**게임개발자전문가과정**

**과 목 명 : 게임기초프로그래밍**

**능력단위 : C/C++ 프로그래밍**

**제출일자 : 2022년 07월 26일**

**포트폴리오 : 파일 입출력 및 템플릿을 적용한 학생관리 프로그램 제작**

**작 성 자 : 김동완**

**<제출내역>**

1. **학생 관리 프로그램 구현 프로젝트**
2. **학생 관리 프로그램 구현 분석 및 세부 문서**

|  |
| --- |
| **결과사진 1 장** |

**< Contents >**

1. **프로젝트 소개 및 개요**
2. 프로젝트 소개
3. 프로젝트 주요 기술
4. **프로젝트 설계 및 다이어그램**
5. 시퀀스 다이어그램
6. 클래스 다이어그램

2-1) 프로젝트에서 사용된 변수 역할

2-2) 프로젝트에서 사용된 함수 역할

1. **최종 결과 및 추가 내용**
2. 범용성/유연성/확장성/간결성 고려하여 추가된 내용
   1. 현재 프로젝트 설계(구현 내용)
   2. 향후 개발 내용
3. 최종 결과
4. **프로젝트 소개 및 개요**
5. **프로젝트 소개**

대상 프로젝트는 파일 입출력 및 템플릿을 적용하여 학생 관리 프로그램을 제작한다. 프로젝트를 개발하면서 C/C++의 기능을 이용하여 데이터를 텍스트 파일에 블록 단위로 저장하고 읽어오는 방법을 익힌다. 학생의 정보를 관리할 때 사용되는 자료구조로 연결리스트를 사용하여 자료구조를 익힌다. 또 연결리스트를 구현하는데 포인터를 사용하여 프로세스와 메모리 사이의 관계를 알 수 있다. 범용성과 확장성을 위하여 연결리스트를 템플릿화하여 구축한다.

1. **프로젝트 주요 기술**

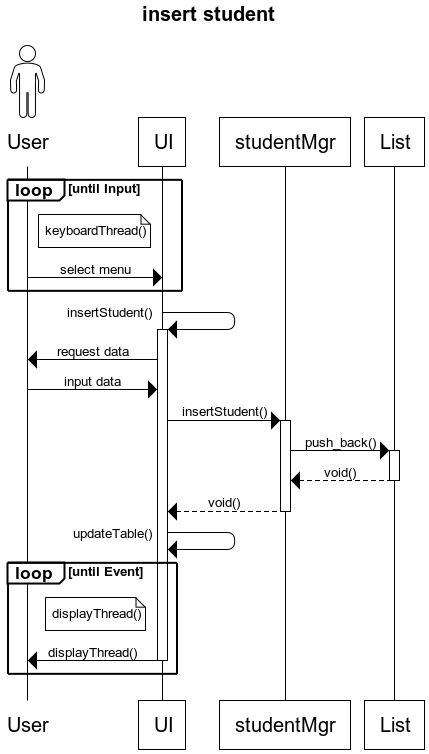
* **파일 입출력**

C에서는 두 가지의 파일 유형을 갖는다. 텍스트 파일과 바이너리 파일이다. 텍스트 파일은 말 그대로 우리가 읽을 수 있는 글자 그대로 파일에 저장된다. 텍스트 파일의 장점은 파일을 열었을 시 모든 사람이 읽을 수 있다는 장점이 있다. 하지만 텍스트 파일의 단점은 바이너리 파일보다 용량이 더 크며 외부에 노출되면 안되는 내용이 존재할 때 문제가 발생한다. 따라서 해당 프로젝트에서는 바이너리 형태로 파일을 저장하였으며 블록 단위 파일 입출력 함수인 fread(), fwrite()을 사용하여 구현하였다.

* **템플릿 연결리스트**

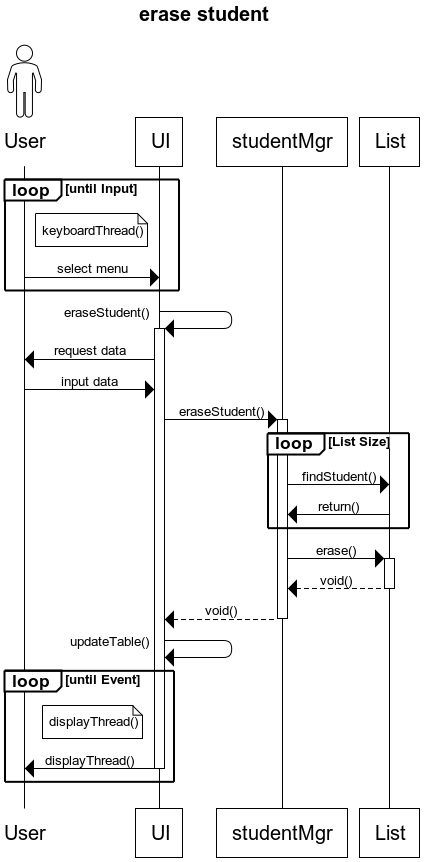
템플릿(Template)은 C++ 프로그래밍 언어의 한 기능으로, 함수와 클래스가 제네릭 형과 동작할 수 있게 도와 준다. 함수나 클래스가 개별적으로 다시 작성하지 않고도 수많은 자료형에서 동작할 수 있게한다. 따라서 템플릿 기능을 적용한다면 범용성을 높일 수 있다. 연결리스트 클래스를 템플릿화 하여 구축함으로써 다른 프로젝트에서 연결리스트가 필요할 경우 새로 구현할 필요 없이 구현한 연결리스트 스크립트를 적용하여 사용할 수 있다. 연결리스트는 Head와 Tail포인터를 사용한 단방향 연결리스트로 구현하였으며 두 개의 포인터 변수를 이용하여 추가/삭제/수정/탐색 등 용이하게 이용할 수 있다.

