**源代码：**

template <typename T1>

friend ostream &operator<< (ostream&, const miniVector<T1>&);

// assignment operator.

// Postcondition: output current vector data.

template <typename T1>

friend istream &operator>> (istream& , miniVector<T1>&);

// assignment operator.

// Postcondition: input vector data.

void insert(const int place, const T& item);

// insert an item at a certain place of the vector.

// Postcondition: the vector size is increased by 1.

void cut(const int place, const T& item);

// delete an item at a certain place of the vector.

// Postcondition: there exist an element equals the item.

// assignment operator.

template <typename T1>

ostream &operator<< (ostream& out, const miniVector<T1>& v)

{

int n = v.size();

for (int i = 0; i < n; i++)

out << v.vArr[i] << " ";

return out;

}

// assignment operator.

template <typename T1>

istream &operator>> (istream& in, miniVector<T1>& v)

{

T1 temp;

do {

in >> temp;

v.push\_back(temp);

} while (getchar() != '\n');

return in;

}

// insert an item at some place of the vector.

template <typename T>

void miniVector<T>::insert(const int place, const T& item)

{

if (vSize == vCapacity){

if (vCapacity == 0){

// if capacity is 0, set capacity to 1.

// set copy to false because there are

// no existing elements.

reserve(1, false);

vArr[0] = item;

// the place is meanless here.

}

else{

reserve(2 \* vCapacity, true);

// if capacity is full,double the capacity.

for (int i = vSize; i > place; i--){

vArr[i] = vArr[i - 1];

}

// move every element after the insert place one

// address behind.

vArr[place] = item;

// after the movement, insert the place number.

}

}

else{

// if vSize < vCapacity , the room is big enough.

// just move the elements and insert the needed one.

for (int i = vSize; i > place; i--){

vArr[i] = vArr[i - 1];

}

// add item to the list.

vArr[place] = item;

}

// update vSize.

vSize++;

}

// delete an item at a certain place of the vector.

template <typename T>

void miniVector<T>::cut(const int place, const T& item)

{

// if inserted place is larger than vSize or the element

// in the place is not equals to the item, output error.

if (place>vSize || vArr[place] != item){

cout << "Find error!"<<endl;

return;

}

// just let every element after the place move one pace forward

// to cover the original data.

else

for (int i = place; i < vSize; i++)

vArr[i] = vArr[i + 1];

// update vSize.

vSize--;

}

**测试实例：**

起始数组1：4 2 1；

重载输出数组1；

删除第一位数 4；

重载输出数组1；

插入第二位数 7；

重载输出数组1；

重载输入数组2；

重载输出数组2。

**实例代码：**

#include<iostream>

#include"d\_vector.h"

using namespace std;

int main(){

miniVector<int> ve1;

ve1.push\_back(4);

ve1.push\_back(2);

ve1.push\_back(1);

cout<<"输出结果是："<<ve1<<endl;

ve1.cut(0,4);

cout<<"输出结果是："<<ve1<<endl;

ve1.insert(1,7);

cout<<"输出结果是："<<ve1<<endl;

miniVector<int> ve2;

cout<<"输入：";

cin>>ve2;

cout<<"输出结果是："<<ve2<<endl;

return 0;

}

**运行截图：**

