目录

**[第三章作业 1](#_Toc27261)**

[第1题 1](#_Toc14528)

[第2题 1](#_Toc18853)

[第3题 2](#_Toc6170)

[第4题 2](#_Toc3268)

[第5题 3](#_Toc9305)

[第6题 3](#_Toc31696)

**[第四章作业](#_Toc24053)** [5](#_Toc24053)

[第1题 5](#_Toc29783)

[第2题 5](#_Toc22385)

[第3题 6](#_Toc25936)

[第4题 6](#_Toc22627)

[第5题 7](#_Toc11680)

[第6题 8](#_Toc5290)

**[附录A MyArray.h 9](#_Toc14301)**

**[附录B MyArray.cpp 12](#_Toc12738)**

**[附录C MyArrayTest.cpp 16](#_Toc5189)**

# 

# 第三章作业

## 第1题

如下函代码是否有编译错误？如果有请说明原因

float g(int i ) {}

int g(int j) {}

答：编译错误，无法仅按返回类型区分不同函数，两者参数列表相同

## 第2题

如下函代码是否有编译错误？如果有请说明原因

int f(int x) {}

int f(int x, int y = 0){}

int f(int x, ...) { }

int test2(){ f(1,2,3); }

答：由于函数体为空，没有返回值，所以会编译报错。

如果假定函数体存在合适内容，那么不会报错，程序可以正确调用第三个函数f，因为传递了三个参数。如果只传递一个参数，才会编译错误（无法区分第一个f和第二个f）  
假定函数体内容如下

## 第3题

如下函代码是否有编译错误？如果有请说明原因

int f(int x) {}

int f(int x, int y = 0) {}

int f(int x, ...) { }

int test3() { f(1); }

答：报错，无法区分第一个f和第二个f，对重载函数的调用不明确

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 严重性 | 代码 | 说明 | 项目 | 文件 | 行 | 禁止显示状态 | 详细信息 |
| 错误 | C2668 | “f”: 对重载函数的调用不明确 | Project1 | C:\Users\muke\source\repos\Project1\源.cpp | 9 |  |  |

## 第4题

函数的get的定义能否让函数test4编译通过并正常运行？如果不行请说明原因，并修改get函数使得 test4能编译通过并正常运行

int get(int\* p, int index)

{

return p[index];

}

void test4()

{

int a[10];

for (int i = 0; i < 10; i++)

{

get(a, i) = i;

}

}

答：不能正常运行。Get函数的返回的是右值，不可进行赋值操作。修改返回为引用即可  
int& get(int\* p, int index)

{

return p[index];

}

## 第5题

给定下面二个重载的函数定义

void f(int\* p) { std::cout << "void f(int \* p)" << std::endl; }

void f(const int \*p){ std::cout << "void f(const int \* p)" << std::endl; }

下面的语句1和语句2分别调用了哪个f

void test5()

{

int a[2] = { 1,2 };

const int b[2] = { 1,2 };

f(a); f(b);

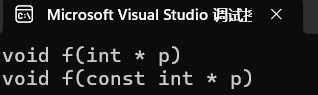
}

答：由于数组名等价于指向第一个元素的指针

所以语句1对应int\*，语句2对应const int \*

· 语句 1：f(a) 调用的是 void f(int\* p)。

· 语句 2：f(b) 调用的是 void f(const int\* p)。



## **第6题**

下面的代码是否可以编译通过？如果不行请说明原因

void print(const char\* p) {}

void print(int i) {}

void test6()

{

void print(int);

print("Hello");

}

答：不通过。由于在函数体内声明print(int)，根据作用域的优先级，print语句将会调用内部参数为int的print，而"Hello" 是const char\*类型，不匹配。

第四章作业

## 第1题

下面代码中对象a1和a2的数据成员i值分别是多少

class A {

public:

int i;

} a1;

void test1() {

static A a2;

std::cout << "a1.i = " << a1.i << std::endl;

std::cout << "a2.i = " << a2.i << std::endl;

