学号：2022141410193 姓名：王枭越

2 Structured Binding

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

template <typename Key, typename Value, typename F>
void update(std::map<Key, Value>& m, F foo) {
// TODO:
    for(auto& [key,value]:m){
        value=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;
    return 0;
}
```

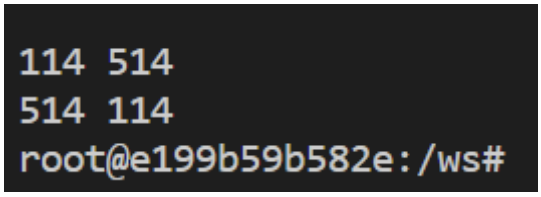
如上，在注释`// TODO:`之后为我所添加的代码，其含义是用结构体绑定去获取m中的每一个键值对，然后根据hash函数改变每个键对应的值。

3 References

```
#include <iostream>

using namespace std;

void _swap(int& a,int& b){
    int c=a;
    a=b;
    b=c;
}

int main(){
    int x=114;
    int y=514;
```

```
    cout<<x<<" "<<y<<endl;
    _swap(x,y);
    cout<<x<<" "<<y<<endl;
    return 0;
}
```

测试代码如上，运行结果如下



含义是通过改变两个数的引用的值做到交换这两个数。

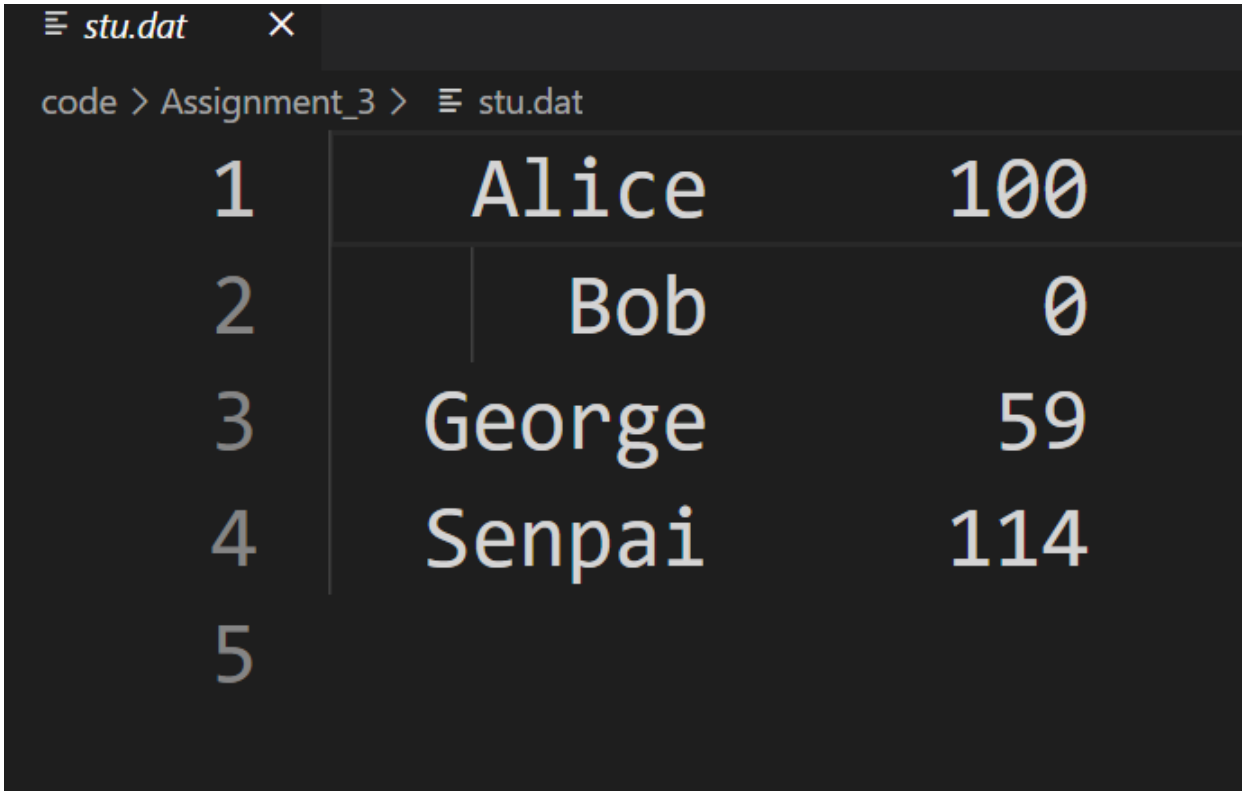
4 Streams

