```
*************************************************************************
* PROGRAMMED BY : Nick Reardon
* CLASS         : CS1D
* SECTION       : MW - 2:30p
* Assignment #6 : Priority Queues
*************************************************************************

                Assignment #6 - Priority Queues

 Implement two priority queues to simulate an afternoon in an
 emergency room. Your software should NOT be dependent on
 the given data.

    1. Priority Queue 1
 Develop software or use the authorÆs priority queue ADT to
 implement a priority queue based on a heap.


    2. Priority Queue 2
 Use the STL priority_queue

 Test both priority queues with the following data assuming
 that the priority queue is built at noon and Doctor DoGood is
 the only doctor on duty. Each patient requires 25 minutes of
 care. The patient who waited the longest has the highest
 priority unless there is a life-threatening scenario. Determine
 the order in which the patients will be treated. You will need
 to create a timer. Output the time each appointment starts
 and ends.
 Assumptions:
    1. When a patient's care is interrupted, they still only get
       25 minutes of care total
    2. If a patientÆs treatment starts at 1:00 and is not
       interrupted, then the next patientÆs treatment will
       start at 1:25
    3. Waiting time is wait time before noon

 |----------------------|----------------------------|--------|
 | Name                 |    Waiting Time (hours)    |  End   |
 |______________________|____________________________|________|
 | Bob Bleeding         |             5              |        |
 |----------------------|----------------------------|--------|
 | Frank Feelingbad     |             3              |        |
 |----------------------|----------------------------|--------|
 | Cathy Coughing       |             2              |        |
 |----------------------|----------------------------|--------|
 | Sam Sneezing         | Life threatening at 1:14 PM |        |
 |----------------------|----------------------------|--------|
 | Paula Pain           |             4              |        |
 |----------------------|----------------------------|--------|
 | Sid Sickly           | Life threatening at 3:01 PM |        |
 |----------------------|----------------------------|--------|
```

```
| Alice Ailment         |              7              |        |
|-----------------------|-----------------------------|--------|
| Irene Ill             |              1              |        |
|-----------------------|-----------------------------|--------|
| Tom Temperature       |              6              |        |
|-----------------------|-----------------------------|--------|


 Due February 26th



**************************************************************************


 Now doing written Priority Queue based on a heap


  -- 12::00 --
 Creating priority queue of waiting patients...

Patient Admitted:
Name: Alice Ailment
Care began at 12:00

Patient Discharge:
Name: Alice Ailment
Care ended at 12:25

Patient Admitted:
Name: Tom Temperature
Care began at 12:25

Patient Discharge:
Name: Tom Temperature
Care ended at 12:50

Patient Admitted:
Name: Bob Bleeding
Care began at 12:50

Patient Care Interrupted:
Name: Sam Sneezing
Care interrupted at 01:14
Minutes in visit remaining: 1

High Priority Patient Recieved
Immediate attention administered:
Name: Sam Sneezing
Care began at 01:14

Patient Discharge:
```

```
Name: Sam Sneezing
Care ended at 01:39

Patient Care Resumed:
Name: Bob Bleeding
Care resumed at 01:39
Minutes in visit remaining: 1

Patient Discharge:
Name: Bob Bleeding
Care ended at 01:40

Patient Admitted:
Name: Paula Pain
Care began at 01:40

Patient Discharge:
Name: Paula Pain
Care ended at 02:05

Patient Admitted:
Name: Frank Feelingbad
Care began at 02:05

Patient Discharge:
Name: Frank Feelingbad
Care ended at 02:30

Patient Admitted:
Name: Cathy Coughing
Care began at 02:30

Patient Discharge:
Name: Cathy Coughing
Care ended at 02:55

Patient Admitted:
Name: Irene Ill
Care began at 02:55

Patient Care Interrupted:
Name: Sid Sickly
Care interrupted at 03:01
Minutes in visit remaining: 19

High Priority Patient Recieved
Immediate attention administered:
Name: Sid Sickly
Care began at 03:01

Patient Discharge:
Name: Sid Sickly
```

