

프로그램 툴

N A M E : 권 성 호

T E L : 010-8874-6452

P A R T : PROGRAMING

Efforts make all doors open



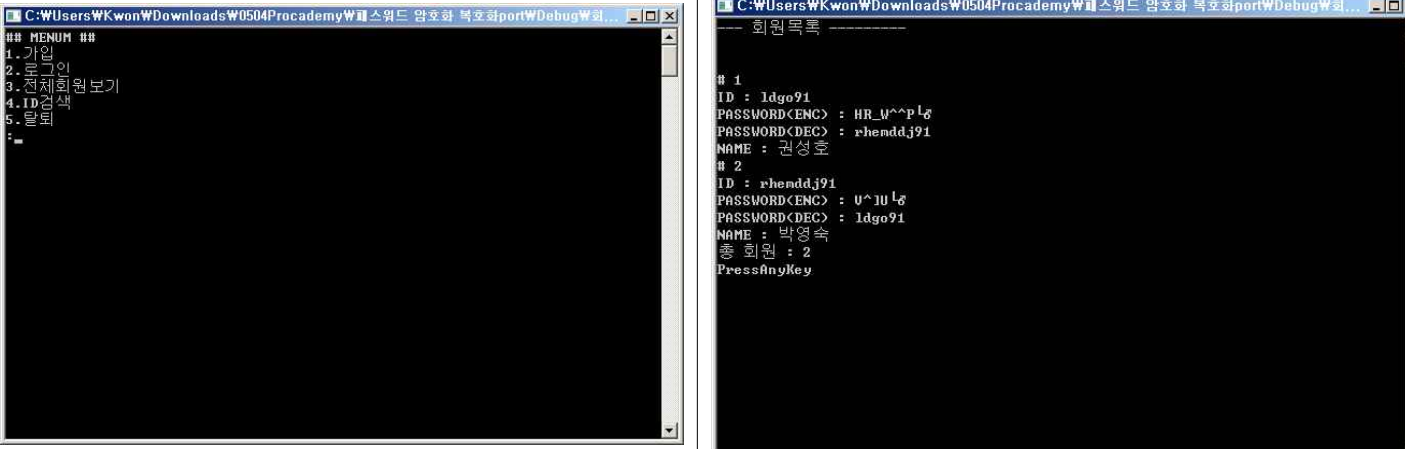
목 차




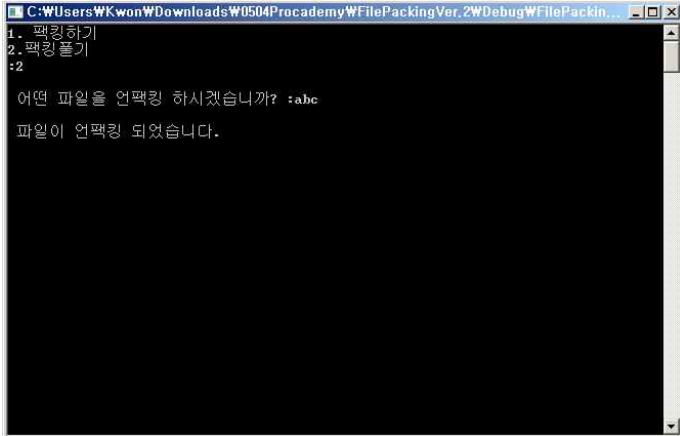
I . 포트폴리오 개요	1
1. 패스워드 암호화 복호화	1
2. 파일패킹 프로그램	2
3. 프로파일러	3
4. STL의 링크드 리스트 구현.....	4
5. new 오버로딩으로 메모리 누수 체크.....	5
6. 모니터링 툴	6
7. A* 알고리즘 툴	7
8. JumpSearch 알고리즘 툴	8
9. CsvToSqlConverter 툴	9
10. 링크	10

