프로젝트 최종 보고서

|  |
| --- |
| 1. 프로그램 개요   * 우편물 배송 관리 프로그램 * 회원 로그인, 우편 및 소포 발송, 우편집중국 관리, 관리자 메뉴 등 * 기존 데이터 로드   2. 전체 논리 구조도  3. 모듈 별 기능   * + 1. App        1. 파일 로드 – 데이터를 읽어와 메모리에 저장함        2. 고객 로그인 – 고객의 아이디, 비밀번호와 비교하여 개인 계정에 로그인 함        3. 관리자 로그인 – 관리자 아이디, 비밀번호와 비교하여 관리자 계정에 로그인 함        4. 로그아웃 – 현재 저장된 로그인 정보를 초기화함        5. 메뉴 출력        6. 종료 – 프로그램을 종료함     2. User        1. 고객 우편물 보내기 – 로그인한 사용자가 우편물을 보냄        2. 고객 우편물 받기 – 로그인 된 사용자에게 온 우편물을 받음        3. 고객 보낸 우편물 목록 – 현재까지 보낸 우편물 목록을 출력함        4. 관리자 우편물 보내기 – 현재 관리중인 우편집중국에 저장된 우편물들을 고객 또는 인근 우편집중국으로 발송함        5. 관리자 우편물 받기 – 분류 대기중인 우편물들을 분류함     3. Mail        1. 가격 계산 – 편지와 소포, 무게에 따라 가격을 계산     4. MailCenter        1. 우편집중국 데이터 저장     5. MailDelivery        1. 이동 횟수 – 메일이 총 이동할 우편집중국 개수를 계산        2. 다음 우편집중국 ID – ID를 반환하여 다음 우편집중국으로 발송   4. 처리 흐름도  5. 각 모듈의 함수    a. App();  /\*======================================= App==============================================  class App  {  private:  vector<User\*> users; //전체 고객 배열  vector<MailCenter\*> mailCenters; //전체 우편집중국 배열  int\*\* router; //현재 우편 집중국이 인근 우편집중국과 연결되어 있는지를 확인  User\* loginedUser; //현재 로그인한 유저를 가리키는 포인터 변수  MailDelivery md;  private:  bool loadUserFiles(); //데이터 파일 로드  bool loadRouterFiles();  bool loadMailCenterIdFiles();  User\* login(int userType, string input\_id, string input\_password); //로그인 함수  bool logout(); //로그아웃 함수  // void saveFiles(); //데이터 파일 저장  void printClientMenu(); //고객 메뉴 출력  void printManagerMenu(); //관리자 메뉴 출력  public:  App(); //파일 로드 및 변수 초기화  ~App(); //소멸자  void run(); //시작 화면 출력  };  ==================================================================================================\*/  b. User  /\*======================================= User==============================================  class User  {  protected:  int userType; // 유저 타입( 0은 일반 고객, 1은 관리자)  string id; //아이디  string password; // 비밀번호  public:  User();  ~User();  void setUserType(int \_UserType);  void setId(string \_Id);  void setPassword(string \_Password);  int getUserType();  string getId();  string getPassword();  virtual vector<Mail\*>\* getReceivedMail() = 0;  virtual vector<Mail\*>\* getSendMail() = 0;  virtual bool send(MailDelivery& md) = 0; // 우편물을 보내는 순수 가상 함수  virtual bool receive(MailDelivery& md) = 0; // 우편물을 받는 순수 가상 함수  };  class Client : public User // User로부터 상속받은 Client 클래스  {  private:  string name; // 이름  string address; // 주소  vector<Mail\*> sendMailList; // 보낸 메일 리스트  vector<Mail\*> reciedvedMailList; // 받은 메일 리스트  public:  Client();  Client(int \_UserType, string \_Id, string \_Password, string \_Name, string \_Address);  ~Client();  void setName(string \_name);  void setAddress(string \_address);  void setClient(int \_UserType, string \_Id, string \_Password, string \_Name, string \_Address);  string getName();  string getAddress();  void printSendedMail();  virtual vector<Mail\*>\* getReceivedMail() { return &reciedvedMailList; };  virtual vector<Mail\*>\* getSendMail() { return &sendMailList; };  virtual bool send(MailDelivery& md); // 우편물 보내기  virtual bool receive(MailDelivery& md); // 우편물 받기  };  class Manager : public User // User로부터 상속받은 Manager 클래스  {  public:  Manager();  Manager(int \_UserType, string \_Id, string \_Password);  ~Manager();  void setCurrentCityName(string \_CurrentCityName);  string getCurrentCityName();  void setManager(int \_UserType, string \_Id, string \_Password);    virtual vector<Mail\*>\* getReceivedMail() { return NULL; };  virtual vector<Mail\*>\* getSendMail() { return NULL; };  virtual bool send(MailDelivery& md); // 우편물 보내기  virtual bool receive(MailDelivery& md); //우편물 받기  private:  string currentCityName; // 현재 도시 이름  };    =============================================================================================\*/  c. Mail  /\*======================================= Mail==============================================  class Mail  {  protected:  string senderId; //보낸 사람 ID  string receiverId; //받는 사람 ID  string senderAddr; //보낸 사람 주소  string receiverAddr; //받는 사람 주소  int mailType; //우편물 종류  int cost; //배송 비용  public:  Mail();  ~Mail();  void setSenderId(string \_SenderId);  void setReceiverId(string \_ReceiverId);  void setSenderAddr(string \_SenderAddr);  void setReceiverAddr(string \_ReceiverAddr);  void setMailType(int \_MailType);  void setCost(int \_Cost);  string getSenderId();  string getReceiverId();  string getSenderAddr();  string getReceiverAddr();  int getMailType();  int getCost();  virtual int calculateCost(int distance) = 0; //배송 비용을 계산하는 순수 가상 함수  virtual void printData() = 0;  };  class Letter : public Mail // Mail로부터 상속받은 Letter 클래스  {  public:  virtual int calculateCost(int distance); // 거리에 따른 배송 비용 계산  virtual void printData();  public:  Letter();  ~Letter();  };  class Package : public Mail // Mail로부터 상속받은 Package 클래스  {  public:  virtual int calculateCost(int distance); // 거리와 무게에 따른 배송 비용 계산  virtual void printData();  Package();  ~Package();  void setWeight(int \_Weight);  int getWeight();  private:  int weight; // 소포 무게  };  ==================================================================================================\*/0  d. MailCenter  /\*======================================= MailCenter==============================================  class MailCenter  {  private:  int id; //우편집중국 ID  string cityName; //도시 이름  int neighborCnt; //인접한 우편집중국 개수  int\* neighborIds; //인접한 우편집중국 아이디  vector<Mail\*> sendMailList; //보낸 우편물 리스트  vector<Mail\*> recievedMailList; //받은 우편물 리스트  vector<Mail\*> localMailList; //내부로 발송될 우편물 리스트  vector<Mail\*> externalMailList; //외부로 발송될 우편물 리스트    public:  MailCenter();  MailCenter(int \_Id, string \_CityName);  ~MailCenter();  void setId(int \_Id);  void setCityName(string \_CityName);  int getId();  string getCityName();  vector<Mail\*>\* getreceivedMailList() { return &recievedMailList; };  vector<Mail\*>\* getlocalMailList() { return &localMailList; };  vector<Mail\*>\* getExternalMailList() { return &externalMailList; };  };  ==================================================================================================\*/  e. MailDelivery  /\*======================================= MailCenter==============================================  class MailDelivery  {  private:  vector<User\*>\* pointerOfUsers;//고객을 가리키는 