VNUHCM - UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY

HOMEWORK REPORT
COURSE: OBJECT-ORIENTED PROGRAMMING

WEEK 07: ASSIGNMENT 02
CODE CHECKING

Instructor:
Nguyen Le Hoang Dung
Ho Tuan Thanh

Student:
Le Trung Kien 23127075 23CLC08

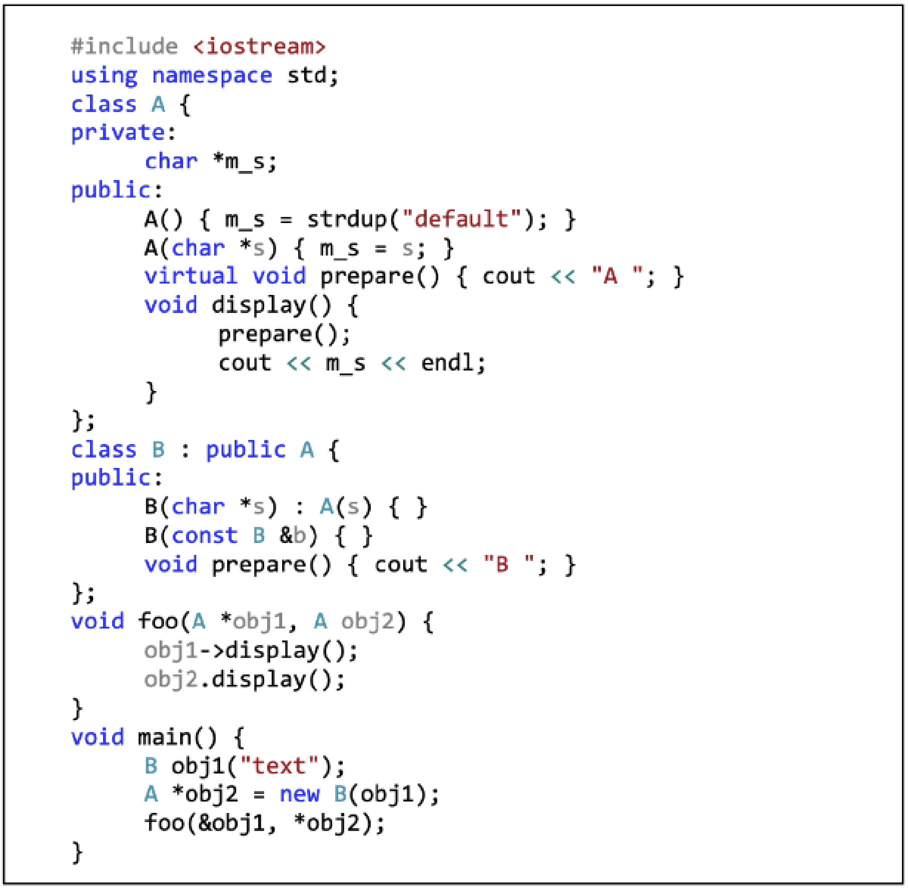
Ho Chi Minh City, 2024

Contents

List of Figures

1 Question 1

In this section, I still use your original provided code.

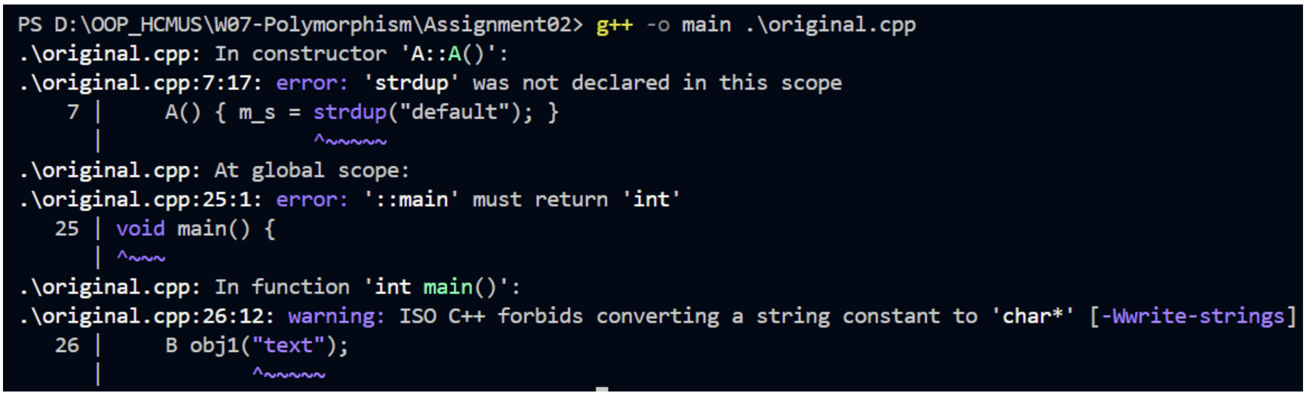


```
#include <iostream>
using namespace std;
class A {
private:
	char *m_s;
public:
	A() { m_s = strdup("default"); }
	A(char *s) { m_s = s; }
	virtual void prepare() { cout << "A "; }
	void display() {
		prepare();
		cout << m_s << endl;
	}
};
class B : public A {
public:
	B(char *s) : A(s) { }
	B(const B &b) { }
	void prepare() { cout << "B "; }
};
void foo(A *obj1, A obj2) {
	obj1->display();
	obj2.display();
}
void main() {
	B obj1("text");
	A *obj2 = new B(obj1);
	foo(&obj1, *obj2);
}
```

