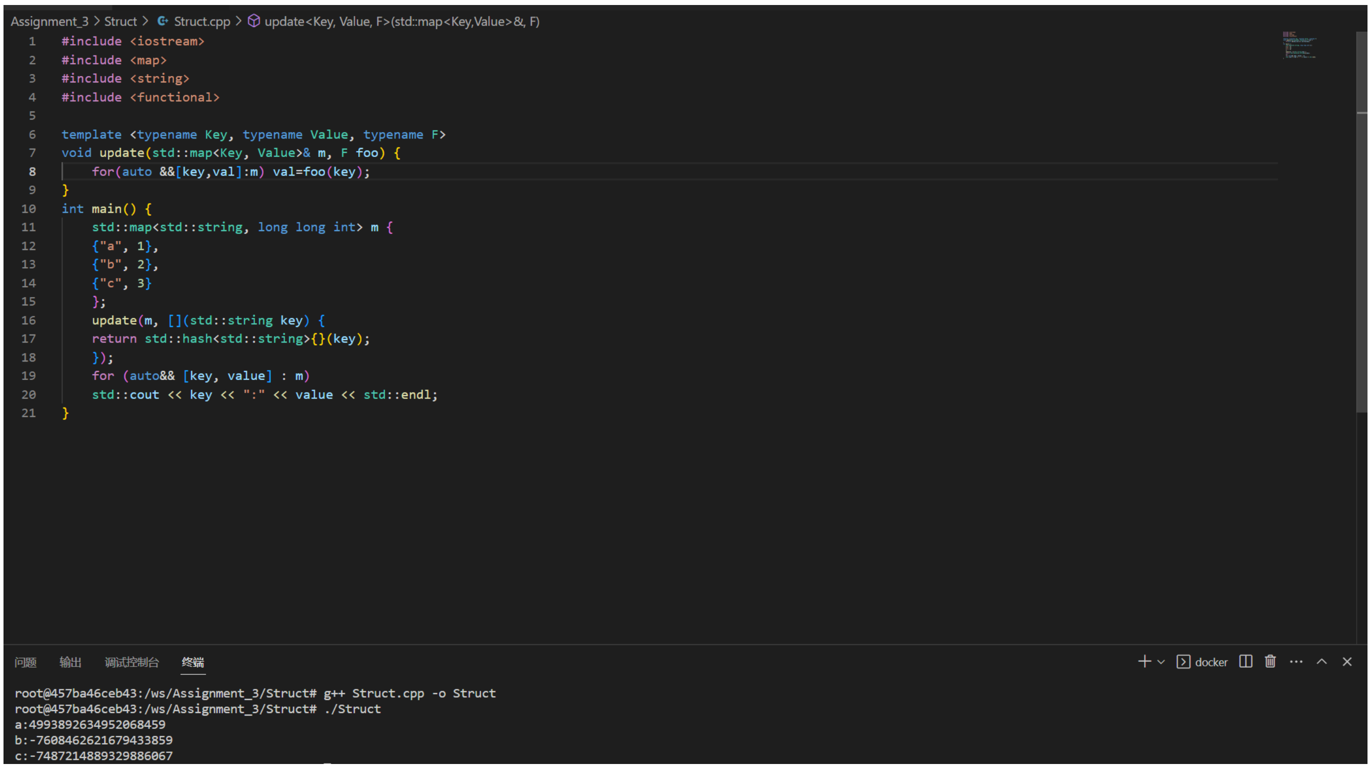
C++面向对象程序设计 作业报告3

报告人：邓雍　学号：2022141220184

一.结构体绑定


Assignment_3 > Struct > Struct.cpp > update<Key, Value, F>(std::map<Key,Value>&, F)
问题
输出
调试控制台
终端
docker

