자료구조와실습  
- 중간 시험 과제 –

동국대학교 컴퓨터공학과  
2016112122 이한빈

스크린샷, 그리기이(가) 표시된 사진

자동 생성된 설명FlightReservationSystem은 Flight을 노드로 하는 이중연결리스트와 Customer를 노드로 하는 이중연결리스트를 갖고 있다.

Flight 클래스

Flight 이중연결리스트에서 Node의 역할을 하는 Flight 클래스는 다음과 같은 멤버 변수와 함수들을 갖고 있다.

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Descreption** |
| **int** | flight\_number | 비행기 번호 |
| **string** | from\_city | 비행기 출발지 |
| **string** | to\_city | 비행기 도착지 |
| **int** | capacity | 최대 탑승 고객 수 |
| **int** | passenger | 예약 고객 수 |
| **int** | standby | 대기 고객 수 |
| **bool** | full | 비행기가 꽉 찼는지 여부 |
| **bool** | flag | customer 객체에서 비행기가 예약됐는지 표시 |
| **Flight\*** | pre\_flight | 이전 비행기 link |
| **Flight\*** | next\_flight | 다음 비행기 link |
| **ReservePassenger** | reserve\_list | 예약고객리스트. head, tail로 front, tail에 접근 가능. |
| **StandbyPassenger** | standby\_list | 대기고객리스트. head, tail로 front, tail에 접근 가능. |

|  |  |  |
| --- | --- | --- |
| **Return Type** | **Function Name** | **Descreption** |
|  | Flight() 생성자 | Flight 클래스의 생성자 |
| **bool** | is\_full() | 예약고객리스트 reserve\_list가 꽉 찼는지 확인 |
| **void** | reserve\_flight(string, Flight\*) | 이름 순으로 비행기에 고객 예약.  reserve 인원 추가. |
| **void** | stand\_flight(string) | 대기 신청한 순서대로 비행기에 고객 대기.  standby 인원 추가. |
| **string** | pop\_standby() | 가장 먼저 대기 신청한 고객을 대기고객 리스트에서 pop. standby 인원 감소. |
| **void** | show\_name(ofstream&) | 파일에 결과 출력 |
| **void** | show\_info(ofstream&) | 파일에 결과 출력 |

Flight 클래스는 Passenger를 노드로 하는 예약고객리스트 ReservePassenger, 대기고객리스트 StandbyPassenger 과 link로 연결되어있다.

Passenger 클래스

Passenger 클래스는 ReservePassenger과 StandbyPassenger의 노드 역할을 한다. 비행기에 탑승한 고객 정보를 갖고 있다. Passenger 클래스는 link를 통해 고객 리스트의 Customer 노드와 연결되어 있다. Passenger 클래스는 아래와 같은 멤버변수와 함수를 갖는다.

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Descreption** |
| **string** | name | 비행기 탑승객의 이름 |
| **Passenger\*** | prev\_passenger | 같은 비행기의 이전 탑승객 link |
| **Passenger\*** | next\_passenger | 같은 비행기의 다음 탑승객 link |
| **bool** | flag | 예약 성공 여부 |
| **Flight\*** | this\_flight | 예약한 비행기편 객체에 대한 link |
| **Customer\*** | cus | Customer 이중연결리스트의 노드 역할을 하는 Customer 객체에 대한 link |

|  |  |  |
| --- | --- | --- |
| **Return Type** | **Function Name** | **Descreption** |
|  | Passenger 생성자 | Passenger 클래스의 생성자 |

ReservePassenger 클래스

ReservePassenger 클래스는 Passenger를 노드로 갖는 예약 고객에 관한 정보를 저장하는 이중연결리스트이다. ReservePassenger에 Passenger 노드가 추가될 때는 이름 순서대로 삽입된다. ReservePassenger 클래스는 아래와 같은 멤버변수와 함수를 갖는다.

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Descreption** |
| **Passenger\*** | head | dummy node: 실질적인 정보 저장 x |
| **Passenger\*** | tail | dummy node: 실질적인 정보 저장 x |

