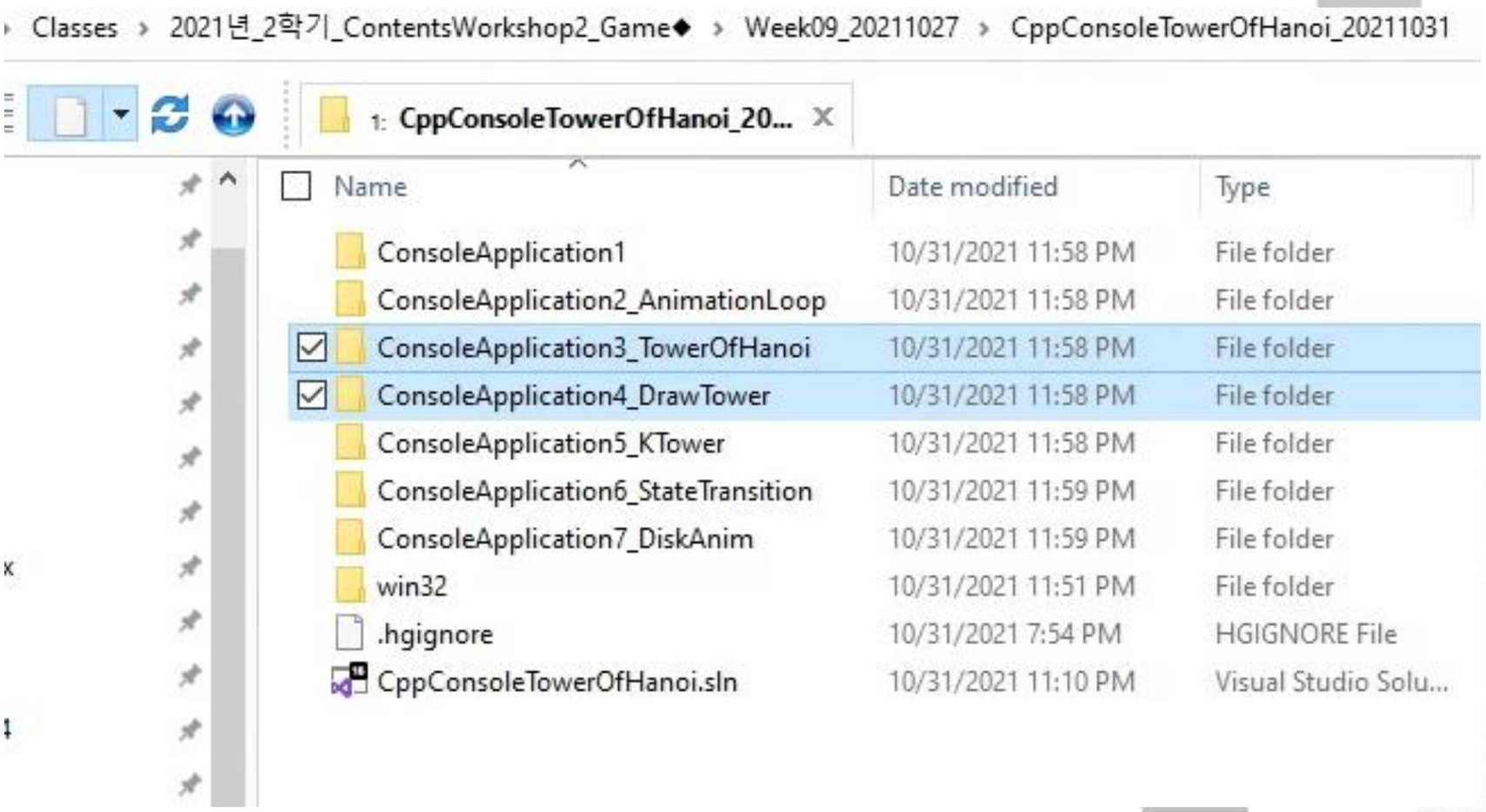
Contents Workshop 2 (Game)

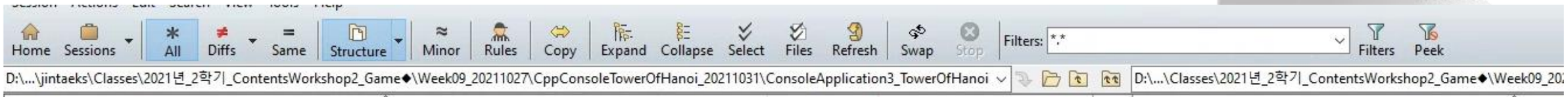
# Console Torus Animation

jintaeks@dongseo.ac.kr

November, 2021

DSU Dongseo University 동서대학교

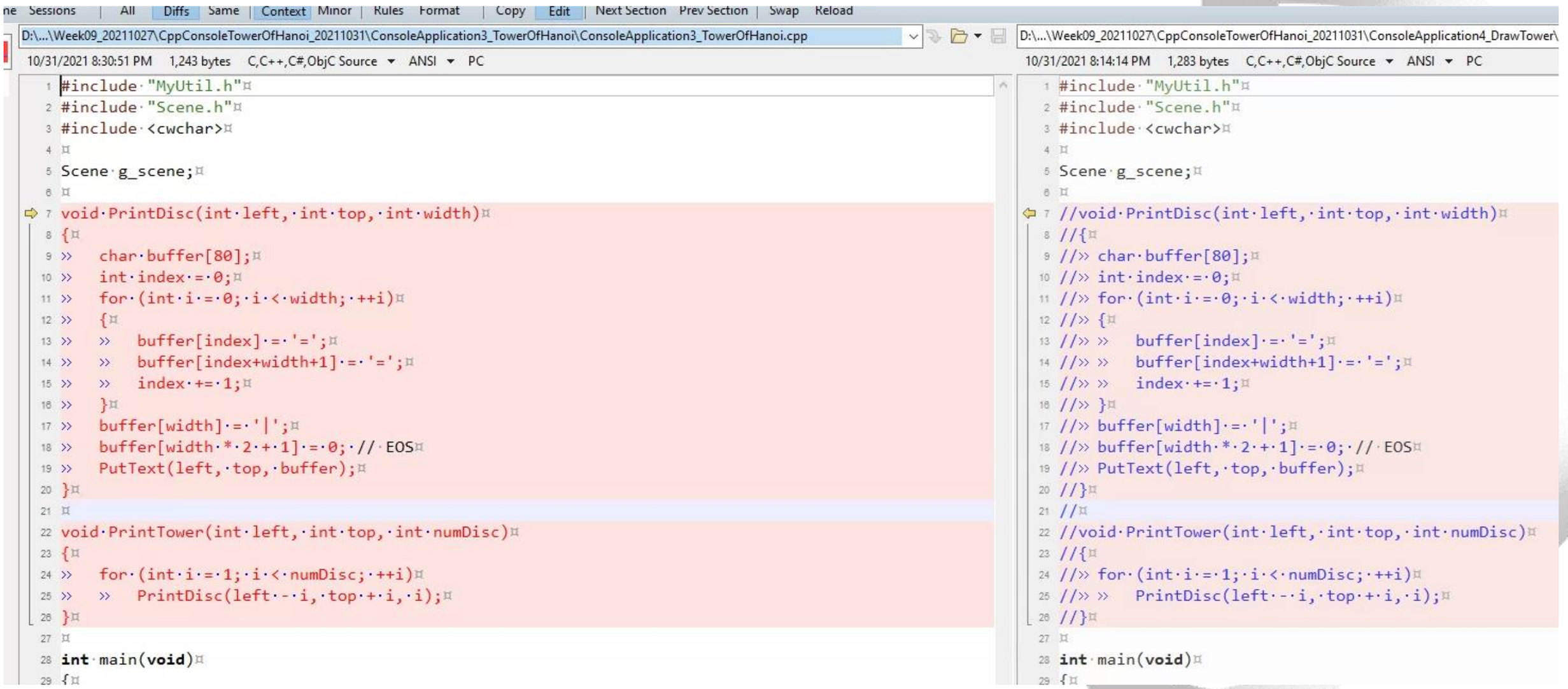

Classes › 2021년_2학기_ContentsWorkshop2_Game◆ › Week09_20211027 › CppConsoleTowerOfHanoi_20211031
1: CppConsoleTowerOfHanoi_20...
Name
Date modified
Type
ConsoleApplication1 10/31/2021 11:58 PM File folder
ConsoleApplication2_AnimationLoop 10/31/2021 11:58 PM File folder
ConsoleApplication3_TowerOfHanoi 10/31/2021 11:58 PM File folder
ConsoleApplication4_DrawTower 10/31/2021 11:58 PM File folder
ConsoleApplication5_KTower 10/31/2021 11:58 PM File folder
ConsoleApplication6_StateTransition 10/31/2021 11:59 PM File folder
ConsoleApplication7_DiskAnim 10/31/2021 11:59 PM File folder
win32 10/31/2021 11:51 PM File folder
.hgignore 10/31/2021 7:54 PM HGIGNORE File
CppConsoleTowerOfHanoi.sln 10/31/2021 11:10 PM Visual Studio Solu...

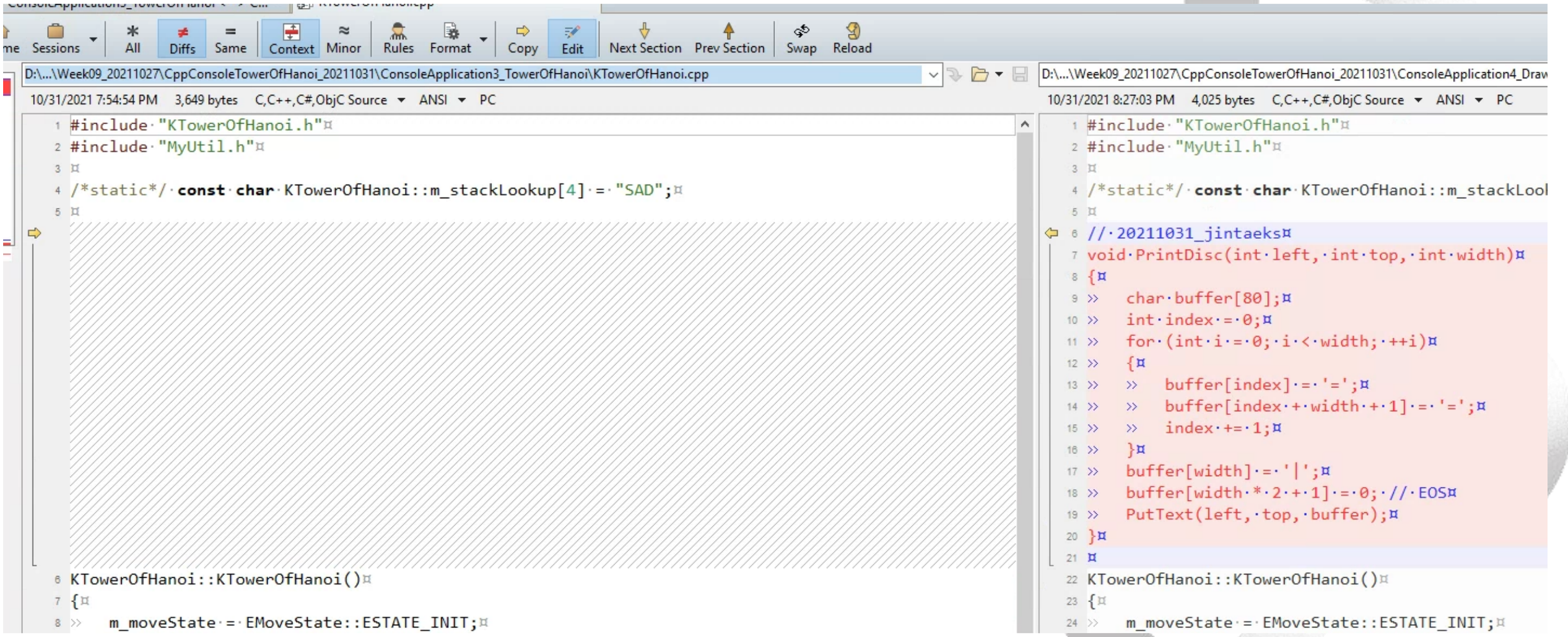



D:\...\jintaeks\Classes\2021년_2학기_ContentsWorkshop2_Game◆\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication3_TowerOfHanoi

D:\...\Classes\2021년_2학기_ContentsWorkshop2_Game◆\Week09_202

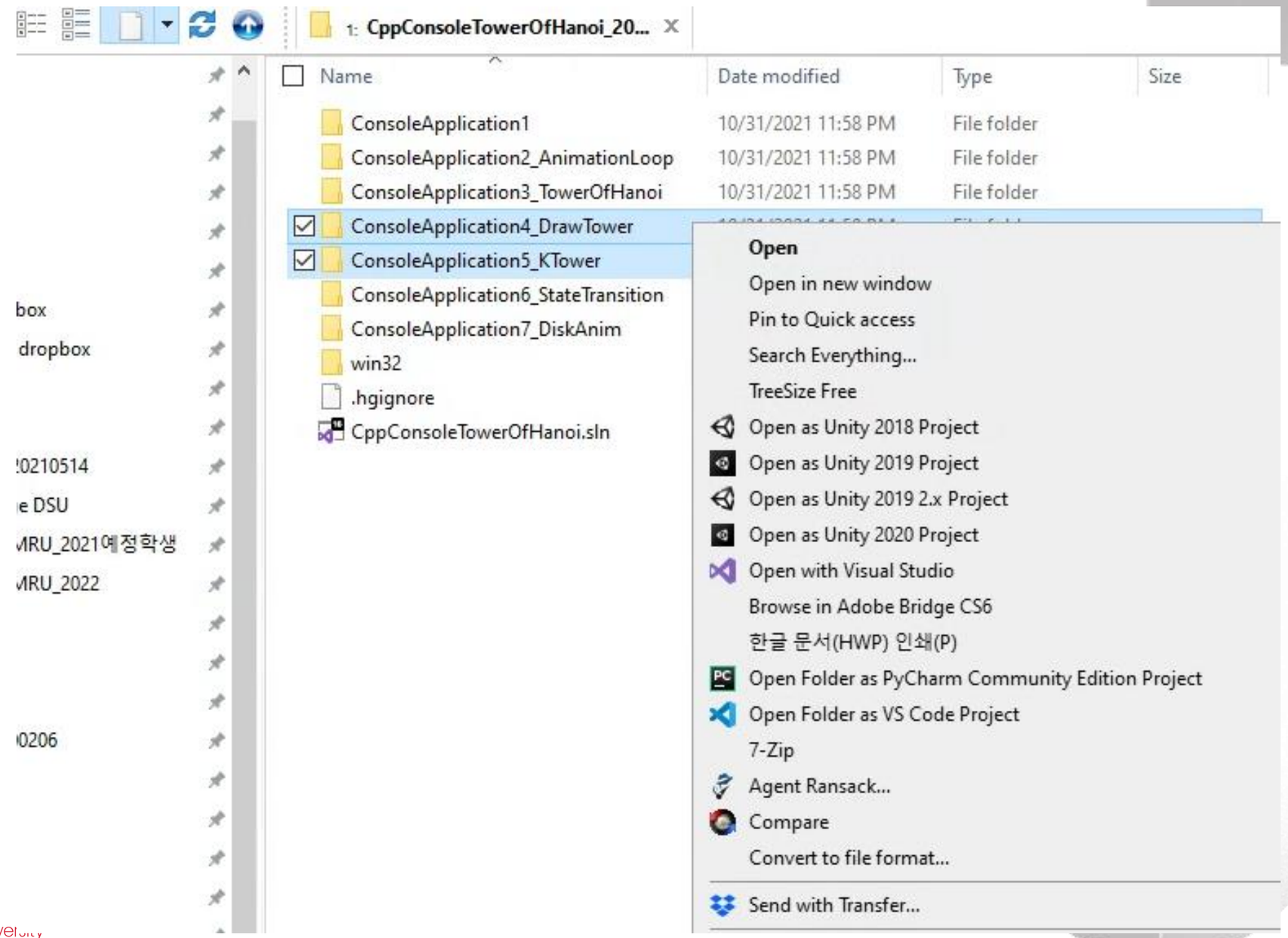
| Name | Size | Modified | Name |
|---|---|---|---|
| ConsoleApplication3_TowerOfHanoi.cpp | 1,243 | 10/31/2021 8:30:51 PM | ConsoleApplication3_TowerOfHanoi.cpp |
| ConsoleApplication3_TowerOfHanoi.vcxproj | 7,730 | 10/31/2021 7:54:54 PM | |
| ConsoleApplication3_TowerOfHanoi.vcxproj.filters | 1,443 | 10/31/2021 7:54:54 PM | |
| ConsoleApplication3_TowerOfHanoi.vcxproj.user | 168 | 10/31/2021 7:54:54 PM | |
| | | | ConsoleApplication4_DrawTower.vcxproj |
| | | | ConsoleApplication4_DrawTower.vcxproj.filters |
| | | | ConsoleApplication4_DrawTower.vcxproj.user |
| KMatrix2.cpp | 141 | 10/31/2021 7:54:54 PM | KMatrix2.cpp |
| KMatrix2.h | 2,326 | 10/31/2021 7:54:54 PM | KMatrix2.h |
| KStack.h | 394 | 10/31/2021 7:54:54 PM | KStack.h |
| KTowerOfHanoi.cpp | 3,649 | 10/31/2021 7:54:54 PM | KTowerOfHanoi.cpp |
| KTowerOfHanoi.h | 695 | 10/31/2021 7:54:54 PM | KTowerOfHanoi.h |
| KVector2.cpp | 496 | 10/31/2021 7:54:54 PM | KVector2.cpp |
| KVector2.h | 1,135 | 10/31/2021 7:54:54 PM | KVector2.h |
| MyUtil.cpp | 3,556 | 10/31/2021 7:54:54 PM | MyUtil.cpp |
| MyUtil.h | 566 | 10/31/2021 7:54:54 PM | MyUtil.h |
| Scene.cpp | 1,222 | 10/31/2021 7:54:54 PM | Scene.cpp |
| Scene.h | 283 | 10/31/2021 7:54:54 PM | Scene.h |