```
157 Care ended at 03:26
158
159 Patient Care Resumed:
160 Name: Irene Ill
161 Care resumed at 03:26
162 Minutes in visit remaining: 19
163
164 Patient Discharge:
165 Name: Irene Ill
166 Care ended at 03:45
167
168
169   --- END OF DAY  05:00 ----
170
171 ____________________________________________________________
172
173  Now doing STL Priority Queue
174
175
176   -- 12::00 --
177  Creating priority queue of waiting patients...
178
179 Patient Admitted:
180 Name: Alice Ailment
181 Care began at 12:00
182
183 Patient Discharge:
184 Name: Alice Ailment
185 Care ended at 12:25
186
187 Patient Admitted:
188 Name: Tom Temperature
189 Care began at 12:25
190
191 Patient Discharge:
192 Name: Tom Temperature
193 Care ended at 12:50
194
195 Patient Admitted:
196 Name: Bob Bleeding
197 Care began at 12:50
198
199 Patient Care Interrupted:
200 Name: Sam Sneezing
201 Care interrupted at 01:14
202 Minutes in visit remaining: 1
203
204 High Priority Patient Recieved
205 Immediate attention administered:
206 Name: Sam Sneezing
207 Care began at 01:14
208
```

```
Patient Discharge:
Name: Sam Sneezing
Care ended at 01:39

Patient Care Resumed:
Name: Bob Bleeding
Care resumed at 01:39
Minutes in visit remaining: 1

Patient Discharge:
Name: Bob Bleeding
Care ended at 01:40

Patient Admitted:
Name: Paula Pain
Care began at 01:40

Patient Discharge:
Name: Paula Pain
Care ended at 02:05

Patient Admitted:
Name: Frank Feelingbad
Care began at 02:05

Patient Discharge:
Name: Frank Feelingbad
Care ended at 02:30

Patient Admitted:
Name: Cathy Coughing
Care began at 02:30

Patient Discharge:
Name: Cathy Coughing
Care ended at 02:55

Patient Admitted:
Name: Irene Ill
Care began at 02:55

Patient Care Interrupted:
Name: Sid Sickly
Care interrupted at 03:01
Minutes in visit remaining: 19

High Priority Patient Recieved
Immediate attention administered:
Name: Sid Sickly
Care began at 03:01

Patient Discharge:
```

```
Name: Sid Sickly
Care ended at 03:26

Patient Care Resumed:
Name: Irene Ill
Care resumed at 03:26
Minutes in visit remaining: 19

Patient Discharge:
Name: Irene Ill
Care ended at 03:45


  --- END OF DAY  05:00 ----

Press any key to continue . . .
```

```cpp
/*****************************************************************************
 * AUTHOR           : Nick Reardon
 * Assignment #6    : Priority Queues
 * CLASS            : CS1D
 * SECTION          : MW - 2:30p
 * DUE DATE         : 02 / 24 / 20
 *****************************************************************************/
#include "main.h"
#include <queue>
using std::cout; using std::endl;


int main()
{

    /*
     * HEADER OUTPUT
     */
    PrintHeader(cout, "Prompt.txt");

    /*****************************************************************************/

    cout << endl << " Now doing written Priority Queue based on a heap" << endl
      << endl;

    const int MAX_CARE_TIME = 25;
    const int MAX_TOTAL_MINUTES = 300;

    ArrayMaxHeap<std::string, int> heap;

    cout << endl << "  -- 12::00 -- \n Creating priority queue of waiting
      patients..." << endl << endl;

    heap.insert("Bob Bleeding", 5);
    heap.insert("Frank Feelingbad", 3);
    heap.insert("Cathy Coughing", 2);
    heap.insert("Paula Pain", 4);
    heap.insert("Alice Ailment", 7);
    heap.insert("Irene Ill", 1);
    heap.insert("Tom Temperature", 6);

    int timer = 0;
    int emergencyTimer = 0;
    bool emergency = false;


    for (int time = 0; time <= MAX_TOTAL_MINUTES; )
    {
        if (!heap.empty())
        {
            switch (time)
            {
```

```
51              case 74:
52                  heap.insert("Sam Sneezing", 1100);
53                  emergency = true;
54                  break;
55
56              case 181:
57
58                  heap.insert("Sid Sickly", 100);
59                  emergency = true;
60                  break;
61              }
62
63              if (emergency)
64              {
65                  if (emergencyTimer == 0)
66                  {
67                      if (!heap.empty())
68                      {
69                          InterruptPatient(heap, time, timer);
70
71                          PriorityPatient(heap, time);
72                      }
73                  }
74
75                  emergencyTimer++;
76                  time++;
77
78                  if (emergencyTimer == 25)
79                  {
80                      DischargePatient(heap, time);
81
82                      heap.remove();
83
84                      if (!heap.empty())
85                      {
86                          ResumePatient(heap, time, timer);