}

如果在test1里定义A a3，是否可以打印出a3.i ？

答：a1和a2的i均为0，由于a1是全局变量，a2是静态变量，其成员变量会初始化为0；

如果定义普通对象a3，由于未初始化其i的值，将会报错（VS2022)

在VsCode环境下，a3.i也被初始化为0

在其他环境下，也有可能输出未定义的值（随机数据）

## **第2题**

下面的代码中语句1和语句2编译是否报错？请说明原因

class A

{

int x;

public:

A(int x) :x(x) {}

};

A a;

A b(1);

答：语句1会报错，因为没有对应的无参构造函数。

语句2不会报错，因为有对应的带参数的构造函数。

## **第3题**

给定A类的定义

class A {

public :int x;

public:

A(int x) :x(x) {}

};

请为class B添加合适的构造函数，使得下面的语句能够编译通过

class B {

const A a;

const A& ra;

public:

//构造函数

} b;

答：B():a(0),ra(a){}

由于在定义 class B 时，后面直接实例化了一个对象 b，并且没有提供参数，因此 B 需要一个无参的默认构造函数。同时，由于 B 包含了 const A a 和 const A& ra，我们必须确保在默认构造函数中初始化这两个成员。因为 const 成员和引用成员一旦定义，就必须立即被初始化，不能留空。

## **第4题**

给定下面类的定义

class A {

int x;

public:

A(int x) :x(x) {}

};

class B {

const int i = 0;

};

class E包含如下数据成员

class E {

A a;

B b;

//则下面class E的构造函数哪些是正确的,哪些是错误的？请说明原因

public:

E() :a(1) {} //构造函数1

E() :b() {} //构造函数2

E() :a(1), b() {} //构造函数3

};

答：13正确，2错误。

1. **正确**。因为 a 被初始化为 1，b 的 i 被默认初始化为 0，符合要求。
2. **错误**，因为 a 没有被初始化，而a又没有无参构造函数
3. **正确**。ab都正确初始化

## **第5题**

下面代码存在哪些编译错误，请说明原因

class A {

A() {}

} a;

class B {

int i;

int j;

public:

explicit B(int x, int y = 0) :i(x), j(y) {}

};

B b1;

b2 = 1;

答：

1. **A的构造函数为私有**，在外部创建对象时无法调用构造函数

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 严重性 | 代码 | 说明 | 项目 | 文件 | 行 | 禁止显示状态 | 详细信息 |
| 错误 | C2248 | “A::A”: 无法访问 private 成员(在“A”类中声明) | Project1 | C:\Users\muke\source\repos\Project1\源.cpp | 8 |  |  |

1. B**没有提供无参构造函数**，B b1将会报错

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 严重性 | 代码 | 说明 | 项目 | 文件 | 行 | 禁止显示状态 | 详细信息 |
| 错误 | C2512 | “B”: 没有合适的默认构造函数可用 | Project1 | C:\Users\muke\source\repos\Project1\源.cpp | 15 |  |  |

