问控制实现数据的“封装”

```
struct Student{
  void setWeight(double value){
     this->weight = value;
  }
  ……
  private:
    double weight;
    ……
};
```

```
struct Student{
   void setWeight(double value){
      if(value > 0 && value < 200)
          this->weight = value;
   }
   ……
   private:
     double weight;
     ……
};
```

```
int main(){
   Student one("mickey",'F',36,50.0,1.66,3);
   cout << one.calBmi() << endl;

   one.setWeight(45.6);
   cout << one.calBmi() << endl;

   return 0;
}
```

外部代码不变

类

- struct 与 class 的差异
  - 由 struct 创建的数据类型
    - 成员的默认访问级别 public
  - 由 class 创建的数据类型
    - 成员的默认访问级别 private
- 唯一的区别

类

```
struct Person{
   double calBmi(){
     return weight/(height*height);
   }
   Person(string name,
          double w,double h){
      this->name = name;
      this->weight = w;
      height = h;
   }

   private:
     string name;
     double weight;
     double height;
};
```

```
class Person{
    string name;
    double weight;
    double height;

  public:
    double calBmi(){
      return weight/(height*height);
    }
    Person(string name,
           double w,double h){
      this->name = name;
      this->weight = w;
      height = h;
   }
};
```

在类外实现成员函数

```
struct Student{
   double calBmi(){
     return weight/(height*height);
   }
   Student(string name,char gender,
           int age,double w,double h){
      this->name = name;
      this->gender = gender;
      this->age = age;
      this->weight = w;
      height = h;
   }
   private:
     string name;
     char   gender;
     int    age;
     double weight;
     double height;
};
```

```
struct Student{
   double calBmi();
   Student(string name,char gender,
           int age,double w,double h);
   private:
     string name;
     char   gender;
     int    age;
     double weight;
     double height;
};

double Student::calBmi(){
   return weight/(height*height);
}

Student::Student(string name,char gender,
                 int age,double w,double h){
     this->name = name;
     this->gender = gender;
     this->age = age;
     this->weight = w;
     height = h;
}
```

在类外实现成员函数

- Student.h
  - 类型 Student 的定义

- Student.cpp
  - 类型 Student 成员函数的具体实现

在类外实现成员函数：多文件

Student.h - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

Student.h

```
1  #ifndef STUDENT_H_INCLUDED
2  #define STUDENT_H_INCLUDED
3
4  struct Student{
5      double calBmi();
6      Student(string name,char gender,
7              int age,double w,double h);
8      private:
9        string name;
10       char   gender;
11       int    age;
12       double weight;
13       double height;
14 };
15
16
17 #endif // STUDENT_H_INCLUDED
18
```

Student.cpp

```
1  #include "Student.h"
2
3  double Student::calBmi(){
4      return weight/(height*height);
5  }
6
7  Student::Student(string name,char gender,
8                   int age,double w,double h){
9      this->name = name;
10     this->gender = gender;
11     this->age = age;
12     this->weight = w;
13     height = h;
14 }
15
```

Logs & others

Code::Blocks | Search results | Cccc | Build log | Build messages | CppCheck | CppCheck messages | Cscope | Debugger | DoxyBlocks | Fortran

File	Line	Message
		=== Build file: "no target" in "no project" (compiler: unknown) ===
		=== Build finished: 0 error(s), 0 warning(s) (0 minute(s), 1 second(s)) ===

D:\workspace\Student.h | Windows (CR+LF) | WINDOWS-936 | Line 20, Column 1 | Insert | Read/Write | default

待续……