一、 Vector

> Vector其实很大程度上和数组一样，只是**数组是固定长度，而vector是不定长度（动态增长）**。 假设我们需要记录明年的测试成绩，但是我们并不知道明年会有多少个学生。那么可以有两种选择，定义一个固定长度的数组，这个长度超过假设的长度， 另一种办法就是使用动态数组，比如是： vector
>
> **vector 在C++ STL(标准模板库)中的一个容器，可以看成是对容器的一种扩展。**在运行时可以改变长度，与数组具有相似的语法，相比数组更高效，提供越界检查

1. 声明和初始化

- 声明

> 使用vector除了要导入`#include`之外，由于它声明于std命名空间里面，所以要配合std命名空间使用

```
#include <vector>
using namespace std;

int main(){

    vector <char> vowels;
    vector <int> test_score;

    // =========================
    vector <char> vowels(5);  //声明一个初始大小为5的char类型vector
    vector <int> test_score(10);
    return 0;
}
```

- 初始化

```
#include <vecotr>
using namespace std;

int mian(){
    //数组定义
    int test_score []{100,99,18,81}

    //vector定义
    vector <char> vowels {'a' , 'e' , 'i' , 'o' ,'u'};
    vector <int> test_score{ 100 ,98,95,90,80};
    vector <double> temperatures{26,20.7};
    "jiangsheng"
    return 0;
}
```

2. 访问vector

> 访问`vector`中的元素有两种方式，一是仍以数组的方式，另一种是使用`vector`提供的`at`函数

- 数组的语法

```
#include <iostream>
#include <vector>
using namespace std;

int main(){
    vector<int> test_score {100,90,85};

    cout << "第一个成绩是: " <<test_score[0] << endl;
    cout << "第二个成绩是: " <<test_score[1] << endl;
    cout << "第三个成绩是: " <<test_score[2] << endl;

    cout << "第三个成绩是: " <<test_score[3] << endl;  //不会检查越界
    return 0 ;
}
```

- vector的语法

```
#include <iostream>
#include <vector>
using namespace std;

int main(){

    vector<int> test_score {100,90,85};

    cout << "第一个成绩是: " <<test_score.at(0) << endl;
    cout << "第二个成绩是: " <<test_score.at(1) << endl;
    cout << "第三个成绩是: " <<test_score.at(2) << endl;

    cout << "第三个成绩是: " <<test_score.at(3) << endl; //抛出越界异常
    return 0 ;
}
```

3. 操作vector

- 修改vector中的元素

```
#include <vector>
using namespace std;

int main(){

    vector<int> test_score {100,90,85};
    test_score.at(0) = 73;

    return 0 ;
}
```

- 往vector中追加元素

```
#include <vector>
using namespace std;

int main(){
    vector<int> test_score {100,90,85};

    test_score.push_back(80); // 100 , 90 , 85 , 80
    test_score.push_back(95); // 100 ,  90 ,  85 ,  80 ,  95
    test_score.insert(test_score.begin(),23);//23
    return 0 ;
}
```

- 越界检查

> 只要当我们使用了vector的语法去获取超出索引的元素时，就会抛出异常。而使用数组的语法去获取元素，则不会进行越界检查

```
#include<vector>
using namespace std;
int main(){
    vector<int> score{95,88};
    score.at( n: 3);
}
```

```
terminate called after throwing an instance of 'std::out_of_range'
  what():  vector::_M_range_check: __n (which is 3) >= this->size() (which is 2)
```

- 遍历vector

```
#include <iostream>
#include <vector>
using namespace std;

int main(){

    //使用下标遍历
```

```
    vector<int> scores{ 100 ,95 ,88 ,80 ,75};
    for (int i = 0; i < scores.size(); ++i) {
        cout << scores[i] << endl;
    }

    //基于范围for遍历
    vector<int> scores{ 100 ,95 ,88 ,80 ,75};
    for(int score : scores){
        cout << score << endl;
    }
    return 0 ;
}
```

4. 二维vector

> 二维vector和二维数组实际上差不太多，二维数组是数组里面装的是数组，二维vector指的是vector里面装的还是vector，在未来碰到矩阵相关的存储操作，多半使用vector来作为媒介。 比如下面的例子，演示了使用vector来存储3个班的考试成绩。每个班的成绩单独使用一个vector来存储。

```
#include <iostream>
#include <vecotr>
using namespace std;

int main(){

    //声明并初始化vector
    vector<vector<int>> scores {
            {95,77,80,85},
            {58,89,93,100},
            {69,73,81,97}
    };

    for (int i = 0; i < scores.size(); ++i) {
        for (int j = 0; j < scores[i].size(); ++j) {
           cout << scores[i][j] <<"\t" ;
        }
        cout << endl;
    }
    return 0 ;
}
```

5 . 练习

使用二维vector记录三个学生的6门学科成绩，并且计算每个学生的总分，平均分。