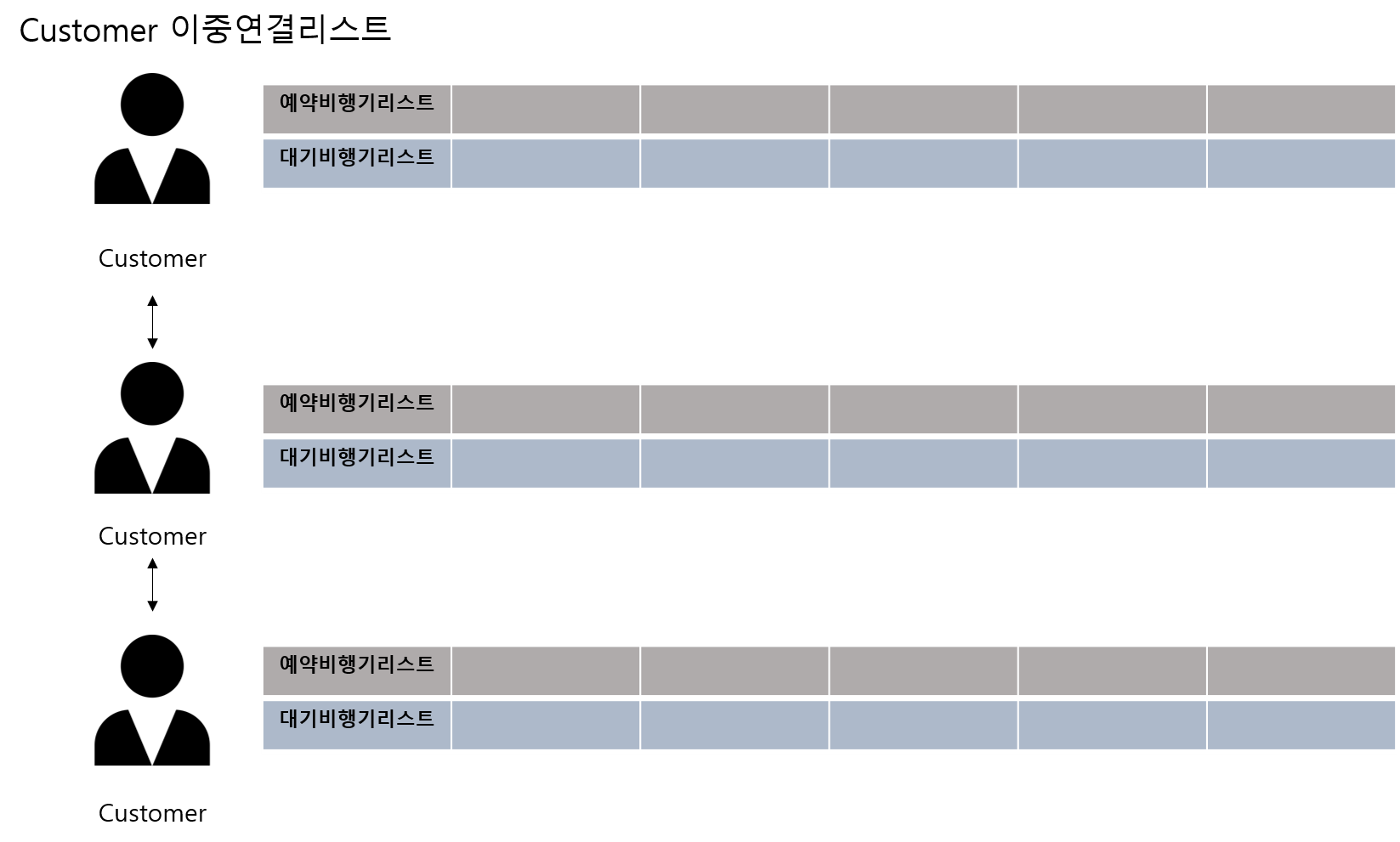
|  |  |  |
| --- | --- | --- |
| **Return Type** | **Function Name** | **Descreption** |
|  | ReservePassenger 생성자 | ReservePassenger 클래스의 생성자 |
| **void** | add(string, Flight\*) | parameter로 받은 string 대로 Passenger 객체를 생성하고 예약한 비행기편과 링크. |
| **bool** | is\_empty() | 예약고객리스트가 비어있는지 확인 |

StandbyPassenger 클래스

StandbyPassenger 클래스는 Passenger를 노드로 갖는 대기 고객에 관한 정보를 저장하는 이중연결리스트이다. StandbyPassenger에 Passenger 노드가 추가될 때는 대기 신청을 한 순서대로 추가되며 대기 고객이 예약고객리스트로 이동할 때는 가장 먼저 대기 신청을 한 고객이 deque 된다. StandbyPassenger 클래스는 아래와 같은 멤버변수와 함수를 갖는다.

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Descreption** |
| **Passenger\*** | head | dummy node: 실질적인 정보 저장 x |
| **Passenger\*** | tail | dummy node: 실질적인 정보 저장 x |

|  |  |  |
| --- | --- | --- |
| **Return Type** | **Function Name** | **Descreption** |
|  | ReservePassenger 생성자 | ReservePassenger 클래스의 생성자 |
| **void** | enque(string) | parameter로 받은 string 대로 Passenger 객체를 생성 |
| **string** | deque() | 가장 먼저 대기신청한 고객을 deque. standby 인원 변동은 Flight class에서 한다. |
| **bool** | is\_empty() | 대기고객리스트가 비어있는지 확인 |
| **bool** | is\_full | 대기고객리스트가 꽉 찼는지 확인 |



Customer 클래스

Customer 클래스는 FlightReservationSystem에서 고객에 대한 정보를 저장하는 이중연결리스트를 만들 때 노드의 역할을 하는 클래스이다. Customer 클래스에는 고객이 예약 또는 대기한 비행기에 대한 link를 갖고 있다. Customer가 이중연결리스트를 만들 때는 이름 순서대로 정렬이 되게끔 이중연결리스트에 Customer 노드를 추가한다.