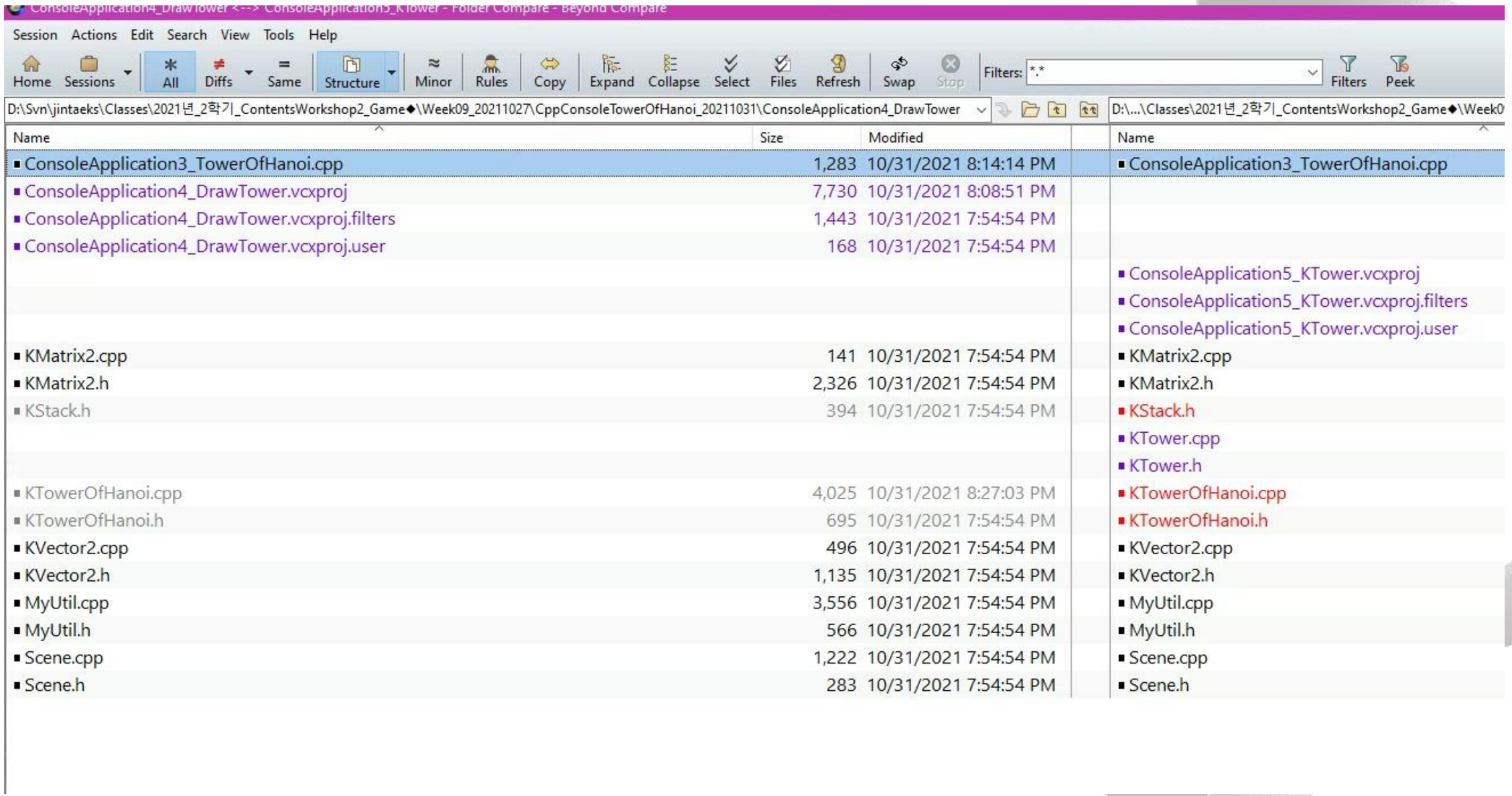

Sessions All Diffs Same Context Minor Rules Format Copy Edit Next Section Prev Section Swap Reload
D:\...\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication3_TowerOfHanoi\ConsoleApplication3_TowerOfHanoi.cpp
10/31/2021 8:30:51 PM 1,243 bytes C,C++,C#,ObjC Source ANSI PC
#include "MyUtil.h"
#include "Scene.h"
#include <cwchar>
Scene g_scene;
void PrintDisc(int left, int top, int width)
{
char buffer[80];
int index = 0;
for (int i = 0; i < width; ++i)
{
buffer[index] = '=';
buffer[index+width+1] = '=';
index += 1;
}
buffer[width] = '|';
buffer[width * 2 + 1] = 0; // EOS
PutText(left, top, buffer);
}
void PrintTower(int left, int top, int numDisc)
{
for (int i = 1; i < numDisc; ++i)
PrintDisc(left--i, top+i, i);
}
int main(void)
{
D:\...\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication4_DrawTower\
10/31/2021 8:14:14 PM 1,283 bytes C,C++,C#,ObjC Source ANSI PC
//void PrintDisc(int left, int top, int width)
//{
//char buffer[80];
//int index = 0;
//for (int i = 0; i < width; ++i)
//{
//buffer[index] = '=';
//buffer[index+width+1] = '=';
//index += 1;
//}
//buffer[width] = '|';
//buffer[width * 2 + 1] = 0; // EOS
//PutText(left, top, buffer);
//}
//void PrintTower(int left, int top, int numDisc)
//{
//for (int i = 1; i < numDisc; ++i)
//PrintDisc(left--i, top+i, i);
//}
int main(void)
{

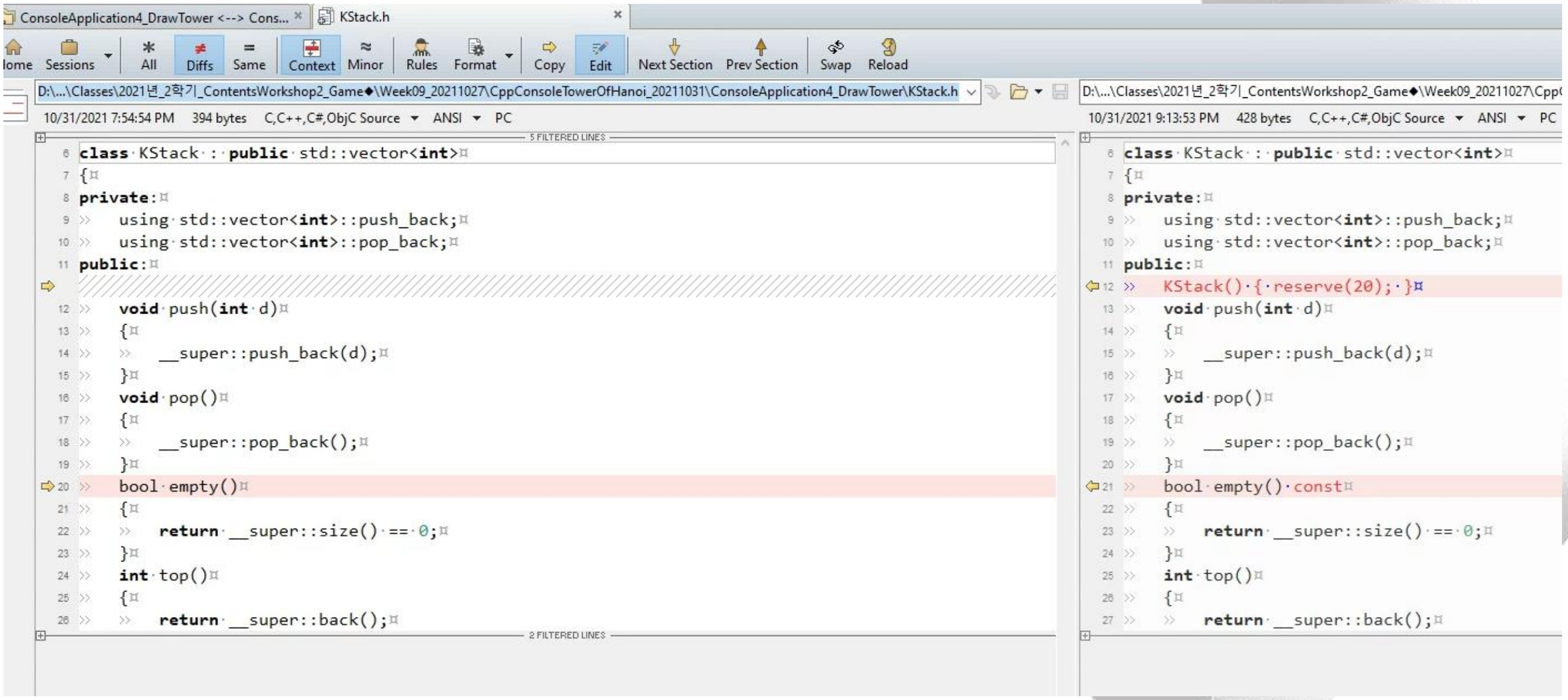

Sessions
All
Diffs
Same
Context
Minor
Rules
Format
Copy
Edit
Next Section
Prev Section
Swap
Reload
D:\...\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication3_TowerOfHanoi\KTowerOfHanoi.cpp
10/31/2021 7:54:54 PM 3,649 bytes C,C++,C#,ObjC Source ANSI PC
#include "KTowerOfHanoi.h"
#include "MyUtil.h"
/*static*/ const char KTowerOfHanoi::m_stackLookup[4] = "SAD";
KTowerOfHanoi::KTowerOfHanoi()
{
m_moveState = EMoveState::ESTATE_INIT;
D:\...\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication4_Draw
10/31/2021 8:27:03 PM 4,025 bytes C,C++,C#,ObjC Source ANSI PC
#include "KTowerOfHanoi.h"
#include "MyUtil.h"
/*static*/ const char KTowerOfHanoi::m_stackLool
//20211031_jintaeks
void PrintDisc(int left, int top, int width)
{
char buffer[80];
int index = 0;
for (int i = 0; i < width; ++i)
{
buffer[index] = '=';
buffer[index + width + 1] = '=';
index += 1;
}
buffer[width] = '|';
buffer[width * 2 + 1] = 0; // EOS
PutText(left, top, buffer);
}
KTowerOfHanoi::KTowerOfHanoi()
{
m_moveState = EMoveState::ESTATE_INIT;

```
            PutText(x, y, str.c_str());
            y += 1;
        }

        char buffer[80];
        int startx = 50;
        int starty = 20;
        const int width = m_numOfDisks * 2 + 1;
        x = startx;
        y = starty;
        for (int i = 0; i < 3; ++i) {
            x += width;
            y = starty;
            for (int n : m_stack[i]) {
                _itoa_s(n, buffer, 10);
                PutText(x, y, buffer);

                y -= 1;
            }
        }
    }
```

```
            PutText(x, y, str.c_str());
            y += 1;
        }

        char buffer[80];
        int startx = 50;
        int starty = 10;
        const int width = m_numOfDisks * 2 + 2;
        x = startx;
        y = starty;
        for (int i = 0; i < 3; ++i) {
            x += width;
            y = starty;
            for (int n : m_stack[i]) {
                _itoa_s(n, buffer, 10);
                PutText(x, y, buffer);
                PrintDisc(x - n, 20 + y, n); //jintaek on 20211031

                y -= 1;
            }
        }
    }
```

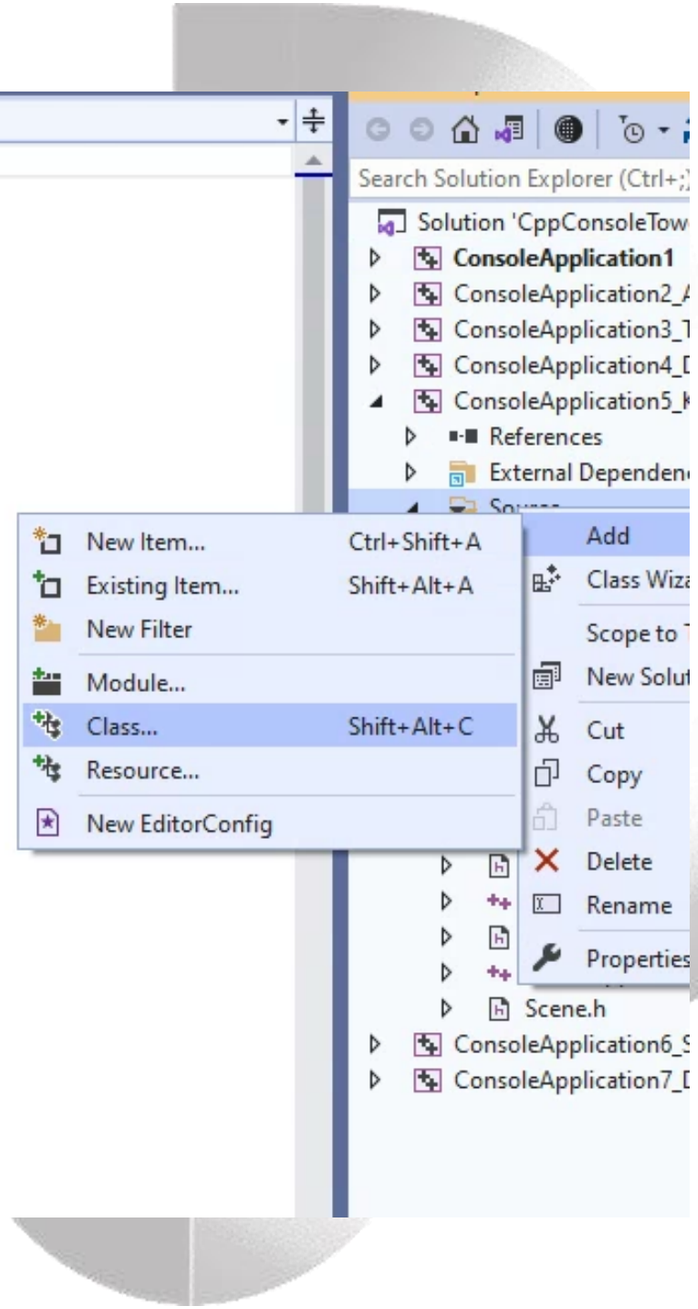

1: CppConsoleTowerOfHanoi_20...
Name
Date modified
Type
Size
ConsoleApplication1
10/31/2021 11:58 PM
File folder
ConsoleApplication2_AnimationLoop
10/31/2021 11:58 PM
File folder
ConsoleApplication3_TowerOfHanoi
10/31/2021 11:58 PM
File folder
ConsoleApplication4_DrawTower
ConsoleApplication5_KTower
ConsoleApplication6_StateTransition
ConsoleApplication7_DiskAnim
win32
.hgignore
CppConsoleTowerOfHanoi.sln
box
dropbox
!0210514
e DSU
MRU_2021예정학생
MRU_2022
!0206
Open
Open in new window
Pin to Quick access
Search Everything...
TreeSize Free
Open as Unity 2018 Project
Open as Unity 2019 Project
Open as Unity 2019 2.x Project
Open as Unity 2020 Project
Open with Visual Studio
Browse in Adobe Bridge CS6
한글 문서(HWP) 인쇄(P)
Open Folder as PyCharm Community Edition Project
Open Folder as VS Code Project
7-Zip
Agent Ransack...
Compare
Convert to file format...
Send with Transfer...


ConsoleApplication4_DrawTower <--> ConsoleApplication5_KTower - Folder Compare - Beyond Compare
Session Actions Edit Search View Tools Help
Home Sessions All Diffs Same Structure Minor Rules Copy Expand Collapse Select Files Refresh Swap Stop Filters: *.* Filters Peek
D:\Svn\jintaeks\Classes\2021년_2학기_ContentsWorkshop2_Game◆\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication4_DrawTower
D:\...\Classes\2021년_2학기_ContentsWorkshop2_Game◆\Week0
Name Size Modified
ConsoleApplication3_TowerOfHanoi.cpp 1,283 10/31/2021 8:14:14 PM
ConsoleApplication4_DrawTower.vcxproj 7,730 10/31/2021 8:08:51 PM
ConsoleApplication4_DrawTower.vcxproj.filters 1,443 10/31/2021 7:54:54 PM
ConsoleApplication4_DrawTower.vcxproj.user 168 10/31/2021 7:54:54 PM
KMatrix2.cpp 141 10/31/2021 7:54:54 PM
KMatrix2.h 2,326 10/31/2021 7:54:54 PM
KStack.h 394 10/31/2021 7:54:54 PM
KTowerOfHanoi.cpp 4,025 10/31/2021 8:27:03 PM
KTowerOfHanoi.h 695 10/31/2021 7:54:54 PM
KVector2.cpp 496 10/31/2021 7:54:54 PM
KVector2.h 1,135 10/31/2021 7:54:54 PM
MyUtil.cpp 3,556 10/31/2021 7:54:54 PM
MyUtil.h 566 10/31/2021 7:54:54 PM
Scene.cpp 1,222 10/31/2021 7:54:54 PM
Scene.h 283 10/31/2021 7:54:54 PM
Name
ConsoleApplication3_TowerOfHanoi.cpp
ConsoleApplication5_KTower.vcxproj
ConsoleApplication5_KTower.vcxproj.filters
ConsoleApplication5_KTower.vcxproj.user
KMatrix2.cpp
KMatrix2.h
KStack.h
KTower.cpp
KTower.h
KTowerOfHanoi.cpp
KTowerOfHanoi.h
KVector2.cpp
KVector2.h
MyUtil.cpp
MyUtil.h
Scene.cpp
Scene.h


ConsoleApplication4_DrawTower <--> Cons...
KStack.h
Sessions
All
Diffs
Same
Context
Minor
Rules
Format
Copy
Edit
Next Section
Prev Section
Swap
Reload
D:\...\Classes\2021년_2학기_ContentsWorkshop2_Game◆\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication4_DrawTower\KStack.h
10/31/2021 7:54:54 PM 394 bytes C,C++,C#,ObjC Source ANSI PC
5 FILTERED LINES
class KStack : public std::vector<int>
{
private:
using std::vector<int>::push_back;
using std::vector<int>::pop_back;
public:
void push(int d)
{
__super::push_back(d);
}
void pop()
{
__super::pop_back();
}
bool empty()
{
return __super::size() == 0;
}
int top()
{
return __super::back();
2 FILTERED LINES
D:\...\Classes\2021년_2학기_ContentsWorkshop2_Game◆\Week09_20211027\Cpp
10/31/2021 9:13:53 PM 428 bytes C,C++,C#,ObjC Source ANSI PC
class KStack : public std::vector<int>
{
private:
using std::vector<int>::push_back;
using std::vector<int>::pop_back;
public:
KStack() { reserve(20); }
void push(int d)
{
__super::push_back(d);
}
void pop()
{
__super::pop_back();
}
bool empty() const
{
return __super::size() == 0;
}
int top()
{
return __super::back();

pplication5_KTower (Global Scope)

