# 2025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

```
#include <iostream>

class Message {
private:
  char* message;

public:
  Message();
  Message(const char* _message);
  ~Message();

  void setMessage (const char* _message);
  char* getMessage (void);
};

std::istream &operator>>(std::istream& stream, Message& obj);
std::ostream &operator<<(std::ostream& stream, Message& obj);
```

**リスト2** Message.cpp

```
#include <stdlib.h>
#include <string.h>
#include <string>
#include "Message.h"

Message::Message(): message(nullptr) {
}

Message::Message(const char* _message) {
  message = new char [strlen(_message) + 1];
  strcpy (message, _message);
}

Message::~Message() {
  if (message != nullptr) delete [] message;
}

void Message::setMessage (const char* _message) {
  if (message != nullptr) delete [] message;
  message = new char [strlen(_message) + 1];
  strcpy (message, _message);
}

char* Message::getMessage (void) {
  return message;
}

std::istream& operator>>(std::istream& stream, Message& obj) {
  std::string buffer;
  std::getline(stream, buffer);
  obj.setMessage(buffer.c_str());
  return stream;
```

```
}

std::ostream& operator<<(std::ostream& stream, Message& obj) {
  stream << obj.getMessage();
  return stream;
}
```

**リスト 3** main.cpp

```
#include <iostream>
#include "Message.h"

int main (int argc, char *argv[]) {
  Message obj1("Hello␣World.");
  Message obj2 = obj1;
  std::cout << obj2 << std::endl;

  return 0;
}
```

**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)

### ヒント

- obj1 のメンバ変数 message と obj2 のメンバ変数 message のアドレスを表示してそれぞれの変数がどのアドレスにあるかを確認してみよう．アドレスの表示は次のようにするとできる．

```
printf ("%p", obj1.message);
```

## 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こと.

- 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

```
#include <string>
#include <vector>

class Message {
private:
  std::vector<std::string> message;

public:
  Message();
  Message(const std::string& message_string);
  Message(const std::vector<std::string>& message_vector);
  ~Message();

  void addMessage (const std::string& message_string);
  std::string getMessage (int message_id);
  void showAllMessages (void);
  int getNMessages (void);
};
```

### 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.