87                      }
88                      emergency = false;
89
90                      emergencyTimer = 0;
91
92                  }
93              }
94              else if (!emergency)
95              {
96                  if (timer == 0)
97                  {
98                      if (!heap.empty())
99                      {
100                         AdmitPatient(heap, time);
101                     }
102                 }
```

```
103
104                 timer++;
105                 time++;
106
107                 if (timer == MAX_CARE_TIME)
108                 {
109                     timer = 0;
110
111                     DischargePatient(heap, time);
112
113                     heap.remove();
114
115                 }
116             }
117
118         }
119         else
120         {
121             time++;
122         }
123     }
124
125     cout << endl << "  --- END OF DAY  " << ConvertTime(MAX_TOTAL_MINUTES, 12) << ↵
        " ----"
126         << endl << endl;
127
128
129     //                                                                          ↵
        ************************************************************************ ↵
        ***********
130
131     //                                                                          ↵
        ************************************************************************ ↵
        ***********
132     cout << std::string(60, '_') << endl;
133     cout << endl << " Now doing STL Priority Queue" << endl << endl;
134
135     std::priority_queue< std::pair< int, std::string>> STL_PrioQ;
136
137     cout << endl << "  -- 12::00 -- \n Creating priority queue of waiting        ↵
        patients... " << endl << endl;
138
139     STL_PrioQ.push(std::make_pair(5, "Bob Bleeding"));
140     STL_PrioQ.push(std::make_pair(3, "Frank Feelingbad"));
141     STL_PrioQ.push(std::make_pair(2, "Cathy Coughing"));
142     STL_PrioQ.push(std::make_pair(4, "Paula Pain"));
143     STL_PrioQ.push(std::make_pair(7, "Alice Ailment"));
144     STL_PrioQ.push(std::make_pair(1, "Irene Ill"));
145     STL_PrioQ.push(std::make_pair(6, "Tom Temperature"));
146
147     timer = 0;
148     emergencyTimer = 0;
```

```
149     emergency = false;
150
151
152     for (int time = 0; time <= MAX_TOTAL_MINUTES; )
153     {
154         if (!STL_PrioQ.empty())
155         {
156             switch (time)
157             {
158             case 74:
159                 STL_PrioQ.push(std::make_pair(999, "Sam Sneezing"));
160                 emergency = true;
161                 break;
162
163             case 181:
164
165                 STL_PrioQ.push(std::make_pair(999, "Sid Sickly"));
166                 emergency = true;
167                 break;
168             }
169
170             if (emergency)
171             {
172                 if (emergencyTimer == 0)
173                 {
174                     if (!STL_PrioQ.empty())
175                     {
176                         cout << "Patient Care Interrupted:" << endl
177                             << "Name: " << STL_PrioQ.top().second << endl
178                             << "Care interrupted at " << ConvertTime(time, 12,
                         false) << endl
179                             << "Minutes in visit remaining: " << 25 - timer
180                             << endl << endl;
181
182                         cout << "High Priority Patient Recieved" << endl
183                             << "Immediate attention administered:" << endl
184                             << "Name: " << STL_PrioQ.top().second << endl
185                             << "Care began at " << ConvertTime(time, 12, false)
186                             << endl << endl;
187                     }
188                 }
189
190                 emergencyTimer++;
191                 time++;
192
193                 if (emergencyTimer == 25)
194                 {
195                     cout << "Patient Discharge:" << endl
196                         << "Name: " << STL_PrioQ.top().second << endl
197                         << "Care ended at " << ConvertTime(time, 12, false)
198                         << endl << endl;
199
```

```
200                     STL_PrioQ.pop();
201
202                     if (!STL_PrioQ.empty())
203                     {
204                         cout << "Patient Care Resumed:" << endl
205                             << "Name: " << STL_PrioQ.top().second << endl
206                             << "Care resumed at " << ConvertTime(time, 12, false) ↵
                          << endl
207                             << "Minutes in visit remaining: " << 25 - timer