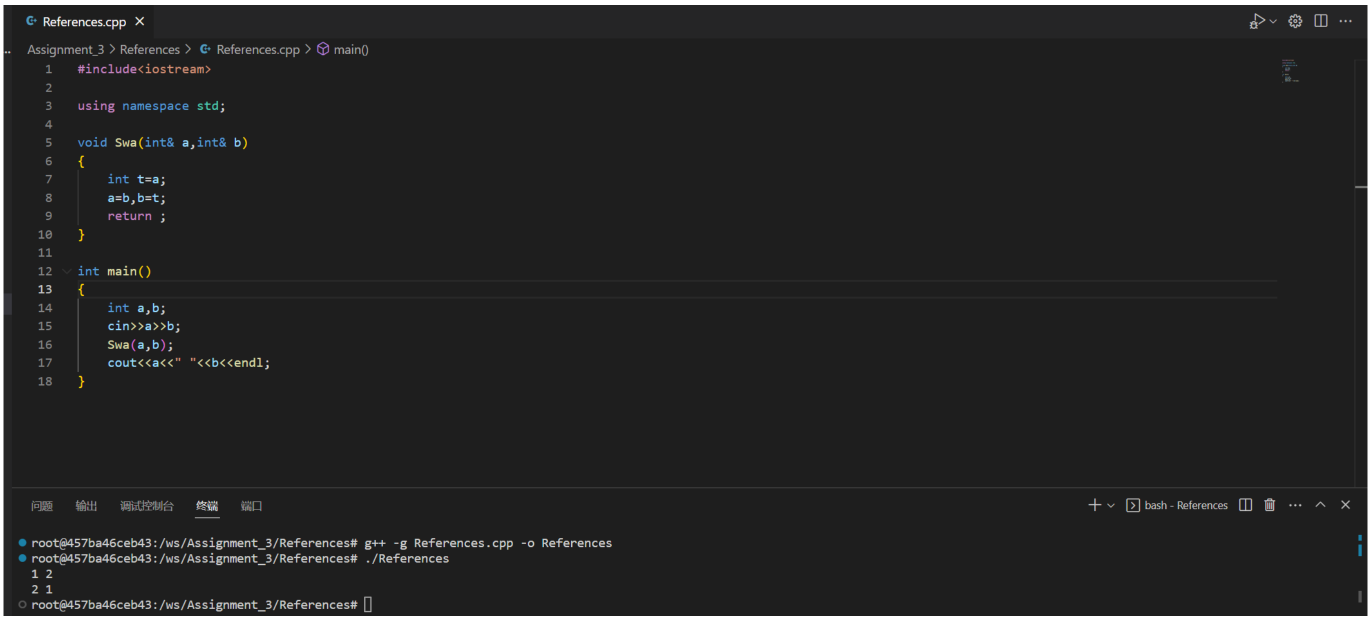
```cpp
#include <iostream>
#include <map>
#include <string>
#include <functional>

template <typename Key, typename Value, typename F>
void update(std::map<Key, Value>& m, F foo) {
    for(auto &&[key,val]:m) val=foo(key);
}
int main() {
    std::map<std::string, long long int> m {
    {"a", 1},
    {"b", 2},
    {"c", 3}
    };
    update(m, [](std::string key) {
    return std::hash<std::string>{}(key);
    });
    for (auto&& [key, value] : m)
    std::cout << key << ":" << value << std::endl;
}
```

```
root@457ba46ceb43:/ws/Assignment_3/Struct# g++ Struct.cpp -o Struct
root@457ba46ceb43:/ws/Assignment_3/Struct# ./Struct
a:4993892634952068459
b:-7608462621679433859
c:-7487214889329886067
```

当存在同时有多个数据的结构时，例如map，pair或者其他结构体，可以通过结构体绑定的方式直接一对一快速取出其中所有元素，如果加上取地址符还可以直接引用。

上图进行了结构体绑定并输出了测试结果

二.References

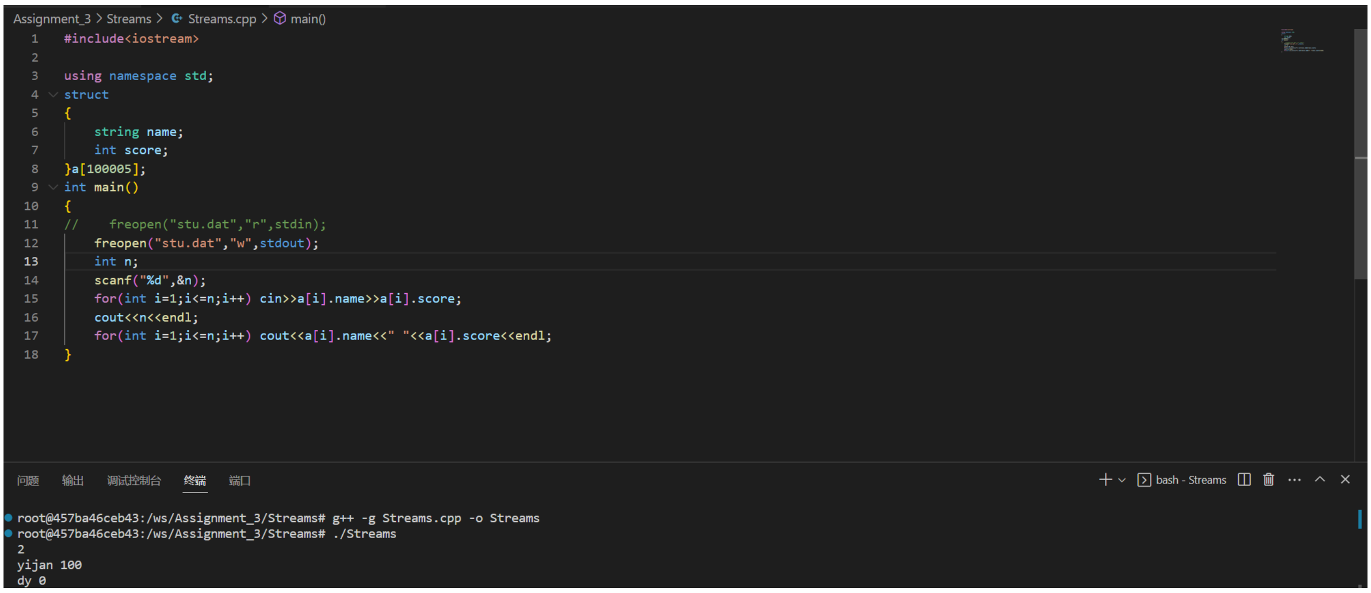

References.cpp
Assignment_3 > References > References.cpp > main()
问题 输出 调试控制台 终端 端口
bash - References

```cpp
#include<iostream>

using namespace std;

void Swa(int& a,int& b)
{
    int t=a;
    a=b,b=t;
    return ;
}

int main()
{
    int a,b;
    cin>>a>>b;
    Swa(a,b);
    cout<<a<<" "<<b<<endl;
}
```

```
root@457ba46ceb43:/ws/Assignment_3/References# g++ -g References.cpp -o References
root@457ba46ceb43:/ws/Assignment_3/References# ./References
1 2
2 1
root@457ba46ceb43:/ws/Assignment_3/References# 
```