1. B 类的构造函数被声明为 **explicit**，这意味着它禁止隐式转换,1无法转换为B类型

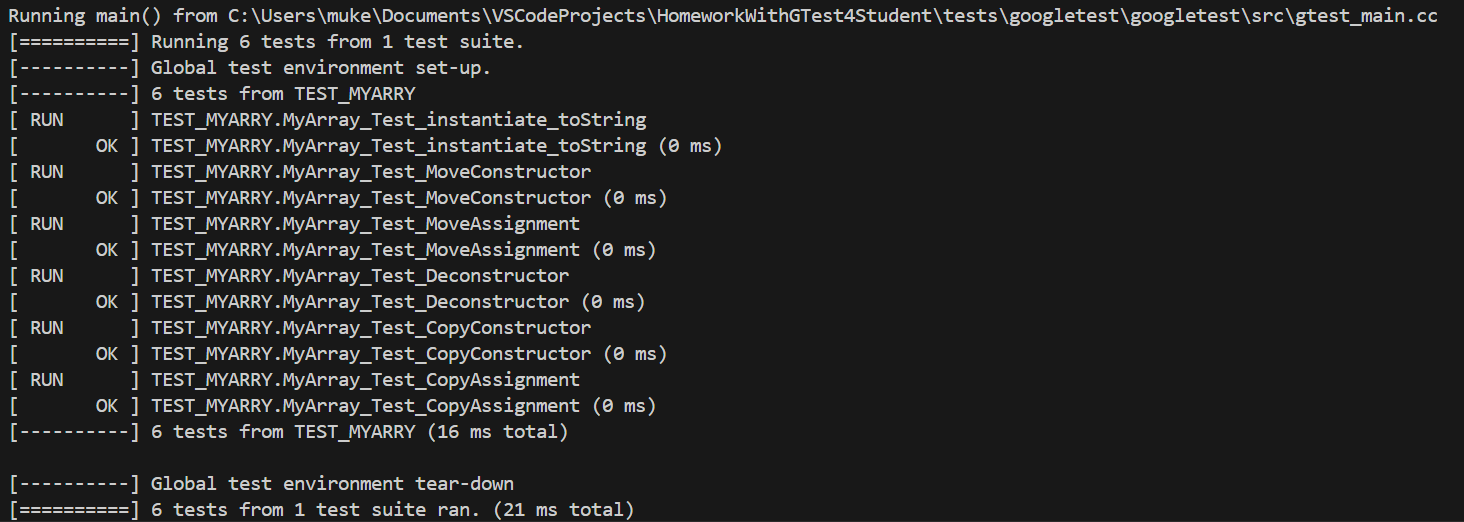
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 严重性 | 代码 | 说明 | 项目 | 文件 | 行 | 禁止显示状态 | 详细信息 |
| 错误 | C2440 | “初始化”: 无法从“int”转换为“B” | Project1 | C:\Users\muke\source\repos\Project1\源.cpp | 18 |  |  |

## **第6题**

编程题

一维整型数组MyArray的定义如下，请实现相应的函数成员

答：运行结果如下，源代码见附录



# 附录A MyArray.h

1. #pragma once
2. #include <string>
3. class MyArray
4. {
5. private:
6. int size = 0;           *// 数组大小*
7. int \*const p = nullptr; *// 指向动态分配的内存，保存数组的内容*
8. public:
9. MyArray(int size = 10); *// 构造函数，参数size指定数组大小*
10. MyArray(const MyArray &old); *// 拷贝构造函数，要求实现深拷贝*
11. MyArray &operator=(const MyArray &rhs); *// 重载=，要求实现深拷贝*
12. MyArray(MyArray &&old) noexcept; *// 移动拷贝*
13. MyArray &operator=(MyArray &&rhs) noexcept; *// 移动=*
14. ~MyArray(); *// 析构函数，要求能防止反复释放资源*
15. int length(); *// 返回数组大小*
16. int &get(int index); *// 返回下标为index的元素，不考虑越界情况*
17. *// 以下方法测试用*
18. *// 一个对象是否为空。如果size或p有一个为0，则返回true*
19. bool isEmpty() const;
20. *// 比较二个MyArray对象是否相等。当二个MyArray对象都不是Empty，size一样，数组的内容完全一样时，这二个MyArray对象才相等*
21. bool equals(const MyArray &other) const;
22. *// 将MyArray数组内容变成字符串。要求数组元素之间用空格分开，形如这样的格式"0 1 2 3 4 5"*
23. *// 如果MyArray对象是Empty，则返回string对象的内容为""*
24. std::string toString();
25. *// 返回内部p指针，仅仅用于测试，不能用于任何其他地方*
26. const int \*const getP();
27. };