208                             << endl << endl;
209                     }
210                     emergency = false;
211
212                     emergencyTimer = 0;
213
214                 }
215             }
216             else if (!emergency)
217             {
218                 if (timer == 0)
219                 {
220                     if (!STL_PrioQ.empty())
221                     {
222                         cout << "Patient Admitted:" << endl
223                             << "Name: " << STL_PrioQ.top().second << endl
224                             << "Care began at " << ConvertTime(time, 12, false)
225                             << endl << endl;
226                     }
227                 }
228
229                 timer++;
230                 time++;
231
232                 if (timer == MAX_CARE_TIME)
233                 {
234                     timer = 0;
235
236                     cout << "Patient Discharge:" << endl
237                         << "Name: " << STL_PrioQ.top().second << endl
238                         << "Care ended at " << ConvertTime(time, 12, false)
239                         << endl << endl;
240
241                     STL_PrioQ.pop();
242
243                 }
244             }
245
246         }
247         else
248         {
249             time++;
250         }
```

```
251     }
252
253     cout << endl << "  --- END OF DAY  " << ConvertTime(MAX_TOTAL_MINUTES, 12) << "
        " ----"
254         << endl << endl;
255
256     system("pause");
257     return 0;
258 }
259
260
261
262 std::string ConvertTime(int totalMinutes, int startHour, bool hours24Style)
263 {
264     int minutes;
265     int hours;
266     std::string output = "";
267
268     minutes = totalMinutes % 60;
269
270     if (!hours24Style)
271     {
272         if (startHour == 12)
273         {
274             hours = totalMinutes / 60;
275             if (hours == 0)
276             {
277                 hours = 12;
278             }
279         }
280         else
281         {
282             hours = startHour + (totalMinutes / 60);
283         }
284     }
285
286
287
288     if ((hours < 10))
289     {
290         output += '0' + std::to_string(hours);
291     }
292     else
293     {
294         output += std::to_string(hours);
295     }
296
297     output += ":";
298
299     if ((minutes < 10))
300     {
301         output += '0' + std::to_string(minutes);
```

```
302     }
303     else
304     {
305         output += std::to_string(minutes);
306     }
307
308
309
310     return output;
311
312 }
313
314 void DischargePatient(ArrayMaxHeap <std::string, int>& heap, int totalMinutes)
315 {
316
317     cout << "Patient Discharge:" << endl
318         << "Name: " << heap.max() << endl
319         << "Care ended at " << ConvertTime(totalMinutes, 12, false)
320         << endl << endl;
321
322
323 }
324
325 void AdmitPatient(ArrayMaxHeap <std::string, int>& heap, int totalMinutes)
326 {
327
328     cout << "Patient Admitted:" << endl
329         << "Name: " << heap.max() << endl
330         << "Care began at " << ConvertTime(totalMinutes, 12, false)
331         << endl << endl;
332 }
333
334 void InterruptPatient(ArrayMaxHeap <std::string, int>& heap, int totalMinutes,
      int currentTimer)
335 {
336
337     cout << "Patient Care Interrupted:" << endl
338         << "Name: " << heap.max() << endl
339         << "Care interrupted at " << ConvertTime(totalMinutes, 12, false) << endl
340         << "Minutes in visit remaining: " << 25 - currentTimer
341         << endl << endl;
342 }
343
344 void ResumePatient(ArrayMaxHeap <std::string, int>& heap, int totalMinutes, int
      currentTimer)
345 {
346
347     cout << "Patient Care Resumed:" << endl
348         << "Name: " << heap.max() << endl
349         << "Care resumed at " << ConvertTime(totalMinutes, 12, false) << endl
350         << "Minutes in visit remaining: " << 25 - currentTimer
351         << endl << endl;
```

```
352 }
353
354 void PriorityPatient(ArrayMaxHeap <std::string, int>& heap, int totalMinutes)
355 {
356
357     cout << "High Priority Patient Recieved" << endl
358         << "Immediate attention administered:" << endl
359         << "Name: " << heap.max() << endl
360         << "Care began at " << ConvertTime(totalMinutes, 12, false)
361         << endl << endl;
362 }
```

```
1  /*****************************************************************************
2   * AUTHOR           : Nick Reardon
3   * Assignment #6    : Priority Queues
4   * CLASS            : CS1D
5   * SECTION          : MW - 2:30p
6   * DUE DATE         : 02 / 24 / 20
