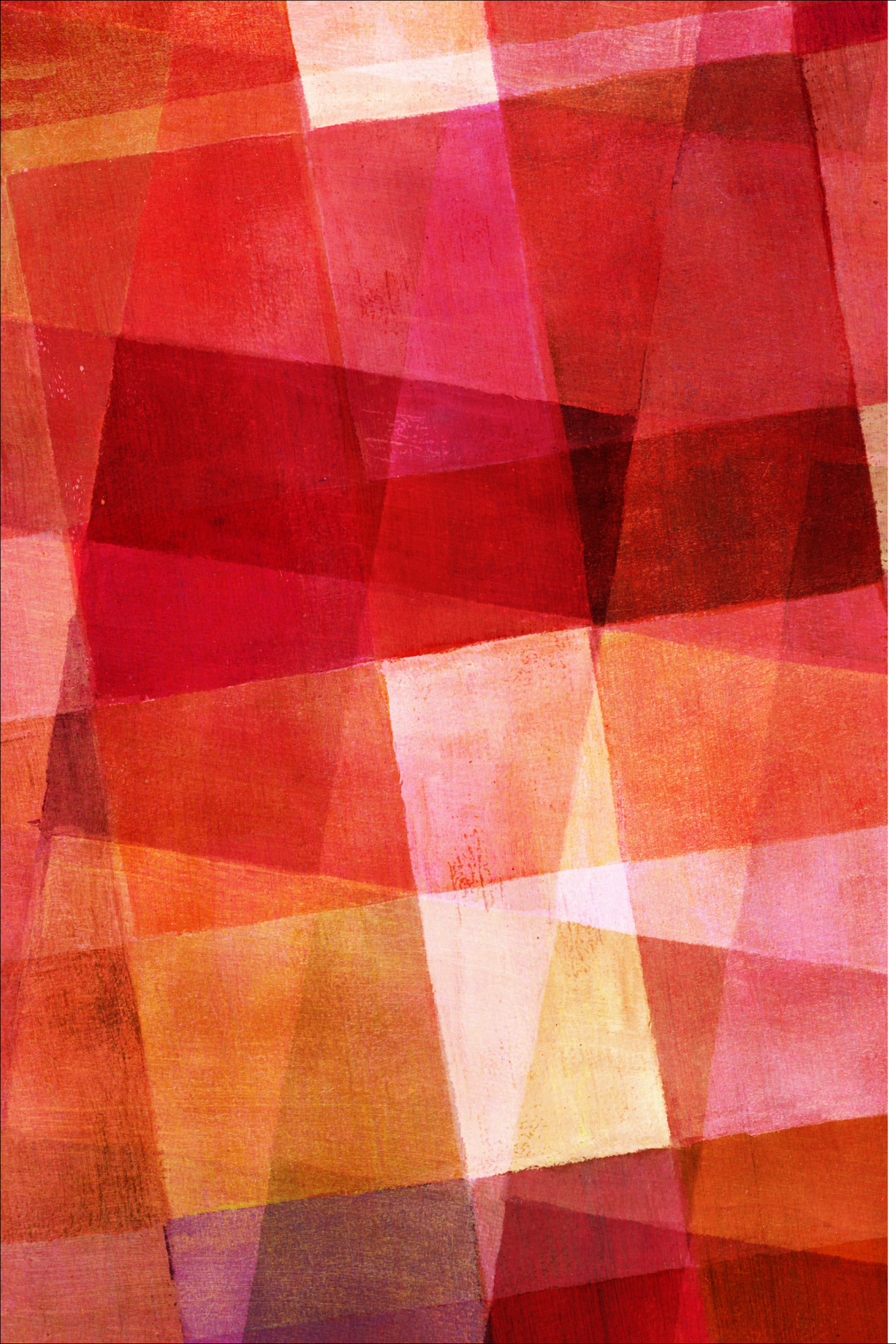
标准模版库

standard template library

容器



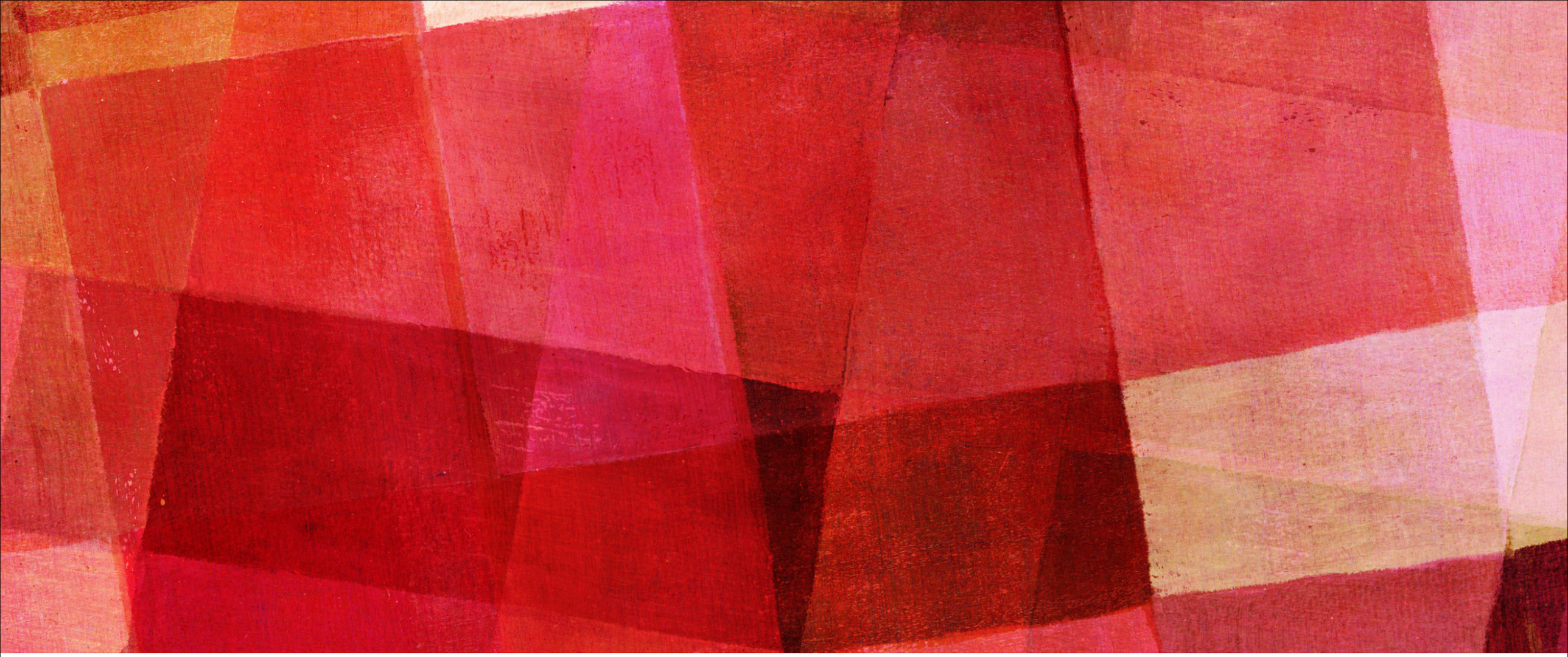
- vector
- set
- multiset
- map
- queue这个相信大家都会
- stack这个相信大家都会

```cpp
vector <int> a, b, c;
void init(int n) {
    a.resize(n, -1);
    b.resize(n, 0);
    c.resize(n, numeric_limits<int>::max());
}

int main() {
    vector <int> tmp = {1, 2, 3, 4 ,5};
    for (int i = 0; i < 10; i++) {
        tmp.push_back(i);
    }
    vector <int> arr(10, -1); //initialize an array of 10 -1
    arr.resize(5, 0); // resize the vector
    cout << arr.size() << endl;
    for (int i = 0; i < (int)arr.size(); i++) { // visit like a normal array
        cout << arr[i] << " " ;
    }

//    arr.erase(arr.begin() + 3, arr.end()); //delete all the element after the third element
    sort (arr.begin(), arr.end());
    arr.erase(unique(arr.begin(), arr.end()), arr.end());
    return 0;
}
```

- 一些细节
- 迭代器是专门为访问容器内的元素而设计的
- c++一般都是左闭右开，[arr.begin(),arr.end())
- *(arr.begin() + k)可以访问下标为k的元素，类似于指针
- vector的迭代器可以进行多步偏移，set，map都没有此功能，只能自增或者自减



