

第 4 课

VECTOR, SET, MAP

薛浩

xuehao0618@outlook.com

阅读

- Programming Abstraction in C++ *Chapter 5*
- CS106B, Summer Quarter 2022
- C++ Tutorials - cplusplus.com

今日话题

- 抽象数据类型
- Vector
- Set
- Map

作业提交

作业 1 建议本周完成

Make slow and steady process each day
—— Keith Schwarz

回顾

以下代码是否违反了我们强调的某些规则？

```
1 int main() {  
2     string s;  
3     cout << s << endl;    // Output: ???  
4  
5     char x = s[-1];  
6     cout << x << endl;    // Output: ???  
7 }
```

抽象数据类型

- 一种定义好的数据类型
- 以某种特定的形式存储数据、管理数据
- 隐藏了内部的实现细节

斯坦福抽象数据类型

本课程将带大家学习使用斯坦福库定义的几种常用的抽象数据类型

**Stanford
抽象数据类型**

**Ordered
Container**

Vector

Grid

Stack

Queue

**Unordered
Container**

Set

Map

Lexicon

斯坦福库

今日话题

- 抽象数据类型
- Vector
- Set
- Map

VECTOR

一组相同类型数据的有序集合。

Vector 有如下几个优点：

- 分配的大小可以任意的改变
- 可以方便地查询容器的大小
- 支持添加和删除元素
- 索引越界有检查机制

不仅可以存储任何基本数据类型，还能存放包括自身的其他容器类型。

需要明确定义元素的基类型

- `Vector<int>` 元素只能为整型
- `Vector<string>` 元素只能为字符串

```
1 #include "vector.h" // Stanford Cppplib for Vector
2 Vector<int> vec;      // declare an empty vector
```

斯坦福库

思考：以下代码有什么问题

```
1 Vector<int> v = {1, 2};  
2 for (int i = 0; i < v.size(); i++) {  
3     v.add(randomInteger(0, 100));  
4 }
```

RANGE-BASED FOR LOOP

```
1 for(type variable: collection) {  
2     body of the loop  
3 }
```

RANGE-BASED FOR LOOP

```
1 Vector<string> v = { "A", "B", "C" };
2
3 for (int i = 0; i < v.size(); i++) {
4     cout << v[i] << endl;
5 }
6
7 for (string elem: v) {
8     cout << elem << endl;
9 }
```

 小试牛刀：编写 MaxPoint 函数

PASS BY REFERENCE

```
1 void dream(Vector<int> numbers) {
2     numbers[1] = 1963;
3 }
4
5 int main() {
6     Vector<int> values = { 1929, 1955, 1964 };
7     dream(values);
8     cout << values << endl;
9     return 0;
10 }
```



```
int main() {  
    Vector<int> values = { 1929, 1955, 1964 };  
    dream(values);  
    cout << values << endl;  
    return 0;  
}
```

```
int main() {  
    Vector<int> values = { 1929, 1955, 1964 };  
    dream(values);  
    cout << values << endl;  
    return 0;  
}
```

1929	1955	1964
------	------	------

values

```
int main() {  
    Vector<int> values = { 1929, 1955, 1964 };  
    dream(values);  
    cout << values << endl;  
    return 0;  
}
```

1929	1955	1964
------	------	------

values

```
int main() {  
    Vector<int> values = { 1929, 1955, 1964 };  
    dream(values);  
    cout << values << endl;  
    return 0;  
}
```

1929 1955 1964

1929	1955	1964
------	------	------

numbers

```
void dream(Vector<int> numbers) {  
    numbers[1] = 1963;  
}
```

1929 1955 1964

1929	1955	1964
------	------	------

numbers

```
void dream(Vector<int> numbers) {  
    numbers[1] = 1963;  
}
```

1929 1963 1964

1929	1963	1964
------	------	------

numbers

```
void dream(Vector<int> numbers) {  
    numbers[1] = 1963;  
}
```

1929 1963 1964

1929	1963	1964
------	------	------

numbers

```
void dream(Vector<int> numbers) {  
    numbers[1] = 1963;  
}
```


1929	1955	1964
------	------	------

values

```
int main() {  
    Vector<int> values = { 1929, 1955, 1964 };  
    dream(values);  
    cout << values << endl;  
    return 0;  
}
```



```
int main() {  
    Vector<int> values = { 1929, 1955, 1964 };  
    dream(values);  
    cout << values << endl;  
    return 0;  
}
```

```
int main() {  
    Vector<int> values = { 1929, 1955, 1964 };  
    dream(values);  
    cout << values << endl;  
    return 0;  
}
```

1929	1955	1964
------	------	------

values

```
int main() {  
    Vector<int> values = { 1929, 1955, 1964 };  
    dream(values);  
    cout << values << endl;  
    return 0;  
}
```

1929	1955	1964
------	------	------

values

```
int main() {  
    Vector<int> values = { 1929, 1955, 1964 };  
    dream(values);  
    cout << values << endl;  
    return 0;  
}
```

1929

1955

1964



```
int m  
V  
d  
c  
r  
}  
}
```

numbers

```
void dream(Vector<int>& numbers) {  
    numbers[1] = 1963;  
}
```

1929	1955	1964
------	------	------

numbers

```
int
```

```
m
```

```
V
```

```
d
```

```
c
```

```
r
```

```
void dream(Vector<int>& numbers) {
```

```
    numbers[1] = 1963;
```

```
}
```


1929	1963	1964
------	------	------

numbers

```
int
```

```
main
```

```
void
```

```
dream
```

```
vector
```

```
numbers
```

```
void dream(Vector<int>& numbers) {
```

```
    numbers[1] = 1963;
```

```
}
```

```
}
```

1929

1963

1964



```
int m  
V  
d  
c  
r  
}  
}
```

numbers

```
void dream(Vector<int>& numbers) {  
    numbers[1] = 1963;  
}
```

1929	1963	1964
------	------	------

values

```
int main() {  
    Vector<int> values = { 1929, 1955, 1964 };  
    dream(values);  
    cout << values << endl;  
    return 0;  
}
```

 小试牛刀：使用引用参数优化 MaxPoint 函数

 课后练习：自行学习 Grid 数据类型

今日话题

- 抽象数据类型
- ~~Vector~~
- Set
- Map

生词本

```
1 Vector<string> wordlist;
2
3 while (true) {
4     string word = getLine("Enter your word: ");
5     if (word == "")
6         break;
7     wordlist.add(word);
8 }
9
10 cout << wordlist << endl;
11 // Output: {"cat", "dog", "cat", "cat"}
```

SET

SET

- 集合类中非常好用的一个类
- 通常用于建模集合 (set) 的数学抽象
- 每个元素是无序的
- 每个元素的值仅出现一次

斯坦福库

 小试牛刀：解决生词本程序冗余问题

今日话题

- 抽象数据类型
- ~~Vector~~
- ~~Set~~
- Map

MAP

MAP

- 概念上与字典类似
- 提供了一个标签称为键 (Key)
- 和一个与键相关联的值 (Value)

MAP

- 概念上与字典类似
- 提供了一个标签称为键 (Key)
- 和一个与键相关联的值 (Value)

MAP

- 概念上与字典类似
- 提供了一个标签称为键 (Key)
- 和一个与键相关联的值 (Value)

斯坦福库

小试牛刀：优化的词频统计程序

今日话题

- 抽象数据类型
- ~~Vector~~
- ~~Set~~
- ~~Map~~



不愤不启，不悱不发，
举一隅不以三隅反，则不复也。
——孔子 《论语·述而》

下一次课

- 程序交互
- Stack
- Queue

THE END