2025第二次作业 选择题

所有题目编译时添加 `--std=c++11`

刷新

1. 【多选题】关于构造函数和析构函数，阅读以下代码，并选出正确的说法：

```
#include <iostream>

class Test {
public:
    int data = -1;
    Test() = default; // (A)
    Test(int dat){ // (B)
        this->data = dat; // (C)
    }
    ~Test(){ // (D)
        std::cout<<data;
    }
};

int main() {
    Test* t = new Test();
    delete t;
    return 0;
}
```

A. 如果将注释 `(A)` 所在的行删除，则该代码**无法**通过编译，原因是 `main` 函数中调用了未定义的缺省构造函数。

B. 如果将注释 `(B)` 所在的行改为 `Test(int dat = 2023){ // (B)`，则该代码**可以**通过编译，运行输出结果为 `-1`。

C. 如果将注释 `(A)` 和 `(C)` 所在的行删除，并将注释 `(B)` 所在的行改为 `Test(int dat = 2023):data(dat){ // (B)`，则该代码 **可以**通过编译，运行输出结果为 `2023`。

D. 如果将注释 `(D)` 所在的行改为 `~Test(int data = 0){ // (D)`，该代码**可以**通过编译，运行输出结果为 `0`。

2. 【单选题】下列说法错误的是

A. 如果类中没有定义任何构造函数，也没有显式删除默认构造函数，则编译器会合成默认构造函数

B. 全局对象在main()函数调用之前构造，在main()函数执行完之前不会被析构

C. 使用全局对象可能会使得代码之间的耦合性增大，因此尽量少用全局变量

D. 假设类A的对象a是类B的数据成员，则类B的对象b执行析构时，会先调用类A的析构函数来析构a，再调用类B的析构函数。

3. 【多选题】在C++中使用new和delete操作符时，下列哪项描述是正确的?

A. 使用new时必须使用delete释放内存。

B. 使用new[]创建的对象数组，可以用delete而不是delete[]释放。

C. 使用delete释放未通过new分配的内存是安全的。

D. new和delete帮助管理动态分配的内存。

4. 【单选题】关于运算符重载，下列说法正确的是：

A. 对于下列代码中的两个运算符重载实现：虽然它们的参数类型都是 `int` 与 `Test` 的组合，这段代码仍**可以通过编译**。这是因为在两个重载实现中， `int` 与 `Test` 在参数中的顺序不同，编译器能够唯一地确定调用哪个运算符实现代码。

```
#include <iostream>
using namespace std;

class Test{
public:
    int data;
    Test(int d):data(d){}
};

Test operator +(const Test& a, const int& b){
    return Test(a.data + b);
}

Test operator +(const int& b, const Test& a){
    return Test(- a.data - b);
}

int main(){
    Test obj(1);
    int data = 2;
    cout << (obj + data).data << endl;
    cout << (data + obj).data << endl;
    return 0;
}
```

B. A 的说法是错误的。这是因为加法满足交换律，因此即便调换 `Test` 和 `int` 的顺序，加法运算的实现仍必须保持一致，编译器在编译时也会检查这一点。如将后一处 `int` 改为 `double`，则可通过编译。

C. A 和 B 的说法都是错误的。运算符重载时，所有参数以及返回值的类型必须完全相同。因此，即便是下列代码中的运算符重载实 现，也**不能通过编译。**

```
class Test{
public:
    int data;
    Test(int d):data(d){}
};

Test operator +(const Test& a, const int& b){
    return Test(a.data + b);
}
```

D. A、B、C 的说法都是错误的。

5. 【多选题】下列说法正确的有

```
#include <iostream>     // cout
#include <cstring>    // strcmp
using namespace std;

char week_name[7][4] = {    "mon", "tu", "wed",
                                "thu", "fri", "sat", "sun"};

class WeekTemperature {
    int temperature[7] = {0};
    int error_temperature = -100;
public:
    int operator[] (const char* name)  // (D)
    {
        for (int i = 0; i < 7; i++) {
            if (strcmp(week_name[i], name) == 0)
                return temperature[i];
        }
        return error_temperature;
    }
};

void ref(int &x) {
    cout << "left " << x << endl;
    // x = 1;  // (C)
}

void ref(int &&x) {
    cout << "right " << x << endl;
    // x = 1;  // (A)
}

int main()
{
    WeekTemperature beijing;
    // beijing["mon"] = -3;  // (B)
    ref(beijing["mon"]);
    cout    << "Monday Temperature: "
            << beijing["mon"] << endl;
    return 0;
}
```

