C++ 程序设计 I

徐东

xu.dong.sh@outlook.com

信息与计算科学

数学系

上海师范大学

2018 年 8 月 28 日

内容

问题

- 问题 1
  - 计算平均值
- 问题 2
  - 找出超出均值的全部数据值
  - 计算标准差
  - 统计投票数

问题

- 问题 1
  - 计算平均值
- 问题 2
  - 找出超出均值的全部数据值
  - 计算标准差
  - 统计投票数
- 两组问题之间的差异

问题

- 问题 1
  - 计算平均值
- 问题 2
  - 找出超出均值的全部数据值
  - 计算标准差
  - 统计投票数
- 两组问题之间的差异
  - 第 2 组问题　　需要保留所有数据

问题

- 从 100 个数据中，找出所有超过平均值的数据。

问题

- 从 100 个数据中，找出所有超过平均值的数据。

- 在程序执行的过程中，需要存储大量的数据。

问题

- 从 100 个数据中，找出所有超过平均值的数据。
- 在程序执行的过程中，需要存储大量的数据。
- 声明 100 个变量　　不现实的做法

问题

- 从 100 个数据中，找出所有超过平均值的数据。
- 在程序执行的过程中，需要存储大量的数据。
- 声明 100 个变量　　不现实的做法
- 高效、有条理的处理方法

问题

- 从 100 个数据中，找出所有超过平均值的数据。
- 在程序执行的过程中，需要存储大量的数据。
- 声明 100 个变量　　不现实的做法
- 高效、有条理的处理方法：数组

问题

- 从 100 个数据中，找出所有超过平均值的数据。
- 在程序执行的过程中，需要存储大量的数据。
- 声明 100 个变量　　不现实的做法
- 高效、有条理的处理方法：数组
- 数组

问题

- 从 100 个数据中，找出所有超过平均值的数据。
- 在程序执行的过程中，需要存储大量的数据。
- 声明 100 个变量　　不现实的做法
- 高效、有条理的处理方法：数组
- 数组
  - 数据结构

问题

- 从 100 个数据中，找出所有超过平均值的数据。
- 在程序执行的过程中，需要存储大量的数据。
- 声明 100 个变量　　不现实的做法
- 高效、有条理的处理方法：数组
- 数组
  - 数据结构
  - 存储一个元素个数固定、元素类型相同的数据集（有序集）

数组声明

- 数组声明语句

数据类型 数组名[n] = {初始值_1, 初始值_2, ···, 初始值_n};

数组声明

- 数组声明语句

$$\text{数据类型}\ \ \text{数组名[n]} = \{\text{初始值\_}1,\ \text{初始值\_}2,\ \cdots,\ \text{初始值\_}n\};$$

- 数组名　　遵循 $C++$ 标识符命名规则

数组声明

- 数组声明语句

$$\text{数据类型}\ \ \text{数组名[n]} = \{\text{初始值\_}1,\ \text{初始值\_}2,\ \cdots,\ \text{初始值\_}n\};$$

- 数组名　　遵循 $C++$ 标识符命名规则
- 中括号 [ ]　　表示声明的是数组

数组声明

- 数组声明语句

数据类型 数组名[n] = {初始值_1, 初始值_2, ···, 初始值_n};

- 数组名　　遵循 $C++$ 标识符命名规则
- 中括号 [ ]　　表示声明的是数组
- n　　必须是整型常量

数组声明

- 数组声明语句

$$\text{数据类型}\ \ \text{数组名}[\mathbf{n}]\ =\ \{\text{初始值\_}1,\ \text{初始值\_}2,\ \cdots,\ \text{初始值\_}n\};$$

- 数组名　　遵循 $C++$ 标识符命名规则
- 中括号 [ ]　　表示声明的是数组
- **n**　　必须是整型常量(数组大小，数组长度)

数组声明

- 数组声明语句

数据类型 数组名[n] = {初始值_1, 初始值_2, ···, 初始值_n};

数组声明

- 数组声明语句

数据类型 数组名[n] = {初始值_1, 初始值_2, ···, 初始值_n};

- 初始值（列表）必须放置在大括号（{}）中

数组声明

- 数组声明语句

$$\text{数据类型}\ \ \text{数组名[n]} = \{\text{初始值\_}1, \text{初始值\_}2, \cdots, \text{初始值\_}n\};$$

- 初始值（列表）必须放置在大括号（{}）中
- 初始值必须以逗号（,）分隔

数组声明

- 数组声明语句

数据类型 数组名[n] = {初始值_1, 初始值_2, ···, 初始值_n};

- 初始值（列表）必须放置在大括号（{}）中
- 初始值必须以逗号（,）分隔
- 初始值的个数 ⩽ 数组长度

数组声明

- 数组声明语句

$$\text{数据类型}\ \ \text{数组名[n]} = \{\text{初始值\_}1,\ \text{初始值\_}2,\ \cdots,\ \text{初始值\_}n\};$$

- 初始值（列表）必须放置在大括号（{}）中
- 初始值必须以逗号（,）分隔
- 初始值的个数 $\leqslant$ 数组长度（不足部分自动赋零）

数组声明

- 数组声明语句

$$\text{数据类型}\ \ \text{数组名}[\ ] = \{\text{初始值\_}1,\ \text{初始值\_}2,\ \cdots,\ \text{初始值\_}n\};$$

- 给出全部初始值　　可省略数组大小

数组声明语句

```
1    int x[10] = {1,2,3,4,5,6,7,8,9,10};
2    int y[10] = {1,2,3};
3    int z[] = {1,2,3};
```