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Descreption** |
| **String** | name | 고객 이름 |
| **Customer\*** | prev\_customer | 이전 고객에 대한 포인터 |
| **Customer\*** | next\_customer | 다음 고객에 대한 포인터 |
| **Flight\*** | reserve\_flight\_head | 예약한 flight list의 head에 대한 link(dummy node) |
| **Flight\*** | reserve\_flight\_tail | 예약한 flight list의 tail에 대한 link(dummy node) |
| **Flight\*** | standby\_flight\_head | 대기 중 flight list의 head에 대한 link(dummy node) |
| **Flight\*** | standby\_flight\_tail | 대기 중 flight list의 tail에 대한 link(dummy node) |
| **bool** | flag | 비행기 예약 여부 |

|  |  |  |
| --- | --- | --- |
| **Return Type** | **Function Name** | **Descreption** |
|  | Customer 생성자 | Customer 클래스의 생성자 |
| **void** | add\_flight(Flight\*) | 비행기를 예약비행기 리스트에 추가 |
| **void** | add\_standby\_flight(Flight\*) | 비행기를 대기비행기 리스트에 추가 |
| **bool** | is\_reserve() | 예약된 비행기가 있으면 true, 없으면 false return |
| **bool** | is\_standby() | 대기한 비행기가 있으면 true, 없으면 false return |

ReservationSystem 클래스

모든 클래스들을 종합하여 다룰 수 있는 프로그램의 중심이 되는 클래스이다. Flight과 Customer 클래스를 노드를 하는 이중연결리스트가 각각 하나씩 존재한다. ReservationSystem 클래스의 함수를 실행하면 txt 파일에 결과를 출력하는 경우 외에는 각 기능을 수행할 때마다 콘솔창에 진행현황을 보여준다. 아래는 ReservationSystem 클래스의 멤버변수와 멤버함수이다.

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Descreption** |
| **Flight\*** | head\_flight | 비행기에 정보를 저장하는 이중연결리스트의 head (dummy node) |
| **Flight\*** | tail\_flight | 비행기에 정보를 저장하는 이중연결리스트의 tail (dummy node) |
| **Customer\*** | head\_customer | 고객정보를 저장하는 이중연결리스트의 head (dummy node) |
| **Customer\*** | tail\_customer | 고객정보를 저장하는 이중연결리스트의 tail (dummy node) |
| **int** | flight\_cnt | 전체 비행기 수 |
| **int** | customer\_cnt | 전체 고객 수 |

|  |  |  |
| --- | --- | --- |
| **Return Type** | **Function Name** | **Descreption** |
| **void** | insert\_flight(int, string, string, int) | 비행기 리스트에 비행기 추가 |
| **void** | delete\_flight(int) | 비행기 리스트에서 비행기 삭제 |
| **bool** | is\_empty() | 비행기 리스트가 비어있는지 확인 |
| **bool** | is\_full() | 비행기 리스트가 꽉 찼는지 확인 |
| **bool** | is\_empty\_customer() | 고객 리스트가 비어있는지 확인 |
| **void** | linkPasCus(Flight\*, string) | Passenger 클래스의 멤버변수 cus가 Customer 객체를 가리키도록 연결 |
| **void** | add\_customer(string, Flight\*) | 예약 고객 추가 |
| **void** | standby\_customer(string, Flight\*) | 대기 고객 추가 |
| **void** | reserve(string, int) | 비행기 예약 혹은 대기 |
| **void** | cancel(string, int) | 비행기 예약 혹은 대기 취소(전체삭제, 혹은 특정 편 삭제 가능) |
| **void** | cancle\_reserve(string, int) | 특정 비행기편 예약 취소 |
| **void** | cancel\_standby(string, itn) | 특정 비행기편 대기 취소 |
| **void** | cancel\_all\_reserve(string) | 모든 예약 및 대기 취소 |
| **void** | delete\_flight\_in\_reserve(Customer\*, int) | Customer 객체에서 예약한 비행기편 삭제 |
| **void** | delete\_flight\_in\_standby(Customer\*, int) | Customer 객체에서 대기 중인 비행기편 삭제 |
| **void** | showAllFlightNumber(string, ofstream&) | 파일에 결과 출력 |
| **void** | showAllCustomerInfo(ofstream&) | 파일에 결과 출력. 테스팅. |
| **void** | showCustomerInfo(string, ofstream&) | 파일에 결과 출력 |
| **void** | showFlightInfo(string, int, ofstream&) | 파일에 결과 출력 |

공통함수

template으로 구현하려고 했으나 프로그램 설계 당시 변수명을 다르게 설계하여 template으로 구현하지 못한 함수이다. 이중연결리스트에서 노드를 삭제하거나 추가할 때 사용하기 위해 만들어졌으며 overloading을 통해 매개변수의 개수에 따라 호출되는 함수를 다르게 하였다.