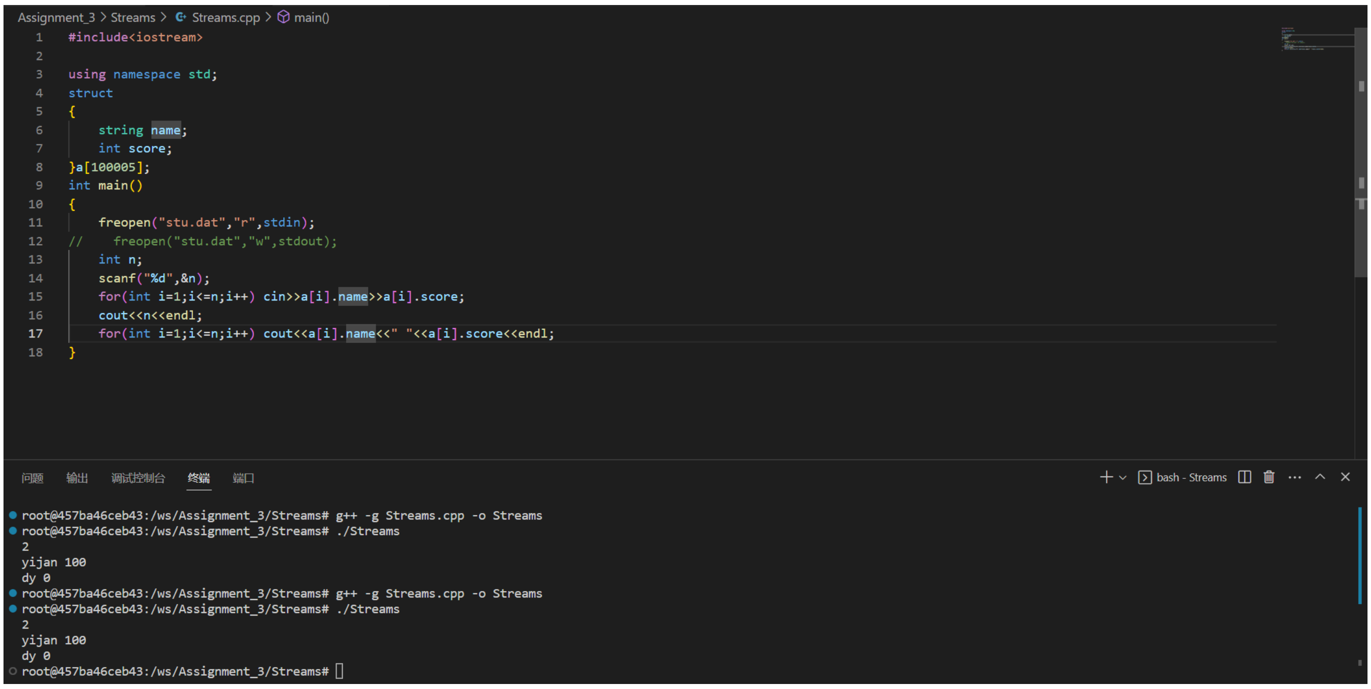
用引用符号取地址，使我们可以直接对地址中的元素进行操作，即使我们在函数内对值进行了修改，在引用的前提下，也是对地址中的信息进行了修改，达到了全局修改的目的

三.Streams


Assignment_3 > Streams > Streams.cpp > main()
问题 输出 调试控制台 终端 端口
bash - Streams

```cpp
#include<iostream>

using namespace std;
struct
{
    string name;
    int score;
}a[100005];
int main()
{
//    freopen("stu.dat","r",stdin);
    freopen("stu.dat","w",stdout);
    int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++) cin>>a[i].name>>a[i].score;
    cout<<n<<endl;
    for(int i=1;i<=n;i++) cout<<a[i].name<<" "<<a[i].score<<endl;
}
```

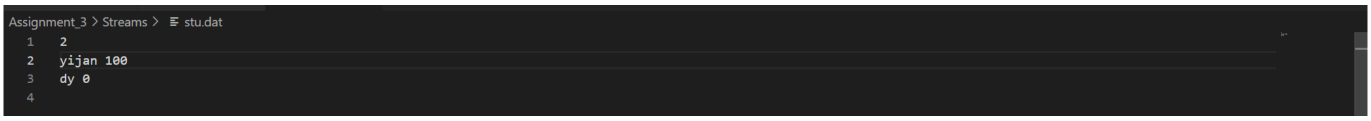
```
root@457ba46ceb43:/ws/Assignment_3/Streams# g++ -g Streams.cpp -o Streams
root@457ba46ceb43:/ws/Assignment_3/Streams# ./Streams
2
yijan 100
dy 0
```



```cpp
#include<iostream>

using namespace std;
struct
{
    string name;
    int score;
}a[100005];
int main()
{
    freopen("stu.dat","r",stdin);
//    freopen("stu.dat","w",stdout);
    int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++) cin>>a[i].name>>a[i].score;
    cout<<n<<endl;
    for(int i=1;i<=n;i++) cout<<a[i].name<<" "<<a[i].score<<endl;
}
```

```
root@457ba46ceb43:/ws/Assignment_3/Streams# g++ -g Streams.cpp -o Streams
root@457ba46ceb43:/ws/Assignment_3/Streams# ./Streams
2
yijan 100
dy 0
root@457ba46ceb43:/ws/Assignment_3/Streams# g++ -g Streams.cpp -o Streams
root@457ba46ceb43:/ws/Assignment_3/Streams# ./Streams
2
yijan 100
dy 0
root@457ba46ceb43:/ws/Assignment_3/Streams# 
```

