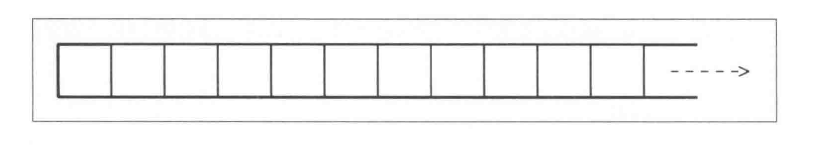
vector类似一个动态数组，有序集合，支持随机访问。

如果在末端添加或删除元素，vector的效率非常高。但如果在其它位置添加或删除元素，效率就不怎么样。



vector的大小（size）和容量（capacity）

容量capactiy是指vector实际能够容纳的元素个数，超过了这个容量，vector会重新分配内部内存。

1. 一旦重新分配内存，vector元素相关的所有reference、pointer、iterator都会失效。
2. 内存重新分配耗时较长。

可以用reserve()保留适当容量，避免重新分配内存。

vector<int> v;

v.reserve(80); // 容量为80

vector<int> v(10); // 容量为10

例：程序stl\_test44

*vector*<int> v1{1, 2, 3, 4, 5};

*vector*<int> v2;

v2.*assign*(v1.*begin*(), v1.*end*());

PrintElements(v2, "assign: ");

// 使用[],front(), back(), at访问vector

// 需要先进行判断确保索引有效

if (v2.*size*() > 5)

{

v2[5] = 3;

v2.*at*(5) = 4;

}

if (!v2.*empty*())

{

*cout* << "front " << v2.*front*() << *endl*;

}

v2.*push\_back*(3);

v2.*push\_back*(3);

PrintElements(v2, "Add: ");

// 删除值为3的元素

v2.*erase*(*remove\_if*(v2.*begin*(), v2.*end*(),

*bind*(*equal\_to*<int>(), *std*::*placeholders*::*\_1*, 3)), v2.*end*());

PrintElements(v2, "Delete: ");

// vector当作C-style数组

*vector*<char> v3;

v3.*resize*(41);

*strcpy*(v3.*data*(), "hello, world");

*cout* << v3.*data*() << *endl*;

*vector*<*string*> sentence;

sentence.*reserve*(5);

sentence.*push\_back*("Hello, ");

sentence.*insert*(sentence.*end*(), {"how", "are", "you", "?"});

*cout* << "original: " << *endl*;

for (const auto& elem : sentence)

{

*cout* << elem.*c\_str*() << " ";

}

*cout* << *endl*;

if (sentence.*size*() > 3)

{

(sentence[1], sentence[3]);

}

sentence.*insert*(*find*(sentence.*begin*(), sentence.*end*(), "?"), "always");

sentence.*back*() = "!";

*cout* << "modify: " << *endl*;

for (const auto& elem : sentence)

{

*cout* << elem.*c\_str*() << " ";

}

*cout* << *endl*;

assign: 1 2 3 4 5

front 1

Add: 1 2 3 4 5 3 3

Delete: 1 2 4 5

hello, world

original:

Hello, how are you ?

modify:

Hello, how are you always !