```
#include <bits/stdc++.h>

using namespace std;

struct stu{
    string name;
    int score;
}a,b[10003];

int main(){
    ofstream out("/ws/code/Assignment_3/stu.dat");
    int n;cin>>n;
    while(n--){
        cin>>a.name>>a.score;
        out<<setw(8)<<a.name<<setw(8)<<a.score<<endl;
    }
    ifstream in("/ws/code/Assignment_3/stu.dat");
    for(int i=0;in>>b[i].name>>b[i].score;i++){
        cout<<setw(8)<<b[i].name<<setw(8)<<b[i].score<<endl;
    }
    return 0;
}
```

以上代码首先定义了临时结构体`a`，并用`ofstream`将每个结构体的信息转进文件`stu.dat`，之后我将`stu.dat`的信息读出在终端上，如下

```
4
Alice 100
Bob 0
George 59
Senpai 114
   Alice     100
     Bob       0
  George      59
  Senpai     114
```

而当我们打开`stu.dat`查看，结果如下



5 STL(Containers)

```cpp
#include <iostream>
#include <vector>

using namespace std;

vector<int> a;
int tmp;

int main(){
    for(int i=0;i<5;i++){
        cin>>tmp;
        a.push_back(tmp);
    }
    for(auto it=a.begin();it!=a.end();it++){
```

```
        cout<<*it<<" ";
    }cout<<endl;
    for(auto it=a.rbegin();it!=a.rend();it++){
        cout<<*it<<" ";
    }cout<<endl;
    return 0;
}
```

利用`vector`自带的正向迭代器`a.begin()`和`a.end()`和反向迭代器`a.rbegin()`和`a.rend()`就可以实现正向和反向遍历整个`vector`

6 Linear Algebra library

先展示代码

`linearalgebra.h`

```
#ifndef LINEARALGEBRA_H
#define LINEARALGEBRA_H

#include <iostream>
#include <vector>
using Matrix = std::vector<std::vector<double>>;
using Vector = std::vector<double>;

class algebra{
    public:
    static Matrix zeros(size_t n, size_t m);
    static Matrix ones(size_t n, size_t m);
    static Matrix random(size_t n, size_t m, double min, double max);
    static void show(const Matrix& matrix);
    static Matrix multiply(const Matrix& matrix, double c);
    static Matrix multiply(const Matrix& matrix1, const Matrix& matrix2);
    static Matrix sum(const Matrix& matrix, double c);
    static Matrix sum(const Matrix& matrix1, const Matrix& matrix2);
    static Matrix transpose(const Matrix& matrix);
    static Matrix minor(const Matrix& matrix, size_t n, size_t m);
    static double determinant(const Matrix& matrix);
    static Matrix inverse(const Matrix& matrix);
    static Matrix concatenate(const Matrix& matrix1, const Matrix& matrix2, int
axis=0);
    static Matrix ero_swap(const Matrix& matrix, size_t r1, size_t r2);
    static Matrix ero_multiply(const Matrix& matrix, size_t r, double c);
    static Matrix ero_sum(const Matrix& matrix, size_t r1, double c, size_t r2);
    static Matrix upper_triangular(const Matrix& matrix);
};
#endif //LINEARALGEBRA_H
```

`linearalgebra.cpp`

```
#include "linearalgebra.h"
```

```cpp
#include <random>
#include <iomanip>
#include <stdexcept>
#define MIN(a,b) (((a)<(b))?(a):(b))
using Matrix = std::vector<std::vector<double>>;
using Vector = std::vector<double>;

const double eps=1e-6;

Matrix algebra::zeros(size_t n, size_t m){
    Matrix ret;
    for(int i=0;i<n;i++){
        Vector tmp;
        for(int j=0;j<m;j++){
            tmp.push_back(0);
        }
        ret.push_back(tmp);
    }
    return ret;
}