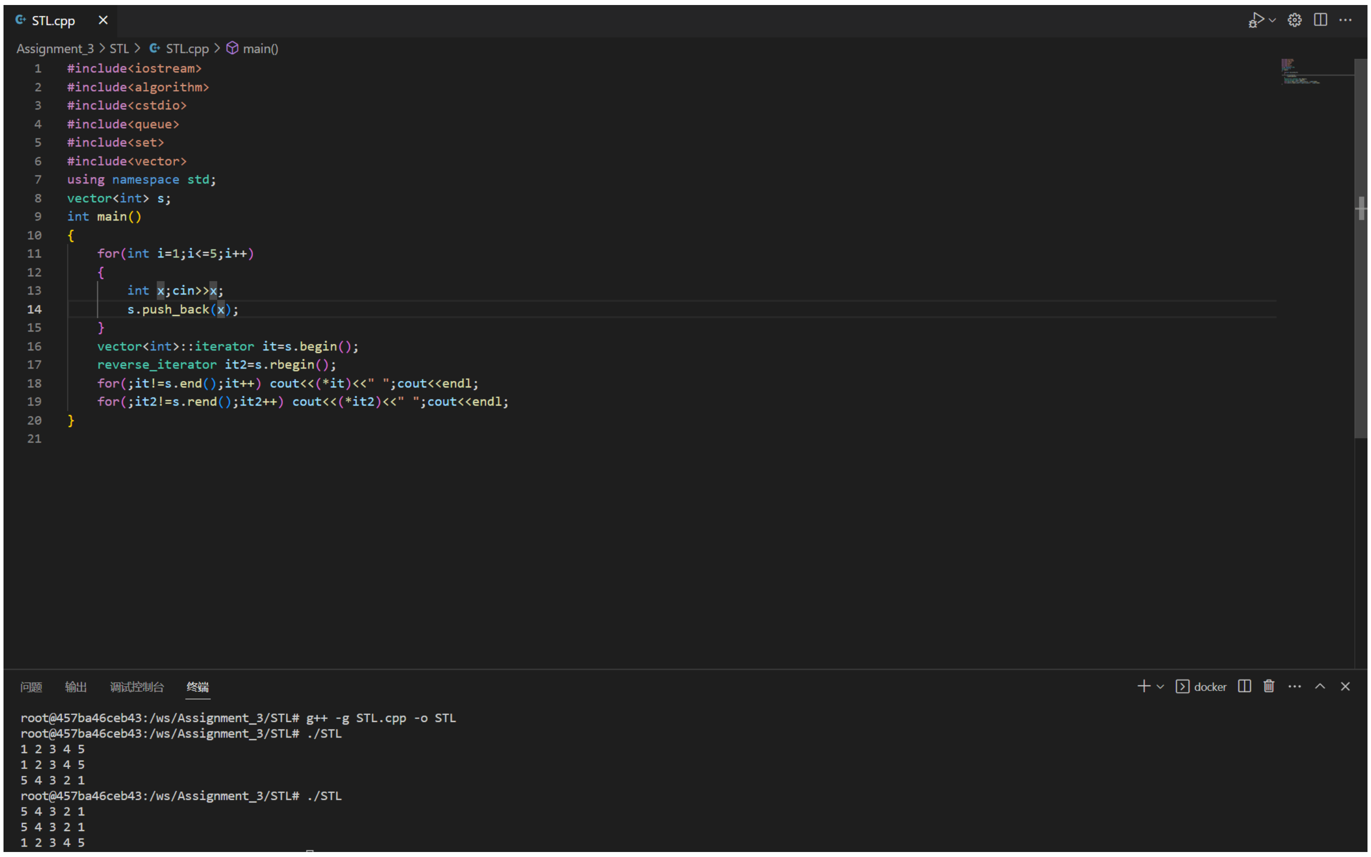
用freopen实现了文件输入输出操作，当启用stdin时，我们会从里面读取数据，当启用stdout，我们会向文件中输出数据



```
2
yijan 100
dy 0

```

附上stu.dat中的数据信息

四.STL(Containers)