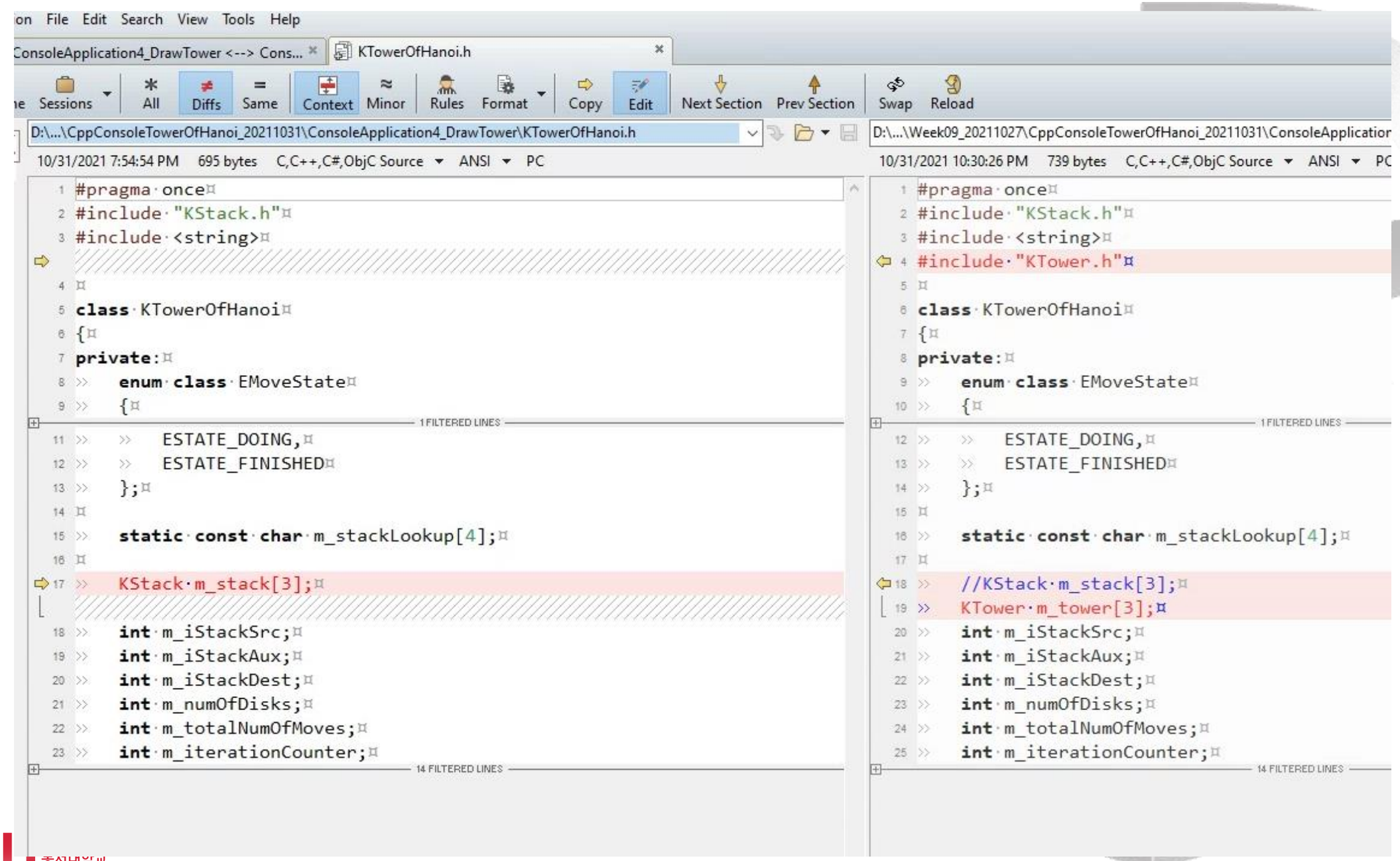
```cpp
#pragma once
#include "KStack.h"
#include "KVector2.h"

class KTower
{
public:
    typedef std::vector<int>::iterator iterator;
private:
    KStack      m_stack;
    KVector2    m_position;

public:
    KTower(int x = 0, int y = 0);
    void PushDisk(int d) { m_stack.push(d); }
    void PopDisk() { m_stack.pop(); }
    bool Empty() const { return m_stack.empty(); }
    int TopDisk() const { return m_stack.back(); }
    iterator begin() { return m_stack.begin(); }
    iterator end() { return m_stack.end(); }

    void Clear() { m_stack.clear(); }
    //KStack& GetStack() { return m_stack; }
    int GetDiskCount() const { return (int)m_stack.size(); }
    bool GetTopDiskPosition( KVector2& vOut );
};
```



KTower.cpp | KTower.h

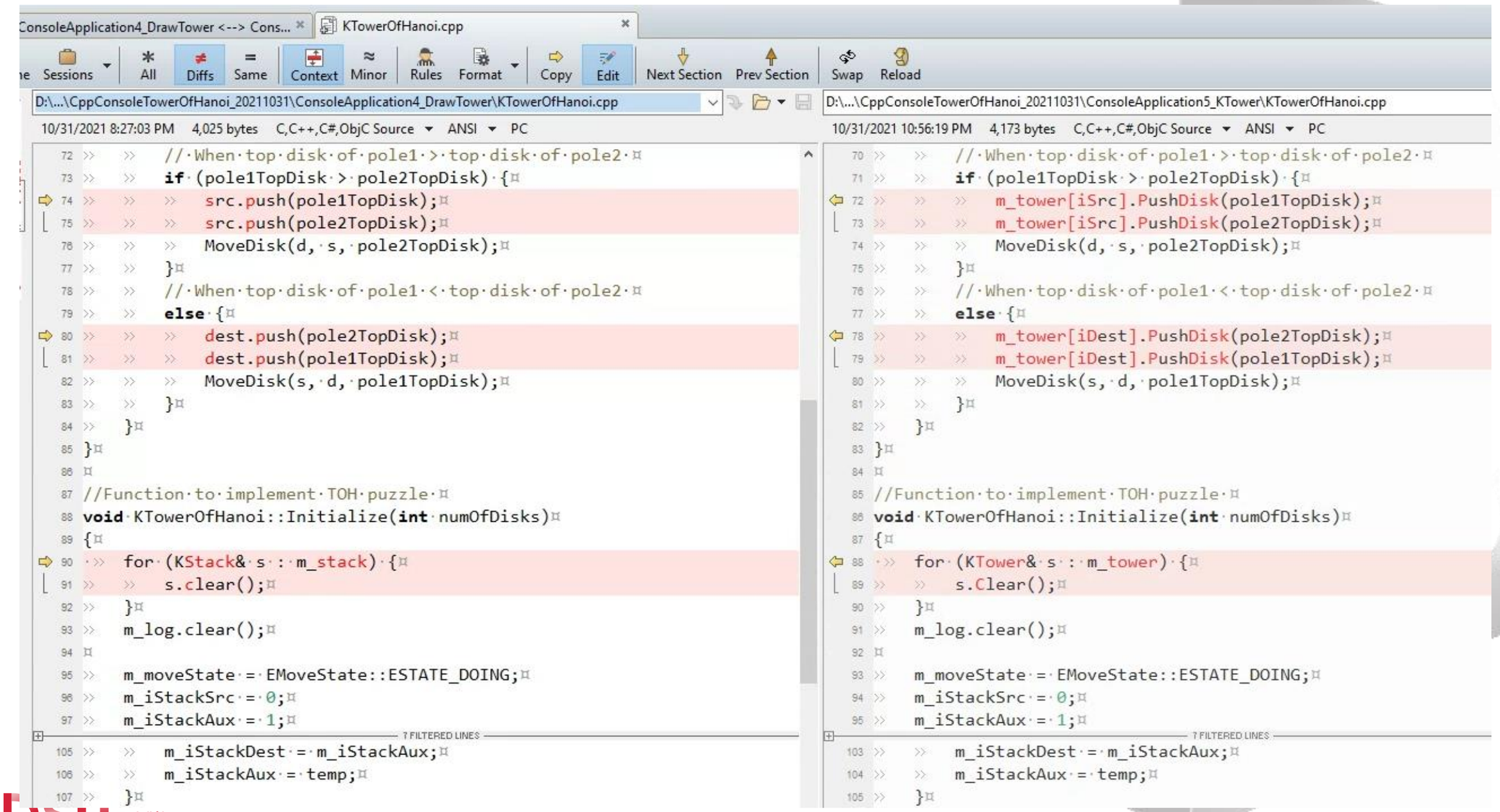
ConsoleApplication5_KTower | (Global Scope)

```cpp
#include "KTower.h"

KTower::KTower(int x, int y)
{
    m_stack.clear();
    m_position = KVector2::zero;
}

bool KTower::GetTopDiskPosition(KVector2& vOut )
{
    if (Empty())
        return false;

    int y = (int)m_position.y - GetDiskCount();
    int x = (int)m_position.x - TopDisk();
    vOut = KVector2(x, y); // set [out] parameter
    return true;
}
```


File Edit Search View Tools Help
ConsoleApplication4_DrawTower <--> Cons...
KTowerOfHanoi.h
Sessions
All
Diffs
Same
Context
Minor
Rules
Format
Copy
Edit
Next Section
Prev Section
Swap
Reload
D:\...\CppConsoleTowerOfHanoi_20211031\ConsoleApplication4_DrawTower\KTowerOfHanoi.h
10/31/2021 7:54:54 PM 695 bytes C,C++,C#,ObjC Source ANSI PC
D:\...\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplicatior
10/31/2021 10:30:26 PM 739 bytes C,C++,C#,ObjC Source ANSI PC
#pragma once
#include "KStack.h"
#include <string>
#include "KTower.h"
class KTowerOfHanoi
{
private:
enum class EMoveState
{
1 FILTERED LINES
ESTATE_DOING,
ESTATE_FINISHED
};
static const char m_stackLookup[4];
KStack m_stack[3];
//KStack m_stack[3];
KTower m_tower[3];
int m_iStackSrc;
int m_iStackAux;
int m_iStackDest;
int m_numOfDisks;
int m_totalNumOfMoves;
int m_iterationCounter;
14 FILTERED LINES

```
//·Function·to·implement·legal·movement·between·¤
//·two·poles·¤
void·KTowerOfHanoi::MoveDisksBetweenTwoPoles(int·iSrc,·int·iDest)¤
{¤
»   KStack&·src·=·m_stack[iSrc];¤
»   KStack&·dest·=·m_stack[iDest];¤
»   char·s·=·m_stackLookup[iSrc];¤
»   char·d·=·m_stackLookup[iDest];¤
¤
»   //·When·pole·1·is·empty·¤
»   if·(src.empty())·{¤
»   »   int·pole2TopDisk·=·dest.top();¤
»   »   dest.pop();¤
»   »   src.push(pole2TopDisk);¤
»   »   MoveDisk(d,·s,·pole2TopDisk);¤
»   }¤
»   //·When·pole2·pole·is·empty·¤
»   else·if·(dest.empty())·{¤
»   »   int·pole1TopDisk·=·src.top();¤
»   »   src.pop();¤
»   »   dest.push(pole1TopDisk);¤
»   »   MoveDisk(s,·d,·pole1TopDisk);¤
»   }¤
»   else·{¤
»   »   int·pole1TopDisk·=·src.top();¤
»   »   int·pole2TopDisk·=·dest.top();¤
»   »   src.pop();¤
»   »   dest.pop();¤
»   »   //·When·top·disk·of·pole1·>·top·disk·of·pole2·¤
»   »   if·(pole1TopDisk·>·pole2TopDisk)·{¤
»   »   »   src.push(pole1TopDisk);¤
»   »   »   src.push(pole2TopDisk);¤
»   »   »   MoveDisk(d,·s,·pole2TopDisk);¤
»   »   }¤
»   »   //·When·top·disk·of·pole1·<·top·disk·of·pole2·¤
```

```
//·Function·to·implement·legal·movement·between·¤
//·two·poles·¤
void·KTowerOfHanoi::MoveDisksBetweenTwoPoles(int·iSrc,·int·iI
{¤
»   char·s·=·m_stackLookup[iSrc];¤
»   char·d·=·m_stackLookup[iDest];¤
¤
»   //·When·pole·1·is·empty·¤
»   if·(m_tower[iSrc].Empty())·{¤
»   »   int·pole2TopDisk·=·m_tower[iDest].TopDisk();¤
»   »   m_tower[iDest].PopDisk();¤
»   »   m_tower[iSrc].PushDisk(pole2TopDisk);¤
»   »   MoveDisk(d,·s,·pole2TopDisk);¤
»   }¤
»   //·When·pole2·pole·is·empty·¤
»   else·if·(m_tower[iDest].Empty())·{¤
»   »   int·pole1TopDisk·=·m_tower[iSrc].TopDisk();¤
»   »   m_tower[iSrc].PopDisk();¤
»   »   m_tower[iDest].PushDisk(pole1TopDisk);¤
»   »   MoveDisk(s,·d,·pole1TopDisk);¤
»   }¤
»   else·{¤
»   »   int·pole1TopDisk·=·m_tower[iSrc].TopDisk();¤
»   »   int·pole2TopDisk·=·m_tower[iDest].TopDisk();¤
»   »   m_tower[iSrc].PopDisk();¤
»   »   m_tower[iDest].PopDisk();¤
»   »   //·When·top·disk·of·pole1·>·top·disk·of·pole2·¤
»   »   if·(pole1TopDisk·>·pole2TopDisk)·{¤
»   »   »   m_tower[iSrc].PushDisk(pole1TopDisk);¤
»   »   »   m_tower[iSrc].PushDisk(pole2TopDisk);¤
»   »   »   MoveDisk(d,·s,·pole2TopDisk);¤
»   »   }¤
»   »   //·When·top·disk·of·pole1·<·top·disk·of·pole2·¤
```

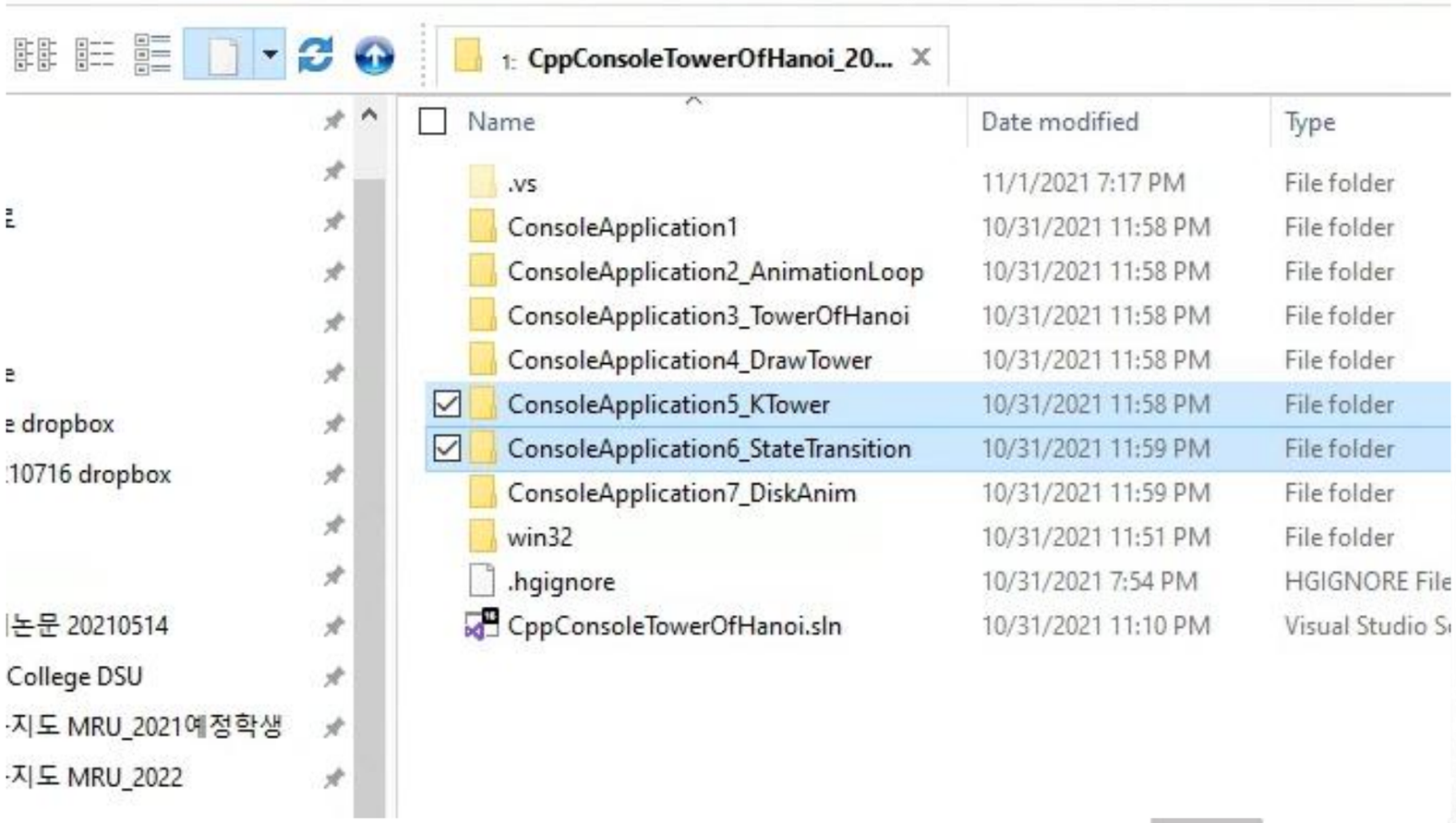

ConsoleApplication4_DrawTower <--> Cons...
KTowerOfHanoi.cpp
Sessions
All
Diffs
Same
Context
Minor
Rules
Format
Copy
Edit
Next Section
Prev Section
Swap
Reload
D:\...\CppConsoleTowerOfHanoi_20211031\ConsoleApplication4_DrawTower\KTowerOfHanoi.cpp
10/31/2021 8:27:03 PM 4,025 bytes C,C++,C#,ObjC Source ANSI PC
D:\...\CppConsoleTowerOfHanoi_20211031\ConsoleApplication5_KTower\KTowerOfHanoi.cpp
10/31/2021 10:56:19 PM 4,173 bytes C,C++,C#,ObjC Source ANSI PC
//When top disk of pole1 > top disk of pole2
if (pole1TopDisk > pole2TopDisk) {
src.push(pole1TopDisk);
src.push(pole2TopDisk);
MoveDisk(d, s, pole2TopDisk);
}
//When top disk of pole1 < top disk of pole2
else {
dest.push(pole2TopDisk);
dest.push(pole1TopDisk);
MoveDisk(s, d, pole1TopDisk);
}
}
}
//Function to implement TOH puzzle
void KTowerOfHanoi::Initialize(int numOfDisks)
{
for (KStack& s : m_stack) {
s.clear();
}
m_log.clear();
m_moveState = EMoveState::ESTATE_DOING;
m_iStackSrc = 0;
m_iStackAux = 1;
7 FILTERED LINES
m_iStackDest = m_iStackAux;
m_iStackAux = temp;
}
m_tower[iSrc].PushDisk(pole1TopDisk);
m_tower[iSrc].PushDisk(pole2TopDisk);
m_tower[iDest].PushDisk(pole2TopDisk);
m_tower[iDest].PushDisk(pole1TopDisk);
for (KTower& s : m_tower) {
s.Clear();