7   *****************************************************************************/
8  #ifndef _MAIN_H_
9  #define _MAIN_H_
10
11 //Standard includes
12 #include <iostream>
13 #include <iomanip>
14 #include <string>
15 #include "PrintHeader.h"
16
17 //Program Specific
18 #include "ArrayHeap.h"
19
20 #endif // _HEADER_H_
21
22 std::string ConvertTime(int totalMinutes, int startHour, bool hours24Style =
     false);
23
24 void DischargePatient(ArrayMaxHeap<std::string, int>& heap, int totalMinutes);
25
26 void AdmitPatient(ArrayMaxHeap<std::string, int>& heap, int totalMinutes);
27
28 void InterruptPatient(ArrayMaxHeap<std::string, int>& heap, int totalMinutes, int
     currentTimer);
29
30 void ResumePatient(ArrayMaxHeap<std::string, int>& heap, int totalMinutes, int
     currentTimer);
31
32 void PriorityPatient(ArrayMaxHeap<std::string, int>& heap, int totalMinutes);
33
```

```
 1  /*****************************************************************************
 2   * AUTHOR           : Nick Reardon
 3   * Assignment #4    : Deque To Queue
 4   * CLASS            : CS1D
 5   * SECTION          : MW - 2:30p
 6   * DUE DATE         : 02 / 10 / 20
 7   *****************************************************************************/
 8  #ifndef _ARRAYHEAP_H_
 9  #define _ARRAYHEAP_H_
10  #include <exception>
11  #include "Except.h"
12
13  enum ERROR_TYPE
14  {
15      DEFUALT,
16      FULL,
17      EMPTY,
18      OUT_OF_RANGE
19  };
20
21  template <typename Type, typename Key>
22  struct heapMember
23  {
24      Type value;
25      Key key;
26
27      heapMember(const Type& newValue, const Key& newKey)
28      {
29          value = newValue;
30          key = newKey;
31      }
32
33      heapMember() {}
34
35      void swap(heapMember& other)
36      {
37          Key tempKey;
38          Type tempType;
39
40          tempKey = other.key;
41          tempType = other.value;
42
43          other.key = this->key;
44          other.value = this->value;
45
46          this->key = tempKey;
47          this->value = tempType;
48
49
50      }
51
52      inline bool operator< (const heapMember& other)
```

```
53      {
54          return (this->key < other.key);
55      }
56
57      inline bool operator> (const heapMember& other)
58      {
59          return (this->key > other.key);
60      }
61
62      inline void operator= (const heapMember& other)
63      {
64          this->key = other.key;
65          this->value = other.value;
66      }
67  };
68
69  template <class Type, class Key>
70  class ArrayMaxHeap
71  {
72  private:
73      heapMember<Type, Key>* heap;
74
75      int currentSize;
76
77      int capacity;
78
79  protected:
80      void sort()
81      {
82          int index = 1;
83
84          int swapIndex;
85
86          while ( (2 * index < currentSize) &&
87              ((heap[index].key < heap[2 * index].key) || (heap[index].key < heap[2
                * index + 1].key)) )
88          {
89              if (heap[2 * index].key > heap[2 * index + 1].key)
90              {
91                  swapIndex = 2 * index;
92              }
93              else
94              {
95                  swapIndex = 2 * index + 1;
96              }
97
98              heap[index].swap(heap[swapIndex]);
99
100             index = swapIndex;
101         }
102     }
103
```

```
104
105
106  public:
107
108      ArrayMaxHeap<Type,Key>(const int newCapacity = 32)
109      {
110          heap = new heapMember<Type, Key>[newCapacity];
111          currentSize = 0;
112          capacity = newCapacity;
113      }
114
115      //VectorHeap<Type>(const VectorHeap<Type>& otherDeque);
116
117      ~ArrayMaxHeap()
118      {
119          delete[] heap;
120      }
121
122      bool empty() const { return currentSize == 0; }
123
124      bool full() const { return currentSize == capacity; }
125
126      int size() const { return size; }
127
128      void insert(const Type& element, const Key& newKey)
129      {
130          if (full())
131          {
132              throw Except("container is full", FULL, 5);
133          }
134
135          currentSize++;
136
137          heap[currentSize].value = element;
138          heap[currentSize].key = newKey;
139
140
141          int index = currentSize;
142
143          while ( (heap[index].key > heap[index / 2].key) && ((index / 2) != 0) )
144          {
145              heap[index].swap(heap[index / 2]);
146
147              index /= 2;
148
149          }
150
151      }
152
153      void remove()
154      {
155          if (empty())
```

```
156         {
