# project6

3220105096

2024 年 4 月 13 日

## 1 题目要求

Given the declaration of a class template Vector as shown below, implement the bodies of all the member functions. Please also write a main function to test all the facilities of the class Vector.

```
template <class T>
class Vector {
public:
  Vector();                       // creates an empty
     vector
  Vector(int size);               // creates a vector
     for holding 'size' elements
  Vector(const Vector& r);        // the copy ctor
  ~Vector();                      // destructs the
     vector
  T& operator[](int index);       // accesses the
     specified element without bounds checking
  T& at(int index);               // accesses the
     specified element, throws an exception of
                                  // type 'std::
                                     out_of_range' when
                                      index <0 or >=
                                     m_nSize
```

```
11    int size() const;                // return the size of
          the container
12    void push_back(const T& x);      // adds an element to
          the end
13    void clear();                    // clears the
         contents
14    bool empty() const;              // checks whether the
          container is empty
15  private:
16    void inflate();                  // expand the storage
          of the container to a new capacity,
17                                     // e.g. 2*m_nCapacity
18    T *m_pElements;                  // pointer to the
         dynamically allocated storage
19    int m_nSize;                     // the number of
         elements in the container
20    int m_nCapacity;                 // the total number
         of elements that can be held in the
21                                     // allocated storage
22  };
```

Listing 1: Vector

# 2 函数实现

## 2.1 构造函数的实现

```
1     Vector<T>::Vector()
2     {
3         m_nSize = 0;
4         m_nCapacity = 1;
5         m_pElements = new T[m_nCapacity];
6     }
7
```

```
8       template <class T>
9       Vector<T>::Vector(int size)
10      {
11          this->m_nSize = size;
12          this->m_nCapacity = size;
13          this->m_pElements = new T[size];
14      }
15
16      template <class T>
17      Vector<T>::Vector(const Vector& r)
18      {
19          this->m_nSize = r.m_nSize;
20          this->m_nCapacity = r.m_nCapacity;
21          this->m_pElements = new T[m_nCapacity];
22          for ( int i = 0; i < m_nSize; ++i )
23          {
24              m_pElements[i] = r.m_pElements[i];
25          }
26      }
```

Listing 2: Vector()

## 2.2 析构函数

```
1       template <class T>
2       Vector<T>::~Vector()
3       {
4
5           delete[] m_pElements;
6           m_nCapacity = 0;
7           m_nSize = 0;
8           m_pElements = nullptr;
9       }
```

Listing 3: Vector

## 2.3 私有函数 inflate ()

```
1   template <class T>
2   void Vector<T>::inflate()
3   {
4       if(m_nCapacity == 0)
5       {
6           m_nCapacity = 1;
7       }
8       else
9       {
10          m_nCapacity *= 2 ;
11      }
12
13      T* newElements = new T[m_nCapacity];
14      for (int i = 0; i < m_nSize; i++)
15      {
16          newElements[i] = m_pElements[i];
17      }
18
19      delete[] m_pElements;
20      m_pElements = newElements;
21  }
```

Listing 4: inflate()

## 2.4 公有函数

```
1   template<class T>
2   int Vector<T>::size() const
```

```
3       {
4           return m_nSize;
5       }
6
7       template <class T>
8       T& Vector<T>::operator[](int index)
9       {
10          return m_pElements[index];
11      }
12
13      template <class T>
14      T& Vector<T>::at(int index)
15      {
16          if (index < 0 || index >= m_nSize)
17          {
18              throw std::out_of_range("Index out of
                    range");
19          }
20          return m_pElements[index];
21      }
22      template <class T>
23      bool Vector<T>::empty() const
24      {
25          return m_nSize == 0;
26      }
27
28      template <class T>
29      void Vector<T>::clear()
30      {
31          m_nCapacity = 0;
32          m_nSize = 0;
33          m_pElements = nullptr;
34          delete[] m_pElements;
```

```
35      }
36
37      template <class T>
38      void Vector<T>::push_back(const T& x)
39      {
40          if (m_nSize == m_nCapacity)
41          {
42              inflate();
43          }
44          m_pElements[m_nSize++] = x;
45      }
46
47  #endif
```

Listing 5: public

# 3 测试

```
1       Vector <int> a1;
2       int num;
3       for ( int i = 0; i < 100; ++i )
4       {
5
6           a1.push_back(i);
7       }
8
9       for ( int i = 0; i < 100; ++i )
10      {
11          std::cout<<a1[i]<<" ";
12      }
13      std::cout<<std::endl;
14      //std::cout<<a1.at(1000)<<std::endl;
15
```

```
16
17      Vector <int> a2(a1);
18      std::cout<<a2.size()<<std::endl;
19
20      Vector <int> a3(10000);
21      std::cout<<a3.size()<<std::endl;
22
23
24      Vector <std::string> str;
25      std::string s0;
26      std::cout<<"Input the first string"<<std::endl;
27      std::cin>>s0;
28      str.push_back(s0);
29      std::cout<<str[0]<<std::endl;
30      std::cout<<"Input the Second string"<<std::endl;
31      std::cin>>s0;
32      str.push_back(s0);
33      std::cout<<str[1]<<std::endl;
34
35      a1.~Vector();
36      a2.clear();
37      str.clear();
38
39      std::cout<<a1.size()<<std::endl;
40      std::cout<<a2.size()<<std::endl;
41      std::cout<<str.size()<<std::endl;
```

Listing 6: main

首先 Vector a1：a1 的预先设置内存为 1

```
1       Vector<T>::Vector()
2       {
3           m_nSize = 0;
4           m_nCapacity = 1;
```

```
5           m_pElements = new T[m_nCapacity];
6       }
```

对 a1 进行 100 次插入操作。超出原有的内存，自动建立新的内存，说明 inflate 执行成功。

对 a1 中的内容执行输出操作，0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99

说明 push_back 函数正常运行。

测试第二个构造函数，

Vector <int> a3(10000);

std::cout«a3.size()«std::endl; 输出为 10000，说明第二个构造函数正常运行。

测试第三个构造函数：

Vector <int> a2(a1);

std::cout«a2.size()«std::endl;

for ( int i = 0; i < 100; ++i ) std::cout«a2[i]«" ";

输出 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99

说明第三个构造函数正确

测试 at

std::cout«a1.at(1000)«std::endl;

抛出报错：

输出错误消息:

```
terminate called after throwing an instance of std::out_of_range
what():  Index out of range
```

说明正确。

测试其他类和 operator[]:

```
Vector <std::string> str;
std::string s0;
std::cout<<"Input the first string"<<std::endl;
std::cin>>s0;
str.push_back(s0);
std::cout<<str.operator[](0)<<std::endl;
std::cout<<"Input the Second string"<<std::endl;
std::cin>>s0;
str.push_back(s0);
std::cout<<str.operator[](1)<<std::endl;
```

输出为:

```
Input the first string
zju
zju
Input the Second string
oop
oop
```

说明 Vector 在类之间通用，并且可以使用 operator[]

最后测试析构函数和 clear 函数

```
a1.~Vector();
a2.clear();
str.clear();


std::cout<<a1.size()<<std::endl;
std::cout<<a2.size()<<std::endl;
std::cout<<str.size()<<std::endl;
```

输出为

```
0
0
```

```
0
```

说明 clear 函数与析构函数正确。