I. 포트폴리오 개요

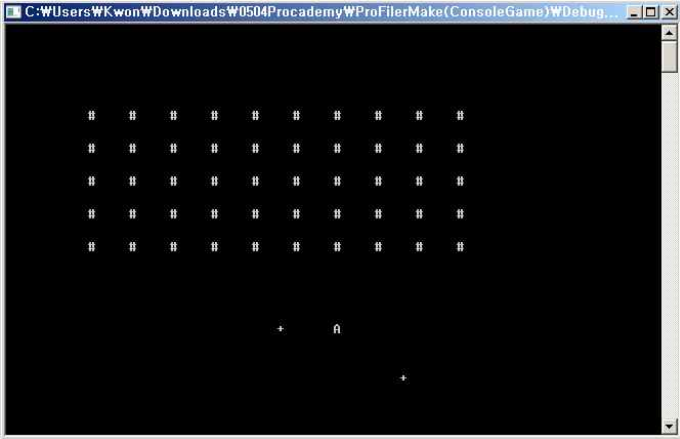
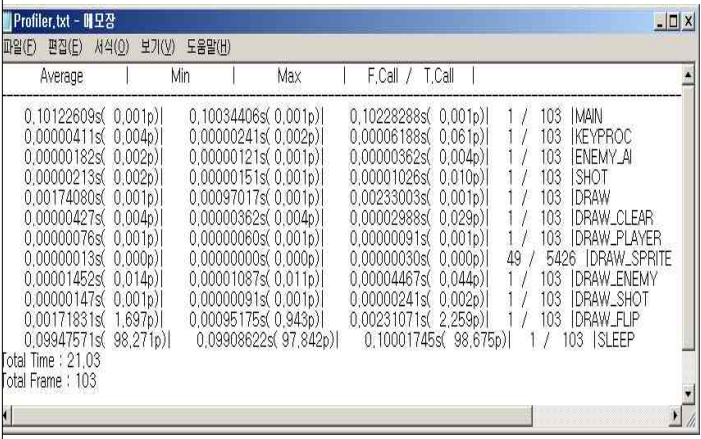
1. 패스워드 암호화 프로그램

프로그램 명	패스워드 암호화 프로그램
플랫폼	PC
프로그램 화면	
	
소개	<ol style="list-style-type: none"> 1. 간단한 회원관리 프로그램 2. 아이디를 키로 변환하여 해쉬테이블로 관리 3. 비밀번호를 XOR연산을 통하여 인코딩, 디코딩 진행
개발 환경	WinConsole
설명	<p>회원 정보를 관리하기 위해서 테이블을 작성하는 것은 필수이다. 리스트로 관리 하는 것보다 키 값으로 바로 접근하는 해시 테이블을 만들면 더욱 접근성이 높아 질 것이다. ID를 Key값으로 변환하는 작업이 들어간다. 또한 비밀번호를 서버에서 알 필요는 전혀 없다. DB에서 비밀번호 자체를 저장하는 경우 보안상 큰 위험이 된다. 서버에서는 비밀번호를 인 코딩한 값을 알고만 있는 것으로 충분하다.</p>

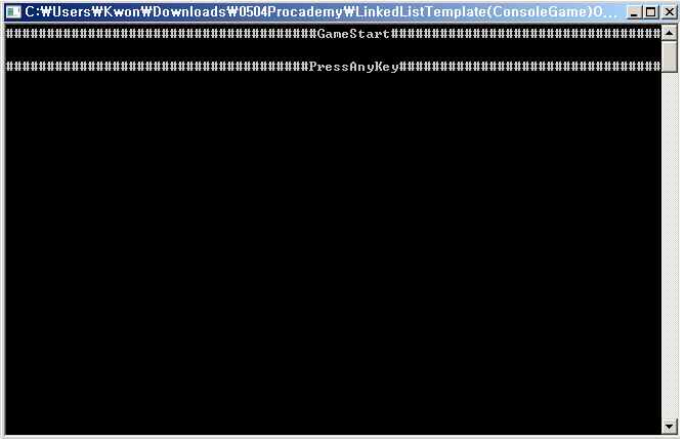
2. 파일패킹 프로그램

프로그램 명	파일패킹 프로그램
플랫폼	PC
프로그램 화면	
<div>   </div>	
소개	1. 여러개의 파일을 하나의 파일로 패킹 2. 간단한 헤드 부착으로 내 패킹 파일인지 확인
개발 환경	WinConsole
설명	데이터를 송수신 하던 가 패치파일을 릴리즈 할 때 여러 개의 데이터 파일을 하나의 파일로 만드는 작업이 필수적이다. 또한 아 패킹된 파일이 나의 파일인지 확인하는 작업과 패킹정보가 필수적으로 들어가야 된다. 이 프로그램은 앞에 헤더를 부착하여 본 프로그램으로 패킹된 파일인지 판단 여부와 개수와 이전 이름을 저장하여 이전에 분할된 파일로 돌려주는 기능을 가지고 있다.