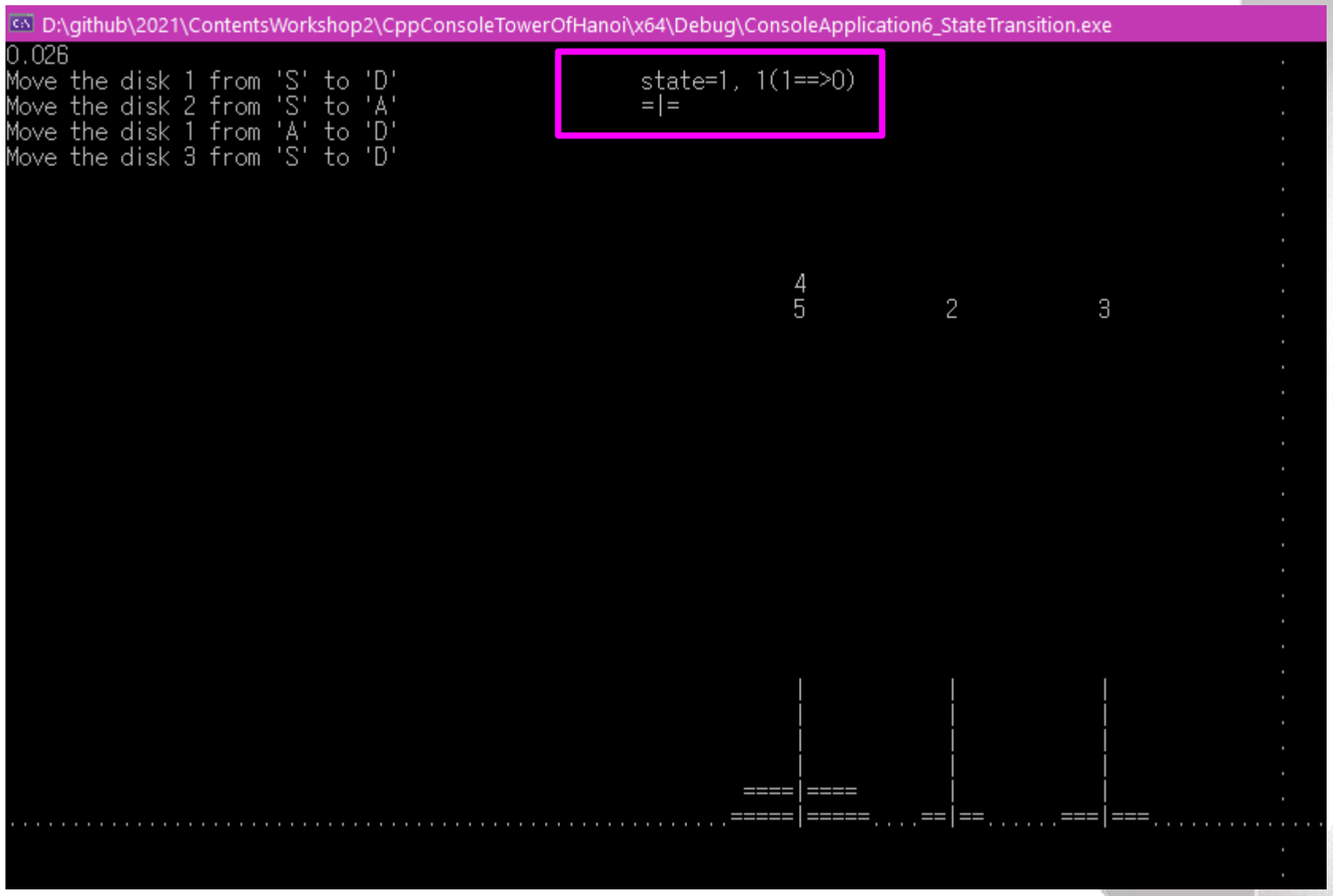
```
88  void KTowerOfHanoi::Initialize(int numOfDisks)
89  {
90      for (KStack& s : m_stack) {
91          s.clear();
92      }
93      m_log.clear();
94
95      m_moveState = EMoveState::ESTATE_DOING;
96      m_iStackSrc = 0;
97      m_iStackAux = 1;
        7 FILTERED LINES
105         m_iStackDest = m_iStackAux;
106         m_iStackAux = temp;
107     }
108     m_numOfDisks = numOfDisks;
109     m_totalNumOfMoves = (int)pow(2, numOfDisks) - 1;
110
111     KStack& src = m_stack[m_iStackSrc];
112     //Larger disks will be pushed first
113     for (int i = m_numOfDisks; i >= 1; i--)
114         src.push(i);
115 }
116
117 //Function to implement TOH puzzle
118 void KTowerOfHanoi::Iterate(double elapsedTime)
119 {
120     if (m_moveState == EMoveState::ESTATE_FINISHED) {
        3 FILTERED LINES
124     m_oneStepTimer += elapsedTime;
125     if (m_oneStepTimer < 1.0) {
126         return;
127     }
128     m_oneStepTimer = 0.0;
129
130     KStack& src = m_stack[m_iStackSrc];
131     KStack& aux = m_stack[m_iStackAux];
132     KStack& dest = m_stack[m_iStackDest];
133     m_iterationCounter += 1;
```

```
86  void KTowerOfHanoi::Initialize(int numOfDisks)
87  {
88      for (KTower& s : m_tower) {
89          s.Clear();
90      }
91      m_log.clear();
92
93      m_moveState = EMoveState::ESTATE_DOING;
94      m_iStackSrc = 0;
95      m_iStackAux = 1;
        7 FILTERED LINES
103         m_iStackDest = m_iStackAux;
104         m_iStackAux = temp;
105     }
106     m_numOfDisks = numOfDisks;
107     m_totalNumOfMoves = (int)pow(2, numOfDisks) - 1;
108
109     KTower& src = m_tower[m_iStackSrc];
110     //Larger disks will be pushed first
111     for (int i = m_numOfDisks; i >= 1; i--)
112         src.PushDisk(i);
113 }
114
115 //Function to implement TOH puzzle
116 void KTowerOfHanoi::Iterate(double elapsedTime)
117 {
118     if (m_moveState == EMoveState::ESTATE_FINISHED) {
        3 FILTERED LINES
122     m_oneStepTimer += elapsedTime;
123     if (m_oneStepTimer < 1.0) {
124         return;
125     }
126     m_oneStepTimer = 0.0;
127
128     m_iterationCounter += 1;
```

```
127 »   }¤
128 »   m_oneStepTimer·=·0.0;¤
129 ¤
130 »   KStack&·src·=·m_stack[m_iStackSrc];¤
131 »   KStack&·aux·=·m_stack[m_iStackAux];¤
132 »   KStack&·dest·=·m_stack[m_iStackDest];¤
133 »   m_iterationCounter·+=·1;¤
134 »   if·(m_iterationCounter·>·m_totalNumOfMoves)·{¤
135 »   »   m_moveState·=·EMoveState::ESTATE_FINISHED;¤
136 »   »   return;¤
137 »   }¤
138 ¤
      20 FILTERED LINES
159 »   const·int·width·=·m_numOfDisks·*·2·+·2;¤
160 »   x·=·startx;¤
161 »   y·=·starty;¤
162 »   for·(int·i·=·0;·i·<·3;·++i)·{¤
163 »   »   x·+=·width;¤
164 »   »   y·=·starty;¤
165 »   »   for·(int·n·:·m_stack[i])·{¤
166 »   »   »   _itoa_s(n,·buffer,·10);¤
167 »   »   »   PutText(x,·y,·buffer);¤
168 »   »   »   PrintDisc(x·-·n,·20·+·y,·n);·//·jintaek·on·20211031¤
169 »   »   »   y·-=·1;¤
170 »   »   }¤
171 »   }¤
      1 FILTERED LINES
1   Operator
```

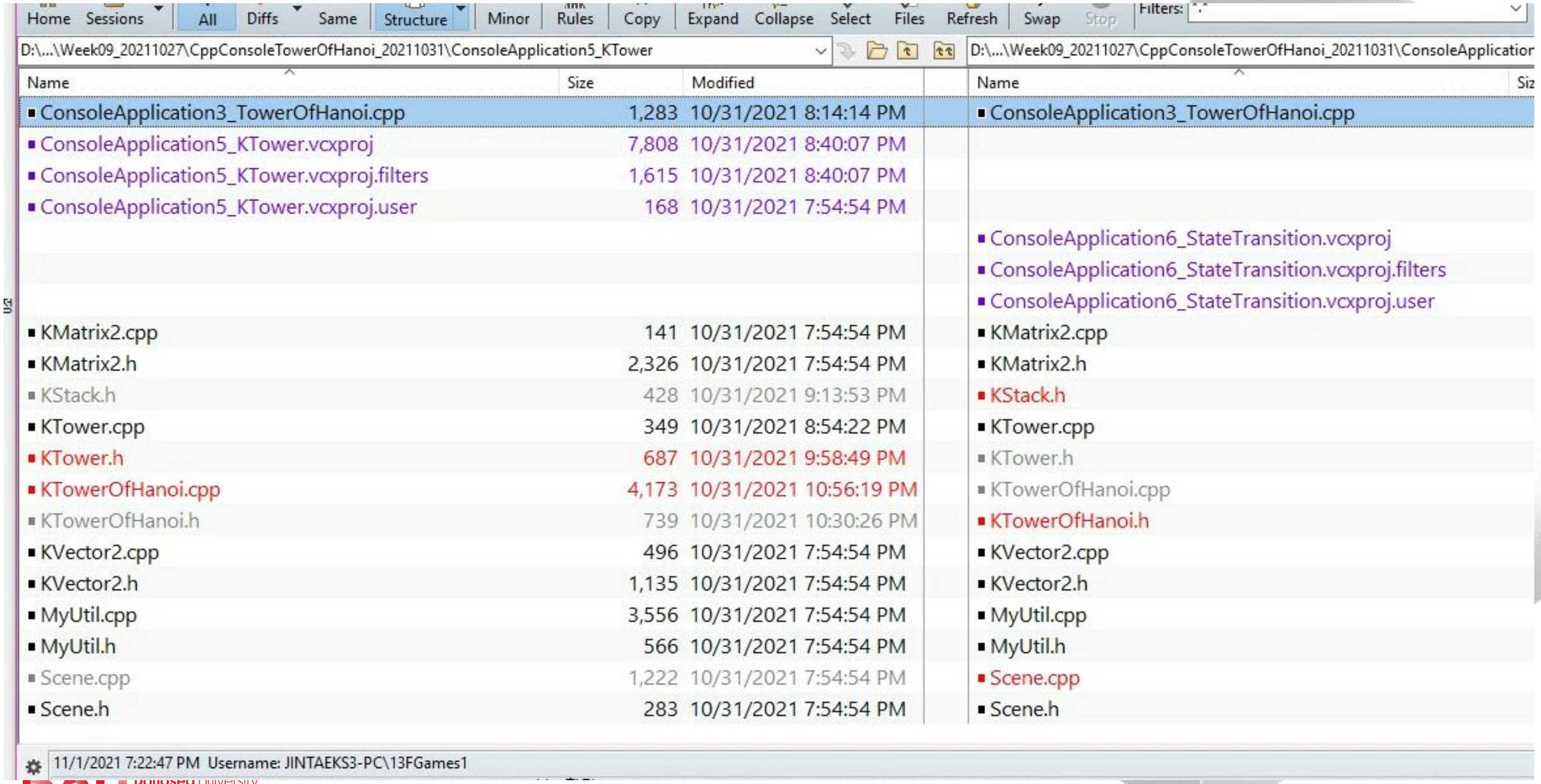
```
125 »   }¤
126 »   m_oneStepTimer·=·0.0;¤
127 ¤
128 »   m_iterationCounter·+=·1;¤
129 »   if·(m_iterationCounter·>·m_totalNumOfMoves)·{¤
130 »   »   m_moveState·=·EMoveState::ESTATE_FINISHED;¤
131 »   »   return;¤
132 »   }¤
133 ¤
      20 FILTERED LINES
154 »   const·int·width·=·m_numOfDisks·*·2·+·2;¤
155 »   x·=·startx;¤
156 »   y·=·starty;¤
157 »   for·(int·i·=·0;·i·<·3;·++i)·{¤
158 »   »   x·+=·width;¤
159 »   »   y·=·starty;¤
160 »   »   for·(int·j·=·0;·j·<·m_numOfDisks·+·1;·++j)¤
161 »   »   »   PutText(x,·20·+·y·-·j,·"|");¤
162 »   »   for·(int·n·:·m_tower[i])·{·//·custom·iterator,·20211031¤
163 »   »   »   _itoa_s(n,·buffer,·10);¤
164 »   »   »   PutText(x,·y,·buffer);¤
165 »   »   »   PrintDisc(x·-·n,·20·+·y,·n);·//·jintaek·on·20211031¤
166 »   »   »   y·-=·1;¤
167 »   »   }¤
168 »   }¤
      1 FILTERED LINES
42: 1   Operator
```

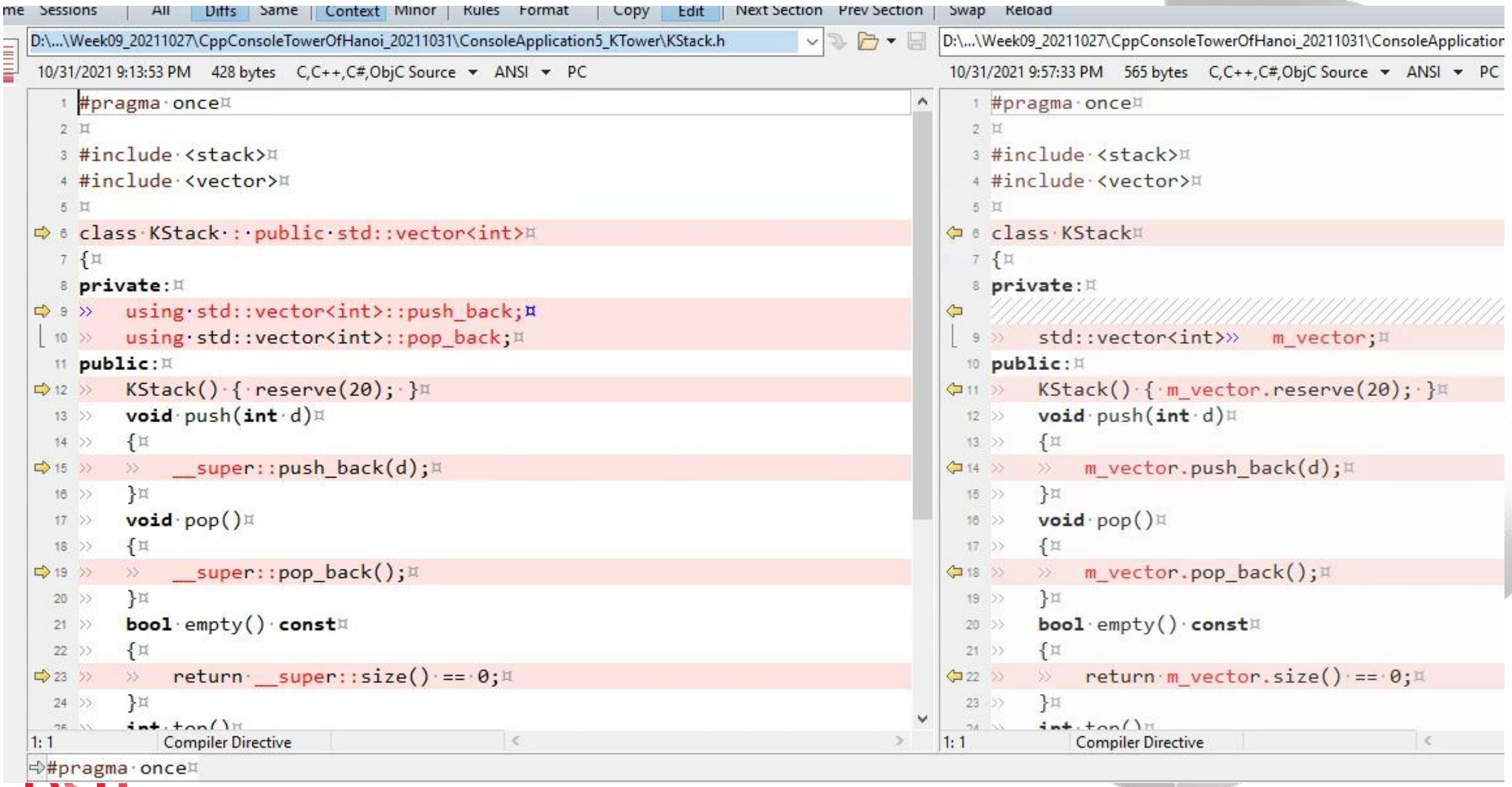

1: CppConsoleTowerOfHanoi_20...
Name
Date modified
Type
.vs
11/1/2021 7:17 PM
File folder
ConsoleApplication1
10/31/2021 11:58 PM
File folder
ConsoleApplication2_AnimationLoop
10/31/2021 11:58 PM
File folder
ConsoleApplication3_TowerOfHanoi
10/31/2021 11:58 PM
File folder
ConsoleApplication4_DrawTower
10/31/2021 11:58 PM
File folder
ConsoleApplication5_KTower
10/31/2021 11:58 PM
File folder
ConsoleApplication6_StateTransition
10/31/2021 11:59 PM
File folder
ConsoleApplication7_DiskAnim
10/31/2021 11:59 PM
File folder
win32
10/31/2021 11:51 PM
File folder
.hgignore
10/31/2021 7:54 PM
HGIGNORE File
CppConsoleTowerOfHanoi.sln
10/31/2021 11:10 PM
Visual Studio S
dropbox
10716 dropbox
논문 20210514
College DSU
지도 MRU_2021예정학생
지도 MRU_2022

