1. 問題

　新幹線の座席予約システムを構築する。以下の条件を満たさなければならない。

・座席は実在の車両に基づくものとする。

・予約結果がコンピュータの電源を切っても残っていること。

・同時に複数のプログラムを実行し、予約できること

・空席確認から予約確定までに情報の矛盾が発生しないこと。

・席は、座席番号を直接指定して予約すること。

それぞれの条件に対し、以下の通りに対応した。

・今回の課題では東海道・山陽新幹線N700系の1号車をモデルとした。

・予約された座席1つにつき1つのテキストファイルを作成することで予約結果を残す。

・実行中の複数のプログラムが、いつでも予約結果を保存したテキストファイルを読み込むことができるように工夫する。

・ユーザーからの入力が必要になる場合に予約状況を更新し、できる限り矛盾をなくす。

・ユーザーが座席の予約状況を確認できるGUIを設計し、列と行がわかるような形式にして入力するようにする。

1. プログラム構成図

　　プログラムファイルは図1に示すものを作成した。

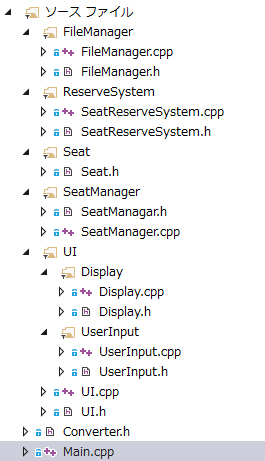


図1：ソリューションエクスプローラー

上から順に

ファイル入出力を管理するCFileManagerクラス、

システム全体をまとめるCSeatReserveSystem、

座席の状態を表すCSeatクラス、

座席全体を管理するCSeatManagerクラス、

ユーザーに予約状況を見せるCDisplayクラス、