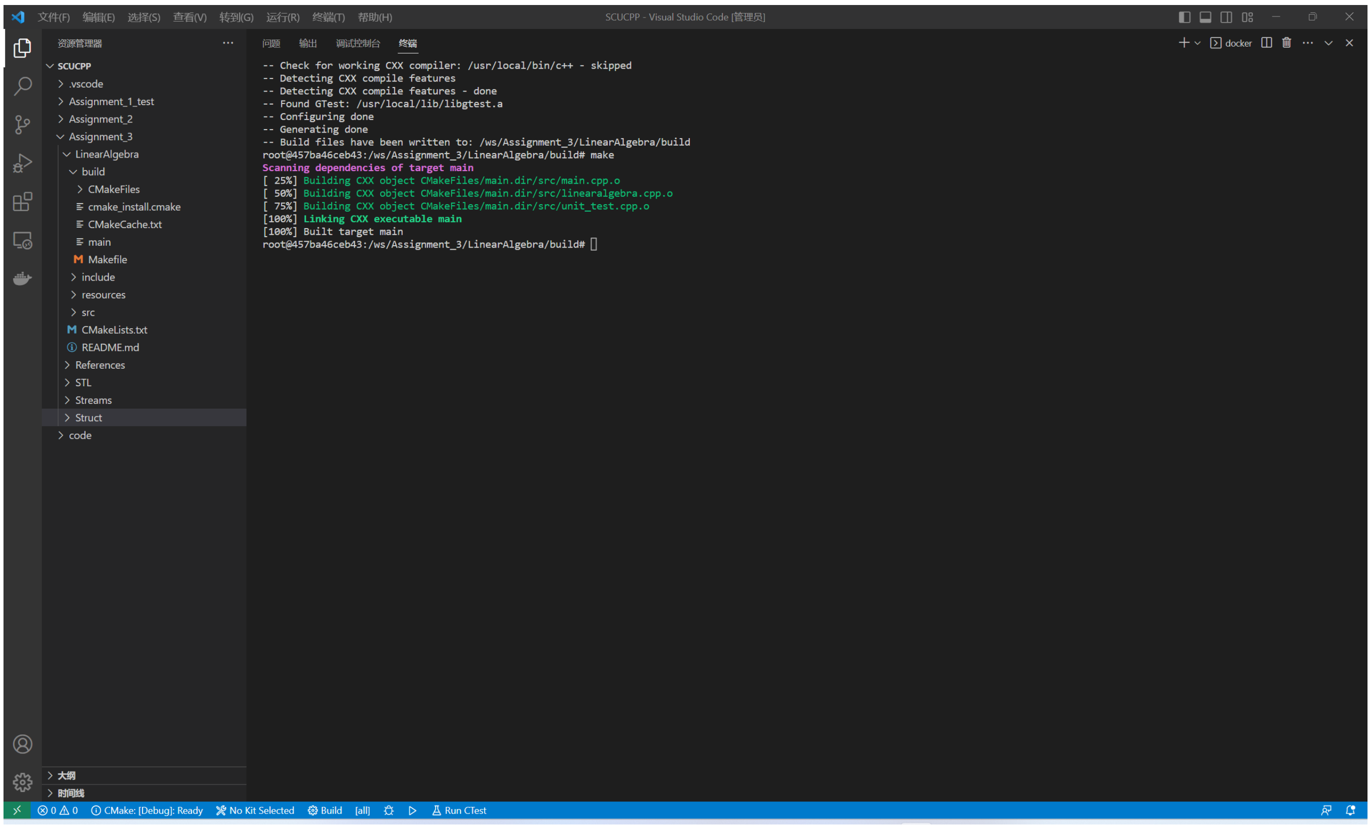

STL.cpp
Assignment_3 > STL > STL.cpp > main()
问题 输出 调试控制台 终端
docker

```cpp
#include<iostream>
#include<algorithm>
#include<cstdio>
#include<queue>
#include<set>
#include<vector>
using namespace std;
vector<int> s;
int main()
{
    for(int i=1;i<=5;i++)
    {
        int x;cin>>x;
        s.push_back(x);
    }
    vector<int>::iterator it=s.begin();
    reverse_iterator it2=s.rbegin();
    for(;it!=s.end();it++) cout<<(*it)<<" ";cout<<endl;
    for(;it2!=s.rend();it2++) cout<<(*it2)<<" ";cout<<endl;
}
```

```
root@457ba46ceb43:/ws/Assignment_3/STL# g++ -g STL.cpp -o STL
root@457ba46ceb43:/ws/Assignment_3/STL# ./STL
1 2 3 4 5
1 2 3 4 5
5 4 3 2 1
root@457ba46ceb43:/ws/Assignment_3/STL# ./STL
5 4 3 2 1
5 4 3 2 1
1 2 3 4 5
```

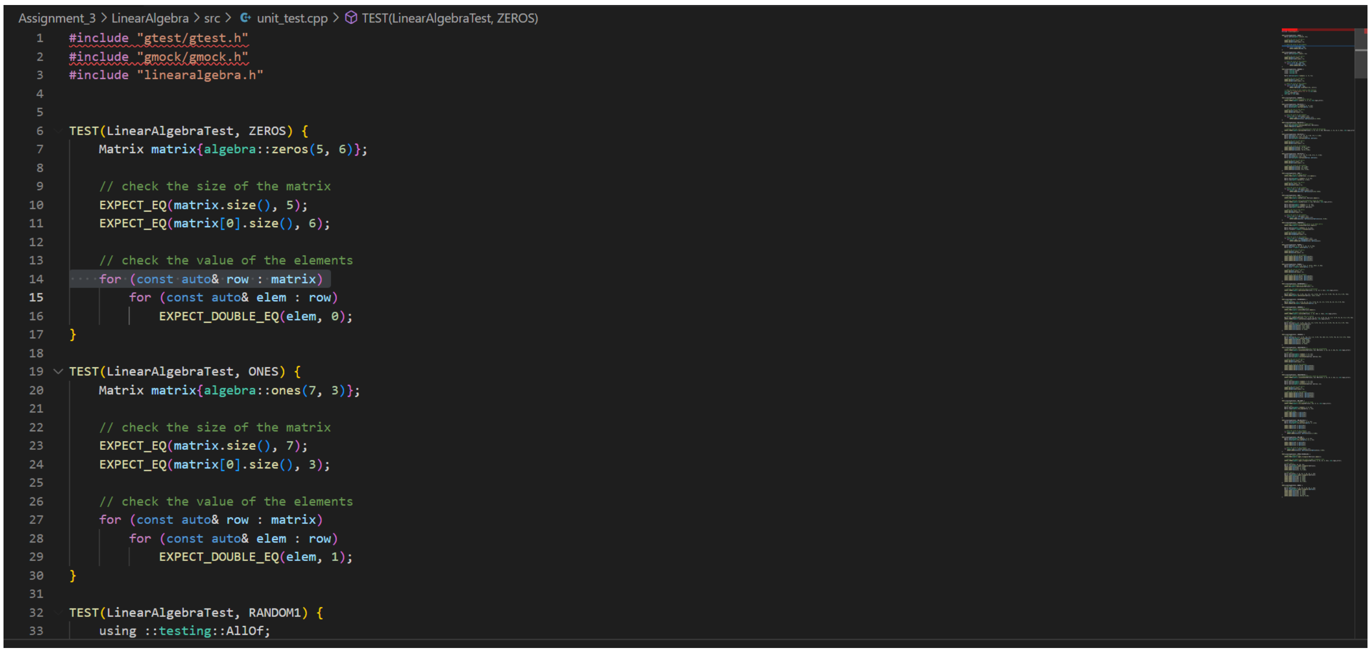
在STL容器中，我们可以使用迭代器来遍历信息，iterator为正向迭代器，reverse_iterator为返向迭代器，均只需地址++判断end即可正序与反序遍历STL中信息