Matrix algebra::ones(size_t n, size_t m){
    Matrix ret;
    for(int i=0;i<n;i++){
        Vector tmp;
        for(int j=0;j<m;j++){
            tmp.push_back(1);
        }
        ret.push_back(tmp);
    }
    return ret;
}

Matrix algebra::random(size_t n, size_t m, double min, double max){
    Matrix ret;
    if(min>max){
        throw std::logic_error("min cannot be greater than max");
        return ret;
    }
    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_int_distribution<int> dis(min, max);
    for(int i=0;i<n;i++){
        Vector tmp;
        for(int j=0;j<m;j++){
            tmp.push_back(dis(gen));
        }
        ret.push_back(tmp);
    }
    return ret;
}
```

```
void algebra::show(const Matrix& matrix){
    for(int i=0;i<matrix.size();i++){
        for(int j=0;j<matrix[i].size();j++){
            std::cout<<std::left<<std::setw(8)<<std::fixed<<std::setprecision(3)
<<matrix[i][j];
        }
        std::cout<<std::endl;
    }
}

Matrix algebra::multiply(const Matrix& matrix, double c){
    Matrix ret;
    for(int i=0;i<matrix.size();i++){
        Vector tmp;
        for(int j=0;j<matrix[i].size();j++){
            tmp.push_back(c*matrix[i][j]);
        }
        ret.push_back(tmp);
    }
    return ret;
}

Matrix algebra::multiply(const Matrix& matrix1, const Matrix& matrix2){
    Matrix ret;
    int n,m,s,t;
    n=matrix1.size();
    if(n)
    m=matrix1[0].size();
    s=matrix2.size();
    if(s)
    t=matrix2[0].size();
    if(n==0&&s==0){
        return ret;
    }else if((n==0&&s!=0)||(n!=0&&s==0)){
        throw std::logic_error("matrices with wrong dimensions cannot be
multiplied");
        return ret;
    }
    if(m!=s){
        throw std::logic_error("matrices with wrong dimensions cannot be
multiplied");
        return ret;
    }
    for(int i=0;i<n;i++){
        Vector tmp;
        for(int k=0;k<t;k++){
            double t=0;
            for(int j=0;j<m;j++){
                t+=matrix1[i][j]*matrix2[j][k];
            }
            tmp.push_back(t);
        }
```

```
        ret.push_back(tmp);
    }
    return ret;
}

Matrix algebra::sum(const Matrix& matrix, double c){
    Matrix ret;
    for(int i=0;i<matrix.size();i++){
        Vector tmp;
        for(int j=0;j<matrix[i].size();j++){
            tmp.push_back(c+matrix[i][j]);
        }
        ret.push_back(tmp);
    }
    return ret;
}

Matrix algebra::sum(const Matrix& matrix1, const Matrix& matrix2){
    Matrix ret;
    int n,m,s,t;
    n=matrix1.size();
    if(n)
    m=matrix1[0].size();
    s=matrix2.size();
    if(s)
    t=matrix2[0].size();
    if(n==0&&s==0){
        return ret;
    }
    if(n!=s||m!=t){
        throw std::logic_error("matrices with wrong dimensions cannot be summed");
        return ret;
    }
    for(int i=0;i<n;i++){
        Vector tmp;
        for(int j=0;j<m;j++){
            tmp.push_back(matrix1[i][j]+matrix2[i][j]);
        }
        ret.push_back(tmp);
    }
    return ret;
}

Matrix algebra::transpose(const Matrix& matrix){
    Matrix ret;
    int n,m;
    n=matrix.size();
    if(n)
    m=matrix[0].size();
    if(!n){
        return ret;
    }
```

```
    for(int i=0;i<m;i++){
        Vector tmp;
        for(int j=0;j<n;j++){
            tmp.push_back(matrix[j][i]);
        }
        ret.push_back(tmp);
    }
    return ret;
}

Matrix algebra::minor(const Matrix& matrix, size_t n, size_t m){
    Matrix ret;
    int s=matrix.size();
    int t=matrix[0].size();
    if(n<0||n>=s||m<0||m>=t)return ret;
    for(int i=0;i<s;i++){
        if(i==n)continue;
        Vector tmp;
        for(int j=0;j<t;j++){
            if(j==m)continue;
            tmp.push_back(matrix[i][j]);
        }
        ret.push_back(tmp);
    }
    return ret;
}

double algebra::determinant(const Matrix& matrix){
    //use minor to implement
    double ret=0;
    int n=matrix.size();
    if(n==0)return 1;
    int m=matrix[0].size();
    int t=1;
    if(n!=m){
        throw std::logic_error("non-square matrices have no determinant");
        return ret;
    }
    if(n==1)return matrix[0][0];
    for(int j=0;j<m;j++){
        ret+=matrix[0][j]*determinant(minor(matrix,0,j))*t;
        t=-t;
    }
    return ret;
}