ユーザーの入力を受け取るCUserInputクラス、

ユーザーインターフェイスをつかさどるCUserInterfaceクラス、

入力されたstringから座席番号として登録してある列挙定数を返すCConverterクラス、

プログラムを実行するmain関数

である。

main関数からSeatReserveSystemクラスの関数を実行する形となっている。

1. ソースプログラム

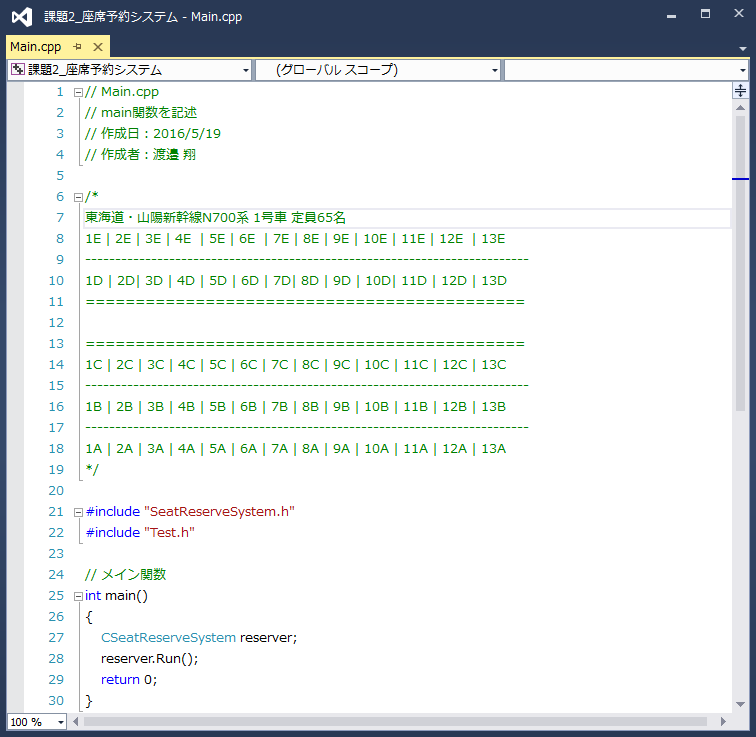


図2：main.cpp

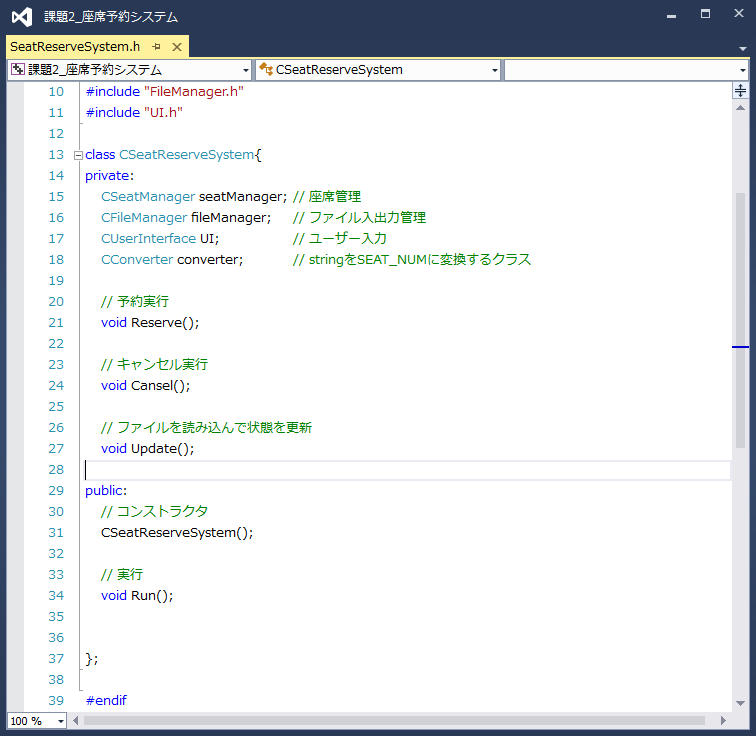


図3：SeatReserveSystem.h

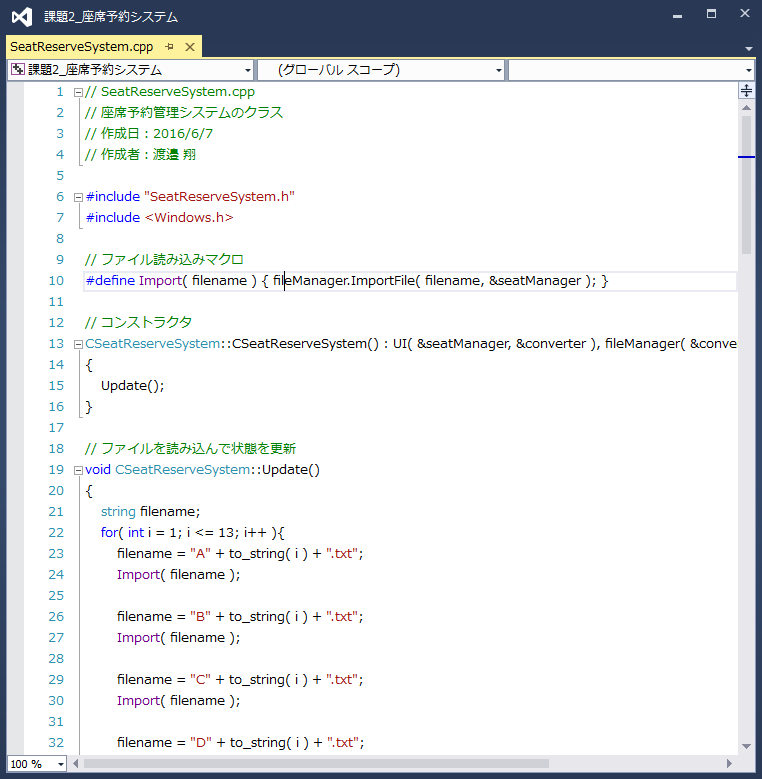


図4：SeatReserveSystem.cpp　1行目から32行目

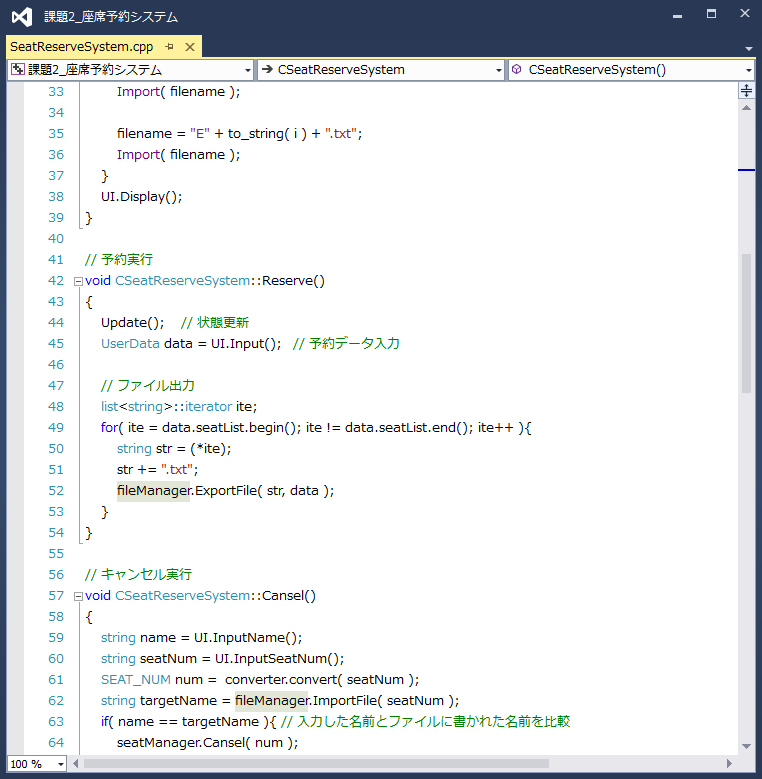


図5：SeatReserveSystem.cpp　33行目から64行目

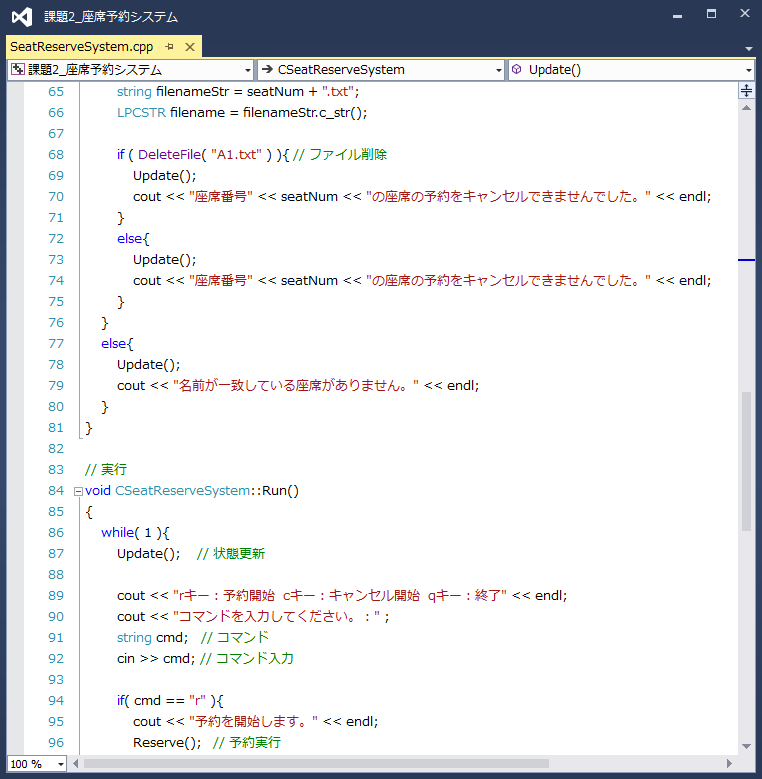


図6：SeatReserveSystem.cpp　65行目から96行目

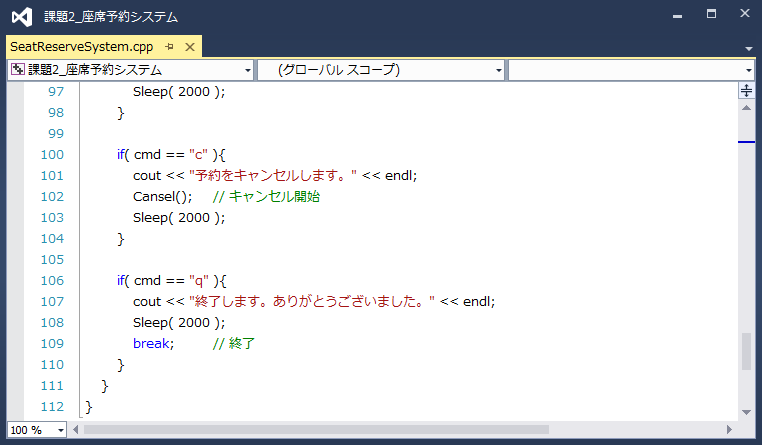
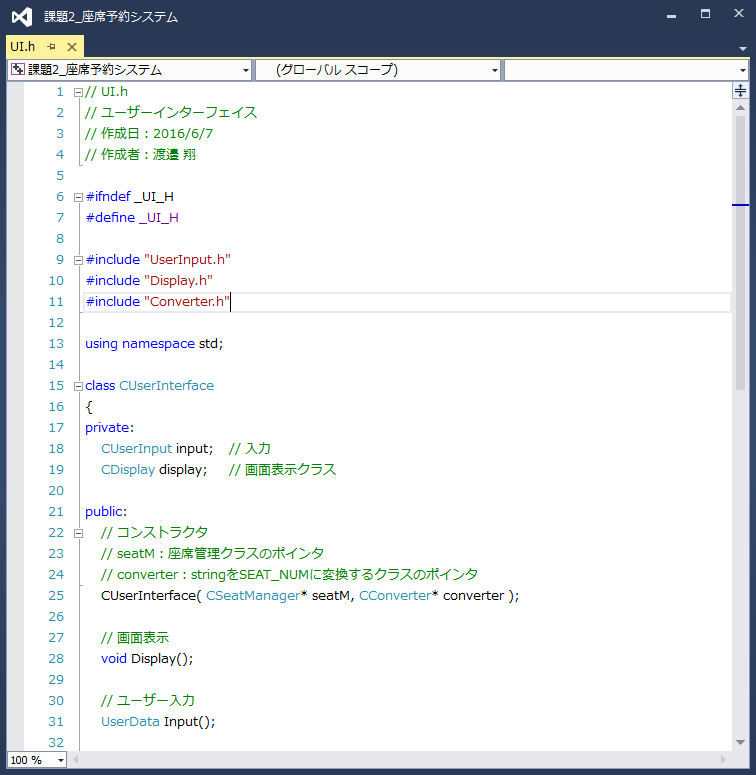
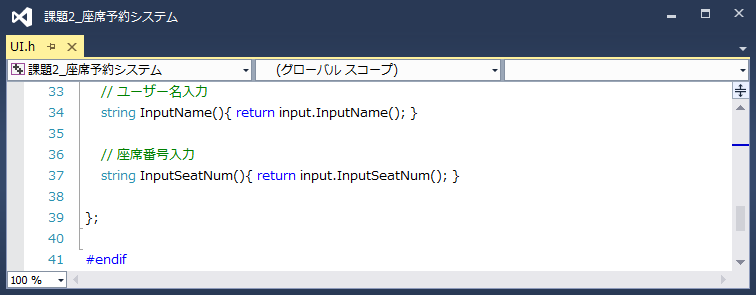


