2024 年ソフトウェア演習2B

第1 回課題

**B243392　　ALCANDER IMAWAN　　2024年6月12日**

**Q1**

**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(); // Constructer

10: ~Message(); //Destructor

11:

12: void setMessage(const char\* message);

13: char\* getMessage(void);

14: };

Message.cpp

1: #include "Message.h"

2: #include <cstring> //for std::strlen and std::strcpy

3:

4: // Constructer initializing

5: Message::Message(){

6: message = nullptr;

7: }

8:

9: //Destructor

10: Message::~Message(){

11: if(message != nullptr)

12: delete[] message;

13: }

14:

15: //this is the function to set Message,

16: //it works by making a new char\* with (msg + 1) as its length

17: //then copy msg to message with strcpy

18: void Message::setMessage(const char\* msg){

19: message = new char[std::strlen(msg) + 1];

20: std::strcpy(message, msg);

21: }

22:

23: //return message

24: char\* Message::getMessage(void){

25: return message;

26: }

27:

main.cpp

1: #include "Message.h"

2:

3: int main (int argc, char \*argv[]){

4: //make a new Message object called obj

5: Message obj;

6: obj.setMessage("Hello World.");

7: std::cout << obj.getMessage() << std::endl;

8:

9: return 0;

10: }

**動作確認**

[a243392@xdev07 q1]$ ./q1

Hello World.

[a243392@xdev07 q1]$

**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(); // Constructer

10: ~Message(); //Destructor

11:

12: void setMessage(const char\* message);

13: const char\* getMessage(void) const;

14:

15: //declaration of stream operators

16: friend std::istream& operator>>(std::istream& stream, Message& obj);

17: friend std::ostream& operator<<(std::ostream& stream, const Message& obj);

18: };

19:

20:

Message.cpp

1: #include "Message.h"

2: #include <cstring> //for std::strlen and std::strcpy

3:

4: // Constructer initializing

5: Message::Message(){

6: message = nullptr;

7: }

8:

9: //definition of extraction operator (>>)

10: std::istream& operator>>(std::istream& stream, Message& obj){

11: //temporary buffer to hold the input, by using buffer, we have better memory management,

12: //safer and more robust(prevent overflows), and dynamic memory allocation

13: char buffer[1024];

14: stream.getline(buffer, 1024);

15:

16: //set the message using setMessage function below

17: obj.setMessage(buffer);

18:

19: return stream;

20: }

21:

22: //definition of the insertion operator(<<)

23: std::ostream& operator<<(std::ostream& stream, const Message& obj){

24: if(obj.getMessage() != nullptr){

25: stream << obj.getMessage();

26: }

27: return stream;

28: }

29:

30: //Destructor

31: Message::~Message(){

32: if(message != nullptr)

33: delete[] message;

34: }

35:

36: //this is the function to set Message,

37: //it works by making a new char\* with (msg + 1) as its length

38: //then copy msg to message with strcpy

39: void Message::setMessage(const char\* msg){

40: message = new char[std::strlen(msg) + 1];

41: std::strcpy(message, msg);

42: }

43:

44: //return message

45: const char\* Message::getMessage(void) const {

46: return message;

47: }

48:

main.cpp

1: #include "Message.h"

2:

3: int main (int argc, char \*argv[]){

4: //make a new Message object called obj

5: Message obj;

6: std::cout << "Input message: ";

7: //use >> operator to input to Message object

8: std::cin >> obj;

9: std::cout << "Output message:" << std::endl;

10: //use << operator to output from Message object

11: std::cout << obj << std::endl;

12:

13: return 0;

14: }

**動作確認**

[a243392@xdev07 q2]$ ./q2

Input message: this is a test for q2

Output message:

this is a test for q2

[a243392@xdev07 q2]$

**Q3**

**プログラム**

Message.hとMessage.cppはQ2と同じプログラムを使っている。

RepeatMessage.h

1: #include <iostream>

2: #include "Message.h"

3:

4: //class RepeatMessage is instanced from Message class,

5: //and all public members of Message class is accessible by RepeatMessage class

6: class RepeatMessage: public Message {

7:

8: private:

9: char\* message;

10: int nloops;

11:

12: public:

13: RepeatMessage(int nloops);

14: ~RepeatMessage();

15: const int getNloops()const;

16: //overload (<<) operator for RepeatMessage class

17: friend std::ostream &operator<<(std::ostream& stream, const RepeatMessage& obj);

18: };

19:

RepeatMessage.cpp

1: #include "RepeatMessage.h"

2: #include <cstring> //for std::strlen and std::strcpy

3:

4: //constructor implementation

5: RepeatMessage::RepeatMessage(int n): Message(), nloops(n){} // Constructer with nloops

6:

7: //definition of the insertion operator(<<) for RepeatMessage

8: std::ostream &operator<<(std::ostream &stream, const RepeatMessage &obj){

9: if(obj.getMessage() != nullptr){

10: for(int i = 0; i < obj.getNloops(); i++){

11: stream << obj.getMessage();

12: }

13: stream << std::endl;

14: }

15: return stream;

16: }

17:

18: //function to get the nloops

19: const int RepeatMessage::getNloops()const{

20: return nloops;

21: }

22:

23: //Destructor

24: RepeatMessage::~RepeatMessage(){

25: if(message != nullptr)

26: delete[] message;

27: }

main.cpp

1: #include "RepeatMessage.h"

2:

3: int main (int argc, char \*argv[]){

4: //make a new Message object called obj

5: RepeatMessage obj(3);

6: std::cout << "Input message: ";

7: std::cin >> obj;

8: std::cout << "Output message:" << std::endl;

9: std::cout << obj;

10:

11: return 0;

12: }

**動作確認**

[a243392@xdev07 q3]$ ./q3

Input message: This is a message␣␣

Output message:

This is a message␣␣This is a message␣␣This is a message␣␣

[a243392@xdev07 q3]$

**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

４ クラスの実装の仕方を理解した.

４ private, protected, public などのアクセス指定子の意味と使い方を理解した.

４ コンストラクタ, デストラクタの実装方法を理解した.

４ メンバ変数の実装方法を理解した.

４ クラスに対する演算子の実装方法を理解した.

３ 変数の参照渡しについて理解し, 値渡しとの違いを説明できる.

４ クラスの継承について理解し, 既存クラスを継承した別のクラスを実装することができる.

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.