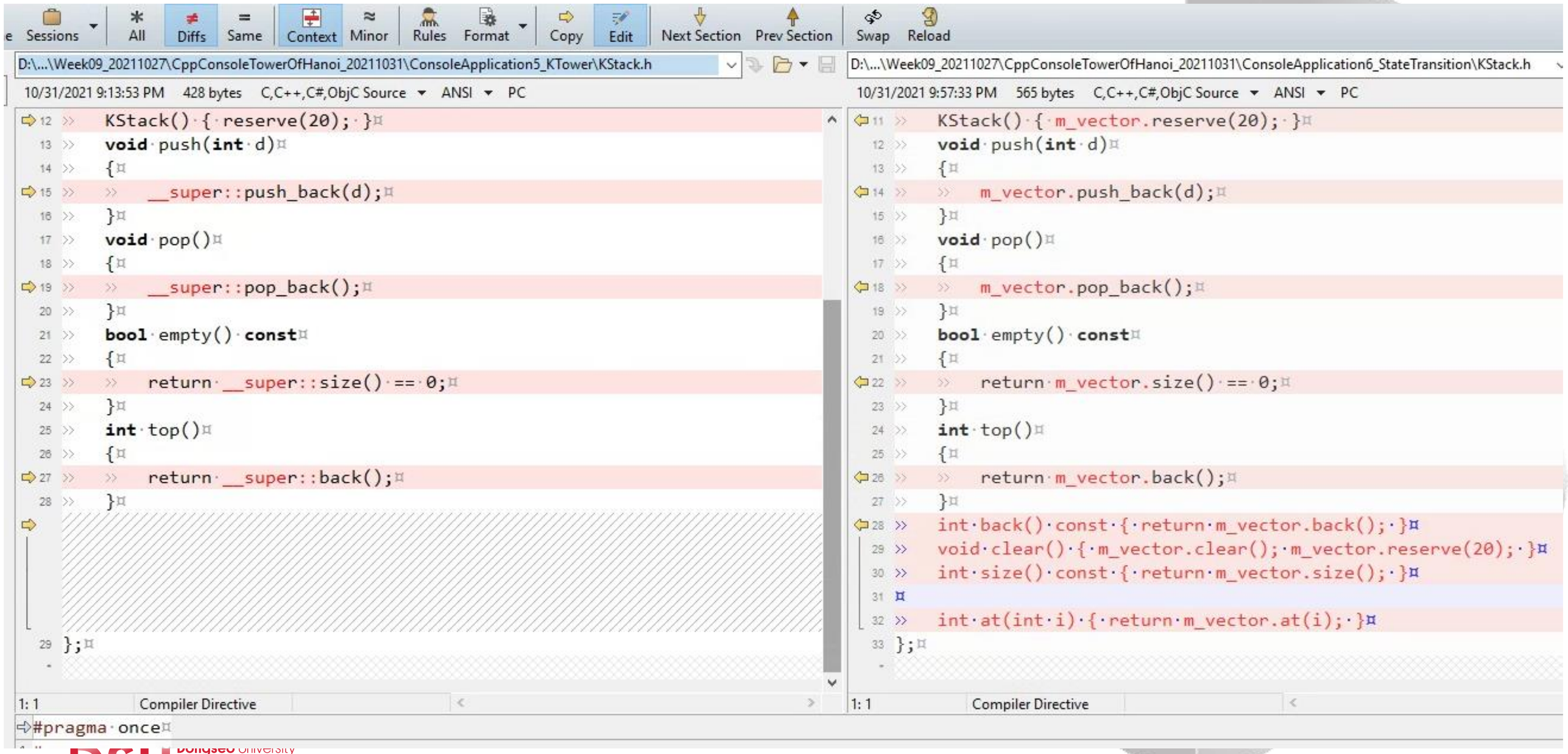


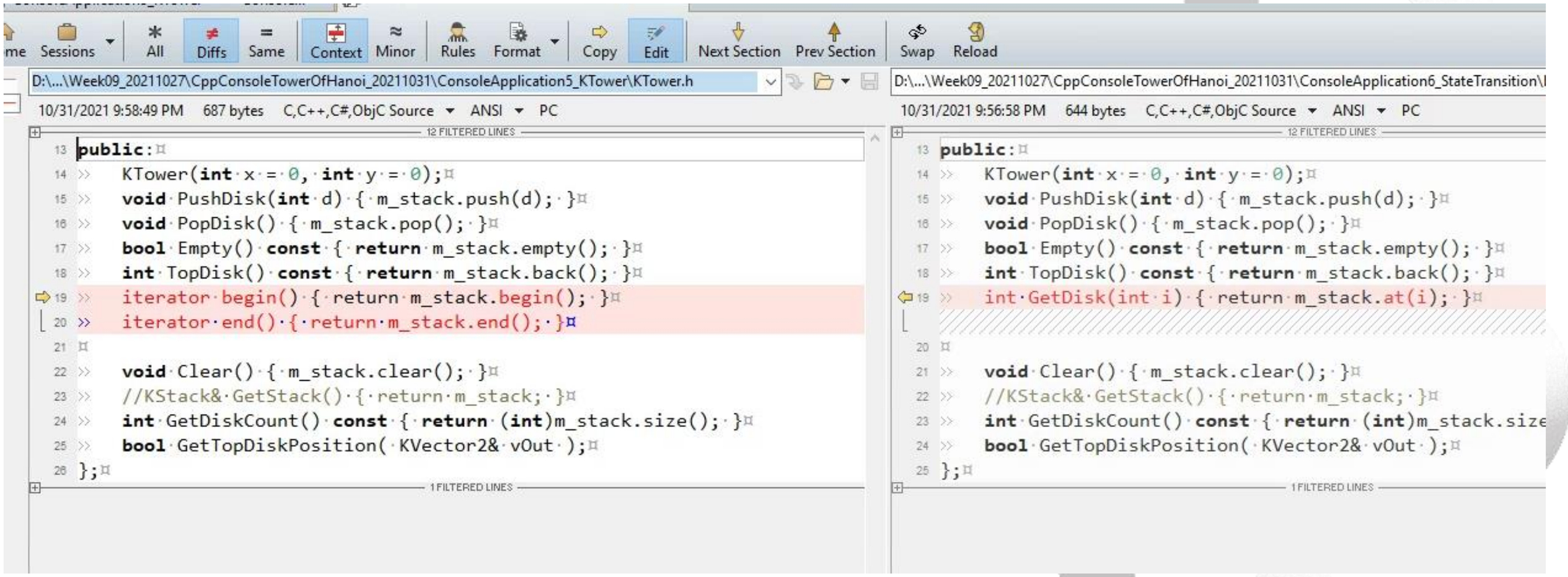



Dongseo University 동서대학교

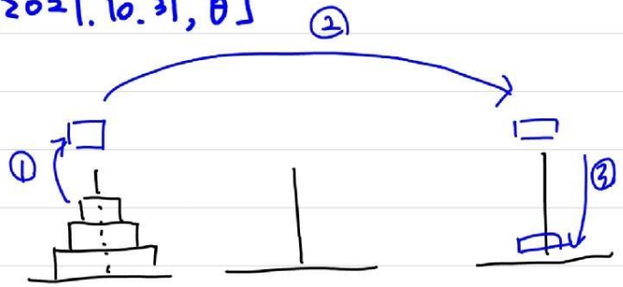

Home Sessions All Diffs Same Structure Minor Rules Copy Expand Collapse Select Files Refresh Swap Stop Filters: *.*
D:\...\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication5_KTower
D:\...\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplicatior
Name Size Modified
ConsoleApplication3_TowerOfHanoi.cpp 1,283 10/31/2021 8:14:14 PM
ConsoleApplication5_KTower.vcxproj 7,808 10/31/2021 8:40:07 PM
ConsoleApplication5_KTower.vcxproj.filters 1,615 10/31/2021 8:40:07 PM
ConsoleApplication5_KTower.vcxproj.user 168 10/31/2021 7:54:54 PM
KMatrix2.cpp 141 10/31/2021 7:54:54 PM
KMatrix2.h 2,326 10/31/2021 7:54:54 PM
KStack.h 428 10/31/2021 9:13:53 PM
KTower.cpp 349 10/31/2021 8:54:22 PM
KTower.h 687 10/31/2021 9:58:49 PM
KTowerOfHanoi.cpp 4,173 10/31/2021 10:56:19 PM
KTowerOfHanoi.h 739 10/31/2021 10:30:26 PM
KVector2.cpp 496 10/31/2021 7:54:54 PM
KVector2.h 1,135 10/31/2021 7:54:54 PM
MyUtil.cpp 3,556 10/31/2021 7:54:54 PM
MyUtil.h 566 10/31/2021 7:54:54 PM
Scene.cpp 1,222 10/31/2021 7:54:54 PM
Scene.h 283 10/31/2021 7:54:54 PM
Name
ConsoleApplication3_TowerOfHanoi.cpp
ConsoleApplication6_StateTransition.vcxproj
ConsoleApplication6_StateTransition.vcxproj.filters
ConsoleApplication6_StateTransition.vcxproj.user
KMatrix2.cpp
KMatrix2.h
KStack.h
KTower.cpp
KTower.h
KTowerOfHanoi.cpp
KTowerOfHanoi.h
KVector2.cpp
KVector2.h
MyUtil.cpp
MyUtil.h
Scene.cpp
Scene.h
11/1/2021 7:22:47 PM Username: JINTAEKS3-PC\13FGames1


Sessions | All | Diffs | Same | Context | Minor | Rules | Format | Copy | Edit | Next Section | Prev Section | Swap | Reload
D:\...\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication5_KTower\KStack.h
10/31/2021 9:13:53 PM 428 bytes C,C++,C#,ObjC Source ANSI PC
#pragma once
#include <stack>
#include <vector>
class KStack : public std::vector<int>
{
private:
using std::vector<int>::push_back;
using std::vector<int>::pop_back;
public:
KStack() { reserve(20); }
void push(int d)
{
__super::push_back(d);
}
void pop()
{
__super::pop_back();
}
bool empty() const
{
return __super::size() == 0;
}
1: 1 Compiler Directive
D:\...\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplicatior
10/31/2021 9:57:33 PM 565 bytes C,C++,C#,ObjC Source ANSI PC
#pragma once
#include <stack>
#include <vector>
class KStack
{
private:
std::vector<int>> m_vector;
public:
KStack() { m_vector.reserve(20); }
void push(int d)
{
m_vector.push_back(d);
}
void pop()
{
m_vector.pop_back();
}
bool empty() const
{
return m_vector.size() == 0;
}
1: 1 Compiler Directive
#pragma once


Sessions
All
Diffs
Same
Context
Minor
Rules
Format
Copy
Edit
Next Section
Prev Section
Swap
Reload
D:\...\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication5_KTower\KStack.h
10/31/2021 9:13:53 PM 428 bytes C,C++,C#,ObjC Source ANSI PC
KStack() { reserve(20); }
void push(int d)
{
__super::push_back(d);
}
void pop()
{
__super::pop_back();
}
bool empty() const
{
return __super::size() == 0;
}
int top()
{
return __super::back();
}
};
1: 1 Compiler Directive
#pragma once
D:\...\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication6_StateTransition\KStack.h
10/31/2021 9:57:33 PM 565 bytes C,C++,C#,ObjC Source ANSI PC
KStack() { m_vector.reserve(20); }
void push(int d)
{
m_vector.push_back(d);
}
void pop()
{
m_vector.pop_back();
}
bool empty() const
{
return m_vector.size() == 0;
}
int top()
{
return m_vector.back();
}
int back() const { return m_vector.back(); }
void clear() { m_vector.clear(); m_vector.reserve(20); }
int size() const { return m_vector.size(); }
int at(int i) { return m_vector.at(i); }
};
1: 1 Compiler Directive


Sessions
All
Diffs
Same
Context
Minor
Rules
Format
Copy
Edit
Next Section
Prev Section
Swap
Reload
D:\...\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication5_KTower\KTower.h
10/31/2021 9:58:49 PM 687 bytes C,C++,C#,ObjC Source ANSI PC
12 FILTERED LINES
public:
KTower(int x = 0, int y = 0);
void PushDisk(int d) { m_stack.push(d); }
void PopDisk() { m_stack.pop(); }
bool Empty() const { return m_stack.empty(); }
int TopDisk() const { return m_stack.back(); }
iterator begin() { return m_stack.begin(); }
iterator end() { return m_stack.end(); }
void Clear() { m_stack.clear(); }
//KStack& GetStack() { return m_stack; }
int GetDiskCount() const { return (int)m_stack.size(); }
bool GetTopDiskPosition( KVector2& vOut );
};
1 FILTERED LINES
D:\...\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication6_StateTransition\
10/31/2021 9:56:58 PM 644 bytes C,C++,C#,ObjC Source ANSI PC
12 FILTERED LINES
public:
KTower(int x = 0, int y = 0);
void PushDisk(int d) { m_stack.push(d); }
void PopDisk() { m_stack.pop(); }
bool Empty() const { return m_stack.empty(); }
int TopDisk() const { return m_stack.back(); }
int GetDisk(int i) { return m_stack.at(i); }
void Clear() { m_stack.clear(); }
//KStack& GetStack() { return m_stack; }
int GetDiskCount() const { return (int)m_stack.size
bool GetTopDiskPosition( KVector2& vOut );
};
1 FILTERED LINES

[2021.10.31, 日]

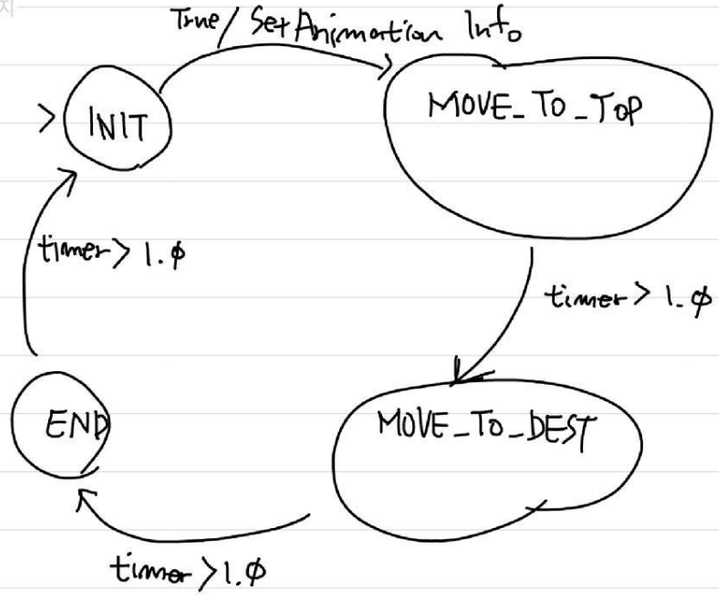



① MOVE_TO_TOP — iAnimSrcTower
② MOVE_TO_DEST — iAnimDestTower
③ END — iAnimDisk
vAnimBegin
vAnimEnd

66 페이지

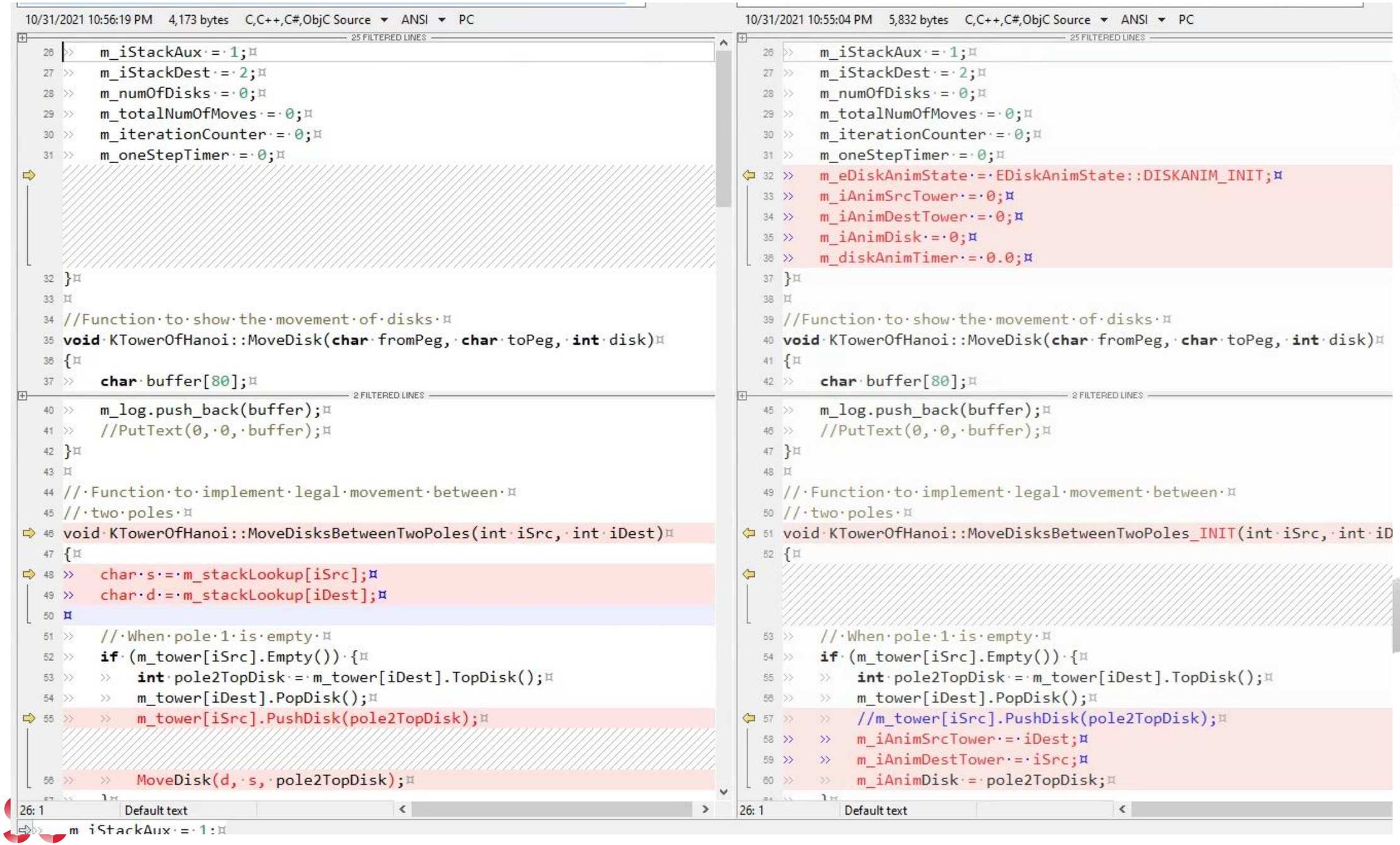
67 페이지

(61)



```
    int m_iterationCounter;
    EMoveState m_moveState;
    std::vector<std::string> m_log;
    double m_oneStepTimer;

public:
    KTowerOfHanoi();
    void Initialize(int numOfDisks);
    void Iterate(double elapsedTime);
    void Draw();
private:
    void MoveDisk(char fromPeg, char toPeg, int disk);
    void MoveDisksBetweenTwoPoles(int iSrc, int iDest);

};
```

```
    int m_iterationCounter;
    EMoveState m_moveState;
    std::vector<std::string> m_log;
    double m_oneStepTimer;

    enum class EDiskAnimState
    {
        DISKANIM_INIT = 0,
        DISKANIM_MOVE_TO_TOP,
        DISKANIM_MOVE_TO_DEST,
        DISKANIM_END,
    };

    EDiskAnimState  m_eDiskAnimState;
    int             m_iAnimSrcTower;
    int             m_iAnimDestTower;
    int             m_iAnimDisk;
    KVector2        m_vAnimBegin;
    KVector2        m_vAnimEnd;
    double          m_diskAnimTimer;

public:
    KTowerOfHanoi();
    void Initialize(int numOfDisks);
    void Iterate(double elapsedTime);
    void Draw();
private:
    void MoveDisk(char fromPeg, char toPeg, int disk);
    void MoveDisksBetweenTwoPoles(int iSrc, int iDest);
    void MoveDisksBetweenTwoPoles_INIT(int iSrc, int iDest);
};
```


