第 3 章 使用批量数据

刘 卉

huiliu@fudan.edu.cn

前　言

- 任务/示例
  - 读取学生的考试和家庭作业成绩, 计算最终成绩.
  - 学习如何存储所有数据, 即使开始并不知道有多少数据需要存储.

3.1 计算学生成绩

- 课程成绩

  0.2 * midterm + 0.4 * final + 0.4 * homework

- 标准库
  - <ios>: 输入输出库, 定义了表示流长度的streamsize类型.
  - <iomanip>: 定义了控制符setprecision, 设置输出的精度.
- 输入操作符>>
  - 左操作数为运算结果⇨链式输入

```
cin >> midterm >> final;
```

- 默认初始化——取决于变量的类型

1) 内置类型的局部变量：若没有显式初始化，则初值不确定，只能用在'='的左边.

```
e.g.    int a, b;
        b = a + 1;   //错误！ a没有值.
```

2) 类的对象：类会说明没有指定初值时如何初始化.

```
e.g.    string name1, name2;   //name1和name2均为""
        name2 = name1 + "abc"   //name2变为"abc"
```

没有显式初始化一个string变量时，会被string类隐式初始化为空字符串.

输出

- setprecision: 控制cout的输出精度(<iomanip>).
- 好习惯: 及时恢复cout的精度.
  - 调用cout的成员函数precision()保存原始精度.

```
streamsize prec = cout.precision(); //streamsize⇨<ios>
cout << "Your final grade is " << setprecision(3)
     << 0.2*midterm + 0.4*final + 0.4*sum/count
     << setprecision(prec) << endl;
```

3.1.1 检测输入的结束

```
while (cin >> x) {……}
```

- 隐式使用了istream类的对象cin作为while语句的条件式.
  - istream类提供了一种转换机制: cin⇨bool值.

    读取成功: true　　读取失败: false
  - 用cin作条件式: 检测从cin的读取是否成功.

- 从cin读取失败的三种情况:
  1) 遇到输入文件的结束标志: Ctrl+z, Ctrl+d(Mac)
  2) 输入类型与要求类型不匹配;
  3) 输入设备的硬件故障.
  - 一旦读取流失败，接下来所有读取流操作都会失败⇨必须重设这个流.
    - Windows: cin.clear()
    - MacOS: cin.clear() + clearerr(stdin)

3.1.2 循环不变式

- We have read *count* grades so far, *sum* is the sum of the first count grades.
  - 在对条件式求值前, 不变式为真.
  - 对条件式求值后:

1) 如果成功读取,
   - 增加count→使得不变式的第一部分为真
   - 执行sum += x→使得不变式的第二部分为真

2) 如果读取失败
   - 并没有读入新的数据, 不变式依然为真.

```
while (cin >> x)
{
    ++count;
    sum += x;
}
```

```
int main()
{
    ……  // ask for and read the student's name, the midterm and final grades
    // ask for the homework grades
    cout << "Enter all your homework grades, followed by end-of-file: ";
    // the number and sum of grades read so far
    int count = 0;
    double sum = 0;
    double x; // a variable into which to read
    while (cin >> x) {  // invariant: we have read count grades so far, and
        ++count;        // sum is the sum of the first count grades
        sum += x;
    }
    // write the result
    streamsize prec = cout.precision();
    cout << "Your final grade is " << setprecision(3)
         << 0.2 * midterm + 0.4 * final + 0.4 * sum / count
         << setprecision(prec) << endl;

    return 0;
}
```

3.2 使用中值取代平均值

- 缺陷
  - 不能保存每次家庭作业成绩.
- 用中值取代平均值
  - 使用中值, 不会因为几次不好的成绩影响最终成绩.
- 必须从根本上改变程序
  - 保存数目不确定的家庭作业成绩→排序→取中值.

3.2.1 用vector保存数据集

- vector——标准库提供的一种容器类型
  - 保存多个相同类型的值, 其大小根据需要增长, 且能高效获取每个元素.

```
// revised version
vector<double> homework;
double x;
// invariant: homework contains all the homework grades read
// so far
while (cin >> x)
    homework.push_back(x);
```

- vector是一个模板类
  - 定义vector对象时, 必须指定它所包含值的类型.

```
vector<double> homework;
```

  - push_back成员函数
    - 在当前vector对象的尾部追加一个新元素; 同时, 使vector对象的大小增1.

```
homework.push_back(x);
```

3.2.2 产生输出

- 找到中值

1) 求homework的长度

```
typedef vector<double>::size_type vec_sz;
vec_sz size = homework.size();
```

- 再次强调: 使用标准库定义的类型表示容器的大小.
- 局部变量size, vector的成员函数size(): 所处的作用域不同⇨同名但不冲突.

2) 检测homework是否包含数据

```
if (size == 0) {
    cout << endl << "You must enter your grades.  "
         "Please try again." << endl;
    return 1;
}
```

3) 排序

指定排序的元素范围

```
sort(homework.begin(), homework.end());
```

- sort函数在<algorithm>中定义, 非降序排列.
- homework.begin(): 返回一个值, 指向homework的第一个元素.
- homework.end(): 指向homework末尾元素的下一个位置.
- 原地排序: 移动元素到合适位置, homework发生改变.

4) 找到中间元素

```
vec_sz mid = size/2;   // vector<double>::size_type mid = size/2
double median;
median = size%2==0?(homework[mid]+homework[mid-1])/2:homework[mid];
```

- 访问homework的元素: 索引

- 计算最终成绩, 输出.
- 无需担心如何获得内存来存储所有作业成绩
  - 标准库已经为我们做好了这些工作!


```
int main()
{
    ……
    vector<double> homework;
    double x;
    // invariant: homework contains all the homework grades read so far
    while (cin >> x)
        homework.push_back(x);
    // check that the student entered some homework grades
    typedef vector<double>::size_type vec_sz;
    vec_sz size = homework.size();
    if (size == 0) {
        cout << endl << "You must enter your grades.  "
             "Please try again." << endl;
        return 1;
    }
    sort(homework.begin(), homework.end()); // sort the grades
    vec_sz mid = size/2; // compute the median homework grade
    double median;
    median = size % 2==0 ? (homework[mid]+homework[mid-1])/2: homework[mid];
    ……  // compute and write the final grade
    return 0;
}
```

3.2.3 值得注意的地方

- vector不会检查索引是否有效
- 程序性能很好
  - 与静态分配内存相比，使用vector对象可以根据输入，动态增长大小, 且性能不比前者差.

小　结

- vector类型——<vector>
  - vector<T> v: 创建一个空的vector, 包含的元素类型为T
  - vector<T>::size_type: 保证可以存放最大可能的vector元素个数
  - v.begin(): 返回一个值, 指向v中的第一个元素
  - v.end(): 指向v中最后一个元素的下一个位置
  - v.push_back(e): 把用e初始化的元素追加到v末尾
  - v[i]: 返回保存在位置i的值
  - v.size(): 返回v中元素个数

- 用输入表达式作为循环条件式
  - while(cin >> x)
- 其它库函数
  - sort(b,e): 对区间[b,e)中的元素按照非降序排列(<algorithm>)
  - s.precision(n)
  - setprecision(n): <iomanip>
  - streamsize: setprecision()需要的参数类型, precision()返回的类型. 定义在<ios>中.