SET

不重复集合

set.cpp buffers

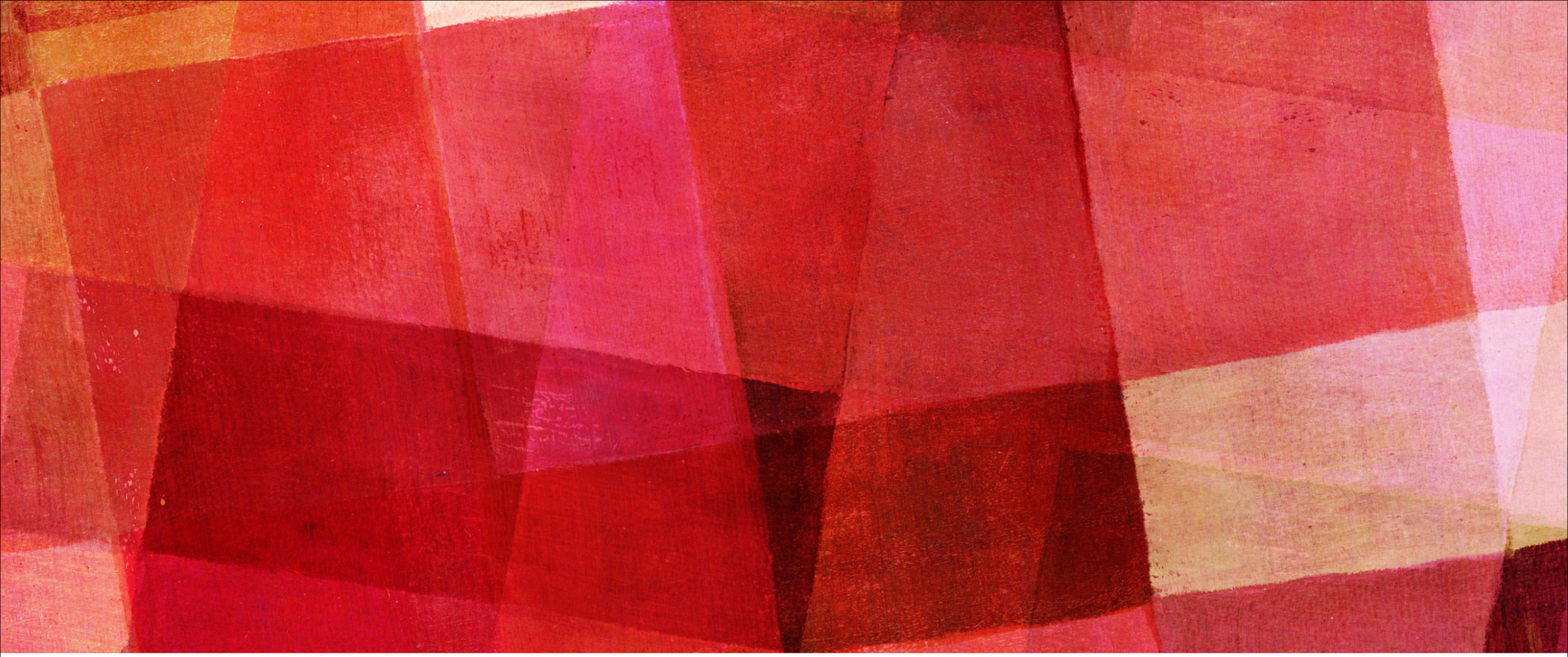
```
12 void output (set <int> &st) {
11     for (set<int>::iterator it = st.begin(); it != st.end(); ++it) {
10         cout<< *it << " " ;
 9     }
 8     cout << endl;
 7 }
 6 int main() {
 5     set <int> st;
 4     st.insert(1);
 3     st.insert(2);
 2     st.insert(2);
 1     st.insert(3);
18     st.insert(4);
 1     output(st);
 2     st.erase(2);
 3     output(st);
 4     set<int>::iterator it = st.find(3);
 5     if (it != st.end()) {
 6         cout << *it << endl;
 7         st.erase(it);
 8         output(st);
 9     }
 9     {
 8         set<int>::iterator it = st.lower_bound(3);
 7         if (it != st.begin()) {
 6             it--;
 5         }
 4         if(it != st.end()) {
 3             cout << *it << endl;
 2         }
 1     }
```

- 插入，删除，查找复杂度均为logn

MULTISET

可重复集合

```cpp
void output (multiset <int> &st) {
    for (set<int>::iterator it = st.begin(); it != st.end(); ++it) {
        cout<< *it << " " ;
    }
    cout << endl;
}
int main() {
    multiset <int> st;
    st.insert(1);
    st.insert(2);
    st.insert(2);
    st.insert(3);
    st.insert(3);
    st.insert(4);
    output(st);
    st.erase(2);
    output(st);
    set<int>::iterator it = st.find(3);
    if (it != st.end()) {
        cout << *it << endl;
        st.erase(it);
        output(st);
    }
    return 0;
}
```



