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

- 某人是否已经是候选人（出现在候选人名单中）
- 是否有人考满分（或者任意一个指定的成绩）
- ...

问题

- 某人是否已经是候选人（出现在候选人名单中）
- 是否有人考满分（或者任意一个指定的成绩）
- ...

- 查找

问题

- 某人是否已经是候选人（出现在候选人名单中）
- 是否有人考满分（或者任意一个指定的成绩）
- ...

- 查找
  - 在数组中寻找特定元素的过程

问题

- 某人是否已经是候选人（出现在候选人名单中）
- 是否有人考满分（或者任意一个指定的成绩）
- ...

- 查找
  - 在数组中寻找特定元素的过程

- 线性查找
- 二分查找

线性查找

- 线性查找
  - 元素逐一比较
  - 匹配成功 (索引值)　返回与查询值匹配的元素在数组中的位置
  - 匹配失败　-1

线性查找

- 线性查找
  - 元素逐一比较
  - 匹配成功　　返回与查询值匹配的元素在数组中的位置(索引值)
  - 匹配失败　　-1

```
1    int hit_index = -1;
2    for(int i = 0; i < N; ++){
3        if(x == data[i]){
4            hit_index = i;
5        }
6    }
```

线性查找 I

```
1  int main(){
2     const int N = 9;
3     int data[N] = {8,10,1,5,16,61,9,11,-1};
4
5     int x = 0;
6     cin >> x;
7
8     int hit_index = -1;
9     for(int i = 0; i < N; ++){
10       if(x == data[i]){ hit_index = i; }
11    }
```

线性查找 II

```
12
13      if(hit_index >= 0){
14          cout << x << "出现在数组␣data␣的第␣"
15              << hit_index + 1 <<"␣位置上␣"
16              << data[hit_index] << endl;
17      }
18      else{
19          cout << "没有在数组␣data␣中找到␣" << x << endl;
20      }
21
22      return 0;
23  }
```

折半查找

```
1    const int N = 9;
2    int data[N] = {-1,1,5,8,9,10,11,16,61};
3
4    ......
```

- 排序数组的查找

折半查找

```
1    const int N = 9;
2    int data[N] = {-1,1,5,8,9,10,11,16,61};
3
4    ......
```

- 排序数组的查找
- 折半查找

折半查找

```
1    const int N = 9;
2    int data[N] = {-1,1,5,8,9,10,11,16,61};
3
4    ......
```

- 排序数组的查找
- 折半查找
  - 数组中的元素必须已经按从小到大的顺序排列（反之亦可）
  - 每次比较后，数组的搜索范围缩小一半。
  - 折半查找比线性查找更高效（针对排序数组）

折半搜索 I

```
1  int main(){
2      const int length = 9;
3      int data[length] = {-1,1,5,8,9,10,11,16,61};
4
5      int x = 0;
6      cout << "请输入查询值 : ";
7      cin >> x;
8
9      cout << "开始搜索..." << endl;
10     int hit_index = -1;
11     int left_index = 0;
12     int right_index = length - 1 ;
13
```

折半搜索 II

```
14      while(left_index <= right_index){
15          int mid_index = (left_index+right_index)/2;
16          if(x == data[mid_index]){
17              hit_index = mid_index;
18              left_index = right_index+1; //结束循环
19          }
20          else if( x > data[mid_index] ){
21              left_index = mid_index+1; //准备搜索后半段
22          }
23          else{
24              right_index = mid_index-1; //准备搜索前半段
25          }
26      }
```

折半搜索 III

```
27      cout << "搜索完毕!" << endl;
28
29      if(hit != -1){
30          cout << x << "␣出现在目标数据组␣-1,1,5,8,9,10,11,16,61␣"
31              << "中的第␣" << hit_index + 1
32              << "␣个位置上。" <<endl;
33      }
34      else{
35          cout << x << "␣没有出现在目标数据组中。" << endl;
36      }
37
38      return 0;
39  }
```

break 语句的作用

- break 语句对查询结果的影响

```
int data[] = {8,10,1,5,16,61,9,11,1};
int hit_index = -1;
for(int i = 0; i < N; ++){
    if(x == data[i]){
        hit_index = i;
        break;
    }
}
```

break 语句的作用

- break 语句对查询结果的影响

```
int data[] = {8,10,1,5,16,61,9,11,1};
int hit_index = -1;
for(int i = 0; i < N; ++){
    if(x == data[i]){
        hit_index = i;
        break;
    }
}
```

- 排序
  - 冒泡排序
  - 选择排序
  - ...

冒泡排序

```
const int size = 6;
int a[size] = 9, 8, 5, 4, 2, 0;

for(int i = 0; i < size - 1; ++i){
    for(int j = 0; j < size - 1 - i; ++j){
        if(a[j] > a[j + 1]){
            int temp = a[j];
            a[j] = a[j + 1];
            a[j + 1] = temp;
        }
    }
}
```

选择排序

```
const int size = 6;
int data[size] = 9,8,5,4,2,0;

for(int i = 0; i < size - 1; ++i){
    int  min_index = i;
    for(int j = i + 1; j < size; ++j){
        if(data[j] < data[min_index]){
            min_index = j;
        }
    }
    int temp = data[i];
    data[i] = data[min_index];
    data[min_index] = temp;
}
```

快速排序

- 基本思想
  1. 选择分界值并据此将数组分成两部分
  2. 对每一部分重复上述过程直至整个数组被排序完毕为止

- 具体实现

  $sort($待排序的起始位置，待排序的结束位置$);$

- 数组元素按从小到大的顺序排列（默认排序方式）
- STL 函数　　$\#include < algorithm >$

sort() 排序

```
#include <algorithm>
using namespace std;
int main(){
    int a[10] = {1, 3, -1, 88, -3, 6, 8, 2};

    sort(a, a + 10);

    for(int i = 0; i < 10; ++i){
       cout << a[i] << endl;
    }
    return 0;
}
```

sort() 排序

```
#include <algorithm>
using namespace std;
int main(){
    int a[10] = {1, 3, -1, 88, -3, 6, 8, 2};

    sort(a, a + 10, greater<int>()); //逆序排列(从大到小)

    for(int i = 0; i < 10; ++i){
        cout << a[i] << endl;
    }
    return 0;
}
```

对随机数进行排序 I

```
1  #include <iostream>
2  #include <stdlib.h>
3  #include <time.h>
4  #include <algorithm>
5  using namespace std;
6
7  int main(){
8      const int length = 9;
9      int data[length] = {0};
10
11     srand(time(0));
```

对随机数进行排序 II

```
12      for(int i = 0 ; i < length; ++i){
13          data[i] = rand() % 1000;
14      }
15
16      for(int i = 0; i < length; ++i){
17          cout << data[i] << "\t";
18      }
19      cout << endl;
20
21      cout << "
            **************************************************
            " << endl;
```

对随机数进行排序 III

```
22
23      sort(data, data + length);
24
25
26      for(int i = 0; i < length; ++i){
27          cout << data[i] << "\t";
28      }
29      cout << endl;
30
31      return 0;
32  }
```

Q & A