**容器类型**

// 现代 c++ 应该使用标准库容器而不是内置数组，int a[]为内置数组

vector<string> a; // 可变数组

deque<string> b; // 双端队列，支持头尾插入

list<string> c; // 双向链表，支持双向顺序访问，可以在认可位置插入，插入速度快

forward\_list<string> d; // 单向链表，支持单向顺序访问

array<string, 10> e; // 固定大小

string f; //

stack<string> g; // 栈

queue<string> h; // 队列

**容器操作**

vector<string> lines;

lines.begin(); // 指向首元素的迭代器

lines.cbegin(); // 指向首元素的const迭代器

lines.end(); // 指向尾元素后一位的迭代器

lines.cend(); // 指向尾元素后一位的const迭代器

lines.rbegin(); // 执行尾元素的反向迭代器

lines.rend(); // 指向首元素前一位的迭代器

lines.crbegin(); // const同理

lines.crend(); // const同理

lines.size(); // 元素数量

lines.max\_size(); // 可保存的最大元素数目

lines.empty(); // 是否容器为空（元素数量为0）

vector<string> lines1;

lines.swap(lines1); // 交换lines和lines1的元素

// 比较

if (lines == lines1)

{

}

if (lines != lines1)

{

}

if (lines < lines1)

{

} // 有些容器没有大小比较操作

if (lines > lines1)

{

}

// 添加移除元素，emplace是新标准操作（部分容器类型支持 push\_front 等操作）

lines.push\_back("ab"); // 添加到元素尾部

lines.emplace\_back("cd"); // 添加到元素尾部

// 在lines.end()指向的元素前面插入元素

lines.insert(lines.end(), "g");

lines.insert(lines.end(), {"d", "l"});

// 移除元素

lines.pop\_back(); // 移除尾元素

lines.erase(lines.begin()); // 移除lines.begin()指向的元素，返回被删除元素的后一个元素的迭代器

lines.erase(lines.begin(), lines.end()); // 移除范围，返回被删除范围的后一个元素的迭代器，这里返回lines的尾后元素

// 改变容器大小

lines.resize(20, "a"); // 将容器大小变为20，如果变大，则多出的部分赋值 "a"

lines.front(); // 首元素引用

lines.back(); // 尾元素引用

lines[0]; // 第一个元素

**迭代器**

vector<string> lines{"ab", "cd", "ef"};

auto begin = lines.begin();

while (begin != lines.end())

{

begin++;

cout << \*begin << endl;

}

**初始化操作**

vector<string> lines{"ab", "cd", "ef"};

vector<string> a(lines); // 拷贝操作

vector<string> b = lines; // 赋值操作

// // 错误，vector<string>无法隐式转换为vector<char\*>

// vector<char\*> c(lines);

vector<char\*> c {"ab", "cd"};

vector<string> d(c.begin(), c.end());

vector<string> e(10, "ab"); // 10个元素赋予默认值ab

// 将 assign2 的元素变为 assign1.begin() - assign1.end()的范围

vector<string> assign1{"ab", "cd"};

vector<string> assign2{"a", "b", "c"};

assign2.assign(assign1.begin(), assign1.end()); // assign2 为 "ab", "cd"

// 交换 swap1 和 swap2 的元素

vector<string> swap1{"ab", "cd"};

vector<string> swap2{"a", "b", "c"};

swap(swap1, swap2);

array<string, 2> f{"ab", "cd"}; // array具有固定大小

array<string, 2> g = f; // 对内置类型无法赋值与拷贝，但array可以

for (auto s : g)

{

cout << s << endl;

}

**字符串操作**

string line("abcdef");

auto s1 = line.substr(0, 2); // s1为 ab

line.insert(line.size(), "g"); // 在line.size()位置插入

line.find("ab"); // 返回 0，如果找不到，则返回 string::npos

line.rfind("ab"); // 找出最后匹配的位置

to\_string(1); // 转字符串

// 字符串传数字

stoi("1");

stol("1");

stoll("1");

stof("1");

stod("1");