10/31/2021 10:56:19 PM 4,173 bytes C,C++,C#,ObjC Source ANSI PC
25 FILTERED LINES
m_iStackAux = 1;
m_iStackDest = 2;
m_numOfDisks = 0;
m_totalNumOfMoves = 0;
m_iterationCounter = 0;
m_oneStepTimer = 0;
}
//Function to show the movement of disks
void KTowerOfHanoi::MoveDisk(char fromPeg, char toPeg, int disk)
{
char buffer[80];
2 FILTERED LINES
m_log.push_back(buffer);
//PutText(0, 0, buffer);
}
// Function to implement legal movement between
// two poles
void KTowerOfHanoi::MoveDisksBetweenTwoPoles(int iSrc, int iDest)
{
char s = m_stackLookup[iSrc];
char d = m_stackLookup[iDest];
// When pole 1 is empty
if (m_tower[iSrc].Empty()) {
int pole2TopDisk = m_tower[iDest].TopDisk();
m_tower[iDest].PopDisk();
m_tower[iSrc].PushDisk(pole2TopDisk);
MoveDisk(d, s, pole2TopDisk);
26: 1
Default text
m_iStackAux = 1;
10/31/2021 10:55:04 PM 5,832 bytes C,C++,C#,ObjC Source ANSI PC
25 FILTERED LINES
m_iStackAux = 1;
m_iStackDest = 2;
m_numOfDisks = 0;
m_totalNumOfMoves = 0;
m_iterationCounter = 0;
m_oneStepTimer = 0;
m_eDiskAnimState = EDiskAnimState::DISKANIM_INIT;
m_iAnimSrcTower = 0;
m_iAnimDestTower = 0;
m_iAnimDisk = 0;
m_diskAnimTimer = 0.0;
}
//Function to show the movement of disks
void KTowerOfHanoi::MoveDisk(char fromPeg, char toPeg, int disk)
{
char buffer[80];
2 FILTERED LINES
m_log.push_back(buffer);
//PutText(0, 0, buffer);
}
// Function to implement legal movement between
// two poles
void KTowerOfHanoi::MoveDisksBetweenTwoPoles_INIT(int iSrc, int iD
{
// When pole 1 is empty
if (m_tower[iSrc].Empty()) {
int pole2TopDisk = m_tower[iDest].TopDisk();
m_tower[iDest].PopDisk();
//m_tower[iSrc].PushDisk(pole2TopDisk);
m_iAnimSrcTower = iDest;
m_iAnimDestTower = iSrc;
m_iAnimDisk = pole2TopDisk;
26: 1
Default text

10/31/2021 10:56:19 PM 4,173 bytes C,C++,C#,ObjC Source ANSI PC

```
46 void KTowerOfHanoi::MoveDisksBetweenTwoPoles(int iSrc, int iDest)
47 {
48     char s = m_stackLookup[iSrc];
49     char d = m_stackLookup[iDest];
50
51     // When pole 1 is empty
52     if (m_tower[iSrc].Empty()) {
53         int pole2TopDisk = m_tower[iDest].TopDisk();
54         m_tower[iDest].PopDisk();
55         m_tower[iSrc].PushDisk(pole2TopDisk);
56         MoveDisk(d, s, pole2TopDisk);
57     }
58     // When pole2 pole is empty
59     else if (m_tower[iDest].Empty()) {
60         int pole1TopDisk = m_tower[iSrc].TopDisk();
61         m_tower[iSrc].PopDisk();
62         m_tower[iDest].PushDisk(pole1TopDisk);
63         MoveDisk(s, d, pole1TopDisk);
64     }
65     else {
66         int pole1TopDisk = m_tower[iSrc].TopDisk();
67         int pole2TopDisk = m_tower[iDest].TopDisk();
68         m_tower[iSrc].PopDisk();
69         m_tower[iDest].PopDisk();
70         // When top disk of pole1 > top disk of pole2
71         if (pole1TopDisk > pole2TopDisk) {
72             m_tower[iSrc].PushDisk(pole1TopDisk);
73             m_tower[iSrc].PushDisk(pole2TopDisk);
74             MoveDisk(d, s, pole2TopDisk);
75         }
```

10/31/2021 10:55:04 PM 5,832 bytes C,C++,C#,ObjC Source ANSI PC

```
51 void KTowerOfHanoi::MoveDisksBetweenTwoPoles_INIT(int iSrc, int i
52 {
53     // When pole 1 is empty
54     if (m_tower[iSrc].Empty()) {
55         int pole2TopDisk = m_tower[iDest].TopDisk();
56         m_tower[iDest].PopDisk();
57         //m_tower[iSrc].PushDisk(pole2TopDisk);
58         m_iAnimSrcTower = iDest;
59         m_iAnimDestTower = iSrc;
60         m_iAnimDisk = pole2TopDisk;
61     }
62     // When pole2 pole is empty
63     else if (m_tower[iDest].Empty()) {
64         int pole1TopDisk = m_tower[iSrc].TopDisk();
65         m_tower[iSrc].PopDisk();
66         //m_tower[iDest].PushDisk(pole1TopDisk);
67         m_iAnimSrcTower = iSrc;
68         m_iAnimDestTower = iDest;
69         m_iAnimDisk = pole1TopDisk;
70     }
71     else {
72         int pole1TopDisk = m_tower[iSrc].TopDisk();
73         int pole2TopDisk = m_tower[iDest].TopDisk();
74         // When top disk of pole1 > top disk of pole2
75         if (pole1TopDisk > pole2TopDisk) {
76             m_tower[iDest].PopDisk();
77             //m_tower[iSrc].PushDisk(pole2TopDisk);
78             m_iAnimSrcTower = iDest;
79             m_iAnimDestTower = iSrc;
80             m_iAnimDisk = pole2TopDisk;
81         }
```

10/31/2021 10:56:19 PM 4,173 bytes C,C++,C#,ObjC Source ANSI PC

```
69      m_tower[iDest].PopDisk();
70      //·When·top·disk·of·pole1·>·top·disk·of·pole2·
71      if·(pole1TopDisk·>·pole2TopDisk)·{
72          m_tower[iSrc].PushDisk(pole1TopDisk);
73          m_tower[iSrc].PushDisk(pole2TopDisk);


74          MoveDisk(d,·s,·pole2TopDisk);
75      }
76      //·When·top·disk·of·pole1·<·top·disk·of·pole2·
77      else·{


78          m_tower[iDest].PushDisk(pole2TopDisk);
```

26: 1 Default text

10/31/2021 10:55:04 PM 5,832 bytes C,C++,C#,ObjC Source ANSI PC

```
74      //·When·top·disk·of·pole1·>·top·disk·of·pole2·
75      if·(pole1TopDisk·>·pole2TopDisk)·{
76          m_tower[iDest].PopDisk();
77          //m_tower[iSrc].PushDisk(pole2TopDisk);
78          m_iAnimSrcTower·=·iDest;
79          m_iAnimDestTower·=·iSrc;
80          m_iAnimDisk·=·pole2TopDisk;
81      }
82      //·When·top·disk·of·pole1·<·top·disk·of·pole2·
83      else·{
84          m_tower[iSrc].PopDisk();
85          //m_tower[iDest].PushDisk(pole1TopDisk);
86          m_iAnimSrcTower·=·iSrc;
87          m_iAnimDestTower·=·iDest;
88          m_iAnimDisk·=·pole1TopDisk;
89      }
90  }
91  }
92
93  void·KTowerOfHanoi::MoveDisksBetweenTwoPoles(int·iSrc,·int·iDest)
94  {
95      char·buffer[80];
96      sprintf_s(buffer,·sizeof(buffer),·"state=%i,·%i(%i==>%i)"
97          ,·m_eDiskAnimState,·m_iAnimDisk,·m_iAnimSrcTower,·m_iAnimDestT
98      PutText(50,·1,·buffer);
99      if·(m_eDiskAnimState·!=·EDiskAnimState::DISKANIM_END)
100         PrintDisc(50,·2,·m_iAnimDisk);
101
102     char·s·=·m_stackLookup[iSrc];
103     char·d·=·m_stackLookup[iDest];
104     if·(m_eDiskAnimState·==·EDiskAnimState::DISKANIM_INIT)
105     {
106         MoveDisksBetweenTwoPoles_INIT(iSrc,·iDest);
107         m_eDiskAnimState·=·EDiskAnimState::DISKANIM_MOVE_TO_TOP;
108         m_diskAnimTimer·=·0.0;
109     }
```

26: 1 Default text

```
void KTowerOfHanoi::MoveDisksBetweenTwoPoles(int iSrc, int iDest)
{
    char buffer[80];
    sprintf_s(buffer, sizeof(buffer), "state=%i, %i(%i==>%i)"
        , m_eDiskAnimState, m_iAnimDisk, m_iAnimSrcTower, m_iAnimDestTower);
    PutText(50, 1, buffer);
    if (m_eDiskAnimState != EDiskAnimState::DISKANIM_END)
        PrintDisc(50, 2, m_iAnimDisk);

    char s = m_stackLookup[iSrc];
    char d = m_stackLookup[iDest];
    if (m_eDiskAnimState == EDiskAnimState::DISKANIM_INIT)
    {
        MoveDisksBetweenTwoPoles_INIT(iSrc, iDest);
        m_eDiskAnimState = EDiskAnimState::DISKANIM_MOVE_TO_TOP;
        m_diskAnimTimer = 0.0;
    }
    else if (m_eDiskAnimState == EDiskAnimState::DISKANIM_MOVE_TO_TOP)
    {
        if (m_diskAnimTimer >= 1.0) {
            m_eDiskAnimState = EDiskAnimState::DISKANIM_MOVE_TO_DEST;
            m_diskAnimTimer = 0.0;
        }
    }
    else if (m_eDiskAnimState == EDiskAnimState::DISKANIM_MOVE_TO_DEST)
    {
        if (m_diskAnimTimer >= 1.0) {
            m_eDiskAnimState = EDiskAnimState::DISKANIM_END;
            m_diskAnimTimer = 0.0;
            // action
            m_tower[m_iAnimDestTower].PushDisk(m_iAnimDisk);
            MoveDisk(s, d, m_iAnimDisk);
        }
    }
    else if (m_eDiskAnimState == EDiskAnimState::DISKANIM_END)
    {
        if (m_diskAnimTimer >= 1.0) {
            m_eDiskAnimState = EDiskAnimState::DISKANIM_INIT;
            m_diskAnimTimer = 0.0;
            m_iterationCounter += 1; // update iteration
        }
    }
}
```

```
86  void KTowerOfHanoi::Initialize(int numOfDisks)
87  {
88      for (KTower& s : m_tower) {
89          s.Clear();
90      }
91      m_log.clear();
92
93      m_moveState = EMoveState::ESTATE_DOING;
94      m_iStackSrc = 0;
95      m_iStackAux = 1;
```

20 FILTERED LINES

```
116 void KTowerOfHanoi::Iterate(double elapsedTime)
117 {
118     if (m_moveState == EMoveState::ESTATE_FINISHED) {
119         return;
120     }
121
122     m_oneStepTimer += elapsedTime;
123     if (m_oneStepTimer < 1.0) {
124         return;
125     }
126     m_oneStepTimer = 0.0;
127
128     m_iterationCounter += 1;
129     if (m_iterationCounter > m_totalNumOfMoves) {
130         m_moveState = EMoveState::ESTATE_FINISHED;
131         return;
132     }
133
134     if (m_iterationCounter % 3 == 1)
135         MoveDisksBetweenTwoPoles(m_iStackSrc, m_iStackDest);
136     else if (m_iterationCounter % 3 == 2)
137         MoveDisksBetweenTwoPoles( m_iStackSrc, m_iStackAux);
138     else if (m_iterationCounter % 3 == 0)
139         MoveDisksBetweenTwoPoles( m_iStackAux, m_iStackDest);
140 }
```

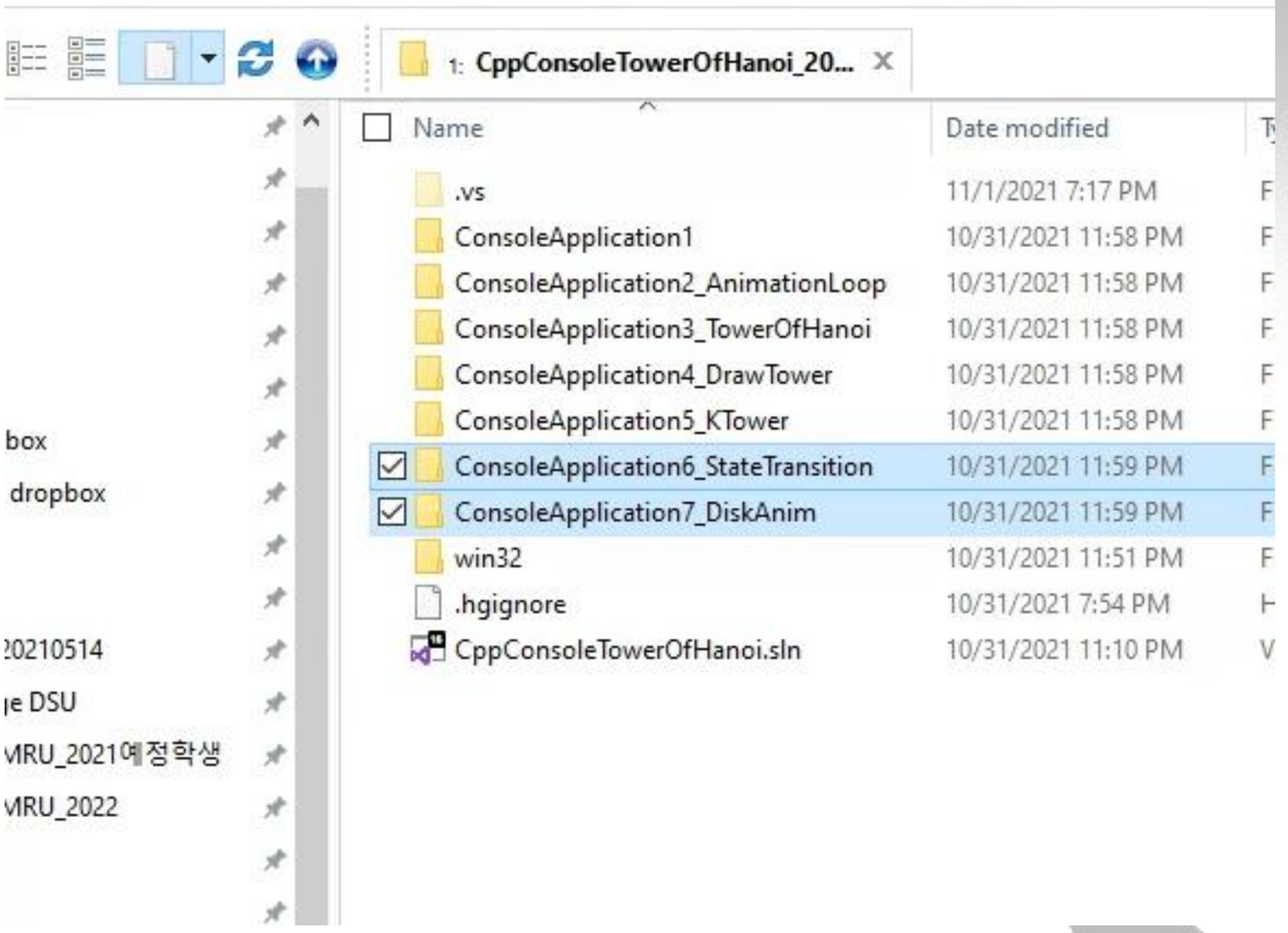
```
138 void KTowerOfHanoi::Initialize(int numOfDisks)
139 {
140     for (int i = 0; i < 3;++i) {
141         m_tower[i].Clear();
142     }
143     m_log.clear();
144
145     m_moveState = EMoveState::ESTATE_DOING;
146     m_iStackSrc = 0;
147     m_iStackAux = 1;
```