図7：SeatReserveSystem.cpp　97行目から最後まで



図8、9：UI.h

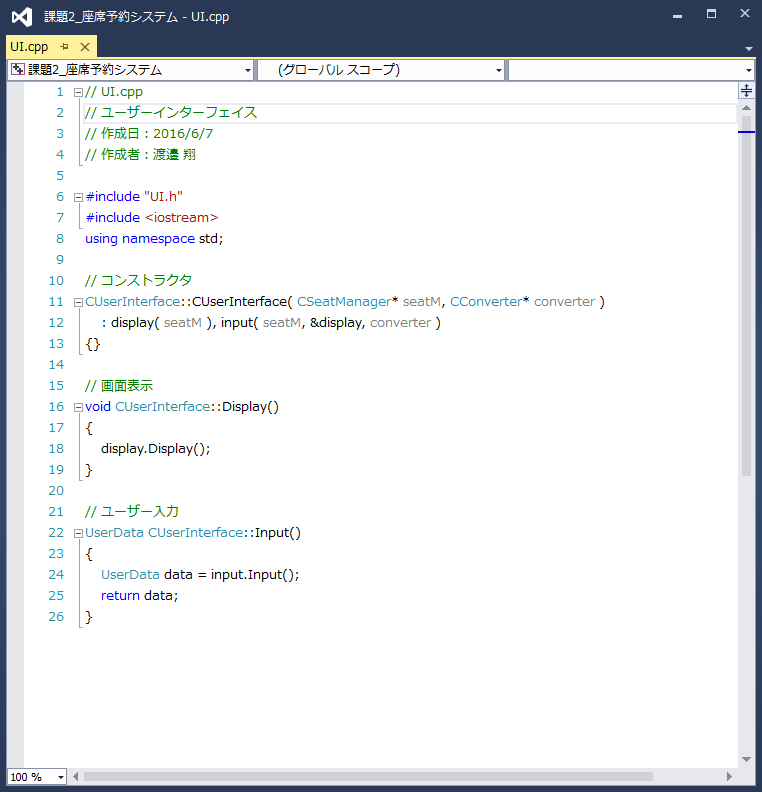


図10：UI.cpp

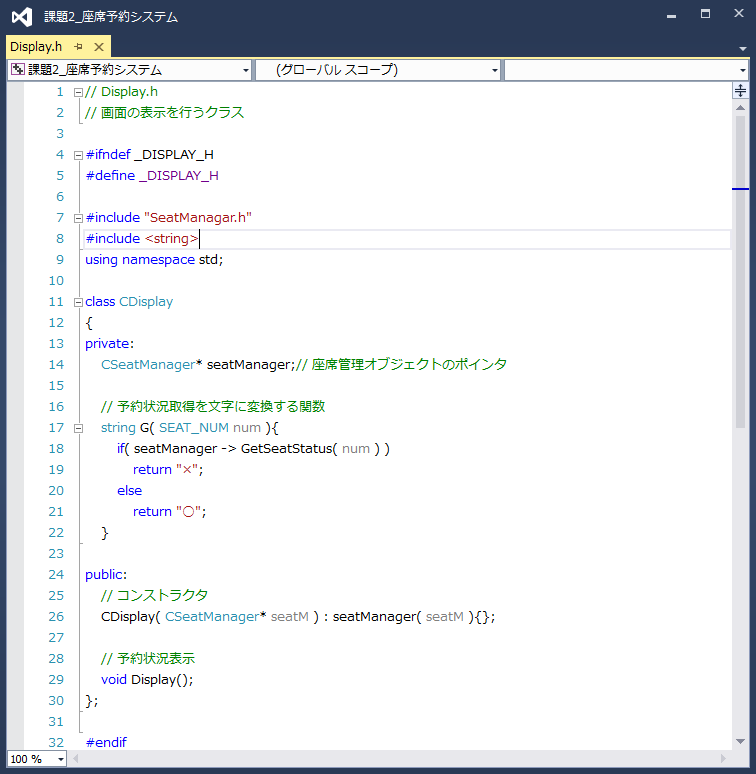


図11：Display.h

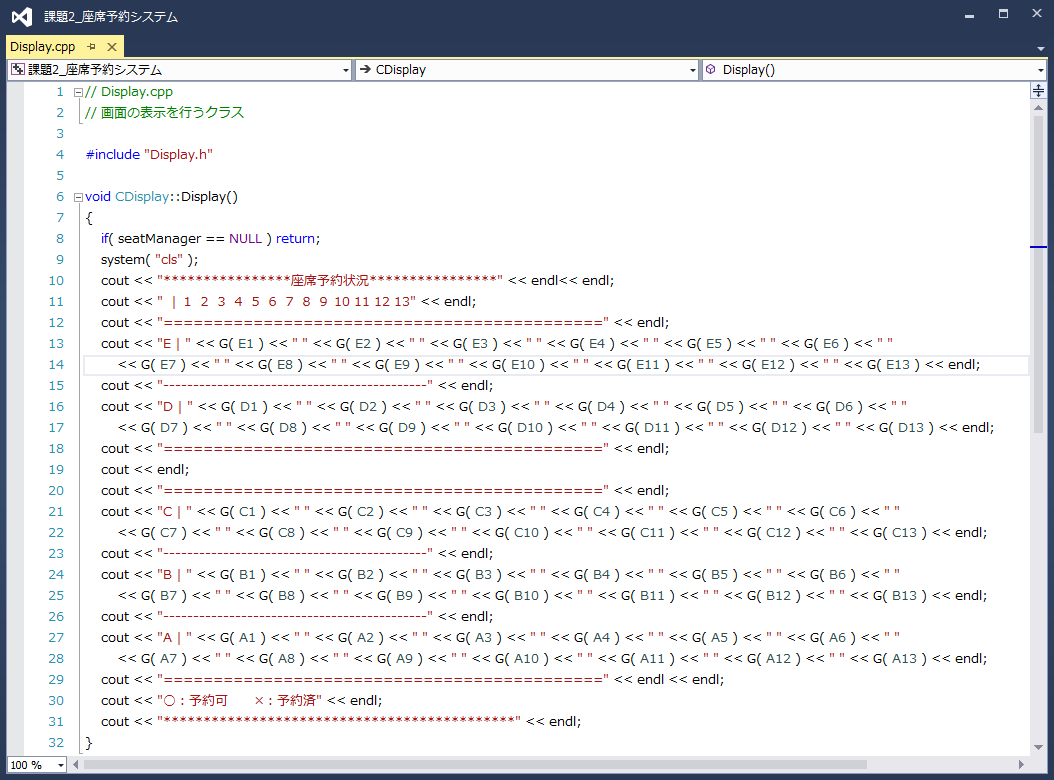


図12：Display.cpp

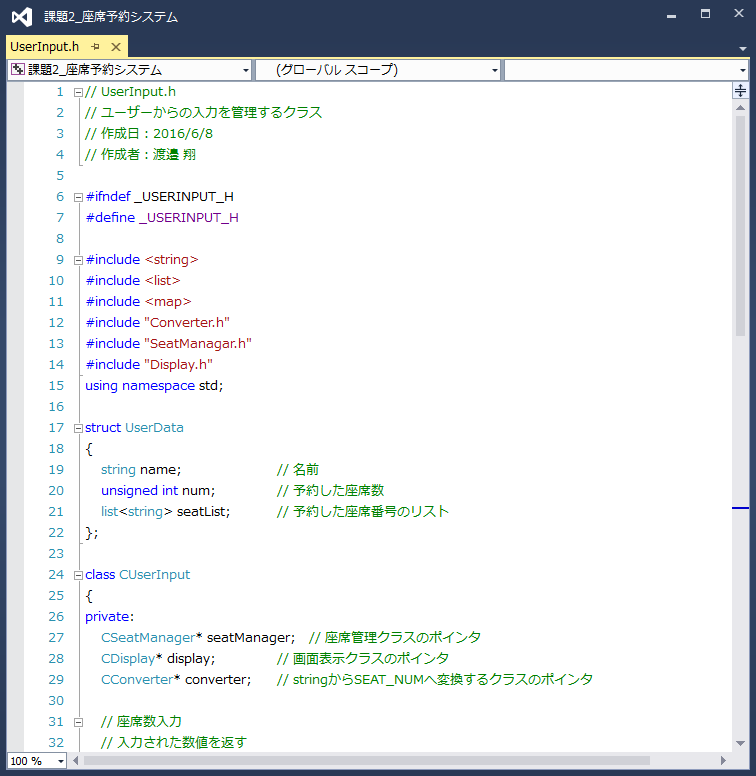


