Aufgabe – Stack

Ein Stack bezeichnet in der Informatik eine Datenstruktur zum Speichern von Elementen. Sie können sich einen Stack vorstellen wie einen Stapel. Dabei können neue Elemente nur oben aufgelegt (geschrieben) und auch nur von dort wieder entnommen (gelesen) werden. Dieses Prinzip wird als LIFO (Last-In-First-Out) bezeichnet: Das Element, welches als letztes geschrieben (auf den Stapel gelegt) wurde, wird als erstes wieder gelesen (vom Stapel genommen).

Unser Stack soll Daten vom Typ CMessage speichern. Zwei Header-Dateien sind gegeben (s. unten).

1. Implementieren Sie in den entsprechenden *.cpp-Dateien die genannten Methoden.
2. Schreiben Sie eine Main-Funktion zum Testen Ihrer Implementierung.

Geben Sie wie gewohnt (einen oder mehrere) Screendumps ab, aus denen Ihre sinnvoll gewählten Testfälle ersichtlich sind.

Hinweis:

Die Methode „display“ dient der Darstellung des gesamten Stacks auf dem Bildschirm – der Stack selbst bleibt dabei unverändert. Im Gegensatz zu anderen Methoden dürfen und sollen Sie hier Funktionen bzw. Operatoren zur Ausgabe auf dem Bildschirm verwenden. (Alle übrigen Bildschirmausgaben finden aus der „main“ heraus statt – und nicht etwa aus „push“ oder „pop“.)

Viel Spaß und Erfolg bei der Bearbeitung!

Datei „CMessage.h“:

```
1  #ifndef CMESSAGE_H_
2  #define CMESSAGE_H_
3  #include <cstring> // Funktion "strcpy_s()"
4
5  const int MSG_MAX_LEN = 255;
6
7  class CMessage
8  {
9  private:
10     int         mID;
11     char        mMsg[MSG_MAX_LEN]; // sog. NULL-terminierter C-String
12
13 public:
14     int     getID();
15     void    setID(int id);
16
17     void    getMsg(char* Msg) // Implizit inline Definition
18     {
19         strcpy_s(Msg, MSG_MAX_LEN, mMsg);
20         // Fuer C-Strings (NULL-terminierte Strings)
21         // funktioniert eine einfache Zuweisung nicht wie gewuenscht!
22     }
23
24     void    setMsg(const char* Msg) // Implizit inline Definition
25     {
26         strcpy_s(mMsg, MSG_MAX_LEN, Msg);
27         // Siehe oben.
28     }
29 };
30
31 #endif
```

Datei „CStack.h“:

```
1   #ifndef CSTACK_H_
2   #define CSTACK_H_
3
4   #include "CMessage.h"
5
6   class CStack
7   {
8   public:
9       CStack(int stacksize);          // Konstruktor.
10      ~CStack ();                     // Destruktor.
11      bool push(const CMessage& msg); // Eine Nachricht auf den Stack legen.
12      bool pop(CMessage& msg);        // Eine Nachricht vom Stack herunter nehmen.
13      int getNumOfMessages(void);     // Aktuelle Anzahl der Nachrichten auf dem Stack.
14      int getNumOfByteNeeded(void);   // Anzahl der Bytes, die vom Stack insgesamt
15                                      // belegt werden.
16      void display(void);             // Gibt den gesamten aktuellen Inhalt des
17                                      // Stacks auf dem Bildschirm aus.
18
19  private:
20      int mSize;                      // Groeße des Stacks.
21      int mStackIndex;                // Anzahk aktueller Nachrichten auf dem Stack.
22      CMessage* mStackPtr;            // Stack-Pointer. Nicht verändern -
23                                      // wird fuer delete-Befehl benoetigt.
24      CMessage* mStackPtrNext;        // Pointer auf die naechste freie Nachricht im Stack.
25  };
26
27  #endif
```

Beispiel Testlauf:

```
Bitte geben Sie die Anzahl der Nachrichten ein, die der Stack maximal aufnehmen kann: 5

 1 = PUSH, 2 = POP, 3 = ENDE. Ihre Wahl: 1
 Ihre Nachricht: Das
 (0, Das) wurde in den Stack geschrieben.
 Stack = [(0, Das)]

 1 = PUSH, 2 = POP, 3 = ENDE. Ihre Wahl: 1
 Ihre Nachricht: Wetter
 (1, Wetter) wurde in den Stack geschrieben.
 Stack = [(0, Das), (1, Wetter)]

 1 = PUSH, 2 = POP, 3 = ENDE. Ihre Wahl: 1
 Ihre Nachricht: ist
 (2, ist) wurde in den Stack geschrieben.
 Stack = [(0, Das), (1, Wetter), (2, ist)]

 1 = PUSH, 2 = POP, 3 = ENDE. Ihre Wahl: 1
 Ihre Nachricht: heute
 (3, heute) wurde in den Stack geschrieben.
 Stack = [(0, Das), (1, Wetter), (2, ist), (3, heute)]

 1 = PUSH, 2 = POP, 3 = ENDE. Ihre Wahl: 1
 Ihre Nachricht: schoen
 (4, schoen) wurde in den Stack geschrieben.
 Stack = [(0, Das), (1, Wetter), (2, ist), (3, heute), (4, schoen)]

 1 = PUSH, 2 = POP, 3 = ENDE. Ihre Wahl: 1
 Ihre Nachricht: und
 Der Stack ist voll! Ihre Nachricht konnte nicht geschrieben werden.

 1 = PUSH, 2 = POP, 3 = ENDE. Ihre Wahl: 2
 (4, schoen) wurde vom Stack genommen.
 Stack = [(0, Das), (1, Wetter), (2, ist), (3, heute)]

 1 = PUSH, 2 = POP, 3 = ENDE. Ihre Wahl: 1
 Ihre Nachricht: gut
 (6, gut) wurde in den Stack geschrieben.
 Stack = [(0, Das), (1, Wetter), (2, ist), (3, heute), (6, gut)]

 1 = PUSH, 2 = POP, 3 = ENDE. Ihre Wahl: 2
 (6, gut) wurde vom Stack genommen.
 Stack = [(0, Das), (1, Wetter), (2, ist), (3, heute)]

 1 = PUSH, 2 = POP, 3 = ENDE. Ihre Wahl: 2
 (3, heute) wurde vom Stack genommen.
 Stack = [(0, Das), (1, Wetter), (2, ist)]

 1 = PUSH, 2 = POP, 3 = ENDE. Ihre Wahl: 2
 (2, ist) wurde vom Stack genommen.
 Stack = [(0, Das), (1, Wetter)]

 1 = PUSH, 2 = POP, 3 = ENDE. Ihre Wahl: 2
 (1, Wetter) wurde vom Stack genommen.
 Stack = [(0, Das)]

 1 = PUSH, 2 = POP, 3 = ENDE. Ihre Wahl: 2
 (0, Das) wurde vom Stack genommen.
 Stack = []

 1 = PUSH, 2 = POP, 3 = ENDE. Ihre Wahl: 2
 Der Stack ist leer! POP konnte nicht ausgefuehrt werden.

 1 = PUSH, 2 = POP, 3 = ENDE. Ihre Wahl: 4
 Ihre Eingabe wurde nicht erkannt. Bitte versuchen Sie es nocheinmal.

 1 = PUSH, 2 = POP, 3 = ENDE. Ihre Wahl: 3
 Vielen Dank - auf Wiedersehen.
```