

2024 年ソフトウェア演習2B

第 2 回課題

B243392 ALCANDER IMAWAN 2024 年 6 月 18 日

Q1

free(): double free detected in tcache 2 のエラーが発生する理由はデフォルトコピーコンストラクタにはメンバ変数の値がそのままコピーされるため、メンバ変数がポインタ変数の場合、アドレスがコピーされてしまう。問題が今回 char* message の変数はポインタであり、コピーコンストラクトして、obj1 と obj2 の message は同じアドレスを指す。デストラクタが呼び出されたとき、obj1 と obj2 のデストラクタがそれぞれ呼び出されて、同じ message を 2 回消そうとする。そのせいで、double free エラーが発生してしまう。

Q2

プログラム

Message.h

```
1: #include <iostream>
2:
3: class Message {
4:
5: private:
6:     char* message;
7:
8: public:
9:     Message(); // Constructor
10:    Message(const char* _message);
11:    //copy constructor
12:    Message(Message& other);
13:    Message& operator=(Message& other);
14:
15:    ~Message(); //Destructor
```

```

16:
17:     void setMessage(const char* message);
18:     char* getMessage(void); //const char* getMessage(void) const;
19: };
20:
21: //declaration of stream operators
22: std::istream& operator>>(std::istream& stream, Message& obj);
23: std::ostream& operator<<(std::ostream& stream, Message& obj);

```

Message.cpp

```

1: #include "Message.h"
2: #include <string.h>
3: #include <string>
4: #include <stdlib.h>
5:
6: // Constructor initializing
7: Message::Message(): message(nullptr){
8: }
9:
10: Message::Message(const char* _message){
11:     message = new char [strlen(_message) + 1];
12:     strcpy(message, _message);
13: }
14:
15: //Destructor
16: Message::~Message(){
17:     if(message != nullptr) delete[] message;
18: }
19:
20: //this is the function to set Message,
21: //it works by making a new char* with (msg + 1) as its length
22: //then copy msg to message with strcpy
23: void Message::setMessage(const char* _message){

```

```

24:     if(message) delete [] message;
25:     message = new char[strlen(_message) + 1];
26:     strcpy(message, _message);
27: }
28:
29: char* Message::getMessage (void){
30:     return message;
31: }
32:
33: //definition of extraction operator (>>)
34: std::istream& operator>>(std::istream& stream, Message& obj){
35:     //temporary buffer to hold the input, by using buffer, we have better memory
management,
36:     //safer and more robust(prevent overflows), and dynamic memory allocation
37:     std::string buffer;
38:     std::getline(stream, buffer);
39:
40:     //set the message using setMessage function below
41:     obj.setMessage(buffer.c_str());
42:
43:     return stream;
44: }
45:
46: //definition of the insertion operator(<<)
47: std::ostream& operator<<(std::ostream& stream, Message& obj){
48:     stream << obj.getMessage();
49:
50:     return stream;
51: }
52:
53: // Copy Constructor
54: Message::Message(Message& other) {
55:     if (other.message) {

```

```

56:         message = new char[strlen(other.message) + 1];
57:         strcpy(message, other.message);
58:     } else {
59:         message = nullptr;
60:     }
61: }
62:
63: // Copy Assignment Operator
64: Message& Message::operator=(Message& other) {
65:     if (this != &other) {
66:         delete[] message; // Free existing resource
67:         if (other.message) {
68:             message = new char[strlen(other.message) + 1];
69:             strcpy(message, other.message);
70:         } else {
71:             message = nullptr;
72:         }
73:     }
74:     return *this;
75: }

```

main.cpp

```

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

```

動作確認

```
[a243392@xdev09 q2]$ ./q2
Hello World.
[a243392@xdev09 q2]$
```

Q3

プログラム

Message.h

```
1: #include <string>
2: #include <vector>
3: #include <iostream>
4:
5: class Message {
6:
7: private:
8:     std::vector<std::string> message;
9:
10: public:
11:     Message(); // Constructor
12:     Message(const std::string& message_string);
13:     Message(const std::vector<std::string>& message_vector);
14:
15:     ~Message(); //Destructor
16:
17:     void addMessage(const std::string& message_string);
18:     std::string getMessage(int message_id);
19:     void showAllMessages(void);
20:     int getNMessages(void);
21: };
```

Message.cpp

```
1: #include "Message.h"
2: #include <string.h>
```

```
3: #include <string>
4: #include <stdlib.h>
5:
6:
7: // Constructor initializing
8: Message::Message(){}
9:
10: //single message constructor
11: Message::Message(const std::string& message_string){
12:     message.push_back(message_string);
13: }
14:
15: //multiple vector messages constructor
16: Message::Message(const std::vector<std::string>& message_vector){
17:     message = message_vector;
18: }
19:
20: Message::~Message(){} //Destructor
21:
22: //add ma message to the list
23: void Message::addMessage(const std::string& message_string){
24:     message.push_back(message_string);
25: }
26:
27: //get a message by message_id
28: std::string Message::getMessage(int message_id){
29:     if(message_id >= 0 && message_id < message.size()){
30:         return message[message_id];
31:     }
32:     else {
33:         return "Message ID not found";
34:     }
35: }
```

```

36:
37: //showing all messages
38: void Message::showAllMessages(void){
39:     for(const auto& msg : message){
40:         std::cout << msg << std::endl;
41:     }
42: }
43:
44: //get the num of messages
45: int Message::getNMessages(void){
46:     return message.size();
47: }

```

main.cpp

```

1: #include "Message.h"
2:
3: int main (int argc, char *argv[]){
4:     //testing default constructor
5:     Message obj1;
6:     obj1.addMessage("Hello World.");
7:     obj1.addMessage("Hello 2nd one\n");
8:     std::cout << "Number of messages: " << obj1.getNMessages() << std::endl;
9:     obj1.showAllMessages();
10:
11:     //testing single message constructor
12:     Message obj2("This is single message constructor\n");
13:     obj2.showAllMessages();
14:
15:     //testing vector of messages constructor
16:     std::vector<std::string> vec = {"1st message", "2nd message", "3rd message"};
17:     Message obj3(vec);
18:     std::cout << "Number of vector messages: " << obj3.getNMessages() << std::endl;
19:     obj3.showAllMessages();

```

```

20:
21:     //testing getMessage
22:     std::cout << "testing getMessage for index 1 vector : " << obj3.getMessage(1) <<
std::endl;
23:
24:     return 0;
25: }

```

動作確認

```

[a243392@xdev09 q3]$ ./q3
Number of messages: 2
Hello World.
Hello 2nd one

```

This is single message constructor

```

Number of vector messages: 3
1st message
2nd message
3rd message
testing getMessage for index 1 vector : 2nd message
[a243392@xdev09 q3]$ █

```

自己チェック項目

- 4 コピーコンストラクタのしくみを理解した.
- 4 コピーコンストラクタを実装することができる.
- 3 vector クラスの使い方を理解した.
- 4 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.
- 4 プログラムに適切なコメントを入れることができる.
- 4 適切な変数名を用いることができる.