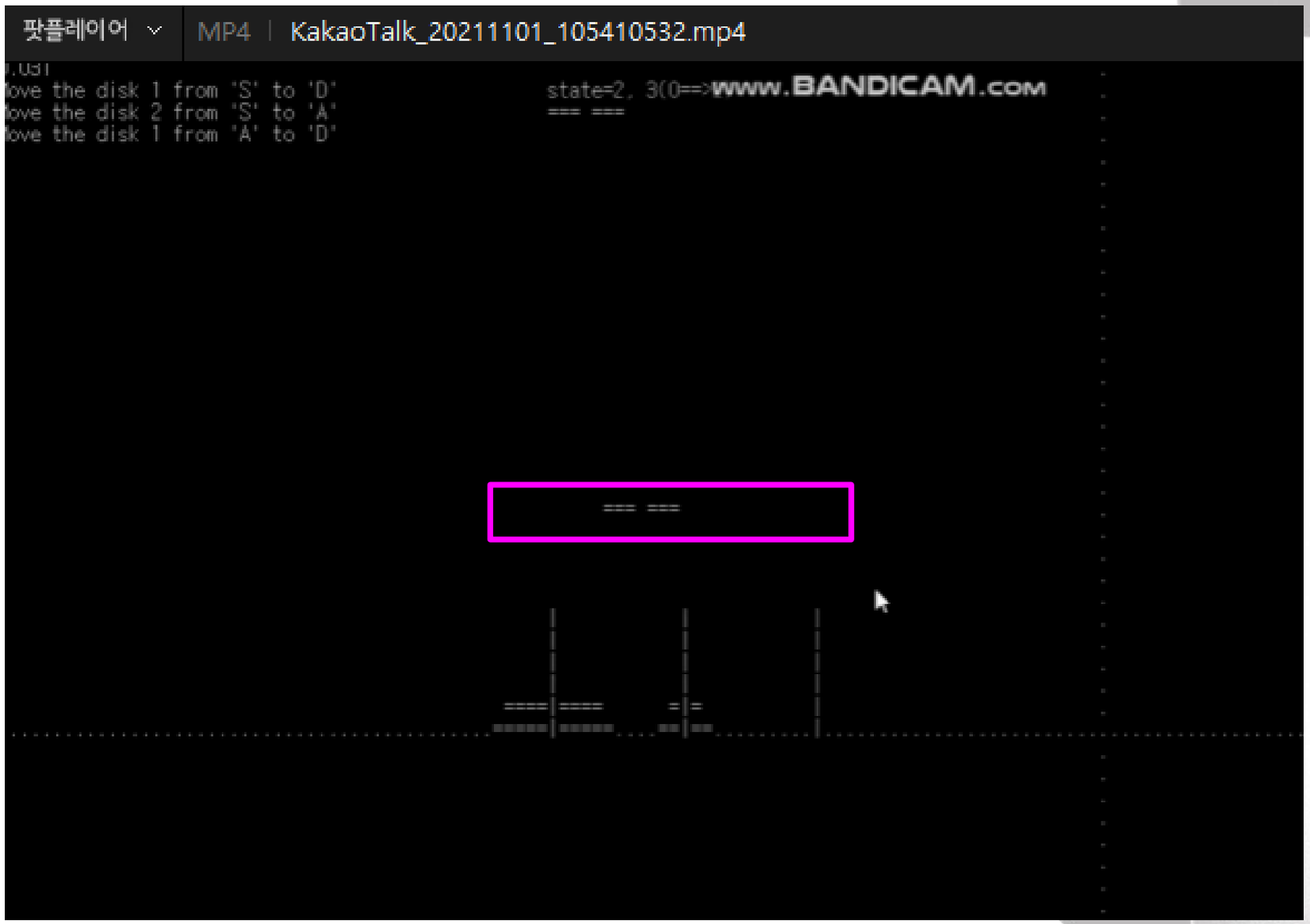
20 FILTERED LINES

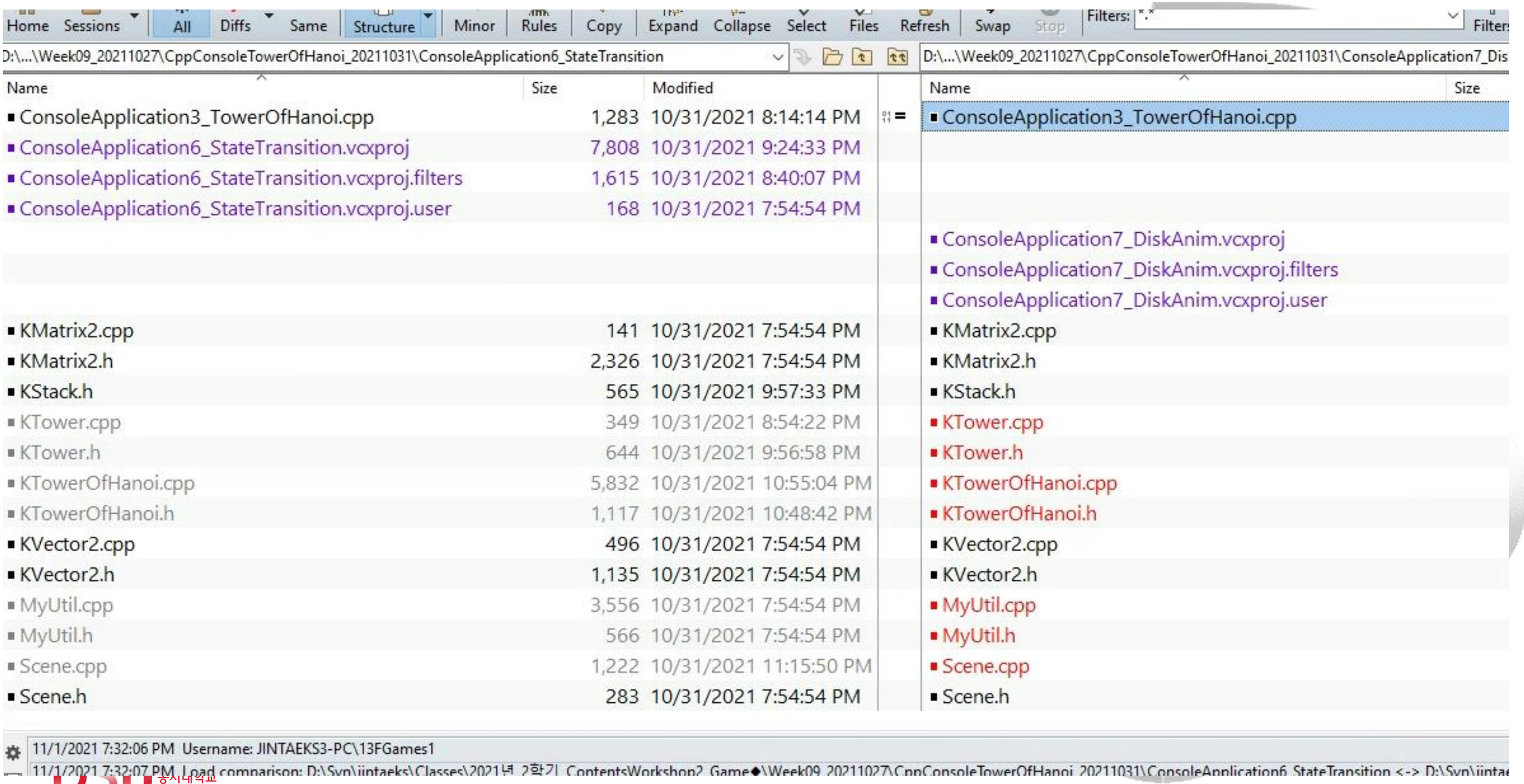
```
168 void KTowerOfHanoi::Iterate(double elapsedTime)
169 {
170     if (m_moveState == EMoveState::ESTATE_FINISHED) {
171         return;
172     }
173
174     //m_oneStepTimer += elapsedTime;
175     //if (m_oneStepTimer < 1.0) {
176     //  return;
177     //}
178     //m_oneStepTimer = 0.0;
179
180     //m_iterationCounter += 1;
181     if (m_iterationCounter >= m_totalNumOfMoves) {
182         m_moveState = EMoveState::ESTATE_FINISHED;
183         return;
184     }
185
186     m_diskAnimTimer += elapsedTime;
187     if (m_iterationCounter % 3 == 0)
188         MoveDisksBetweenTwoPoles(m_iStackSrc, m_iStackDest);
189     else if (m_iterationCounter % 3 == 1)
190         MoveDisksBetweenTwoPoles( m_iStackSrc, m_iStackAux);
191     else if (m_iterationCounter % 3 == 2)
192         MoveDisksBetweenTwoPoles( m_iStackAux, m_iStackDest);
193 }
```

```
    m_iterationCounter += 1;
    if (m_iterationCounter > m_totalNumOfMoves) {
        m_moveState = EMoveState::ESTATE_FINISHED;
        return;
    }

    if (m_iterationCounter % 3 == 1)
        MoveDisksBetweenTwoPoles(m_iStackSrc, m_iStackDest);
    else if (m_iterationCounter % 3 == 2)
        MoveDisksBetweenTwoPoles( m_iStackSrc, m_iStackAux);
    else if (m_iterationCounter % 3 == 0)
        MoveDisksBetweenTwoPoles( m_iStackAux, m_iStackDest);
}

void KTowerOfHanoi::Draw()
{
    int x = 0;
    11 FILTERED LINES
    y = starty;
    for (int i = 0; i < 3; ++i) {
        x += width;
        y = starty;
        for (int j = 0; j < m_numOfDisks + 1; ++j)
            PutText(x, 20 + y - j, "|");

        for (int n : m_tower[i]) { // custom iterator, 20211031
            _itoa_s(n, buffer, 10);
            PutText(x, y, buffer);
            PrintDisc(x - n, 20 + y, n); // jintaek on 20211031
            y -= 1;
        }
    }
    1 FILTERED LINES
```

```
    //m_iterationCounter += 1;
    if (m_iterationCounter >= m_totalNumOfMoves) {
        m_moveState = EMoveState::ESTATE_FINISHED;
        return;
    }

    m_diskAnimTimer += elapsedTime;
    if (m_iterationCounter % 3 == 0)
        MoveDisksBetweenTwoPoles(m_iStackSrc, m_iStackDest);
    else if (m_iterationCounter % 3 == 1)
        MoveDisksBetweenTwoPoles( m_iStackSrc, m_iStackAux);
    else if (m_iterationCounter % 3 == 2)
        MoveDisksBetweenTwoPoles( m_iStackAux, m_iStackDest);
}

void KTowerOfHanoi::Draw()
{
    int x = 0;
    11 FILTERED LINES
    y = starty;
    for (int i = 0; i < 3; ++i) {
        x += width;
        y = starty;
        for (int j = 0; j < m_numOfDisks + 1; ++j)
            PutText(x, 20 + y - j, "|");
        for (int j=0;j<m_tower[i].GetDiskCount();++j) {
            const int n = m_tower[i].GetDisk(j);
            _itoa_s(n, buffer, 10);
            PutText(x, y, buffer);
            PrintDisc(x - n, 20 + y, n); // jintaek on 20211031
            y -= 1;
        }
    }
    1 FILTERED LINES
```


1: CppConsoleTowerOfHanoi_20...
box
dropbox
20210514
e DSU
MRU_2021예정학생
MRU_2022
Name
Date modified
.vs
11/1/2021 7:17 PM
ConsoleApplication1
10/31/2021 11:58 PM
ConsoleApplication2_AnimationLoop
10/31/2021 11:58 PM
ConsoleApplication3_TowerOfHanoi
10/31/2021 11:58 PM
ConsoleApplication4_DrawTower
10/31/2021 11:58 PM
ConsoleApplication5_KTower
10/31/2021 11:58 PM
ConsoleApplication6_StateTransition
10/31/2021 11:59 PM
ConsoleApplication7_DiskAnim
10/31/2021 11:59 PM
win32
10/31/2021 11:51 PM
.hgignore
10/31/2021 7:54 PM
CppConsoleTowerOfHanoi.sln
10/31/2021 11:10 PM


팟플레이어
MP4
KakaoTalk_20211101_105410532.mp4
ove the disk 1 from 'S' to 'D'
ove the disk 2 from 'S' to 'A'
ove the disk 1 from 'A' to 'D'
state=2, 3(0==>
www.BANDICAM.com
=== ===
=== ===


Home Sessions All Diffs Same Structure Minor Rules Copy Expand Collapse Select Files Refresh Swap Stop Filters: *.* Filters
D:\...\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication6_StateTransition
D:\...\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication7_Dis
Name Size Modified
ConsoleApplication3_TowerOfHanoi.cpp 1,283 10/31/2021 8:14:14 PM
ConsoleApplication6_StateTransition.vcxproj 7,808 10/31/2021 9:24:33 PM
ConsoleApplication6_StateTransition.vcxproj.filters 1,615 10/31/2021 8:40:07 PM
ConsoleApplication6_StateTransition.vcxproj.user 168 10/31/2021 7:54:54 PM
KMatrix2.cpp 141 10/31/2021 7:54:54 PM
KMatrix2.h 2,326 10/31/2021 7:54:54 PM
KStack.h 565 10/31/2021 9:57:33 PM
KTower.cpp 349 10/31/2021 8:54:22 PM
KTower.h 644 10/31/2021 9:56:58 PM
KTowerOfHanoi.cpp 5,832 10/31/2021 10:55:04 PM
KTowerOfHanoi.h 1,117 10/31/2021 10:48:42 PM
KVector2.cpp 496 10/31/2021 7:54:54 PM
KVector2.h 1,135 10/31/2021 7:54:54 PM
MyUtil.cpp 3,556 10/31/2021 7:54:54 PM
MyUtil.h 566 10/31/2021 7:54:54 PM
Scene.cpp 1,222 10/31/2021 11:15:50 PM
Scene.h 283 10/31/2021 7:54:54 PM
Name Size
ConsoleApplication3_TowerOfHanoi.cpp
ConsoleApplication7_DiskAnim.vcxproj
ConsoleApplication7_DiskAnim.vcxproj.filters
ConsoleApplication7_DiskAnim.vcxproj.user
KMatrix2.cpp
KMatrix2.h
KStack.h
KTower.cpp
KTower.h
KTowerOfHanoi.cpp
KTowerOfHanoi.h
KVector2.cpp
KVector2.h
MyUtil.cpp
MyUtil.h
Scene.cpp
Scene.h
11/1/2021 7:32:06 PM Username: JINTAEKS3-PC\13FGames1
11/1/2021 7:32:07 PM Load comparison: D:\Svn\jintaeks\Classes\2021년 2학기 ContentsWorkshop2 Game◆\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication6_StateTransition <-> D:\Svn\jintae

D:\...\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication6_StateTransition\KTower.h

10/31/2021 9:56:58 PM 644 bytes C,C++,C#,ObjC Source ANSI PC

5 FILTERED LINES

```
{
public:
    typedef std::vector<int>::iterator iterator;
private:
    KStack      m_stack;
    KVector2    m_position;

public:
    KTower(int x = 0, int y = 0);
    void PushDisk(int d) { m_stack.push(d); }
    void PopDisk() { m_stack.pop(); }
    bool Empty() const { return m_stack.empty(); }
```

1 FILTERED LINES

```
    int GetDisk(int i) { return m_stack.at(i); }

    void Clear() { m_stack.clear(); }
    //KStack& GetStack() { return m_stack; }
    int GetDiskCount() const { return (int)m_stack.size(); }
    bool GetTopDiskPosition( KVector2& vOut );

};
```

D:\...\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication7_DiskAnim\KT

10/31/2021 11:35:50 PM 771 bytes C,C++,C#,ObjC Source ANSI PC

5 FILTERED LINES

```
{
public:
    typedef std::vector<int>::iterator iterator;
private:
    KStack      m_stack;
    KVector2    m_position;
    int         m_numOfDisks;

public:
    KTower(int x = 0, int y = 0);
    void PushDisk(int d) { m_stack.push(d); }
    void PopDisk() { m_stack.pop(); }
    bool Empty() const { return m_stack.empty(); }
```

1 FILTERED LINES

```
    int GetDisk(int i) { return m_stack.at(i); }

    void Clear() { m_stack.clear(); }
    //KStack& GetStack() { return m_stack; }
    int GetDiskCount() const { return (int)m_stack.si
    bool GetTopDiskPosition( KVector2& vOut );
    KVector2 GetPosition() { return m_position; }
    void Init(int x, int y, int numOfDisk);
    void Draw();
};
```

)/31/2021 8:54:22 PM 349 bytes C,C++,C#,ObjC Source ANSI PC

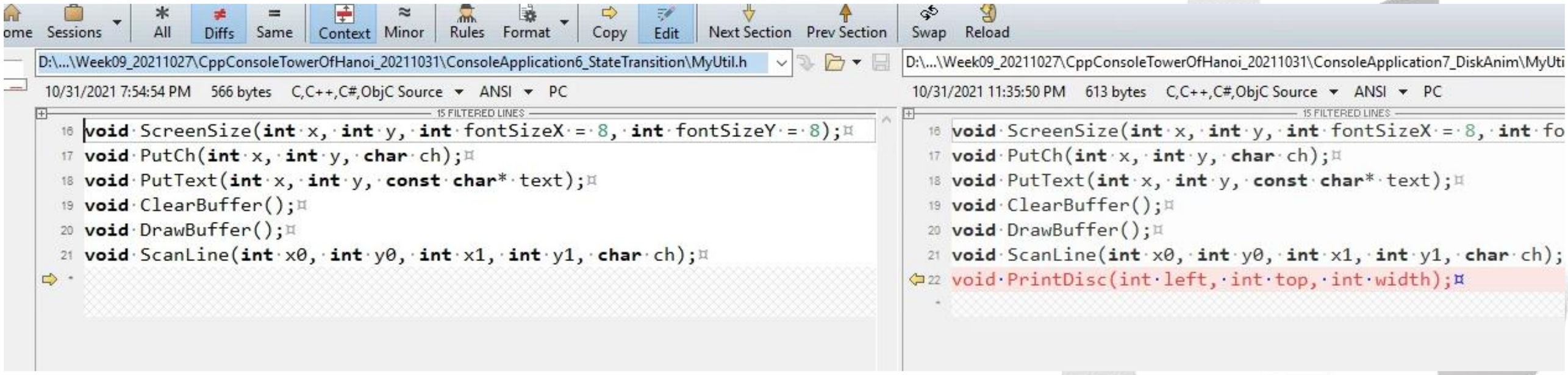
```cpp
#include "KTower.h"

KTower::KTower(int x, int y)
{
    m_stack.clear();
    m_position = KVector2::zero;

}

bool KTower::GetTopDiskPosition(KVector2& vOut )
{
    if (Empty())
        return false;

    int y = (int)m_position.y - GetDiskCount();
    int x = (int)m_position.x - TopDisk();
    vOut = KVector2(x, y); //set [out] parameter
    return true;
}
```