Matrix algebra::inverse(const Matrix& matrix){
    //use minor to implement
    Matrix ret;
    int n,m;
    n=matrix.size();
    if(n)
```

```
    m=matrix[0].size();
    if(!n){
        return ret;
    }
    if(n!=m){
        throw std::logic_error("non-square matrices have no inverse");
        return ret;
    }
    double det=determinant(matrix);
    if(-eps<det&&det<eps){
        throw std::logic_error("singular matrices have no inverse");
        return ret;
    }
    for(int i=0,p=1;i<n;i++,p=-p){
        Vector tmp;
        for(int j=0,q=p;j<m;j++,q=-q){
            double t=determinant(minor(matrix,i,j))/det;
            tmp.push_back(q*t);
        }
        ret.push_back(tmp);
    }
    return transpose(ret);
}

Matrix algebra::concatenate(const Matrix& matrix1, const Matrix& matrix2, int axis){
    Matrix ret;
    int n,m,s,t;
    n=matrix1.size();
    if(n)
    m=matrix1[0].size();
    s=matrix2.size();
    if(s)
    t=matrix2[0].size();
    if(n==0&&s==0){
        return ret;
    }
    if((n==0&&s!=0)&&(n!=0&&s==0)){
        throw std::logic_error("matrices with wrong dimensions cannot be
concatenated");
        return ret;
    }
    if(!axis){
        if(m!=t){
            throw std::logic_error("matrices with wrong dimensions cannot be
concatenated");
            return ret;
        }
        for(int i=0;i<n;i++){
            ret.push_back(matrix1[i]);
        }
        for(int i=0;i<s;i++){
            ret.push_back(matrix2[i]);
```

```
        }
    }else if(axis==1){
        if(n!=s){
            throw std::logic_error("matrices with wrong dimensions cannot be
concatenated");
            return ret;
        }
        for(int i=0;i<n;i++){
            Vector tmp;
            for(int j=0;j<m;j++){
                tmp.push_back(matrix1[i][j]);
            }
            for(int j=0;j<t;j++){
                tmp.push_back(matrix2[i][j]);
            }
            ret.push_back(tmp);
        }
    }
    return ret;
}

Matrix algebra::ero_swap(const Matrix& matrix, size_t r1, size_t r2){
    Matrix ret=matrix;
    int n=matrix.size();
    if(r1>=n||r1<0||r2>=n||r2<0){
        throw std::logic_error("r1 or r2 inputs are out of range");
        return ret;
    }
    Vector tmp=ret[r1];
    ret[r1]=ret[r2];
    ret[r2]=tmp;
    return ret;
}

Matrix algebra::ero_multiply(const Matrix& matrix, size_t r, double c){
    Matrix ret=matrix;
    int n=matrix.size();
    if(r>=n||r<0){
        throw std::logic_error("r input is out of range");
        return ret;
    }
    Vector tmp;
    for(int i=0;i<ret[r].size();i++)tmp.push_back(ret[r][i]*c);
    ret[r]=tmp;
    return ret;
}