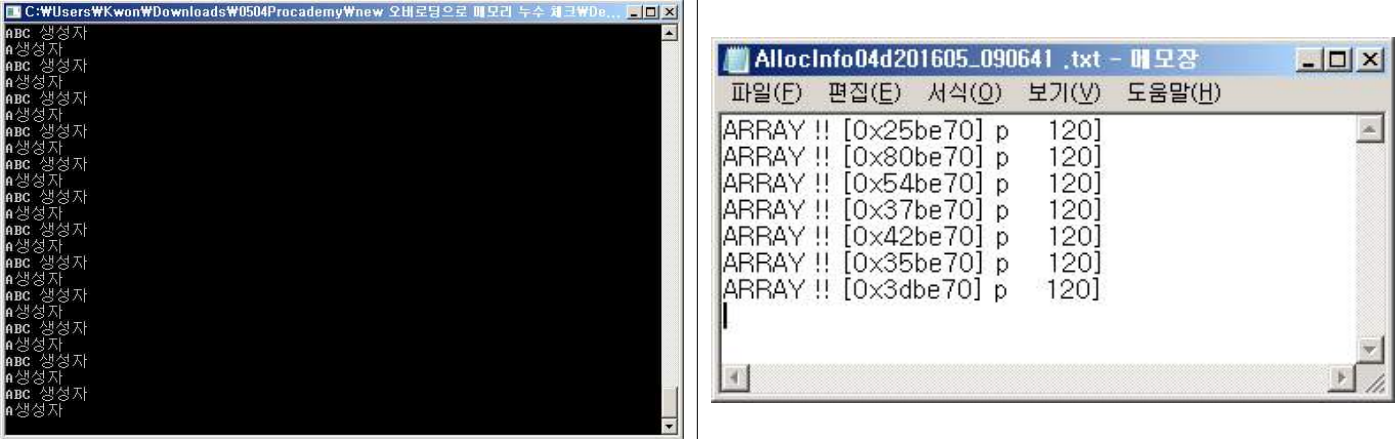
3. 프로파일러

프로그램 명	프로파일러
플랫폼	PC
프로그램 화면	
 	
소개	<ol style="list-style-type: none"> 1. 절차지향적인 프로그램에서 각각의 모듈의 속도를 측정 2. 프로그램이 종료되면 Profiler.txt로 정보가 저장됨
개발 환경	WinConsole
설명	<p>프로그램의 속도는 20%의 코드가 속도의 80%를 먹는다는 말이 있다. 이 20%의 코드를 어떻게 찾을 것인가? 바로 코드의 속도를 측정하는 프로파일러를 만들어야 할 것이다. 이 프로그램은 각각의 함수 명을 등록하여 시작과 끝 지점에 각 항목별로 호출하여 프레임 당 시간을 측정한다. 그 후 프로그램이 종료되는 시점에서 지금까지 저장된 정보를 파일 출력한다.</p>