五.Linear Algebra library

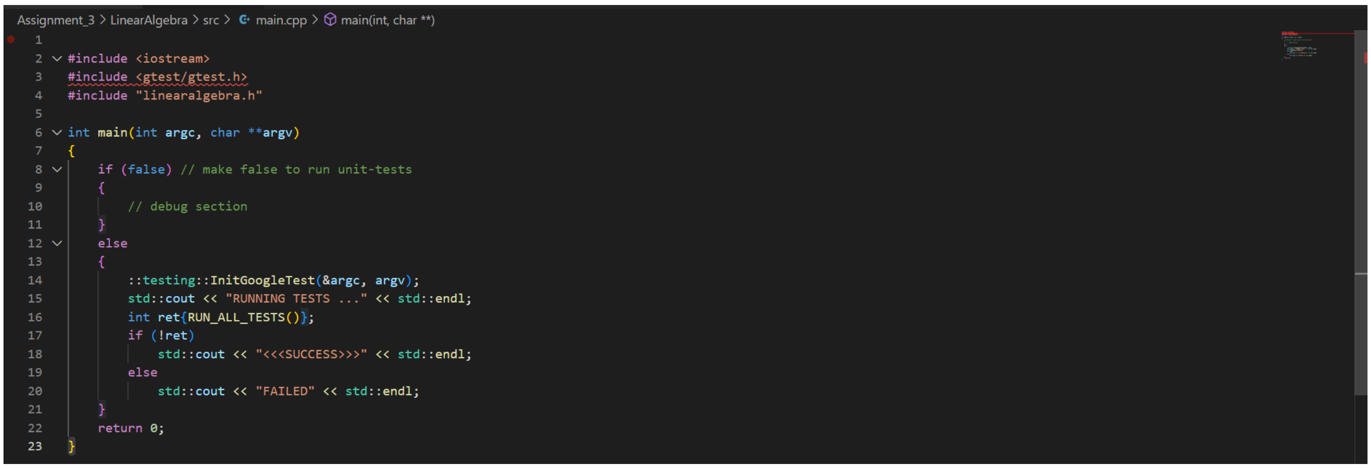


```
-- Check for working CXX compiler: /usr/local/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found GTest: /usr/local/lib/libgtest.a
-- Configuring done
-- Generating done
-- Build files have been written to: /ws/Assignment_3/LinearAlgebra/build
root@457ba46ceb43:/ws/Assignment_3/LinearAlgebra/build# make
Scanning dependencies of target main
[ 25%] Building CXX object CMakeFiles/main.dir/src/main.cpp.o
[ 50%] Building CXX object CMakeFiles/main.dir/src/linearalgebra.cpp.o
[ 75%] Building CXX object CMakeFiles/main.dir/src/unit_test.cpp.o
[100%] Linking CXX executable main
[100%] Built target main
root@457ba46ceb43:/ws/Assignment_3/LinearAlgebra/build# 
```

按照操作说明执行后，我们获得了可执行文件main


Assignment_3 > LinearAlgebra > src > unit_test.cpp > TEST(LinearAlgebraTest, ZEROS)