図13：UserInput.h　1行目から32行目まで

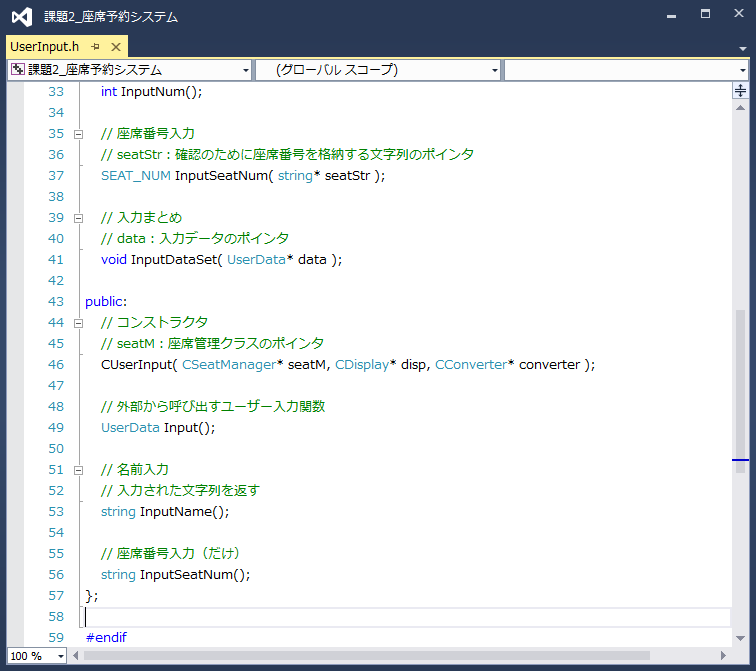


図14：UserInput.h　33行目から最後まで

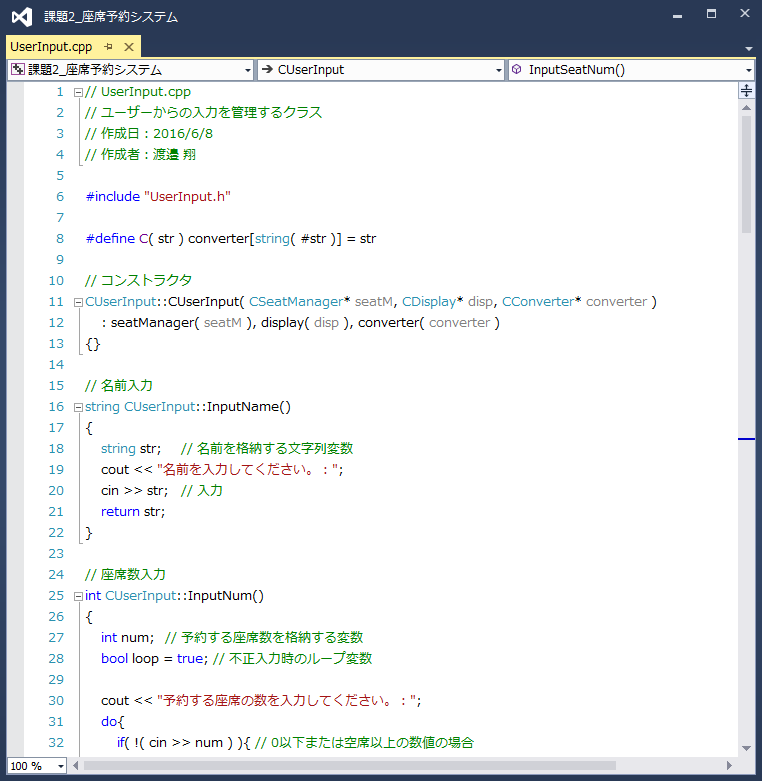


図15：UserInput.cpp　1行目から32行目まで

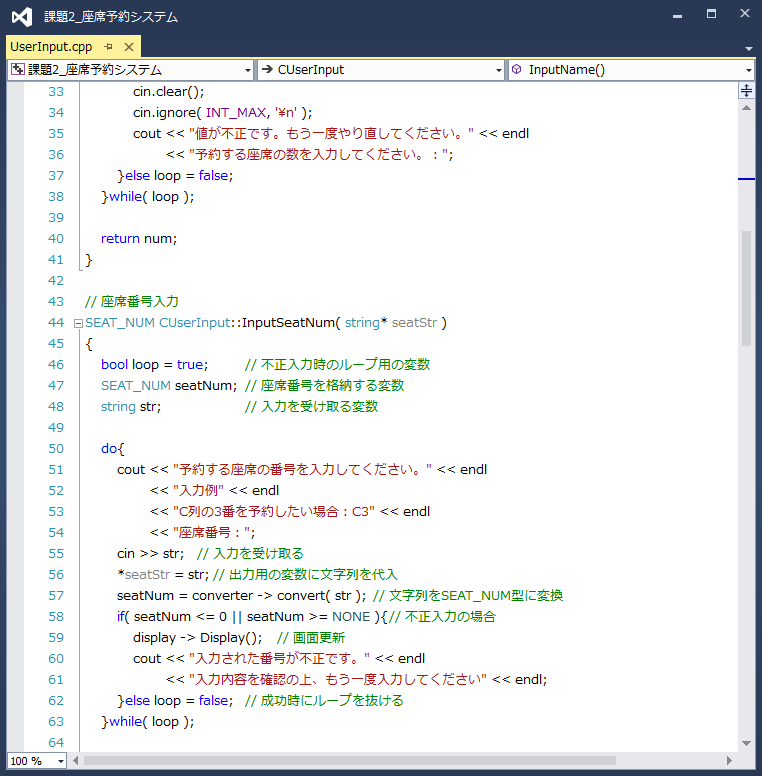


図16：UserInput.cpp　33行目から64行目まで