포인터  vector<MailCenter\*>\* pointerOfMailCenters;//우편집중국을 가리키는 포인터  int\*\* pointerOfRouter;//경로를 가리키는 포인터  public:  MailDelivery();  ~MailDelivery();  bool clientSendMail(User\* curUser, Mail\* mail);//고객의 우편물 보내기  bool managerSendMail(string cityName);//관리자의 우편물 보내기  bool managerReceiveMail(string cityName);//관리자의 우편물 받기  vector<User\*>\* getpointerOfUsers() {return pointerOfUsers; };  int getTotalDistance(int sourceId, int destinationId);//출발지에서 목적지까지 총 이동 횟수를 계산하여 반환  int getNextMailCenterId(int currentPos, int destinationId);//현재 우편집중국에서 목적지 우편집까지 가는 경로의 다음 우편집중국 ID를 받아옴  int getMailCenterIdByCityName(string cityName);//도시 이름을 이용해 ID를 받아옴  int getMailCenterIdByAddress(string address);  string getCityNameByMailCenterId(int id);  void setDBPointers(vector<User\*>\* pUser, vector<MailCenter\*>\* pMailCenter, int\*\* pRouter)//App 클래스에 저장된 데이터들의 주소값을 복사해옴  {  pointerOfUsers = pUser;  pointerOfMailCenters = pMailCenter;  pointerOfRouter = pRouter;  }  };  ==================================================================================================\*/  6. 설계 원칙  설계 요구 조건에 따라 충실히 설계한다.   1. User 클래스를 상속받은 사용자 클래스 Client와 우편물 관리자 클래스 Manager가 있다. 2. Mail 클래스를 상속받은 편지 클래스 Letter와 소포 클래스 Package가 있다. 3. App은 User DB, Router DB, Mail Center DB를 가지고 있다.  - User.txt 파일에는 User를 상속받은 Client와 Manager가 섞여 저장되어 있으며 공통적으로 부모인 User가 가지고 있는 userType, ID, pw를 가지고 있다  - MailCenterID.txt 파일에는 전국의 우편집중국과 각 우편집중국의 ID 필드가 있다.  - Router.txt는 우편집중국들 간에 연결정보를 가지고 있다. 4. User는 아이디, 비밀번호, 사용자 타입을 가지고 있다. 5. User를 상속받은 Client는 이름, 주소, 보낸 우편물 목록, 받은 우편물 목록을 가지고 있다. 6. User를 상속받은 Manager는 추가적인 맴버변수는 가지고 있지 않다. 7. Client와 Manager는 User클래스의 순수가상함수 send( )와 receive( ) 함수를 overridding하고 있어 서로 다른 기능을 수행한다. 8. App는 User와 MailCenter 객체들을 총괄하여 관리하고 필요한 함수를 호출한다. 9. 프로그램 실행 시에 User DB, Router DB, Mail Center ID DB를 App 클래스의 각 컨테이너에 로드한다. 10. 프로그램 종료 시에 갱신된 정보들을 파일에 다시 쓰지 않는다. 이번 프로젝트는 난이도를 고려하여 모든 우편물 송수신을 메모리에서만 동작하는 것으로 고려하고, 파일 입출력을 하지 않는다. 단, 앞서 언급한 우편물 관리에 필요한 모든 DB들은 프로그램 시작 시 읽어와야 한다.   7. 개발도구  Microsoft Visual Studio 2013 C++  8. 사용 매뉴얼     1. 시작 메뉴             (고객 로그인 시) (관리자 로그인 시)   1. 고객 로그인 / 관리자 로그인 : 입력 데이터와 저장된 ID, 비밀번호를 확인하여 개인 계정에 접속, 개인 정보와 메뉴 출력 (예외 처리 : 일치하는 ID가 없을 경우, 비밀번호가 일치하지 않을 경우, 이미 로그인 된 상태에서 재 로그인 할 경우, 지정된 숫자 외 다른 문자를 입력 받았을 경우, 고객 로그인에서 관리자 계정으로 로그인 시도 할 경우, 관리자 로그인에서 고객 계정으로 로그인 시도할 경우) 2. 종료 : 프로그램을 종료   b. 고객 우편물 보내기         1. 보낼 우편물 종류 선택하기 : 편지와 소포 두 가지 종류의 우편물 중 하나를 선택하여 보낼 수 있음. 2. 받는 사람의 이름을 입력 (예외 처리 : 없는 사람의 이름을 입력, 자기 자신의 이름을 입력) 3. 받는 사람의 주소를 입력 (예외 처리 : 없는 주소를 입력) 4. 소포 선택했을 경우 무게 입력 (예외 처리 : 잘못된 값 입력) 5. 전송 완료 : 해당 지역의 우편집중국의 받은 우편물 리스트로 보냄   c. 보낸 우편물 목록     1. 모든 Client는 보낸 메일 리스트와 받은 메일 리스트를 멤버 변수로 갖고 있음 2. 고객이 편지 또는 소포를 전송 할 때, 보낸 우편물 목록에 추가. 우편물 종류, 보낸 사람, 주소, 받는 사람, 주소 등을 확인 가능   .  e. 우편집중국 우편물 수신    1. 우편집중국에 도착한 우편물들의 종류와 목적지를 출력하고 다른 우편집중국으로  배송이 필요한 우편물들은 다음 우편집중국의 이름을 출력함.    2. 우편집중국에 도착한 우편물들을 지역 내부로 보낼지, 다른 우편집중국으로 보낼지  분류함  f. 우편집중국 발송     1. 지역 내 발송 시 고객에게 직접 발송되었음을 출력함. 외부 발송 시 적지와 함께 발송된 우편물들을 출력함      1. 지역 내 리스트 내 우편물은 고객의 받은 메일 리스트로 이동. 다른 지역 발송 우편물은 다른 우편집중국 받은 우편물로 이동.   G, 고객 우편물 수신     1. 로그인한 고객에게 도착한 우편물을 확인 가능      1. 고객의 받은 우편물 리스트를 출력 |
|  |