# 附录B MyArray.cpp

1. #include "..\..\include\ch4\_homework\MyArray.h"
2. #include <iostream>
3. *// 构造函数*
4. MyArray::MyArray(int size) : size(size), p(new int[size])
5. {
6. for (int i = 0; i < size; ++i)
7. {
8. p[i] = i + 1;
9. }
10. }
11. *// 拷贝构造函数（深拷贝）*
12. MyArray::MyArray(const MyArray &old) : size(old.size), p(new int[old.size])
13. {
14. for (int i = 0; i < size; ++i)
15. {
16. p[i] = old.p[i];
17. }
18. }
19. *// 移动构造函数*
20. MyArray::MyArray(MyArray &&old) noexcept : size(old.size), p(old.p)
21. {
22. \*(const\_cast<int \*\*>(&old.p)) = nullptr; *// 防止旧对象释放资源*
23. old.size = 0;
24. }
25. *// 拷贝赋值运算符（深拷贝）*
26. MyArray &MyArray::operator=(const MyArray &rhs)
27. {
28. if (this == &rhs)
29. {
30. return \*this; *// 防止自赋值*
31. }
32. if (size != rhs.size)
33. {
34. delete[] p;                                    *// 释放旧资源*
35. \*(const\_cast<int \*\*>(&p)) = new int[rhs.size]; *// 分配新内存*
36. size = rhs.size;
37. }
38. for (int i = 0; i < size; ++i)
39. {
40. p[i] = rhs.p[i];
41. }
42. return \*this;
43. }
44. *// 移动赋值运算符*
45. MyArray &MyArray::operator=(MyArray &&rhs) noexcept
46. {
47. if (this != &rhs)
48. {
49. delete[] p;
50. \*(const\_cast<int \*\*>(&p)) = rhs.p;
51. size = rhs.size;
52. \*(const\_cast<int \*\*>(&rhs.p)) = nullptr;
53. rhs.size = 0;
54. }
55. return \*this;
56. }
57. *// 析构函数*
58. MyArray::~MyArray()
59. {
60. if(p == nullptr)
61. {
62. size=0;
63. return;
64. }
65. else
66. {
67. delete[] p;
68. \*(const\_cast<int \*\*>(&p)) = nullptr;
69. size = 0;
70. }
71. }
72. *// 返回数组大小*
73. int MyArray::length()
74. {
75. return size;
76. }
77. *// 返回下标为index的元素*
78. int &MyArray::get(int index)
79. {
80. return p[index];
81. }
82. *// 判断数组是否为空*
83. bool MyArray::isEmpty() const
84. {
85. return size == 0 || p == nullptr;
86. }
87. *// 比较两个MyArray对象是否相等*
88. bool MyArray::equals(const MyArray &other) const
89. {
90. if (size != other.size || isEmpty() || other.isEmpty())
91. {
92. return false;
93. }
94. for (int i = 0; i < size; ++i)
95. {
96. if (p[i] != other.p[i])
97. {
98. return false;
99. }
100. }
101. return true;
102. }
103. *// 将MyArray数组内容转换为字符串*
104. std::string MyArray::toString()
105. {
106. if (isEmpty())
107. {
108. return "";
109. }
110. std::string result;
111. for (int i = 0; i < size; ++i)
112. {
113. result += std::to\_string(p[i]);
114. if (i < size - 1)
115. result += " ";
116. }
117. return result;
118. }
119. *// 返回内部指针p，仅用于测试*
120. const int \*const MyArray::getP()
121. {
122. return p;
123. }