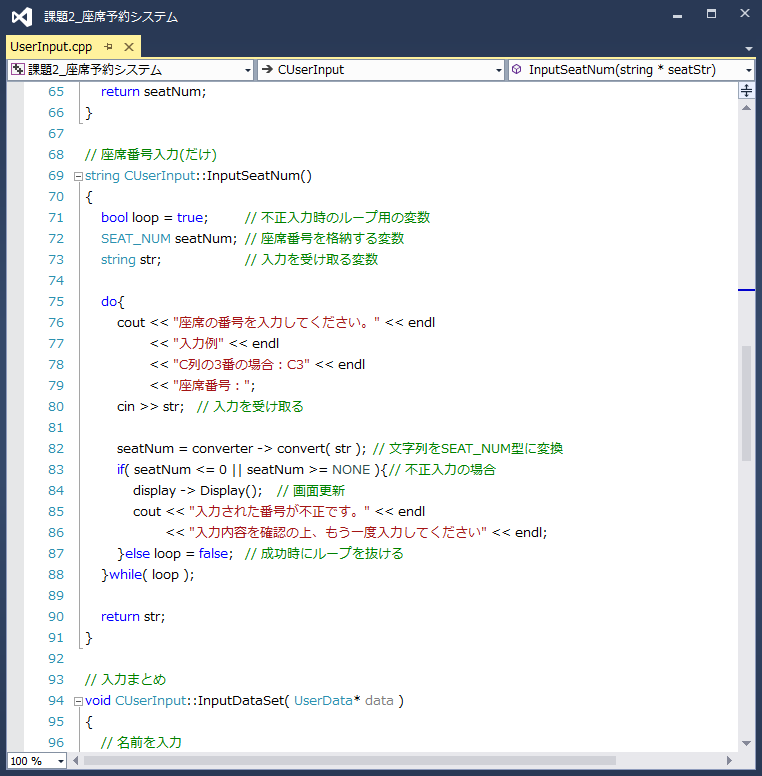


図17：UserInput.cpp　65行目から96行目まで

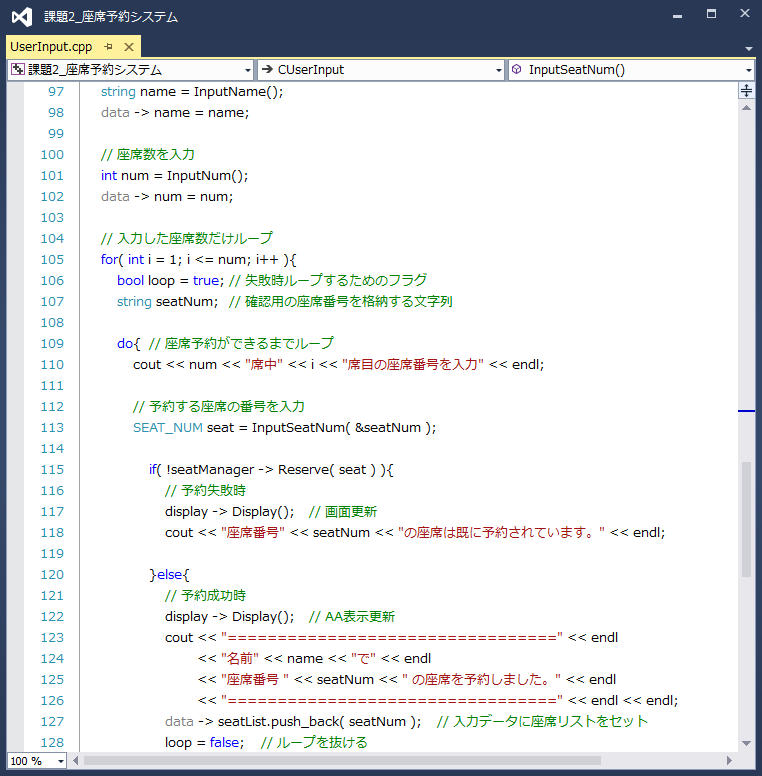


図18：UserInput.cpp　97行目から128行目まで

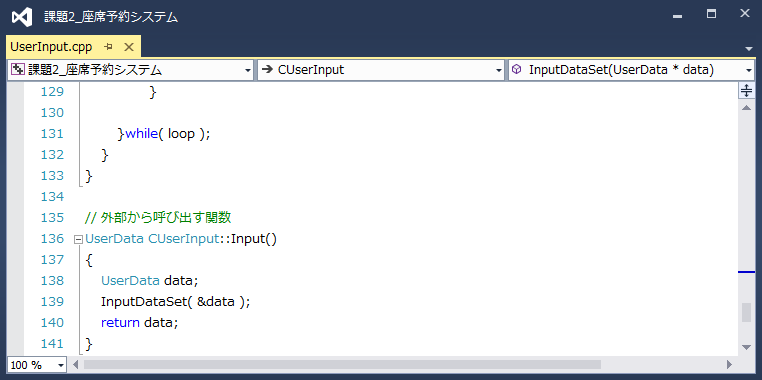
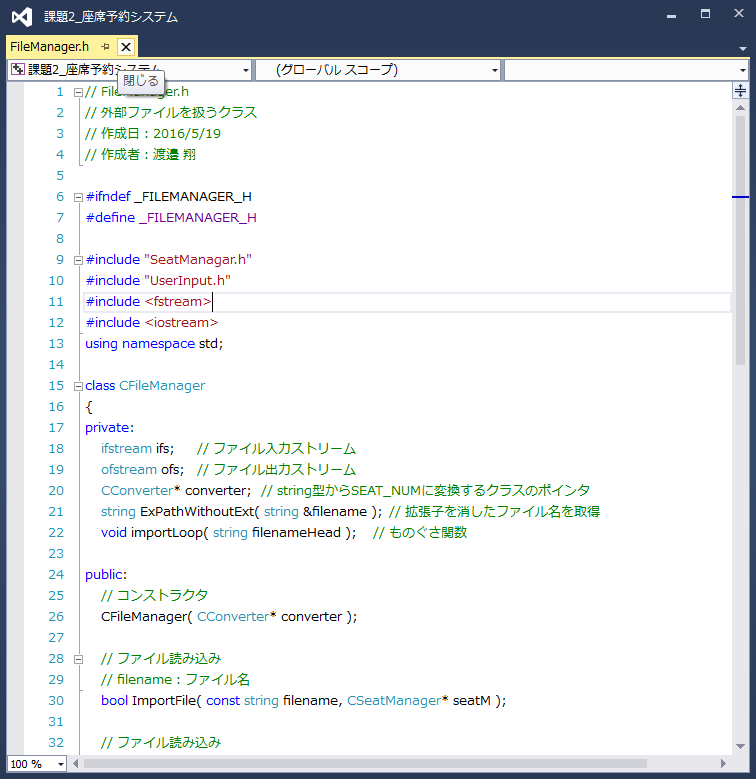