数组声明语句

```
1    int x[10] = {1,2,3,4,5,6,7,8,9,10};
2    int y[10] = {1,2,3};
3    int z[] = {1,2,3};
```

- 一维静态数组

数组声明语句

```
1    int x[10] = {1,2,3,4,5,6,7,8,9,10};
2    int y[10] = {1,2,3};
3    int z[] = {1,2,3};
```

- 一维静态数组
  - 声明之后，数组大小固定。

数组声明语句

```
1    int x[10] = {1,2,3,4,5,6,7,8,9,10};
2    int y[10] = {1,2,3};
3    int z[] = {1,2,3};
```

- 一维静态数组
  - 声明之后，数组大小固定。
  - 类似 向量

处理数组（元素）的方式

- 访问数组元素的方式

数组名[索引值]

- 数组名后的中括号（[]）　　不能省略
- 索引值
  - 代表元素在数组中的位置（必须出现在中括号 [] 中）
  - 从 0 开始计数
  - 整数值（整型常量、整型变量、计算结果为整型的表达式）

一维静态数组的声明和使用

```
1       int x[10] = {0};
2
3       for(int i = 0; i < 10; ++i){
4           cin >> x[i];
5       }
```

一维静态数组的声明和使用

```
1    int x[10] = {0};
2
3    for(int i = 0; i < 10; ++i){
4        cin >> x[i];
5    }
```

- （数组）一次声明，逐个使用（数组元素）。

数组元素的索引值不能超过合法范围

```
1  int x[10] = {0};
2  for(int i = 0; i < 10; ++i){
3     cin >> x[i];
4  }
```

```
int x[10] = {0};
for(int i = 0; i <= 10; ++i){
    cin >> x[i];
}
```

数组元素的索引值不能超过合法范围

```
1  int x[10] = {0};
2  for(int i = 0; i < 10; ++i){
3     cin >> x[i];
4  }
```

```
int x[10] = {0};
for(int i = 0; i <= 10; ++i){
    cin >> x[i];
}
```

数组元素的索引值不能超过合法范围

```
1  int x[10] = {0};
2  for(int i = 0; i < 10; ++i){
3     cin >> x[i];
4  }
```

```
int x[10] = {0};
for(int i = 0; i <= 10; ++i){
    cin >> x[i];
}
```

- 越界访问

数组元素的索引值不能超过合法范围

```
1  int x[10] = {0};
2  for(int i = 0; i < 10; ++i){
3     cin >> x[i];
4  }
```

```
int x[10] = {0};
for(int i = 0; i <= 10; ++i){
    cin >> x[i];
}
```

- 越界访问
  - 数组元素的索引值超出合法范围（存在安全隐患，bug）

问题

- 从 100 个数据中，找出所有超过平均值的数据。

- 提示
  - 区分“数组”和“数组元素”
  - (数组) 一次声明，逐个使用 (数组元素)。
  - 数组大小固定 (整型常量)
  - 索引值从 0 开始 (整型值)
  - 避免越界访问 (编译器不报错)

问题

- 模拟发 4 张扑克牌

- 提示
  - 数组大小　　52
  - 0 ~ 12　　13 张黑桃
  - 13 ~ 25　　13 张红桃
  - 26 ~ 38　　13 张方块
  - 39 ~ 51　　13 张梅花

  - cardNumber / 13　　决定牌的花色
  - cardNumber % 13　　决定具体花色中的哪张牌

模拟发牌 I

```
#include<iostream>
#include<cstdlib>
#include<ctime>
using namespace std;

int main(){
    int deck[52] = {0};
    string suits[] = {”Spades”,”Hearts”,”Diamonds”,”
        Clubs”};
    string ranks[] = {”Ace”,”2”,”3”,”4”,”5”,”6”,”7”,”8”,
        ”9”,”10”,
”Jack”,”Queen”,”King”};
```

模拟发牌 II

```
11
12      for(int i = 0; i < 52; ++i){
13          deck[i] = i;
14      }
15
16      //洗牌
17      srand(time(0));
18      for(int i = 0; i < 52; ++i){
19          int index = rand() % 52;
20          int temp = deck[i];
21          deck[i] = deck[index];
22          deck[index] = temp;
23      }
```

模拟发牌 III

```
24
25      //发牌
26      for(int i = 0; i < 4; ++i){
27          int cardNumber = deck[i];
28          string mysuit = suits[cardNumber / 13];
29          string myrank = ranks[cardNumber % 13];
30          cout << mysuit << " : " << myrank << endl;
31      }
32
33      return 0;
34  }
```

问题

- 统计每个小写字母出现的次数

- 提示
  - 随机生成 100 个小写英文字母
  - 创建一个具有 26 个 int 值的数组 counts
    - 每个元素存放一个字母出现的次数
    - counts[0] 记录'a' 出现的次数
    - counts[1] 记录'b' 出现的次数
    - ...
    - counts[25] 记录'z' 出现的次数

问题

- 计算多项式的值
- 秦九韶算法

问题

- 数组的复制

问题

- 数组的复制
- 函数 memcpy()

问题

- 数组的复制
- 函数 memcpy()

```
1    int a[5] = {1, 2, 3, 4, 5};
2    int b[10] = {0};
3
4    memcpy(b, a, sizeof(a));
```

问题

- 数组的复制

- 函数 memcpy()

```
1    int a[5] = {1, 2, 3, 4, 5};
2    int b[10] = {0};
3
4    memcpy(b, a, sizeof(a));
```

- 把数组 a 中指定字节大小的数据复制到数组 b 中

Q & A