```
//vector的模拟实现
#define _CRT_SECURE_NO_WARNINGS
#include<iostream>
#include<assert.h>
using namespace std;

namespace bit
{
	template<typename T>
	class vector
	{
	public:
		typedef T* iterator;  //vector中迭代器是原生指针
		typedef const T* const_iterator;
	public:
		iterator begin()
		{return start;}
		iterator end()
		{return finish;}
	public:
		vector() :start(nullptr), finish(nullptr), end_of_storage(nullptr)
		{}
		~vector()
		{
			delete[]start;
			start = finish = end_of_storage = nullptr;
		}
	public:
		size_t size()const
		{return finish - start;	}  //两个指针相减得到的是指针区间上的元素个数
		size_t capacity()const
		{return end_of_storage - start;	}
	public:
		T& operator[](size_t i)
		{
			assert(i < size());
			return start[i];
		}
		void push_back(const T& value)
		{
			insert(end(), value);
		}
	public:
		iterator insert(iterator pos, const T& x)
```

```
    {
        if (size() == capacity())
        {
            int new_capacity = capacity() == 0 ? 1 : capacity() * 2;
            reserve(new_capacity);
        }

        iterator p = finish;
        if (pos == nullptr)
            * finish = x;
        else
        {
            while (p != pos)
            {
                *p = *(p - 1);
                p--;
            }
            *pos = x;
        }
        finish++;
        return pos;
    }
public:
    void resize(size_t n, const T& value = T())
    {
        if (n < size())
        {
            finish = start + n;
            return;
        }
        if (n > capacity())
        {
            reserve(n);
        }

        iterator p = finish;
        finish = finish + n - size();
        while (p != finish)
        {
            *p = value;
            p++;
        }

        //int len = n - size();
        //for(int i=0; i<len; ++i)
        //{
        //    *finish++ = value;
```

```
            //}
        }
        void reserve(size_t n)
        {
            if (n > capacity())
            {
                T* tmp = new T[n];
                int old_size = size();
                for (int i = 0; i < old_size; ++i)
                {
                    tmp[i] = start[i];
                }
                delete[]start;
                start = tmp;
                finish = start + old_size;
                end_of_storage = start + n;
            }
        }
    public:
        iterator start;  //vector元素的起始位置
        iterator finish;  //真实存储的元素结尾
        iterator end_of_storage;  //vector真实容量
    };
};


int main()
{
    bit::vector<int> v;
    cout << "size = " << v.size() << endl;
    cout << "capacity = " << v.capacity() << endl;
    v.push_back(1);
    cout << "size = " << v.size() << endl;
    cout << "capacity = " << v.capacity() << endl;

    for (int i = 0; i < v.size(); ++i)
    {
        cout << v[i] << " ";
    }
    cout << endl;
    return 0;
}
```

size

capacity

resize

reserve

insert

push_back

pop_back