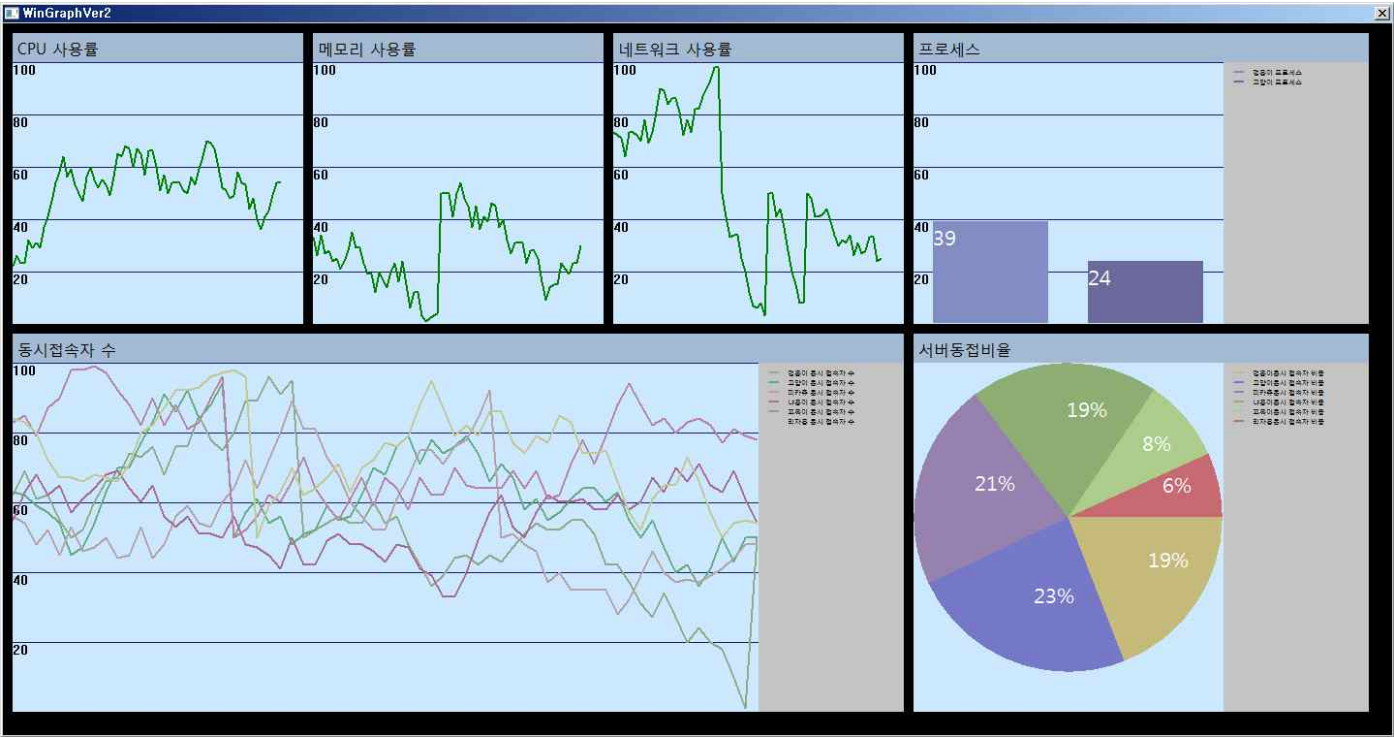
4. STL의 링크드 리스트 구현

프로그램 명	MiniMini
플랫폼	PC
프로그램 화면	
	<pre> 4 5 template<class DATA> 6 class CLinkedList 7 { 8 private: 9 struct Node { ... }; 10 private: 11 Node *m_Head; 12 Node *m_Rear; 13 public: 14 CLinkedList() { ... } 15 16 void Push(DATA data) { ... } 17 18 void Remove(const DATA& data) { ... } 19 20 public: 21 //friend class CIterator; 22 class CIterator 23 { 24 public: 25 Node* m_currentNode; 26 public: 27 CIterator() { ... } 28 CIterator(Node* node) { ... } 29 CIterator(CIterator &cls) { ... } 30 CIterator& operator = (CIterator &p) { ... } 31 32 public: 33 Node* getNode() { ... } 34 35 DATA* operator->() { ... } 36 37 DATA& operator*() { ... } 38 CIterator operator++() { ... } 39 CIterator operator++(int) { ... } 40 CIterator operator--() { ... } 41 CIterator operator--(int) { ... } 42 bool operator==(CIterator& cls) { ... } 43 44 bool operator!=(CIterator& cls) { ... } 45 }; 46 47 ... </pre>
소개	<ol style="list-style-type: none"> 1. 간단히 객체지향 적으로 짜여진 콘솔게임에 객체를 직접 관리하자. 2. STL의 LinkedList를 직접 구현하자.
개발 환경	WinConsole
설명	<p>게임을 만들다 보면 객체를 담은 컨테이너를 사용할 때가 있다. STL에서는 링크드 리스트를 기본으로 제공하고 있다. 하지만 쓸 때 없는 기능 등으로 속도가 저하될 경우도 있다. 링크드 리스트인 경우 직접 구현해 보았다. 중첩 클래스를 사용하여 반복자를 구현 하였으며 각각 상황에 맞는 연산자 오버로딩 또한 구현하였다.</p>

