Week 7

Exercise 60

../60–4/main.ih

```
#define ERR(msg) printf("%s : %d", (msg), __LINE__)

#include "client/client.ih"
#include "warehouse/warehouse.ih"

#include <iostream>
#include <string>
#include <thread>



using namespace std;
```

../60–4/main.cc

```
#include "main.ih"

int main(int argc, char const **argv)
{
  Warehouse warehouse;

  vector<Client> clients;

  for (size_t idx = 1; idx < argc; ++idx)
    clients.push_back(Client(&warehouse, argv[idx]));
//adding clients to the clients vector

  thread addThread(&Warehouse::addlines, ref(warehouse));
//thread that adds lines to the queue

  vector<thread> threads;

  for (auto &client: clients)
    threads.push_back(thread(&Client::printProduct, ref(client)));

//addings a thread that takes lines from the queue in the warehouse and prints
//it in the file corresponding to the client.


  for (auto &it: threads)
    it.join();

  addThread.join();  //joining threads

  for (auto client: clients)  //printing nr of lines per file
    cout << client.size() << '\n';

}
```

../60–4/client/client.h

```
#ifndef INCLUDED_CLIENT_
#define INCLUDED_CLIENT_

#include "../warehouse/warehouse.ih"

class Client
{
  Warehouse *d_warehouse;
```

```
9     std::string d_outputFile;
10    size_t d_nrlines = 0;
11
12    public:
13      Client(Warehouse *warehouse, std::string const &outputFile);
14
15      size_t size();        //returns nr of lines printed
16      void printProduct(); //prints strings to file
17    private:
18  };
19
20  #endif
```

../60–4/client/client.ih

```
1   #include "client.h"
2   #include <fstream>
3   #include <iostream>
4   #include <chrono>
5   #include <thread>
6
7
8   using namespace std;
```

../60–4/client/c_client.cc

```
1   #include "client.ih"
2
3   Client::Client(Warehouse *warehouse, string const &outputFile)
4   : d_warehouse(warehouse),
5     d_outputFile(outputFile)
6   {
7
8   }
```

../60–4/client/printProduct.cc

```
1   #include "client.ih"
2
3   void Client::printProduct()
4   {
5     ofstream outputStream(d_outputFile);
6
7     while(!d_warehouse -> empty() || !d_warehouse -> isitfinished())
8     {
9       string tmp = d_warehouse -> getProduct();
10
11      if(!tmp.empty())
12      {
13        outputStream << tmp << '\n';
14        ++d_nrlines;
15      }
16    }
17  }
```

../60–4/client/size.cc

```
1   #include "client.ih"
2
3   size_t Client::size()
4   {
5     return d_nrlines;
6   }
```

../60–4/warehouse/warehouse.h

```
#ifndef INCLUDED_WAREHOUSE_
#define INCLUDED_WAREHOUSE_

#include <queue>
#include <string>
#include <mutex>
#include <condition_variable>

class Warehouse
{
  std::queue<std::string> d_queue;
  std::mutex wMutex;
  std::condition_variable condition;
  bool d_finished = false;

  public:
    Warehouse();

    std::string &front(); //returns string that has been in queue the longest

    bool empty(); //checks whether queue is empty

    std::string getProduct(); //used by clients to retrieve a string from
                              // the queue

    bool isitfinished(); //checks if there is more input to come

    void addlines(); //processes input to queue and calls finished when there
                     //is no more input

  private:
    std::string next(); //removes and returns string from queue
    void addProduct(std::string const &line); //adds a string to the queue
    void finished(); //sets d_finished to true and notifies all waiting

};

#endif
```

../60–4/warehouse/warehouse.ih

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


using namespace std;
```

../60–4/warehouse/addLines.cc

```
#include "warehouse.ih"

void Warehouse::addlines()
{
  string inputString;
  while (getline(cin, inputString))      // While there is still user input
    addProduct(inputString);      // Push that input to the queue

  finished();                     // When input is done, signal that
}
```

../60–4/warehouse/addProduct.cc

```
#include "warehouse.ih"
```

```
2
3  void Warehouse::addProduct(string const &line)
4  {
5    lock_guard<mutex> lk(wMutex);
6    d_queue.push(line);
7
8    if (d_queue.size() == 1)
9      condition.notify_all(); //notify waiting clients a string is available
10
11 }
```

../60–4/warehouse/c_warehouse.cc

```
1  #include "warehouse.ih"
2
3  Warehouse::Warehouse()
4  //:
5  {
6  }
```

../60–4/warehouse/empty.cc

```
1  #include "warehouse.ih"
2
3  bool Warehouse::empty()
4  {
5    return d_queue.empty();
6  }
```

../60–4/warehouse/finished.cc

```
1  #include "warehouse.ih"
2
3  void Warehouse::finished()
4  {
5    d_finished = true;
6    condition.notify_all(); //notify waiting processes there will be no more
7                            //products so they should stop waiting for one.
8  }
```

../60–4/warehouse/front.cc

```
1  #include "warehouse.ih"
2
3  string &Warehouse::front()
4  {
5    return d_queue.front();
6  }
```

../60–4/warehouse/getProduct.cc

```
1  #include "warehouse.ih"
2
3  string Warehouse::getProduct()
4  {
5    unique_lock<mutex> ul(wMutex);
6    while (empty() && !d_finished)
7      condition.wait(ul);
8
9    if (!empty())
10     return next();
11  return {};  //return empty string, which can happen if no more strings
12               //are going to be available
13 }
```

../60–4/warehouse/isitfinished.cc

```
1  #include "warehouse.ih"
2
3  bool Warehouse::isitfinished()
4  {
5    return d_finished;
6  }
```

../60–4/warehouse/next.cc

```
1  #include "warehouse.ih"
2
3  string Warehouse::next()
4  {
5    string front = d_queue.front();      // Get element from queue
6    d_queue.pop();                       // Remove that element
7
8    return front;                        // Return it
9  }
```