図20：UserInput.cpp　129行目から最後まで



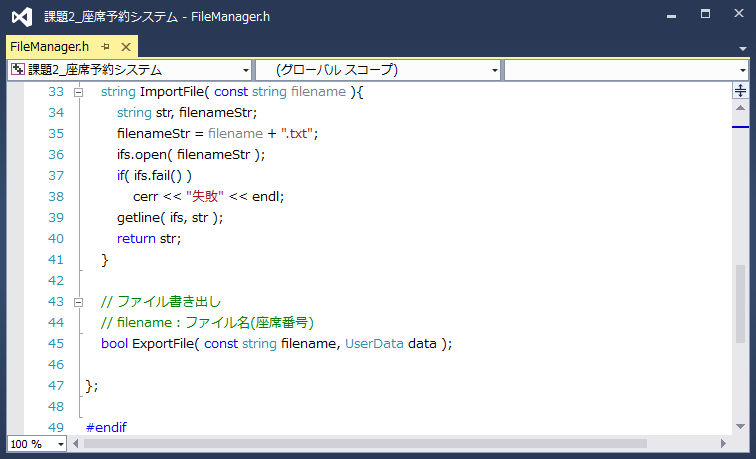
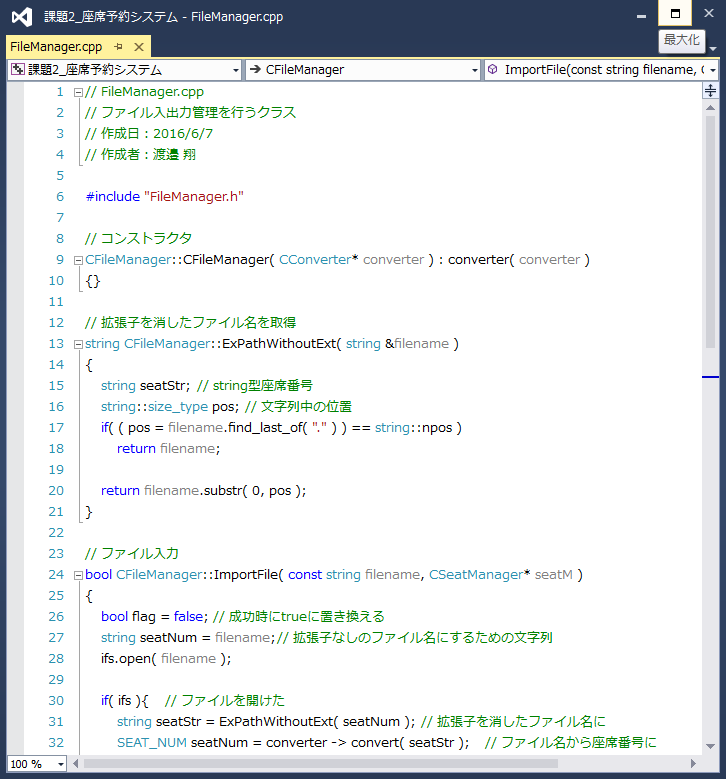