157             throw(Except("Cannot remove - heap is empty", EMPTY, 5));
158         }
159 
160         heap[1] = heap[currentSize];
161         currentSize--;
162         sort();
163     }
164 
165 
166 
167     Type max() const
168     {
169         Type temp = heap[1].value;
170         return temp;
171     }
172 
173     void printAll(std::ostream& output) const
174     {
175         if (empty())
176         {
177             throw(Except("Cannot print - heap is empty", EMPTY, 5));
178         }
179 
180         int current = 0;
181         int levelSize = 1;
182 
183         for (int i = 1; i < currentSize; i++)
184         {
185             output << heap[i].value << '(' << heap[i].key << ')' << "  ";
186 
187             current++;
188 
189             if (current == levelSize)
190             {
191                 current = 0;
192 
193                 levelSize = levelSize * 2;
194 
195                 output << '\n';
196             }
197         }
198 
199         output << "\n\n";
200     }
201 
202 };
203 
204 
205 #endif //!_ARRAYHEAP_H_
```

```
/*****************************************************************************
 * AUTHOR           : Nick Reardon
 * Assignment #6    : Priority Queues
 * CLASS            : CS1D
 * SECTION          : MW - 2:30p
 * DUE DATE         : 02 / 24 / 20
 *****************************************************************************/
#ifndef _PRINTHEADER_H_
#define _PRINTHEADER_H_

#include <iostream>
#include <iomanip>
#include <ostream>
#include <string>
#include <fstream>

/*****************************************************************************
* PrintHeader
* ---------------------------------------------------------------------------
* This function will output a class header through the use of ostream.
* It also will output the program description
* ---------------------------------------------------------------------------
*  Call
* ------
* The function call requires 1 parameters. The following example uses an
* output file in the ostream parameter. Ex:
*
*       PrintHeader (oFile);
*
* ---------------------------------------------------------------------------
*  Output
* --------
*   The function will output as follows. Ex:
*
*       ********************************************************
*       * PROGRAMMED BY : Parsa Khazravi and Nick Reardon
*       * CLASS         : CS1B
*       * SECTION       : MW: 7:30pm
*       * Lab #3        : Functions - GCD
*       ********************************************************
*
* ---------------------------------------------------------------------------
* CONSTANTS
* ---------------------------------------------------------------------------
* OUTPUT - USED FOR CLASS HEADING
* ---------------------------------------------------------------------------
* PROGRAMMER        : Name(s) of programmer(s)  - Nick Reardon
* SECTION           : Class times               - MW - 7:30p
* CLASS             : Class label               - CS1B
* PROGRAM_NUM       : # of the program
* PROGRAM_NAME      : Title of the program
* PROGRAM_TYPE      : Type of program - Lab, Assignment, etc.
```

```
53  *
54  * -------------------------------------------------------------------------
55  * MAX_OUTPUT        : Max movies to be output at once
56  *************************************************************************/
57  const std::string PROGRAMMER = "Nick Reardon";
58  const std::string SECTION = "MW - 2:30p";
59  const std::string CLASS = "CS1D";
60  const int PROGRAM_NUM = 6;
61  const std::string PROGRAM_NAME = "Priority Queues";
62  const std::string PROGRAM_TYPE = "Assignment";
63
64
65  void PrintHeader(std::ostream &output, std::string inputText)
66  {
67      std::string typeNum = PROGRAM_TYPE + " #" + std::to_string(PROGRAM_NUM);
68
69      output << std::left
70          << std::string(76, '*')
71          << std::endl
72          << "* PROGRAMMED BY : " << PROGRAMMER << std::endl
73          << "* " << std::setw(14) << "CLASS" << ": " << CLASS << std::endl
74          << "* " << std::setw(14) << "SECTION" << ": " << SECTION << std::endl
75          << "* " << std::setw(14) << typeNum << ": " << PROGRAM_NAME << std::endl
76          << std::string(76, '*')
77          << std::endl << std::endl
78          << std::string(((60 - typeNum.length() - PROGRAM_NAME.length() ) / 2), '
            ')
79          << typeNum + " - " + PROGRAM_NAME
80          << std::endl << std::endl
81          << std::ifstream(inputText).rdbuf()
82          << std::endl
83          << std::string(76, '*')
84          << std::endl << std::endl;
85
86  }
87
88  #endif //_PRINTHEADER_H_
```