# 附录C MyArrayTest.cpp

1. #include <gtest/gtest.h>
2. #include "../include/ch4\_homework/MyArray.h"
3. #include <iostream>
4. using namespace std;
5. */\**
6. 在实现好MyArray类基础上，编写下面的代码来测试验证MyArray的功能
7. \*/
8. */\**
9. 请构建构建大小为size的MyArray对象a，要求元素的值分别为1 2 3 4 ... size
10. 并返回该对象 ，用于测试
11. \*/
12. MyArray buildArrayInstace(int size)
13. {
14. MyArray a(size);
15. return a;
16. }
17. */\**
18. 测试MyArray的实例化以及toString函数
19. \*/
20. TEST(TEST\_MYARRY, MyArray\_Test\_instantiate\_toString)
21. {
22. *// given*
23. MyArray a = buildArrayInstace(5);
24. *// when*
25. *// 得到对象a的字符串表示*
26. std::string content = a.toString();
27. *// then*
28. std::string target("1 2 3 4 5");
29. ASSERT\_EQ(target, content);
30. }
31. */\**
32. 测试MyArray的移动构造函数
33. \*/
34. TEST(TEST\_MYARRY, MyArray\_Test\_MoveConstructor)
35. {
36. *// given*
37. MyArray a = buildArrayInstace(10);
38. const int \*const pa = a.getP(); *// 保留a的内部指针*
39. *// when*
40. MyArray b(std::move(a));
41. *// 检测内容是否相等*
42. std::string target("1 2 3 4 5 6 7 8 9 10");
43. *// then*
44. ASSERT\_EQ(target, b.toString());
45. *// 检查是否为浅拷贝*
46. *// then*
47. ASSERT\_EQ(pa, b.getP());
48. *// 检查a对象是不是为可以安全析构*
49. *// then*
50. ASSERT\_EQ(0, a.length());
51. ASSERT\_EQ(nullptr, a.getP());
52. }
53. */\**
54. 测试MyArray的move assignment（移动=）
55. \*/
56. TEST(TEST\_MYARRY, MyArray\_Test\_MoveAssignment)
57. {
58. *// given*
59. MyArray a = buildArrayInstace(5);
60. MyArray b = buildArrayInstace(10);
61. const int \*const pa = a.getP(); *// 保留a的内部指针*
62. *// when*
63. b = std::move(a);
64. *// 检测内容是否相等*
65. std::string target("1 2 3 4 5");
66. *// then*
67. ASSERT\_EQ(target, b.toString());
68. *// 检查是否为浅拷贝*
69. *// then*
70. ASSERT\_EQ(pa, b.getP());
71. *// 检查a对象是不是为可以安全析构*
72. *// then*
73. ASSERT\_EQ(0, a.length());
74. ASSERT\_EQ(nullptr, a.getP());
75. }
76. */\**
77. 测试MyArray的析构函数
78. \*/
79. TEST(TEST\_MYARRY, MyArray\_Test\_Deconstructor)
80. {
81. *// given*
82. MyArray a = buildArrayInstace(5);
83. int rtn = 0;
84. *// when*
85. a.~MyArray();
86. a.~MyArray();
87. rtn = 1; *// 如果析构函数实现正确，会执行到rtn = 1;*
88. *// then*
89. ASSERT\_EQ(0, a.length());
90. ASSERT\_EQ(nullptr, a.getP());
91. ASSERT\_EQ(1, rtn);
92. }
93. */\**
94. 测试MyArray的拷贝构造函数
95. \*/
96. TEST(TEST\_MYARRY, MyArray\_Test\_CopyConstructor)
97. {
98. *// given*
99. MyArray a = buildArrayInstace(10);
100. *// 检测内容是否相等*
101. *// when*
102. MyArray b(a);
103. *// then*
104. ASSERT\_EQ(true, a.equals(b));
105. *// 检查是否为深拷贝*
106. *// when*
107. a.~MyArray(); *// 析构对象a*
108. *// then*
109. std::string target("1 2 3 4 5 6 7 8 9 10");
110. ASSERT\_EQ(target, b.toString()); *// 这时对象b的内容应该还在，深拷贝*
111. }
112. */\**
113. 测试MyArray的copy assignment（拷贝=）
114. \*/
115. TEST(TEST\_MYARRY, MyArray\_Test\_CopyAssignment)
116. {
117. *// given*
118. MyArray a = buildArrayInstace(10);
119. MyArray b = buildArrayInstace(5);
120. *// 检测内容是否相等*
121. *// when*
122. a = b;
123. *// then*
124. ASSERT\_EQ(true, a.equals(b));
125. *// 检查是否为深拷贝*
126. *// when*
127. b.~MyArray();
128. *// then*
129. std::string target("1 2 3 4 5");
130. ASSERT\_EQ(target, a.toString());
131. }