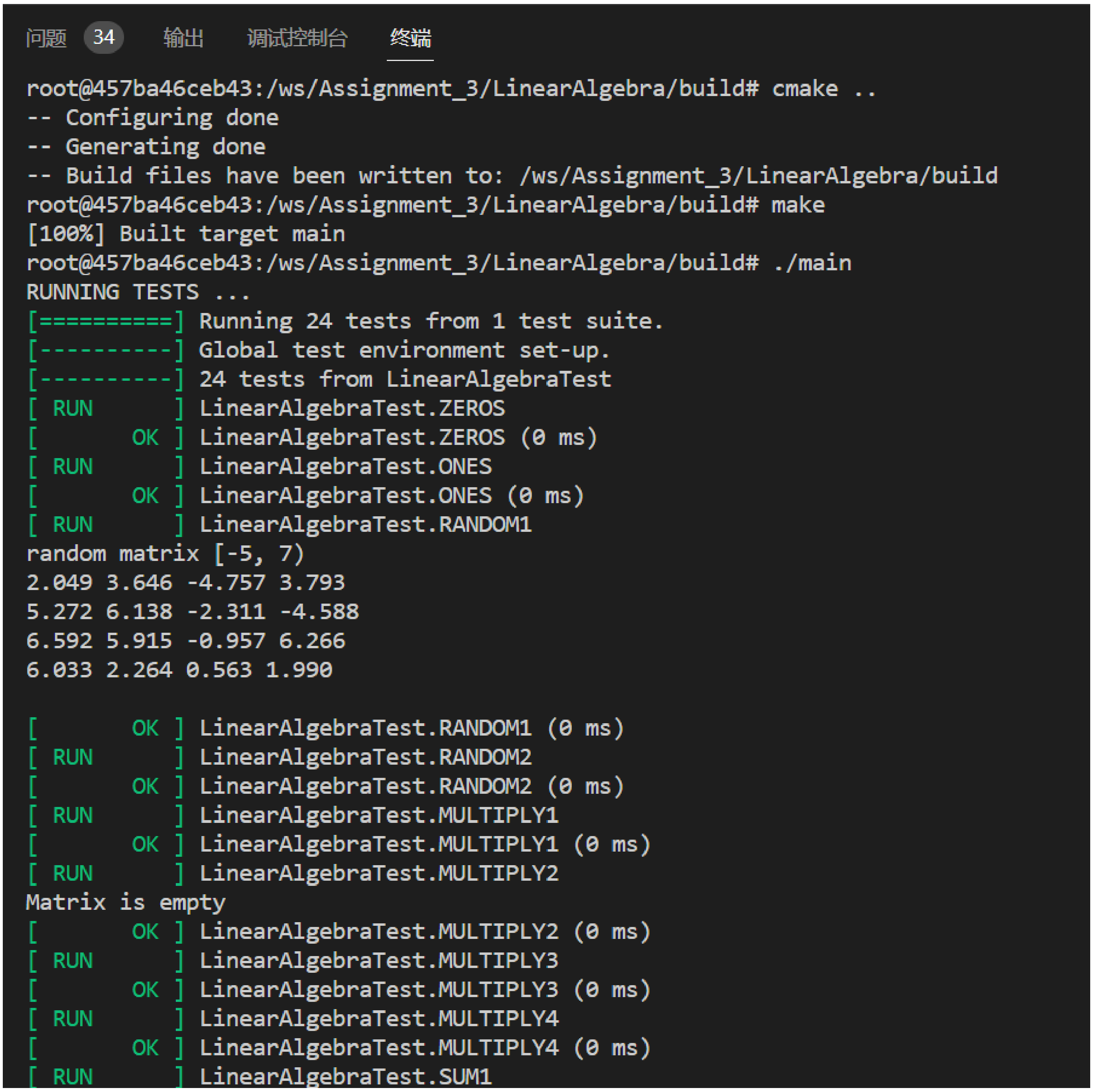
```cpp
#include "gtest/gtest.h"
#include "gmock/gmock.h"
#include "linearalgebra.h"


TEST(LinearAlgebraTest, ZEROS) {
    Matrix matrix{algebra::zeros(5, 6)};

    // check the size of the matrix
    EXPECT_EQ(matrix.size(), 5);
    EXPECT_EQ(matrix[0].size(), 6);

    // check the value of the elements
    for (const auto& row : matrix)
        for (const auto& elem : row)
            EXPECT_DOUBLE_EQ(elem, 0);
}

TEST(LinearAlgebraTest, ONES) {
    Matrix matrix{algebra::ones(7, 3)};

    // check the size of the matrix
    EXPECT_EQ(matrix.size(), 7);
    EXPECT_EQ(matrix[0].size(), 3);

    // check the value of the elements
    for (const auto& row : matrix)
        for (const auto& elem : row)
            EXPECT_DOUBLE_EQ(elem, 1);
}

TEST(LinearAlgebraTest, RANDOM1) {
    using ::testing::AllOf;
```