MAP

映射

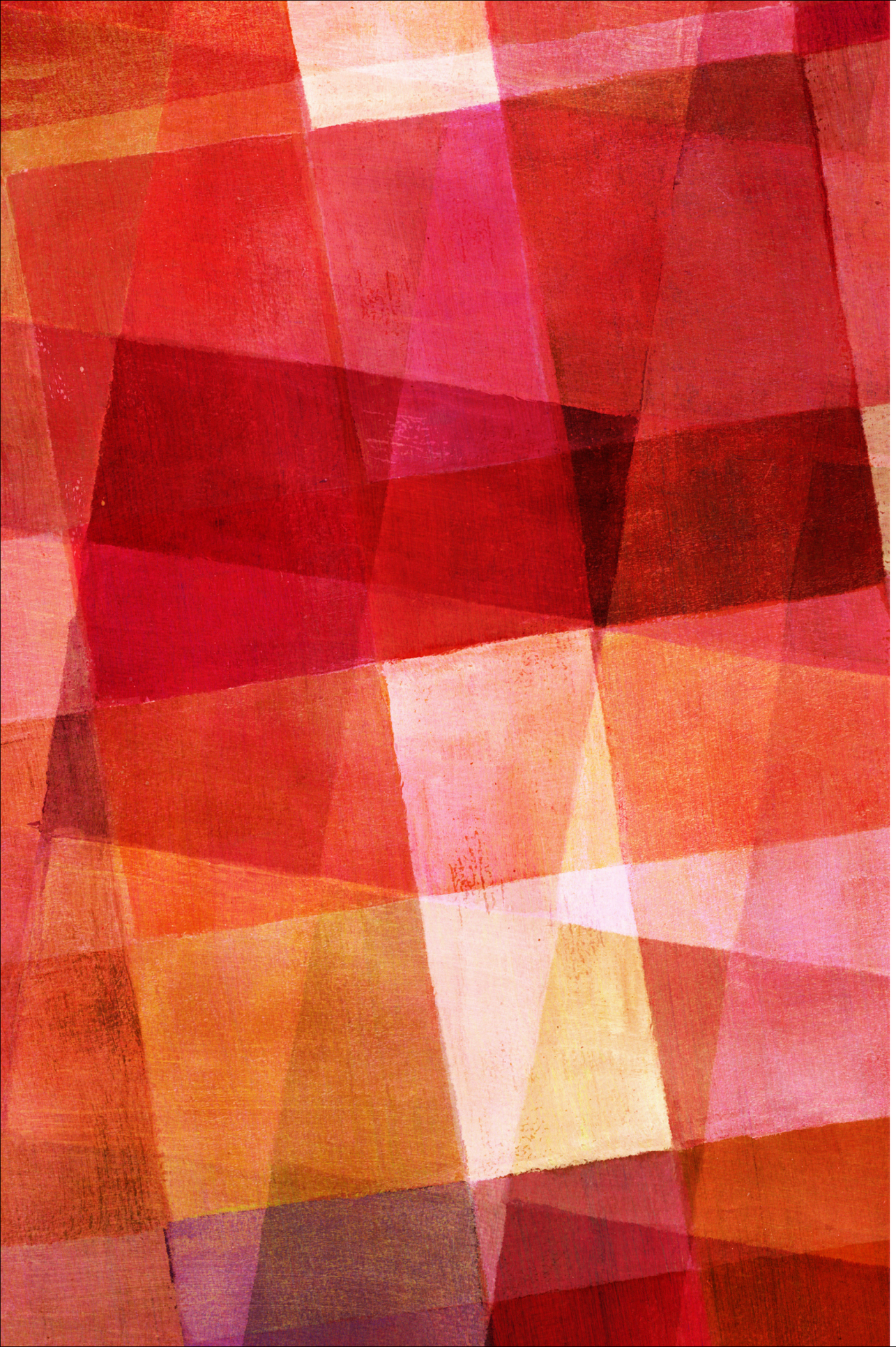
```cpp
//平衡树启发式合并
void merge(map<int,int> &tree, map<int,int>& other) {
    if((int)tree.size() < (int)other.size()) {
        tree.swap(other);
    }
    for (auto it : other) {
        tree[it.first] += it.second;
    }
}
int main() {
    map<string,int> mp;
    mp["hello"] = 1;
    mp["hello"] += 2;
    mp["world"] = 10;
    for (map<string, int> ::iterator it = mp.begin(); it != mp.end(); ++it) {
        cout << it->first << " " << it->second << endl;
    }
    //更多的类型之间的映射
    map<pair<int,int>, string> mp2;
    map<vector<int>, int> mp3;
```

算法

algorithm

- random_shuffle(a.begin(), a.end())
- rotate(first, mid, last)// [mid,last)之间的元素挪到最前面
- reverse(a.begin(), a.end())//反转vector 或者string
- copy(first, last, target)//[first,last)复制到target迭代器开头
- count(first, last, value)//[first,last)之间有多少个元素为value
- prev_permutation(first, last)
- next_permutation(first,last)
- is_sorted(first, last) //判断是否有序
- nth_element(first, nth, last)//将小于第n个元素的元素分到左边，大于第n个元素的元素分到右边，均摊复杂度线性
- fill(a, a + n, value)//清空数组为value
- *min_element(a.begin(), a.end()), *max_element(a.begin(), a.end())
- merge(first1, last1, first2, last2, result)//. 合并有序序列

作业



将优化进行到极致

- sort源码阅读