

2024 年 ソフトウェア演習 2B

第 2 回課題

1 課題 1

ソースコード 1, 2 で実装した Message クラスに対してソースコード 3 に示すプログラムを実行したところ, 実行結果 1 のようにエラーが発生した. その理由を説明しなさい.

リスト 1 Message.h

```
1 #include <iostream>
2
3 class Message {
4 private:
5     char* message;
6
7 public:
8     Message();
9     Message(const char* _message);
10    ~Message();
11
12    void setMessage (const char* _message);
13    char* getMessage (void);
14 };
15
16 std::istream &operator>>(std::istream& stream, Message& obj);
17 std::ostream &operator<<(std::ostream& stream, Message& obj);
```

リスト 2 Message.cpp

```
1 #include <stdlib.h>
2 #include <string.h>
3 #include <string>
4 #include "Message.h"
5
6 Message::Message(): message(nullptr) {
7 }
8
9 Message::Message(const char* _message) {
10     message = new char [strlen(_message) + 1];
11     strcpy (message, _message);
12 }
13
14 Message::~Message() {
15     if (message != nullptr) delete [] message;
16 }
17
18 void Message::setMessage (const char* _message) {
19     if (message) delete [] message;
20     message = new char [strlen(_message) + 1];
21     strcpy (message, _message);
22 }
23
24 char* Message::getMessage (void) {
25     return message;
26 }
27
28 std::istream& operator>>(std::istream& stream, Message& obj) {
29     std::string buffer;
30     std::getline(stream, buffer);
31     obj.setMessage(buffer.c_str());
32     return stream;
```

```

33 }
34
35 std::ostream& operator<<(std::ostream& stream, Message& obj) {
36     stream << obj.getMessage();
37     return stream;
38 }

```

リスト 3 main.cpp

```

1 #include <iostream>
2 #include "Message.h"
3
4 int main (int argc, char *argv[]) {
5     Message obj1("Hello World.");
6     Message obj2 = obj1;
7     std::cout << obj2 << std::endl;
8
9     return 0;
10 }

```

実行結果 1

```

% g++ main.cpp Message.cpp -o program1
% ./program1
Hello World.
free(): double free detected in tcache 2
中止 (コアダンプ)

```

課題のねらい

- (標準の) コピーコンストラクタのしくみを理解する

採点基準

- エラーの原因を正しく説明しているか (20 点)

2 課題 2

Message クラスに対するコピーコンストラクタを実装して, リスト 3 のプログラムを実行してもエラーがでないようにしなさい.

課題のねらい

- コピーコンストラクタを正しく実装できるようにする

採点基準

- コピーコンストラクタを正しく実装できているか (20 点)
- 動作確認の様子を示しているか (10 点)

3 課題 3

Message クラスのメンバ変数を char 型のポインタ変数から string クラスの変数を要素として持つ vector クラスの変数 `std::vector<std::string>` に置き換えたものに修正しなさい。なお、クラスの定義はリスト 4 として、以下の機能を持つものとする。

- 複数の文字列を保持し、それらを string クラスのインスタンスを要素として持つ vector クラスのインスタンスで管理する。
- メンバ関数 `addMessage` で文字列を登録する。この関数を複数回実行することで、複数のメッセージを登録することができるようにする。
- メンバ関数 `getMessage` では関数の引数にメッセージ ID を指定して、その ID (添字) に対応する文字列を関数の戻り値として返す。
- メンバ関数 `showAllMessages` では保持しているすべての文字列を表示する。
- メンバ関数 `getNMessages` では、保持している文字列数を関数の戻り値として返す。

リスト 4 Message.h

```
1 #include <string>
2 #include <vector>
3
4 class Message {
5 private:
6     std::vector<std::string> message;
7
8 public:
9     Message();
10    Message(const std::string& message_string);
11    Message(const std::vector<std::string>& message_vector);
12    ~Message();
13
14    void addMessage (const std::string& message_string);
15    std::string getMessage (int message_id);
16    void showAllMessages (void);
17    int getNMessages (void);
18 };
```

課題のねらい

- vector クラスの使い方を理解する

採点基準

- 課題の仕様を満たしている (20 点)
- クラスの汎用性があるか (クラスとして必要な機能を実装しているかなど) (15 点)
- 動作確認の様子を示しているか (5 点)

自己チェック項目

以下の項目について、1 から 4 までの 4 段階で自己評価しなさい。

4. 十分に理解した 3. 少し不安が残るが理解した 2. 十分には理解できていない 1. まったく理解できない

☐ コピーコンストラクタのしくみを理解した。

- ☐ コピーコンストラクタを実装することができる.
- ☐ `vector` クラスの使い方を理解した.
- ☐ インデント（字下げ）など、一貫したスタイルでプログラムが書ける.
- ☐ プログラムに適切なコメントを入れることができる.
- ☐ 適切な変数名を用いることができる.