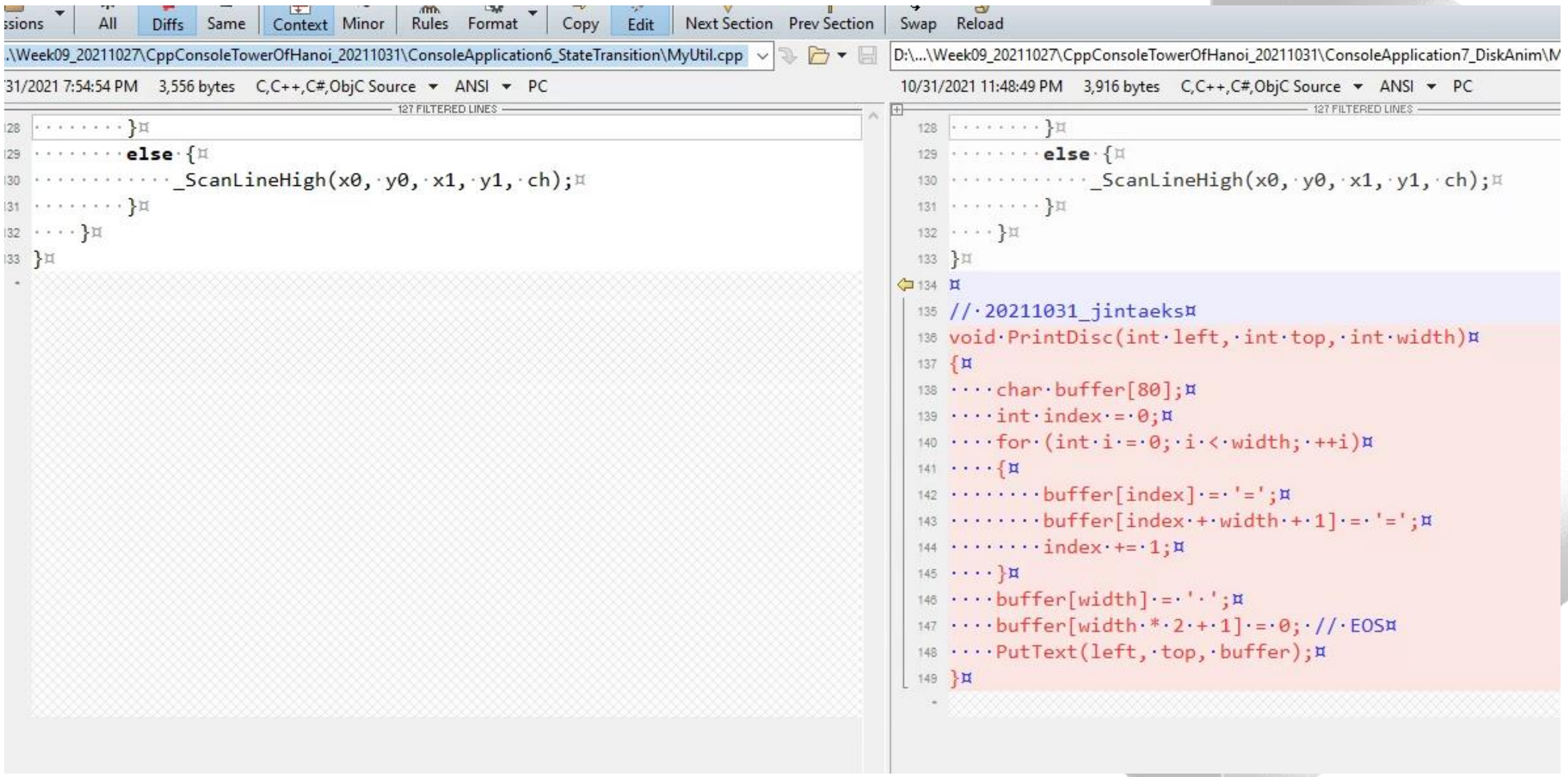
10/31/2021 11:48:15 PM 906 bytes C,C++,C#,ObjC Source ANSI PC

```cpp
#include "KTower.h"
#include "MyUtil.h"

KTower::KTower(int x, int y)
{
    m_stack.clear();
    m_position = KVector2::zero;
    m_numOfDisks = 0;
}

bool KTower::GetTopDiskPosition(KVector2& vOut )
{
    if (Empty())
        return false;

    int y = (int)m_position.y - GetDiskCount();
    int x = (int)m_position.x - TopDisk();
    vOut = KVector2(x, y); //set [out] parameter
    return true;
}

void KTower::Init(int x, int y, int numOfDisk)
{
    m_position = KVector2(x, y);
    m_numOfDisks = numOfDisk;
}

void KTower::Draw()
{
```

```
    return false;

    int y = (int)m_position.y - GetDiskCount();
    int x = (int)m_position.x - TopDisk();
    vOut = KVector2(x, y); // set [out] parameter
    return true;
}
```

```
    return false;

    int y = (int)m_position.y - GetDiskCount();
    int x = (int)m_position.x - TopDisk();
    vOut = KVector2(x, y); // set [out] parameter
    return true;
}

void KTower::Init(int x, int y, int numOfDisk)
{
    m_position = KVector2(x, y);
    m_numOfDisks = numOfDisk;
}

void KTower::Draw()
{
    int x = (int)m_position.x;
    int y = (int)m_position.y;

    char buffer[80];
    for (int j = 0; j < GetDiskCount(); ++j) {
        const int n = GetDisk(j);
        _itoa_s(n, buffer, 10);
        //PutText(x, y, buffer);
        PrintDisc(x - n, y, n); // jintaek on 20211031
        y -= 1;
    }
    y = (int)m_position.y;
    for (int j = 0; j < m_numOfDisks + 1; ++j)
        PutText(x, y - j, "|");
}
```


Sessions
All
Diffs
Same
Context
Minor
Rules
Format
Copy
Edit
Next Section
Prev Section
Swap
Reload
D:\...\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication6_StateTransition\MyUtil.h
10/31/2021 7:54:54 PM 566 bytes C,C++,C#,ObjC Source ANSI PC
15 FILTERED LINES
16 void ScreenSize(int x, int y, int fontSizeX = 8, int fontSizeY = 8);
17 void PutCh(int x, int y, char ch);
18 void PutText(int x, int y, const char* text);
19 void ClearBuffer();
20 void DrawBuffer();
21 void ScanLine(int x0, int y0, int x1, int y1, char ch);
D:\...\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication7_DiskAnim\MyUti
10/31/2021 11:35:50 PM 613 bytes C,C++,C#,ObjC Source ANSI PC
15 FILTERED LINES
16 void ScreenSize(int x, int y, int fontSizeX = 8, int fo
17 void PutCh(int x, int y, char ch);
18 void PutText(int x, int y, const char* text);
19 void ClearBuffer();
20 void DrawBuffer();
21 void ScanLine(int x0, int y0, int x1, int y1, char ch);
22 void PrintDisc(int left, int top, int width);


All
Diffs
Same
Context
Minor
Rules
Format
Copy
Edit
Next Section
Prev Section
Swap
Reload
.\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication6_StateTransition\MyUtil.cpp
31/2021 7:54:54 PM 3,556 bytes C,C++,C#,ObjC Source ANSI PC
127 FILTERED LINES
}
else {
_ScanLineHigh(x0, y0, x1, y1, ch);
}
}
}
D:\...\Week09_20211027\CppConsoleTowerOfHanoi_20211031\ConsoleApplication7_DiskAnim\M
10/31/2021 11:48:49 PM 3,916 bytes C,C++,C#,ObjC Source ANSI PC
127 FILTERED LINES
}
else {
_ScanLineHigh(x0, y0, x1, y1, ch);
}
}
}
//20211031_jintaeks
void PrintDisc(int left, int top, int width)
{
char buffer[80];
int index = 0;
for (int i = 0; i < width; ++i)
{
buffer[index] = '=';
buffer[index + width + 1] = '=';
index += 1;
}
buffer[width] = '.';
buffer[width * 2 + 1] = 0; //EOS
PutText(left, top, buffer);
}

```
51 void KTowerOfHanoi::MoveDisksBetweenTwoPoles_INIT(int iSrc, int iDest)
52 {
53     // When pole 1 is empty
54     if (m_tower[iSrc].Empty()) {
55         int pole2TopDisk = m_tower[iDest].TopDisk();
56         m_tower[iDest].PopDisk();
57         //m_tower[iSrc].PushDisk(pole2TopDisk);
58         m_iAnimSrcTower = iDest;
59         m_iAnimDestTower = iSrc;
60         m_iAnimDisk = pole2TopDisk;
61     }
62     // When pole2 pole is empty
63     else if (m_tower[iDest].Empty()) {
64         int pole1TopDisk = m_tower[iSrc].TopDisk();
65         m_tower[iSrc].PopDisk();
66         //m_tower[iDest].PushDisk(pole1TopDisk);
67         m_iAnimSrcTower = iSrc;
68         m_iAnimDestTower = iDest;
69         m_iAnimDisk = pole1TopDisk;
70     }
71     else {
72         int pole1TopDisk = m_tower[iSrc].TopDisk();
73         int pole2TopDisk = m_tower[iDest].TopDisk();
74         // When top disk of pole1 > top disk of pole2
75         if (pole1TopDisk > pole2TopDisk) {
76             m_tower[iDest].PopDisk();
77             //m_tower[iSrc].PushDisk(pole2TopDisk);
78             m_iAnimSrcTower = iDest;
79             m_iAnimDestTower = iSrc;
```

```
35 void KTowerOfHanoi::MoveDisksBetweenTwoPoles_INIT(int iSrc, int i
36 {
37     // When pole 1 is empty
38     if (m_tower[iSrc].Empty()) {
39         int pole2TopDisk = m_tower[iDest].TopDisk();
40         m_tower[iDest].GetTopDiskPosition(m_vAnimBegin);
41         m_tower[iDest].PopDisk();
42         //m_tower[iSrc].PushDisk(pole2TopDisk);
43         m_iAnimSrcTower = iDest;
44         m_iAnimDestTower = iSrc;
45         m_iAnimDisk = pole2TopDisk;
46     }
47     // When pole2 pole is empty
48     else if (m_tower[iDest].Empty()) {
49         int pole1TopDisk = m_tower[iSrc].TopDisk();
50         m_tower[iSrc].GetTopDiskPosition(m_vAnimBegin);
51         m_tower[iSrc].PopDisk();
52         //m_tower[iDest].PushDisk(pole1TopDisk);
53         m_iAnimSrcTower = iSrc;
54         m_iAnimDestTower = iDest;
55         m_iAnimDisk = pole1TopDisk;
56     }
57     else {
58         int pole1TopDisk = m_tower[iSrc].TopDisk();
59         int pole2TopDisk = m_tower[iDest].TopDisk();
60         // When top disk of pole1 > top disk of pole2
61         if (pole1TopDisk > pole2TopDisk) {
62             m_tower[iDest].GetTopDiskPosition(m_vAnimBegin);
63             m_tower[iDest].PopDisk();
64             //m_tower[iSrc].PushDisk(pole2TopDisk);
65             m_iAnimSrcTower = iDest;
66             m_iAnimDestTower = iSrc;
```

```
73      int pole2TopDisk = m_tower[iDest].TopDisk();
74      //When top disk of pole1 > top disk of pole2
75      if (pole1TopDisk > pole2TopDisk) {
76          m_tower[iDest].PopDisk();
77          //m_tower[iSrc].PushDisk(pole2TopDisk);
78          m_iAnimSrcTower = iDest;
79          m_iAnimDestTower = iSrc;
80          m_iAnimDisk = pole2TopDisk;
81      }
82      //When top disk of pole1 < top disk of pole2
83      else {
84          m_tower[iSrc].PopDisk();
85          //m_tower[iDest].PushDisk(pole1TopDisk);
86          m_iAnimSrcTower = iSrc;
87          m_iAnimDestTower = iDest;
88          m_iAnimDisk = pole1TopDisk;
89      }
90  }
91 }
92
93 void KTowerOfHanoi::MoveDisksBetweenTwoPoles(int iSrc, int iDest)
94 {
95      char buffer[80];
96      sprintf_s(buffer, sizeof(buffer), "state=%i, %i(%i==>%i)"
```

12 FILTERED LINES

```
59      int pole2TopDisk = m_tower[iDest].TopDisk();
60      //When top disk of pole1 > top disk of pole2
61      if (pole1TopDisk > pole2TopDisk) {
62          m_tower[iDest].GetTopDiskPosition(m_vAnimBegin);
63          m_tower[iDest].PopDisk();
64          //m_tower[iSrc].PushDisk(pole2TopDisk);
65          m_iAnimSrcTower = iDest;
66          m_iAnimDestTower = iSrc;
67          m_iAnimDisk = pole2TopDisk;
68      }
69      //When top disk of pole1 < top disk of pole2
70      else {
71          m_tower[iSrc].GetTopDiskPosition(m_vAnimBegin);
72          m_tower[iSrc].PopDisk();
73          //m_tower[iDest].PushDisk(pole1TopDisk);
74          m_iAnimSrcTower = iSrc;
75          m_iAnimDestTower = iDest;
76          m_iAnimDisk = pole1TopDisk;
77      }
78  }
79
80      //set animation info
81      //m_tower[m_iAnimSrcTower].GetTopDiskPosition(m_vAnimBegin);
82      m_vAnimEnd = m_tower[m_iAnimSrcTower].GetPosition();
83      m_vAnimEnd.x = m_vAnimBegin.x;
84      m_vAnimEnd.y -= m_numOfDisks*2;
85 }
86
87 void KTowerOfHanoi::MoveDisksBetweenTwoPoles(int iSrc, int iDest)
88 {
89      char buffer[80];
90      sprintf_s(buffer, sizeof(buffer), "state=%i, %i(%i==>%i)"
```

12 FILTERED LINES

```
93  void KTowerOfHanoi::MoveDisksBetweenTwoPoles(int iSrc, int iDest)
94  {
95      char buffer[80];
96      sprintf_s(buffer, sizeof(buffer), "state=%i, %i(%i==>%i)"
        12 FILTERED LINES
109     }
110     else if (m_eDiskAnimState == EDiskAnimState::DISKANIM_MOVE_TO_TOP)
111     {
112         if (m_diskAnimTimer >= 1.0) {
113             m_eDiskAnimState = EDiskAnimState::DISKANIM_MOVE_TO_DEST;
114             m_diskAnimTimer = 0.0;
115         }
116     }
117     else if (m_eDiskAnimState == EDiskAnimState::DISKANIM_MOVE_TO_DEST)
118     {
119         if (m_diskAnimTimer >= 1.0) {
120             m_eDiskAnimState = EDiskAnimState::DISKANIM_END;
        38 FILTERED LINES
159     m_totalNumOfMoves = (int)pow(2, numOfDisks) - 1;
160
161     KTower& src = m_tower[m_iStackSrc];
162     //Larger disks will be pushed first
```

```
87  void KTowerOfHanoi::MoveDisksBetweenTwoPoles(int iSrc, int iDest)
88  {
89      char buffer[80];
90      sprintf_s(buffer, sizeof(buffer), "state=%i, %i(%i==>%i)"
        12 FILTERED LINES
103     }
104     else if (m_eDiskAnimState == EDiskAnimState::DISKANIM_MOVE_TO_T
105     {
106         if (m_diskAnimTimer >= 1.0) {
107             m_eDiskAnimState = EDiskAnimState::DISKANIM_MOVE_TO_DES
108             m_diskAnimTimer = 0.0;
109             m_vAnimBegin = m_vAnimEnd;
110             m_vAnimEnd = m_tower[m_iAnimDestTower].GetPosition();
111             m_vAnimEnd.x -= m_iAnimDisk;
112             m_vAnimEnd.y = m_vAnimBegin.y;
113         }
114     }
115     else if (m_eDiskAnimState == EDiskAnimState::DISKANIM_MOVE_TO_D
116     {
117         if (m_diskAnimTimer >= 1.0) {
118             m_eDiskAnimState = EDiskAnimState::DISKANIM_END;
        38 FILTERED LINES
157     m_totalNumOfMoves = (int)pow(2, numOfDisks) - 1;
158
159     KTower& src = m_tower[m_iStackSrc];
160     //Larger disks will be pushed first
```

```
void KTowerOfHanoi::Initialize(int numOfDisks)
{
    for (int i = 0; i < 3;++i) {
        m_tower[i].Clear();
    }
    m_log.clear();

    m_moveState = EMoveState::ESTATE_DOING;
    m_iStackSrc = 0;
    m_iStackAux = 1;
    m_iStackDest = 2;
    m_iterationCounter = 0;

    //If number of disks is even, then interchange
    //destination pole and auxiliary pole
    if (numOfDisks % 2 == 0) {
        char temp = m_iStackDest;
        m_iStackDest = m_iStackAux;
        m_iStackAux = temp;
    }
    m_numOfDisks = numOfDisks;
    m_totalNumOfMoves = (int)pow(2, numOfDisks) - 1;

    KTower& src = m_tower[m_iStackSrc];
    //Larger disks will be pushed first
    for (int i = m_numOfDisks; i >= 1; i--)
        src.PushDisk(i);
```

```
void KTowerOfHanoi::Initialize(int numOfDisks)
{
    for (int i = 0; i < 3;++i) {
        m_tower[i].Clear();
    }
    m_log.clear();

    m_moveState = EMoveState::ESTATE_DOING;
    m_iStackSrc = 0;
    m_iStackAux = 1;
    m_iStackDest = 2;
    m_iterationCounter = 0;

    //If number of disks is even, then interchange
    //destination pole and auxiliary pole
    if (numOfDisks % 2 == 0) {
        char temp = m_iStackDest;
        m_iStackDest = m_iStackAux;
        m_iStackAux = temp;
    }
    m_numOfDisks = numOfDisks;
    m_totalNumOfMoves = (int)pow(2, numOfDisks) - 1;

    KTower& src = m_tower[m_iStackSrc];
    //Larger disks will be pushed first
    for (int i = m_numOfDisks; i >= 1; i--)
        src.PushDisk(i);

    int startx = 50;
    int starty = 30;
    const int width = m_numOfDisks * 2 + 2;
    int x = startx;
    int y = starty;
    for (int i = 0; i < 3; ++i) {
        m_tower[i].Init(x, y, m_numOfDisks);
        x += width;
    }
```

D:\...\CppConsoleTowerOfHanoi_20211031\ConsoleApplication6_StateTransition\KTowerOfHanoi.cpp

10/31/2021 10:55:04 PM 5,832 bytes C,C++,C#,ObjC Source ANSI PC

```
168 void KTowerOfHanoi::Iterate(double elapsedTime)
169 {
170     if (m_moveState == EMoveState::ESTATE_FINISHED) {
    27 FILTERED LINES
198     int y = 1;
199     for (std::string& str : m_log) {
200         PutText(x, y, str.c_str());
201         y += 1;
202     }
203
204     char buffer[80];
205     int startx = 50;
206     int starty = 10;
207     const int width = m_numOfDisks * 2 + 2;
208     x = startx;
209     y = starty;
210     for (int i = 0; i < 3; ++i) {
211         x += width;
212         y = starty;
213         for (int j = 0; j < m_numOfDisks + 1; ++j)
214             PutText(x, 20 + y--j, "|");
215         for (int j=0;j<m_tower[i].GetDiskCount();++j) {
216             const int n = m_tower[i].GetDisk(j);
217             _itoa_s(n, buffer, 10);
218             PutText(x, y, buffer);
219             PrintDisc(x--n, 20 + y, n); // jintaek on 20211031
220             y -= 1;
221         }
222     }
223 }
```

D:\...\CppConsoleTowerOfHanoi_20211031\ConsoleApplication7_DiskAnim\KTowerOfHanoi.cpp

10/31/2021 11:43:41 PM 6,253 bytes C,C++,C#,ObjC Source ANSI PC

```
176 void KTowerOfHanoi::Iterate(double elapsedTime)
177 {
178     if (m_moveState == EMoveState::ESTATE_FINISHED) {
    27 FILTERED LINES
206     int y = 1;
207     for (std::string& str : m_log) {
208         PutText(x, y, str.c_str());
209         y += 1;
210     }
211
212     for (int i = 0; i < 3; ++i) {
213         m_tower[i].Draw();
214     }
215
216     if (m_eDiskAnimState == EDiskAnimState::DISKANIM_MOVE_TO_TOP)
217     {
218         KVector2 pos = KVector2::Lerp(m_vAnimBegin, m_vAnimEnd, m_disk
219         PrintDisc(pos.x, pos.y, m_iAnimDisk);
220     }
221     else if (m_eDiskAnimState == EDiskAnimState::DISKANIM_MOVE_TO_DEST
222     {
223         KVector2 pos = KVector2::Lerp(m_vAnimBegin, m_vAnimEnd, m_disk
224         PrintDisc(pos.x, pos.y, m_iAnimDisk);
225     }
226 }
```

# THANKS!

## Any questions?

MY **BRIGHT** FUTURE

DSU Dongseo University 동서대학교