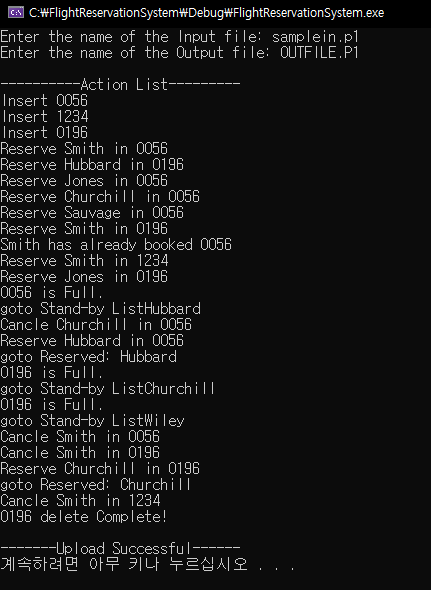
|  |  |  |
| --- | --- | --- |
| **Return Type** | **Function Name** | **Descreption** |
| **void** | connect\_link(prev, next) | 이중연결리스트에서 노드를 삭제할 때 링크 연결해주는 함수 |
| **void** | connect\_link(prev, curr, next) | 이중연결리스트에서 노드를 추가할 때 링크를 연결해주는 함수 |

기타함수

main 함수를 작동하게 하는 함수들이다.

|  |  |  |
| --- | --- | --- |
| **Return Type** | **Function Name** | **Descreption** |
|  | split() | string을 분할하는데 사용되는 함수이다. |
| **string\*** | fileIn | 사용자로부터 txt 파일을 입력 받아 프로그램에서 file을 open 하고 파일의 문자열을 string 배열로 반환한다. 이후, flie을 close한다. |
| **void** | update(ReservationSystem&, vector<string>, string\*, ofstream&) | 파일 입력을 통해 입력 받은 문장을 반복하여 비행기 예약시스템을 진행한다. |
| **ofstream** | fileOut() | 파일을 출력하기 위한 경로를 입력 받고 파일을 출력 파일을 open한다. |

RESULT:



콘솔창에는 위와 같이 실행한 ACTION에 대한 정보를 출력한다.

OUTFILE.P1에는 다음과 같은 정보를 출력한다.

|  |
| --- |
| ==============================  총 비행기: 3대  [Flight NO.0056]  Reserved List:  Stand-by List :  ------------------------------  [Flight NO.0196]  Reserved List:  Stand-by List :  ------------------------------  [Flight NO.1234]  Reserved List:  Stand-by List :  ------------------------------  ==============================  ==============================  총 비행기: 3대  [Flight NO.0056]  From : Lafayette  To : Detroit  Capacity : 4명  Reserved : 4명  Stand-by : 1명  ------------------------------  [Flight NO.0196]  From : Dayton  To : Indianapolis  Capacity : 3명  Reserved : 3명  ------------------------------  [Flight NO.1234]  From : Chicago  To : Baltimore  Capacity : 6명  Reserved : 1명  ------------------------------  ==============================  [Flight NO.0056]  Reserved List: Churchill Jones Sauvage Smith  Stand-by List : Hubbard  ------------------------------  [Customer Name: Smith]  Reserved Flight:  [Flight NO.0056]  From: Lafayette  To : Detroit  Reservation: OK  [Flight NO.0196]  From: Dayton  To : Indianapolis  Reservation: OK  [Flight NO.1234]  From: Chicago  To : Baltimore  Reservation: OK  ------------------------------  ==============================  총 비행기: 3대  [Flight NO.0056]  Reserved List: Hubbard Jones Sauvage Smith  Stand-by List :  ------------------------------  [Flight NO.0196]  Reserved List: Hubbard Jones Smith  Stand-by List : Churchill Wiley  ------------------------------  [Flight NO.1234]  Reserved List: Smith  Stand-by List :  ------------------------------  ==============================  [Flight NO.0196]  Reserved List: Churchill Hubbard Jones  Stand-by List : Wiley  ------------------------------  ==============================  총 비행기: 2대  [Flight NO.0056]  Reserved List: Hubbard Jones Sauvage  Stand-by List :  ------------------------------  [Flight NO.1234]  Reserved List:  Stand-by List :  ------------------------------  ============================== |