Matrix algebra::ero_sum(const Matrix& matrix, size_t r1, double c, size_t r2){
    Matrix ret=matrix;
    int n=matrix.size();
    if(r1>=n||r1<0||r2>=n||r2<0){
        throw std::logic_error("r1 or r2 inputs are out of range");
```

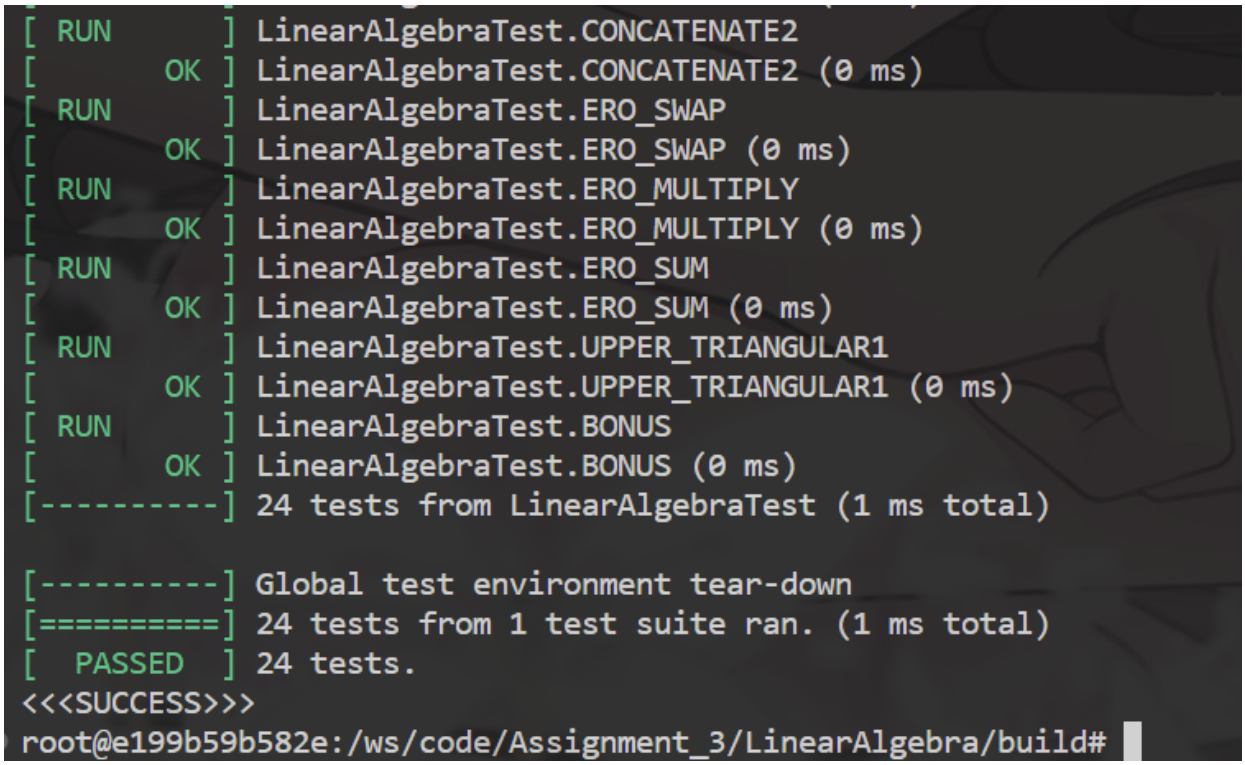
```
        return ret;
    }
    Vector tmp;
    for(int i=0;i<ret[r1].size();i++)tmp.push_back(ret[r1][i]*c+ret[r2][i]);
    ret[r2]=tmp;
    return ret;
}

Matrix algebra::upper_triangular(const Matrix& matrix){
    Matrix ret=matrix;
    int n,m;
    n=matrix.size();
    if(n)
    m=matrix[0].size();
    if(!n){
        return ret;
    }
    if(n!=m){
        throw std::logic_error("non-square matrices have no upper triangular form");
        return ret;
    }
    int d=0;
    for(int k=0;k<MIN(n,m);k++){
        int i=k;
        i-=d;
        if(ret[i][i+d]==0){
            bool isok=false;
            for(int _i=i+1;_i<n;_i++){
                if(ret[_i][i+d]!=0){
                    ret=ero_swap(ret,i,_i);
                    isok=true;
                    break;
                }
            }
            if(isok){
                for(int _i=i+1;_i<n;_i++){
                    ret=ero_sum(ret,i,-(ret[_i][i])/(ret[i][i]),_i);
                }
            }else{
                d++;
            }
        }else{
            for(int _i=i+1;_i<n;_i++){
                ret=ero_sum(ret,i,-(ret[_i][i])/(ret[i][i]),_i);
            }
        }
    }
    return ret;
}
```

以上代码能够完全通过测试，如下

```
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)
0.000   -4.000  -5.000  -2.000
-1.000  2.000   1.000   6.000
6.000   4.000   4.000   -4.000
4.000   4.000   1.000   6.000