図20、21：FileManager.h　

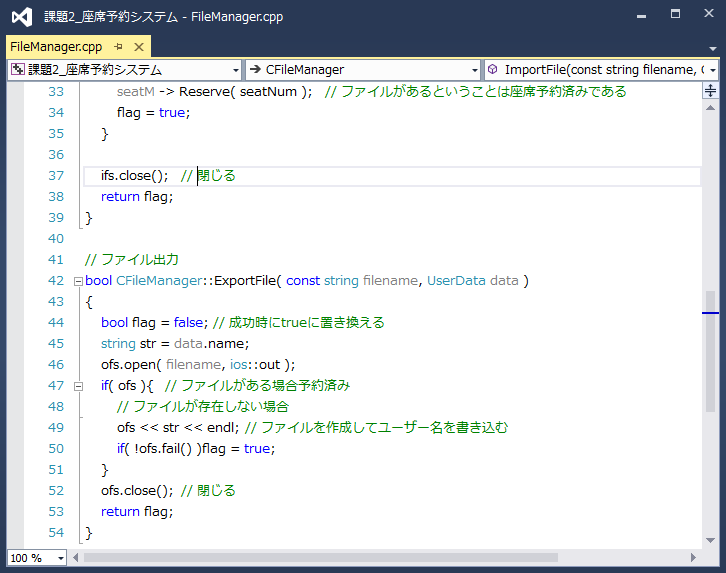


図22、23：FileManager.cpp

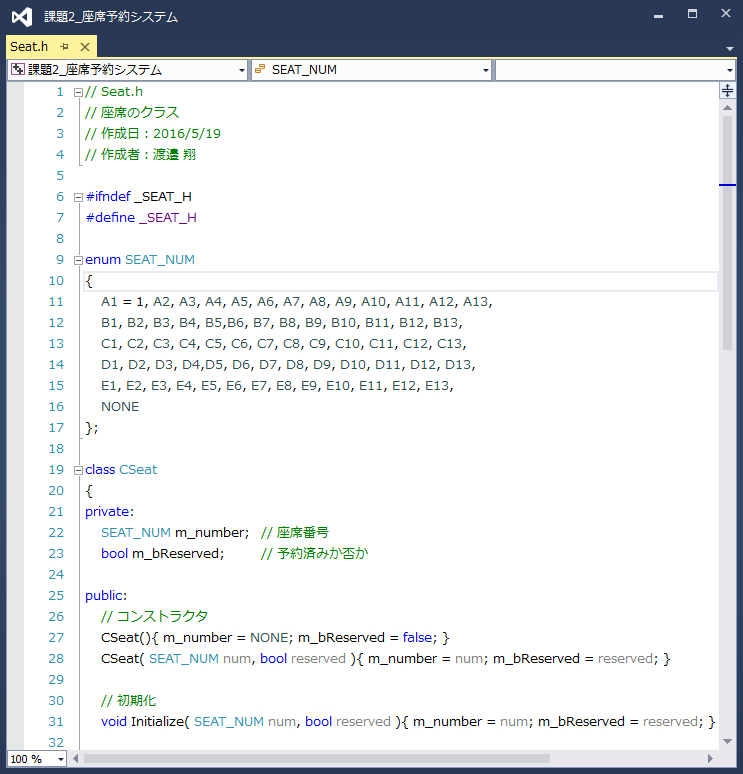
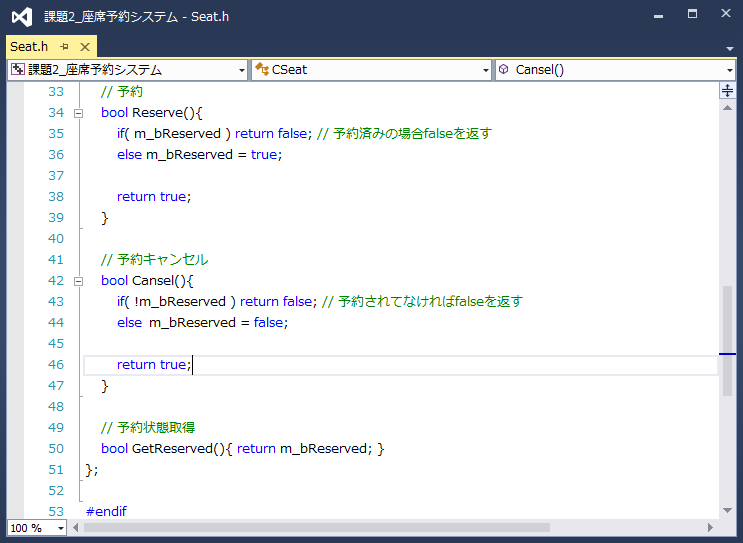
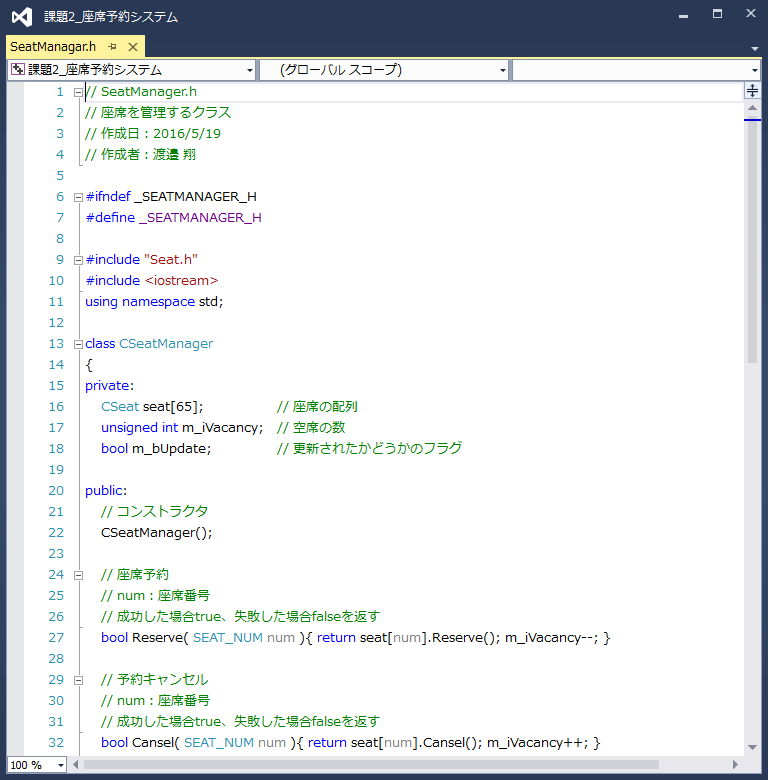


図24、25：Seat.h



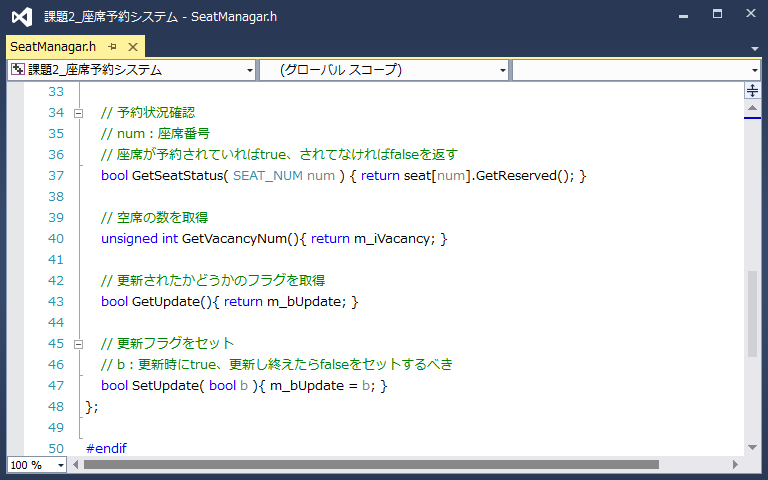


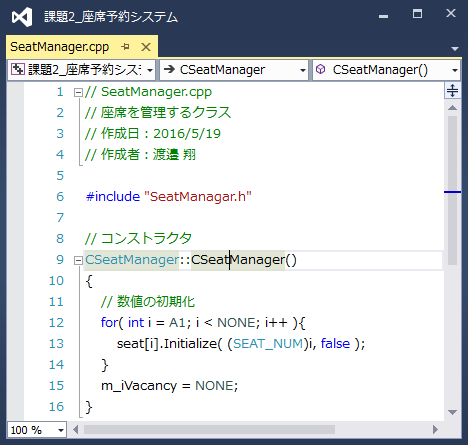
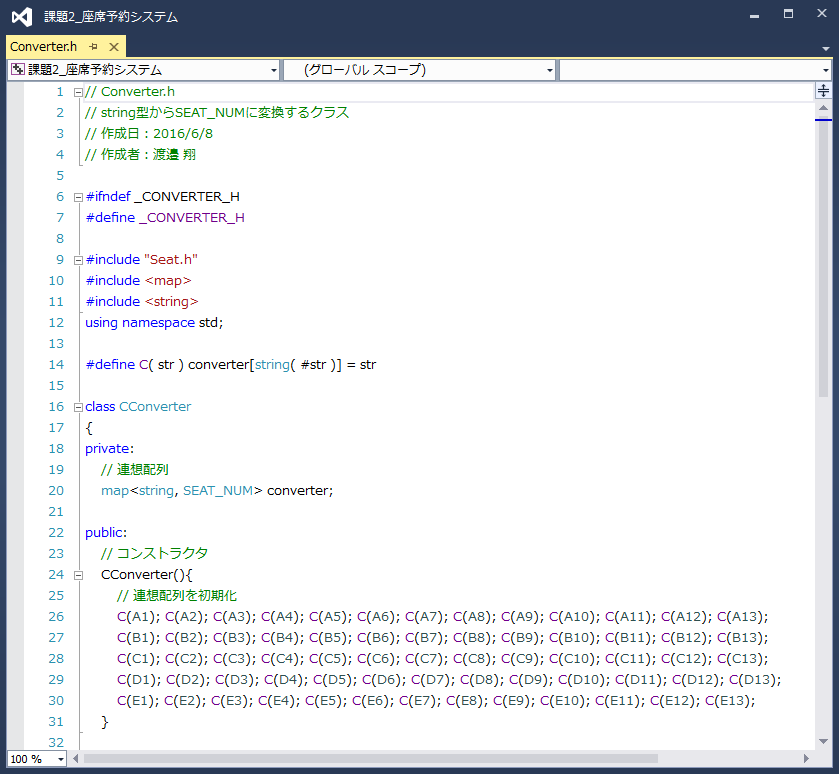
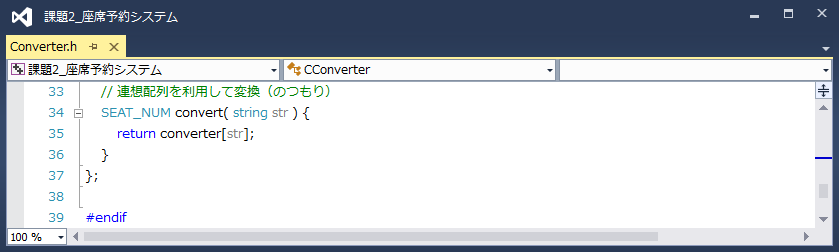
図26、27：SeatManager.h

