语法笔记

STL

各种容器的成员函数

vector

使用条件

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

初始化：

```
vector<int> v(10);  // 创建10个元素的数组v，初始化为默认值(0)
vector<int> v(10, 2);  // 创建10个元素的数组v，初始化为2
vector<int> v = {1, 2, 3, 4, 5};  // 使用数组初始化
vector<int> v2(v.begin(), v.end());
```

访问元素的方法：

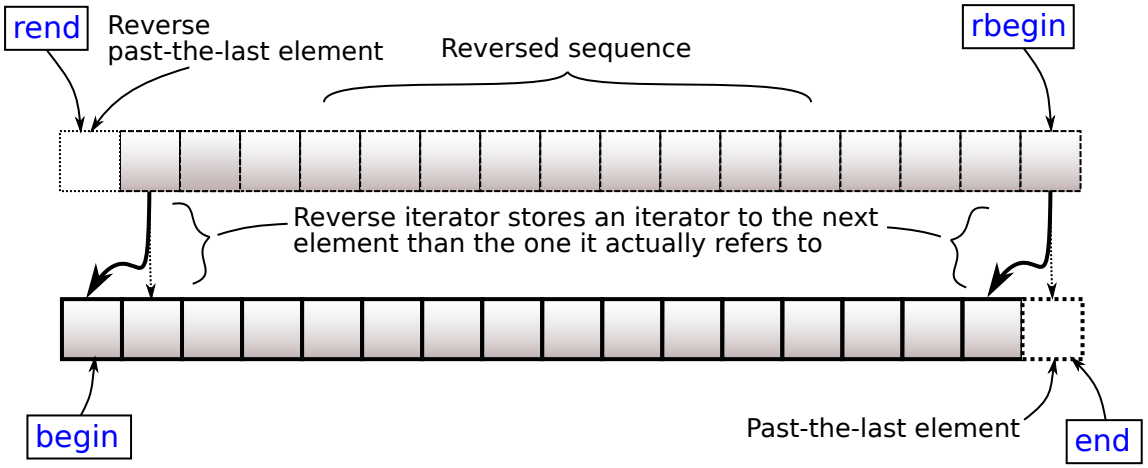
1. 通过下标：`v[i]`；
2. 通过迭代器：`*(v.begin() + i)`。

成员函数：

```
vector& operator=(const vector& other);  // 赋值给容器
void assign(size_type count, const T& value);  // 可以对已经初始化的容器重新赋值
// 元素访问
reference at(size_type pos);  // 返回pos位置元素的引用，带边界检查
reference operator[](size_type pos);
reference front();  // 访问第一个元素
reference back();  // 访问最后一个元素
T* data();  // 返回指向内存中数组第一个元素的指针
// 迭代器
iterator begin();  // 返回指向起始的迭代器
iterator end();  // 返回指向末尾的迭代器
iterator rbegin();  // 返回指向起始的逆向迭代器
iterator rend();  // 返回指向末尾的逆向迭代器
// 容量
bool empty();  // 检查容器是否为空
size_type size();  // 返回容纳的元素数
size_type max_size();  // 返回根据系统或库实现限制的容器可保有的元素最大数量
```

```
void reserve(size_type new_cap);  // 增加vector的容量到大于或等于new_cap的值。不会修改size。若new_cap > capacity，则会重新分配存储空间，且所有迭代器失效。
size_type capacity();  // 返回容器当前已为之分配空间的元素数
// 修改器
void clear();  // 清除所有元素，size变为0，capacity不变。
iterator insert(iterator pos, const T& value);  // 插入元素，返回指向被插入value的迭代器
iterator insert(const_iterator pos, size_type count, const T& value);  // 插入count个value，返回第一个被插入元素的迭代器，若count == 0返回pos
iterator erase(iterator pos);  // 移除位于pos的元素
iterator erase(iterator first, iterator last);  // 移除范围[first, last)中的元素
void push_back(const T& value);  // 将元素添加到容器末尾，初始化新元素为value的副本
void push_back(T&& value);  // 移动value进新元素
template< class... Args >
constexpr reference emplace_back(Args&&... args);  // 将元素添加到容器末尾，返回被插入元素的引用
void pop_back();  // 移除容器的末元素
void resize(size_type count);  // 重设容器大小以容纳count个元素
void swap(vector& other);  // 将内容与other的交换
```

关于逆向迭代器的说明图和使用示例如下。




```
    for (auto it = v.rbegin(); it != v.rend(); ++it) {  // 使用逆向迭代器遍历数组
        cout << *it << endl;
    }
```

stack

queue

deque

priority_queue

string

set, unordered_set

map, unordered_map

头文件algorithm

常用头文件

cstdlib

cctype

cmath

complex

其他

数与字符串的转换

字符串转为数

```
#include <cstdlib>

double atof(const char* str);
int atoi(const char* str);
long int atol(const char* str);
long long int atoll(const char* str);
```

格式化读取字符串内容的例子：

```
#include <cstdio>

char info[] = "ID: 120 Name: John Age: 25";
int ID;
char name[20];
int age;
sscanf(info, "ID: %d Name: %s Age: %d", &sno, name, &age);
```

数转为字符串

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

```
string to_string(int n);
string to_string(int val);
string to_string(long val);
string to_string(long long val);
string to_string(unsigned val);
string to_string(unsigned long val);
string to_string(unsigned long long val);
string to_string(float val);
string to_string(double val);
string to_string(long double val);
```

格式化输出到字符串的例子：

```
#include <cstdio>

char buffer[50];
int num = 10;
sprintf(buffer, "Value of num = %d.", num);
```

格式说明符：

- **%d**: 读取整数。
- **%f**: 读取浮点数。
- **%s**: 读取字符串。
- **%c**: 读取字符。
- **%x**: 读取十六进制整数。
- **%o**: 读取八进制整数。
- **%u**: 读取无符号整数。