[       OK ] LinearAlgebraTest.RANDOM1 (0 ms)
[ RUN      ] LinearAlgebraTest.RANDOM2
[       OK ] LinearAlgebraTest.RANDOM2 (0 ms)
[ RUN      ] LinearAlgebraTest.MULTIPLY1
[       OK ] LinearAlgebraTest.MULTIPLY1 (0 ms)
[ RUN      ] LinearAlgebraTest.MULTIPLY2
[       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
[       OK ] LinearAlgebraTest.INVERSE1 (0 ms)
[ RUN      ] LinearAlgebraTest.INVERSE2
[       OK ] LinearAlgebraTest.INVERSE2 (0 ms)
[ RUN      ] LinearAlgebraTest.CONCATENATE1
[       OK ] LinearAlgebraTest.CONCATENATE1 (0 ms)
[ RUN      ] LinearAlgebraTest.CONCATENATE2
[       OK ] LinearAlgebraTest.CONCATENATE2 (0 ms)
[ RUN      ] LinearAlgebraTest.ERO_SWAP
[       OK ] LinearAlgebraTest.ERO_SWAP (0 ms)
[ RUN      ] LinearAlgebraTest.ERO_MULTIPLY
[       OK ] LinearAlgebraTest.ERO_MULTIPLY (0 ms)
```

```
[ RUN      ] LinearAlgebraTest.ERO_SUM
[       OK ] LinearAlgebraTest.ERO_SUM (0 ms)
[ RUN      ] LinearAlgebraTest.UPPER_TRIANGULAR1
[       OK ] LinearAlgebraTest.UPPER_TRIANGULAR1 (0 ms)
[ RUN      ] LinearAlgebraTest.BONUS
[       OK ] LinearAlgebraTest.BONUS (0 ms)
[----------] 24 tests from LinearAlgebraTest (1 ms total)

[----------] Global test environment tear-down
[==========] 24 tests from 1 test suite ran. (1 ms total)
[  PASSED  ] 24 tests.
<<<SUCCESS>>>
```

```
root@e199b59b582e:/ws/code/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)
0.000   -4.000  -5.000  -2.000
-1.000  2.000   1.000   6.000
6.000   4.000   4.000   -4.000
4.000   4.000   1.000   6.000

[       OK ] LinearAlgebraTest.RANDOM1 (0 ms)
[ RUN      ] LinearAlgebraTest.RANDOM2
[       OK ] LinearAlgebraTest.RANDOM2 (0 ms)
[ RUN      ] LinearAlgebraTest.MULTIPLY1
[       OK ] LinearAlgebraTest.MULTIPLY1 (0 ms)
[ RUN      ] LinearAlgebraTest.MULTIPLY2
[       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
[       OK ] LinearAlgebraTest.INVERSE1 (0 ms)
[ RUN      ] LinearAlgebraTest.INVERSE2
[       OK ] LinearAlgebraTest.INVERSE2 (0 ms)
[ RUN      ] LinearAlgebraTest.CONCATENATE1
[       OK ] LinearAlgebraTest.CONCATENATE1 (0 ms)
```


[RUN] LinearAlgebraTest.CONCATENATE2
[OK] LinearAlgebraTest.CONCATENATE2 (0 ms)
[RUN] LinearAlgebraTest.ERO_SWAP
[OK] LinearAlgebraTest.ERO_SWAP (0 ms)
[RUN] LinearAlgebraTest.ERO_MULTIPLY
[OK] LinearAlgebraTest.ERO_MULTIPLY (0 ms)
[RUN] LinearAlgebraTest.ERO_SUM
[OK] LinearAlgebraTest.ERO_SUM (0 ms)
[RUN] LinearAlgebraTest.UPPER_TRIANGULAR1
[OK] LinearAlgebraTest.UPPER_TRIANGULAR1 (0 ms)
[RUN] LinearAlgebraTest.BONUS
[OK] LinearAlgebraTest.BONUS (0 ms)
[----------] 24 tests from LinearAlgebraTest (1 ms total)
[----------] Global test environment tear-down
[==========] 24 tests from 1 test suite ran. (1 ms total)
[PASSED] 24 tests.
<<<SUCCESS>>>
root@e199b59b582e:/ws/code/Assignment_3/LinearAlgebra/build#