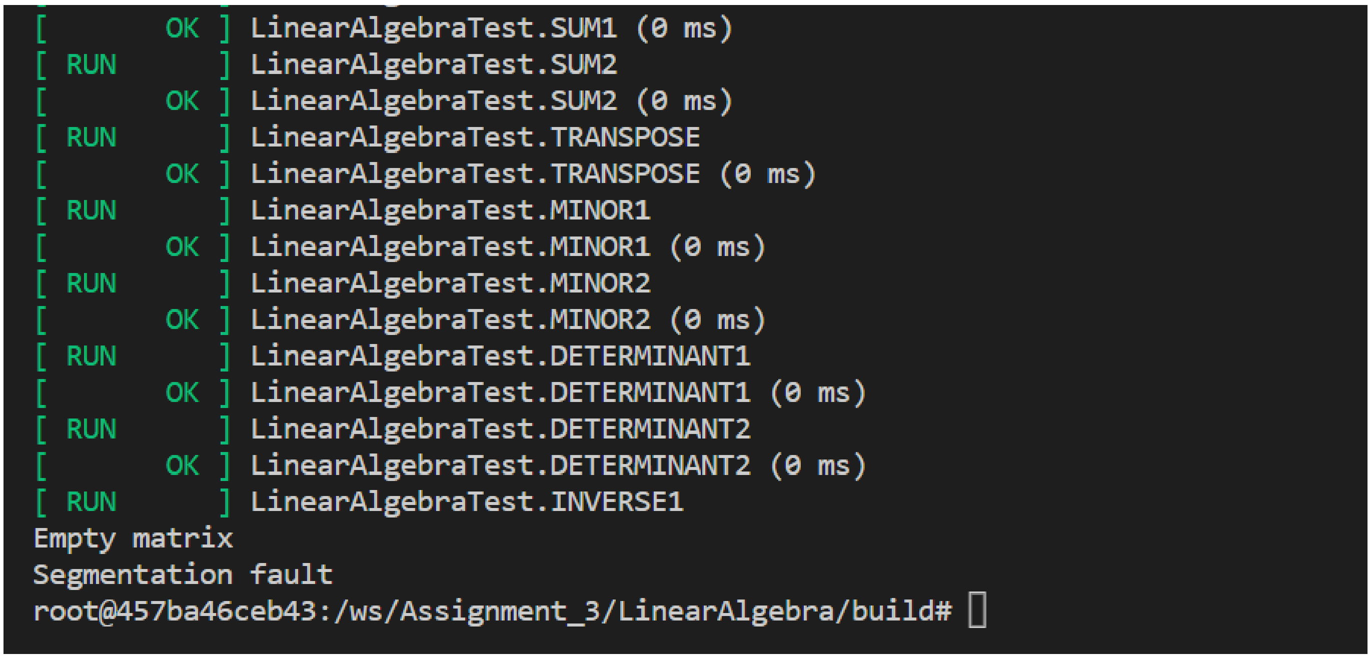
Assignment_3 > LinearAlgebra > src > main.cpp > main(int, char **)

```cpp

#include <iostream>
#include <gtest/gtest.h>
#include "linearalgebra.h"

int main(int argc, char **argv)
{
    if (false) // make false to run unit-tests
    {
        // debug section
    }
    else
    {
        ::testing::InitGoogleTest(&argc, argv);
        std::cout << "RUNNING TESTS ..." << std::endl;
        int ret{RUN_ALL_TESTS()};
        if (!ret)
            std::cout << "<<<SUCCESS>>>" << std::endl;
        else
            std::cout << "FAILED" << std::endl;
    }
    return 0;
}
```

注释并修改得到单元测试所需文件



```
root@457ba46ceb43:/ws/Assignment_3/LinearAlgebra/build# cmake ..
-- Configuring done
-- Generating done
-- Build files have been written to: /ws/Assignment_3/LinearAlgebra/build
root@457ba46ceb43:/ws/Assignment_3/LinearAlgebra/build# make
[100%] Built target main
root@457ba46ceb43:/ws/Assignment_3/LinearAlgebra/build# ./main
RUNNING TESTS ...
[==========] Running 24 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 24 tests from LinearAlgebraTest
[ RUN      ] LinearAlgebraTest.ZEROS
[       OK ] LinearAlgebraTest.ZEROS (0 ms)
[ RUN      ] LinearAlgebraTest.ONES
[       OK ] LinearAlgebraTest.ONES (0 ms)
[ RUN      ] LinearAlgebraTest.RANDOM1
random matrix [-5, 7)
2.049 3.646 -4.757 3.793
5.272 6.138 -2.311 -4.588
6.592 5.915 -0.957 6.266
6.033 2.264 0.563 1.990

[       OK ] LinearAlgebraTest.RANDOM1 (0 ms)
[ RUN      ] LinearAlgebraTest.RANDOM2
[       OK ] LinearAlgebraTest.RANDOM2 (0 ms)
[ RUN      ] LinearAlgebraTest.MULTIPLY1
[       OK ] LinearAlgebraTest.MULTIPLY1 (0 ms)
[ RUN      ] LinearAlgebraTest.MULTIPLY2
Matrix is empty
[       OK ] LinearAlgebraTest.MULTIPLY2 (0 ms)
[ RUN      ] LinearAlgebraTest.MULTIPLY3
[       OK ] LinearAlgebraTest.MULTIPLY3 (0 ms)
[ RUN      ] LinearAlgebraTest.MULTIPLY4
[       OK ] LinearAlgebraTest.MULTIPLY4 (0 ms)
[ RUN      ] LinearAlgebraTest.SUM1
```



```
[       OK ] LinearAlgebraTest.SUM1 (0 ms)
[ RUN      ] LinearAlgebraTest.SUM2
[       OK ] LinearAlgebraTest.SUM2 (0 ms)
[ RUN      ] LinearAlgebraTest.TRANSPOSE
[       OK ] LinearAlgebraTest.TRANSPOSE (0 ms)
[ RUN      ] LinearAlgebraTest.MINOR1
[       OK ] LinearAlgebraTest.MINOR1 (0 ms)
[ RUN      ] LinearAlgebraTest.MINOR2
[       OK ] LinearAlgebraTest.MINOR2 (0 ms)
[ RUN      ] LinearAlgebraTest.DETERMINANT1
[       OK ] LinearAlgebraTest.DETERMINANT1 (0 ms)
[ RUN      ] LinearAlgebraTest.DETERMINANT2
[       OK ] LinearAlgebraTest.DETERMINANT2 (0 ms)
[ RUN      ] LinearAlgebraTest.INVERSE1
Empty matrix
Segmentation fault
root@457ba46ceb43:/ws/Assignment_3/LinearAlgebra/build# 
```

程序运行通过，代码附在附件中。