A. 该段代码运行输出是 `right 0\nMonday Temperature: 0\n` （`\n` 代表换行符，下同）。

B. 如果复原 `(A)` 处代码（即取消注释，下同），则代码无法通过编译，因为 `x` 是右值，不能被赋值。

C. 如果复原 `(A)` 处代码，则代码运行输出为 `right 1\nMonday Temperature: 1\n`。

D. 如果将 `(D)` 处代码改为 `int& operator[] (const char* name)`，复原 `(B)` `(C)` 处代码，则代码运行输出为 `left -3\nMonday Temperature: 1\n`。

6. 【单选题】考虑以下C++类定义，哪个描述是正确的?

```
class MyClass {
    int a;
public:
    MyClass(int x) : a(x) {}
    friend void showA(MyClass& x);
    void increment() { ++a; }
};

void showA(MyClass& x) {
    cout << "MyClass a=" << x.a << endl;
}

void incrementAndShowA(MyClass& x) {
    x.increment();
    showA(x);
}
```

A. `showA` 是 `MyClass` 的成员函数。

B. 友元函数 `showA` 可以访问 `MyClass` 的私有成员 `a`。

C. `incrementAndShowA` 函数可以直接修改 `MyClass` 对象的私有成员 `a` 的值，不通过任何公有或友元函数。

D. `incrementAndShowA` 尝试直接访问 `MyClass` 的私有成员 `a`，这将导致编译错误。

7. 【单选题】考虑以下C++类定义，选择正确的选项。

```
class Test {
public:
    static const int num = 10;
    int array[num];
    Test() {}
};
```

A. 类定义因为静态成员 `num` 而编译失败。

B. 静态常量成员 `num` 可以在类内初始化。

C. `array` 的大小必须在构造函数中确定。

D. `num` 必须在类外进行初始化。

8. 【多选题】下列说法中，错误的是：

A. 使用 `new[]` 构造的对象可以使用 `delete` 释放内存，如下面代码所示。

```
class Test{};

int main() {
Test* a = new Test[10];
delete a;
return 0;
}
```

B. 下列代码可以通过编译，但由于在函数 `func` 中返回了局部对象的引用，因此可能在编译时引发警告（Warning），或引起运行时 错误。

```
class Test{};

Test& func(){
Test t;
return t;
}

int main() {
Test& t = func();
return 0;
}
```

C. `int` 类型的常量静态数据成员可以在类内初始化，如下代码所示。

```
class Test{
public:
    static const int x = 0;
};

int main(){
    return 0;
}
```

D. 使用 `delete` 释放指针 `p` 指向的内存单元后， `p` 将指向 `nullptr` 。即下列代码能够正常运行：

```
#include <cassert>

class Test{};

int main(){
Test* p = new Test();
delete p;
assert(p == nullptr);
return 0;
}
```

提交格式

请你提交一个文本文件，第i行是一个由A\B\C\D组成的字符串，代表第i题的答案。

若你不想提交第i题，请将第i行留空。

你提交的文本文件至少要有8行，且前8行必须由A\B\C\D组成，否则将被认为是无效提交。

你可以通过测试点的详细信息，看到评测对你提交文件的解析。若存在问题，请联系助教。

以下给出一个合法的提交答案的例子：

```
A
BC
C
A
B
C
CD
C
```

评测器对第一行的解析结果如下：

```
Valid answer: ['A']
```

注意：作业截止之前本题的评测器只检查提交格式是否正确，不检查具体答案。只要提交格式正确就显示100分。每道题的具体答案会在作业截止之后重新评测，以最后一次提交的答案为准

语言和编译选项

#	名称	编译器	额外参数	代码长度限制
0	answer	cp		1048576 B

递交历史

#	状态	时间
319013	Accepted	2025-03-25 19:56:33

1

当前没有提交权限！