5. new 오버로딩으로 메모리 누수 체크

프로그램 명	new 오버로딩으로 메모리 누수 체크
플랫폼	PC
프로그램 화면	
	
소개	<ol style="list-style-type: none"> 1. new로 동적 할당된 변수가 해제가 되었는지 알아내보자. 2. 해제가 되지 않았다면 파일 출력 3. 할당된 라인과 파일 명을 생성할 때 저장해 두자.
개발 환경	WinConsole
설명	메모리를 할당할 때 항상 주의를 하여야 한다. 서버에서의 경우 항상 프로세스가 활성화 된 상태이기 때문에 로직 상 메모리 누수가 발생하게 되면 치명적인 문제가 될 수 있다. 이러한 문제를 개발환경에서 막고자 메모리 누수를 막기 위해 동적 할당된 변수의 정보를 관리하는 프로그램을 만들어야 한다.

6. 모니터링 툴

프로그램 명	모니터링 툴								
플랫폼	PC								
프로그램 화면									
 <p>The screenshot displays the WinGraphVer2 monitoring tool interface. It features several charts and graphs: <ul style="list-style-type: none"> CPU 사용률: A line graph showing CPU usage percentage over time, fluctuating between approximately 40% and 80%. 메모리 사용률: A line graph showing memory usage percentage, fluctuating between approximately 20% and 60%. 네트워크 사용률: A line graph showing network usage percentage, fluctuating between approximately 20% and 100%. 프로세스: A bar chart showing the usage of various processes. The 'System' process is highlighted with a value of 39, and the 'smss.exe' process is highlighted with a value of 24. 동시접속자 수: A line graph showing the number of concurrent users, fluctuating between approximately 20 and 100. 서버동접비율: A pie chart showing the distribution of server connections. The data is as follows: <table border="1"> <thead> <tr> <th>Category</th> <th>Percentage</th> </tr> </thead> <tbody> <tr> <td>System</td> <td>21%</td> </tr> <tr> <td>smss.exe</td> <td>23%</td> </tr> <tr> <td>Other processes</td> <td>19%, 8%, 6%, 19%</td> </tr> </tbody> </table> </p>		Category	Percentage	System	21%	smss.exe	23%	Other processes	19%, 8%, 6%, 19%
Category	Percentage								
System	21%								
smss.exe	23%								
Other processes	19%, 8%, 6%, 19%								
소개	1. 모니터링 툴로 서버의 상황을 파악하자. 3. 여러개의 자식윈도우를 생성하여 여러 가지 도표를 표현 하자. 2. 위험한 상황까지 가게 되면 경보음과 부모 윈도우의 색상이 바뀐다.								
개발 환경	WinAPI								
설명	<p>서버의 종류는 무척 많습니다. 채팅서버, 게임서버, DB서버등 그 중 하나로 각 서버의 상황을 파악할 수 있는 모니터링 서버에서 사용 될 프로그램을 만들었습니다. 윈도우 API의 숙지를 목적으로 만들었으며 각 자식윈도우들은 하나의 도표를 표시합니다.</p>								