Figure 1: Original code

1.1 Compiling errors

When compiling, this is what is printed to the console:



```
PS D:\OOP_HCMUS\W07-Polymorphism\Assignment02> g++ -o main .\original.cpp
.\original.cpp: In constructor 'A::A()':
.\original.cpp:7:17: error: 'strdup' was not declared in this scope
    7 |     A() { m_s = strdup("default"); }
      |                 ^~~~~~
.\original.cpp: At global scope:
.\original.cpp:25:1: error: '::main' must return 'int'
   25 | void main() {
      | ^~~~
.\original.cpp: In function 'int main()':
.\original.cpp:26:12: warning: ISO C++ forbids converting a string constant to 'char*' [-Wwrite-strings]
   26 |     B obj1("text");
      |            ^~~~~~
```

Figure 2: Compiling errors

1.2 Brief explanation

1.2.1 In constructor `’A::A()’`

```
.\original.cpp:7:17: error: ’strdup’ was not declared in this scope
```

⇒ At line 7, strdup must be declared by including `cstring` library.

```
// Error
    A() { m\_s = strdup("default"); }
```

```
// Solution
#include <cstring> // Include required library
```

1.2.2 At global scope

```
.\original.cpp:25:1: error: ’::main’ must return ’int’
```

⇒ At line 25, the function `main` in C++ must return int data type. However, the function `main` is declared in the original code with void data type. We must change it into int and return 0 at the end of the function `main`.

```
// Error
void main() {
    ...
}
```

```
// Solution
int main() {
    ...
    return 0;
}
```

1.2.3 In function `’int main()’`

```
.\original.cpp:26:12: warning: ISO C++ forbids converting a string constant to
’char*’ [-Wwrite-strings]
```

⇒ In line 26, the data type of `"text"` is `const char*` while the data type of the parameter of `class B`’s parameterized constructor is `char*` (line 17). It is mentioned in the warning that: `"ISO C++ forbids converting a string constant to ’char*’"`. The reason is that it can lead to undefined behavior.

```
// Warning
    B obj1("text");
```

```
// Solution
    char s[] = "text";
    B obj1(s);
```

2 Question 2

In question 1, we have gone over 3 compile-time errors. Now, we will move on to other issues in the provided code.

2.1 Memory issues and corrections

2.1.1 Memory managing in `class A`

The parameterized constructor `A(char *s)` does not copy the string passed to it but instead assigns the pointer `m_s` to `s`. If the `s` is modified or deallocated outside the class, `m_s` will be affected too. Therefore, we need to use `strdup` to copy `s` to `m_s`.

```
// Error
    A(char *s) { m_s = s; }
```

```
// Solution
    A(char *s) { m_s = strdup(s); }
```

2.1.2 Memory leak in `class A` and `class B`

The data member `m_s` is allocated in the default constructor and copy constructor by using `strdup` without deallocating. Therefore, it is essential to create a virtual destructor in `class A` to ensure the proper deallocation in both classes.

```
// Solution: Create destructor
    virtual ~A() { free(m_s); }
```

2.1.3 Memory leak in function `main`

```
// Error
    A *obj2 = new B(obj1); // Without deallocation
```

```
// Solution
    delete obj2; // added this before ending main function
```

2.1.4 Output

The output that is printed to the console is:

```
B text
A default
```

2.2 Other issues and corrections

2.2.1 Missing keyword override

Function `display()` is virtual in `class A` but doesn't have keyword `override` in `class B`.

```
\\ Error
class A {
    ...
public:
    ...
    virtual void prepare() { cout << "A "; }
    ...
};

class B : public A {
public:
    ...
    void prepare() { cout << "B "; }
};
```

```
\\ Solution
class A {
    ...
public:
    ...
    virtual void prepare() { cout << "A "; }
    ...
};

class B : public A {
public:
    ...
    void prepare() override { cout << "B "; }
};
```

2.2.2 Empty copy constructor in `class B`

The copy constructor in `class B` is declared without doing anything in it. So, we need to add a copy constructor in `class A` using deep copy and use it for the copy constructor in `class B`.

```
// Error
```

```
class B : public A {
public:
    B(const B& b) { }
};
```

```
// Solution
class A {
    ...
public:
    ...
    A(const A& other) { this->m_s = strdup(other.m_s); }
    ...
};
class B : public A {
public:
    ...
    B(const B& b) : A(b) { }
    ...
};
```

2.3 Final output

The output that is printed to the console is:

```
B text
A text
```