図28：SeatManager.cpp



図29、30：Converter.h

1. 実行結果

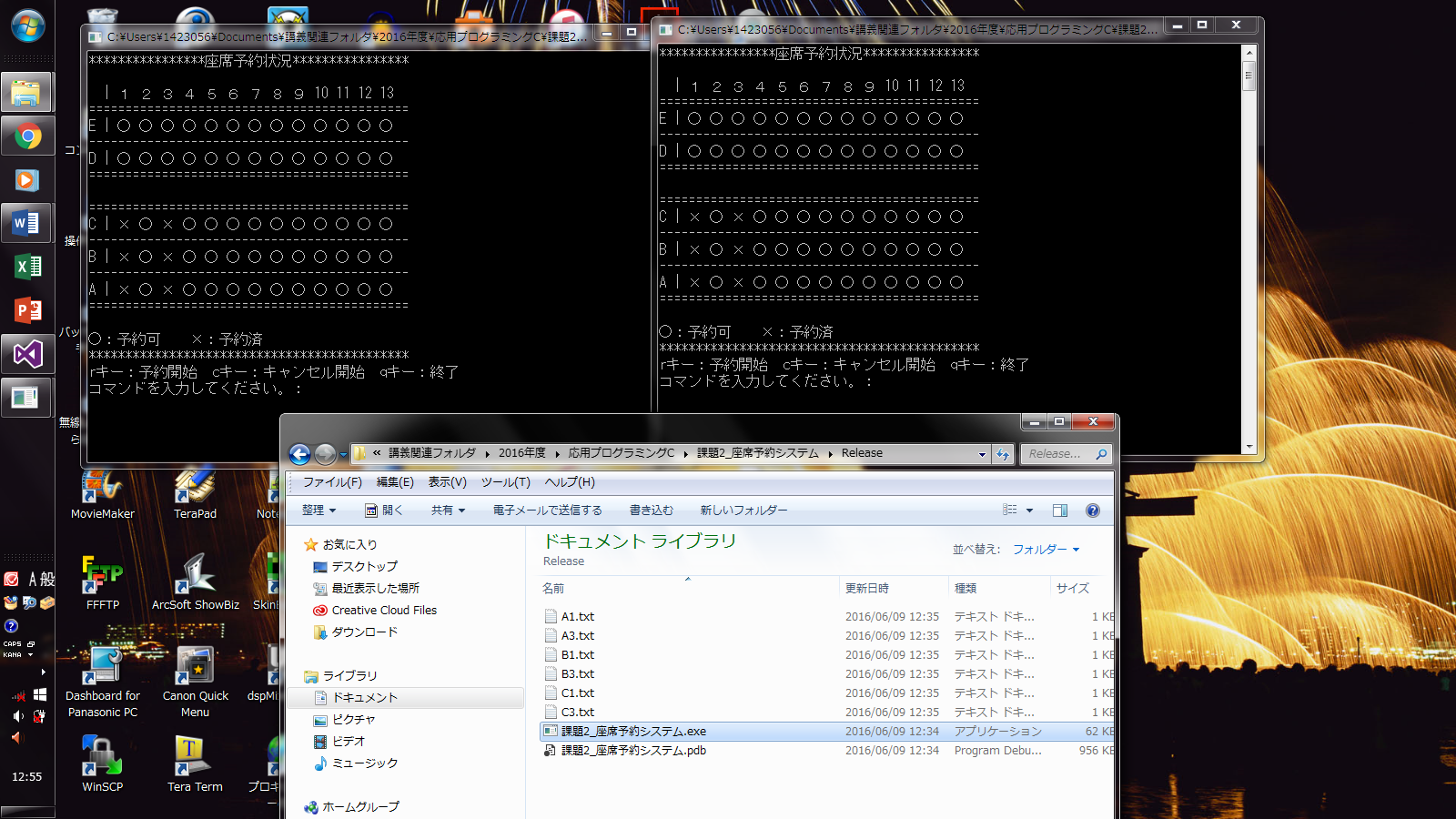


図31：実行画面

予約した際に、座席番号をファイル名としてテキストファイルを作成し、ファイルの内容には実行中に登録した名前を保存している。

ユーザーが何らかの入力を行う度に存在する予約ファイルをすべて読み込みSeatManagerクラスが更新される。

　予約時にSeatManagerクラス内のデータで指定した席が予約されていた場合は予約エラーが起き、存在しない場合は予約できる仕組みとなっている。

同じパソコン内でつプログラムを動作させ、予約状況に矛盾がないことを確かめた。同時に操作できないので完全にデバッグができておらず、予約するタイミングによっては予期せぬエラーが発生する可能性があると考えられる。

また、キャンセル機能を実装しようとしたがファイルをうまく削除できなかったため、断念せざるを得なかった。

1. 考察

　予約システムを実際に作成してみて、世の中で使われているシステムがいかに複雑で高度なものなのか実感した。他のユーザーとの兼ね合いだけでなくデータベースの扱いや、インターフェイスのわかりやすさ等、商用のソフトウェアを作成する場合は考えなければならないことが多いと思った。

　今回ファイルの削除ができなかったのはstring型をDeleteFile関数に対応できる型に変換することができなかったことが主な原因である。プロジェクトの設定でUnicord形式かマルチバイト形式のどちらかを選択でき、どちらにするかで文字列の扱い方が変わってくる。それぞれに対応した文字列型の使い方や、相互変換の方法の理解と実践を深めていく必要があると感じた。