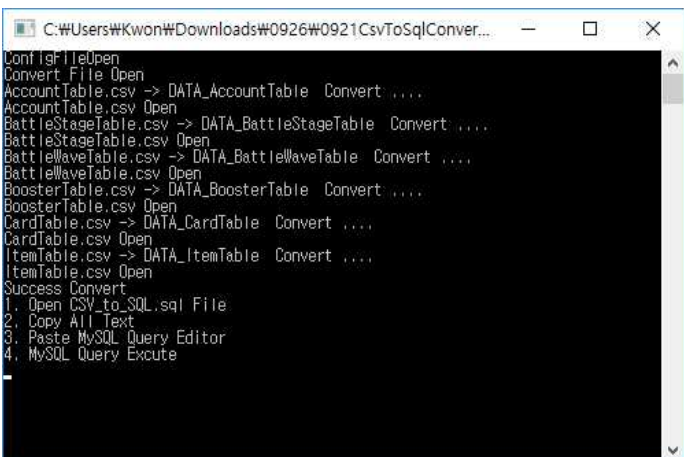

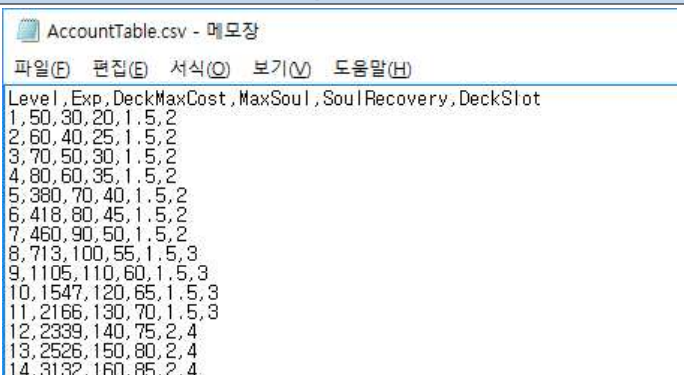
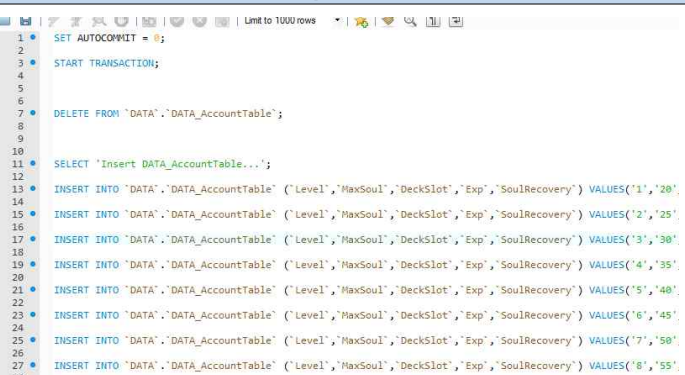
7. A* 알고리즘 툴

프로그램 명	A* 알고리즘 툴
플랫폼	PC
프로그램 화면	
<div> <div> <div>0629AStar찾기</div> <div>파일(F) 도움말(H)</div> </div> <div> </div> </div>	
소개	<ol style="list-style-type: none"> 1. WinAPI로 A*알고리즘을 구현하자 2. 오른쪽 버튼으로 처음과 시작을 지정할 수 있습니다. 3. 왼쪽 버튼으로 장애물을 설치합니다. 4. ESC로 화면을 지웁니다.
개발 환경	WinAPI
설명	A*알고리즘을 이용하여 WinAPI로 UI환경의 툴을 만들었습니다. 이 프로그램을 더 확장하여 맵 에디터를 만들 수 있을 것입니다.

8. JumpSearch 알고리즘 툴

프로그램 명	JumpSearch 알고리즘 툴
플랫폼	PC
프로그램 화면	
	
소개	<ol style="list-style-type: none"> 1. WinAPI로 A*알고리즘보다 빠른 JumpSearch 알고리즘을 구현하자. 2. 오른쪽 버튼으로 처음과 시작을 지정할 수 있습니다. 3. 왼쪽 버튼으로 장애물을 설치합니다. 4. ESC로 화면을 지웁니다.
개발 환경	WinAPI
설명	<p>A*알고리즘보다 빠른 JumpSearch알고리즘입니다. 노드를 건너뛰며 OpenList에 추가되는 로직이 A*알고리즘에 추가되었으며 나머지 부분은 A*알고리즘과 거의 같습니다.</p>

9. CsvToSqlConverter 툴

프로그램 명		CsvToSqlConverter 툴
플렛 폼		PC
프로그램 화면		
		
실행 화면		설정 파일
		
Csv파일		변경된 Sql파일
소개	1. 설정파일에서 컬럼을 골라 알맞은 쿼리문을 만들자. 2. Csv의 컬럼순서와 설정파일의 컬럼순서는 동일하지 않아도 됩니다. 3. 클라이언트DB쿼리용 서버DB쿼리용 Sql을 만들 수 있게 컬럼 명만 같다면 순서에 맞게 생성됩니다.	
개발 환경	WinConsole	
설명	MySql의 배치프로그램을 사용하여도 좋지만 유동적인 면에 있어서는 제가 만든 이 툴이 더 효율적입니다. Converter tool을 만들어 기획자가 만든 데이터 Csv파일을 서버프로그래머가 효율적으로 DB작업을 할 수 있게 해줍니다.	

10. 링크

프로젝트 파일 및 실행파일:

<https://drive.google.com/file/d/0BzusZZDBAoL3ODNjR0Z6dXpJUE0/view?usp=sharing>