1. **프로젝트 설계 및 다이어그램**
2. **시퀀스 다이어그램(Sequence Diagram)**
   1. **Insert Student**

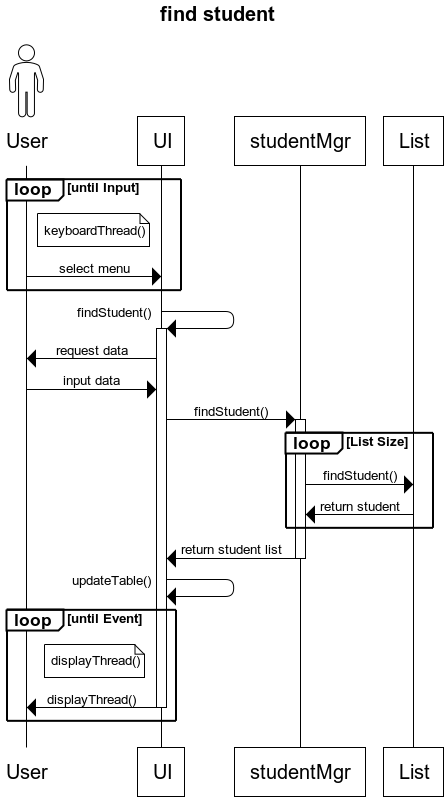
****

[그림 1-1] 시퀀스 다이어그램(Sequence Diagram)

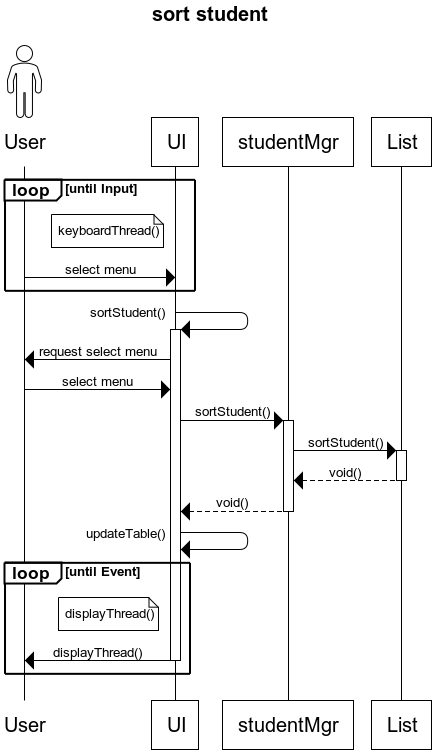
* 1. **Erase Student**

****

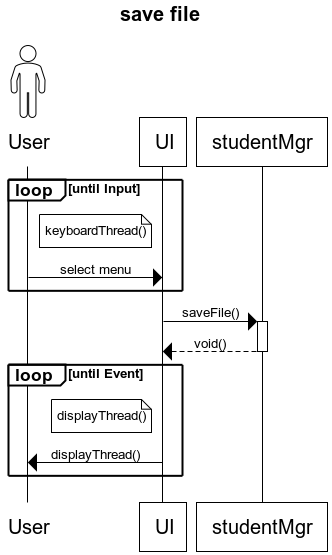
* 1. **Find Student**

****

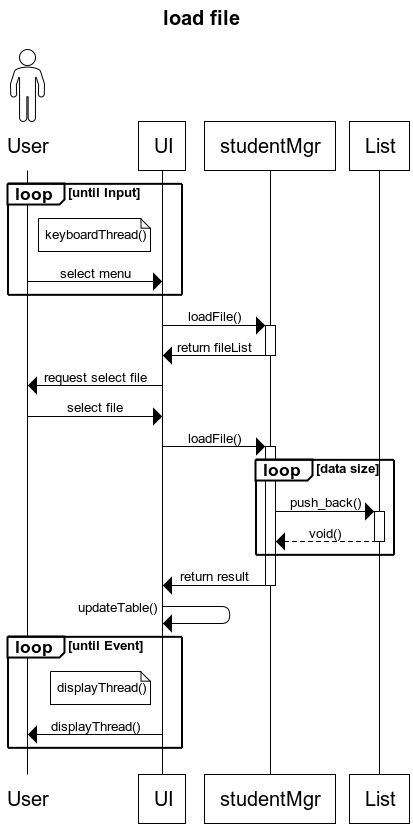
* 1. **Sort Student**

****

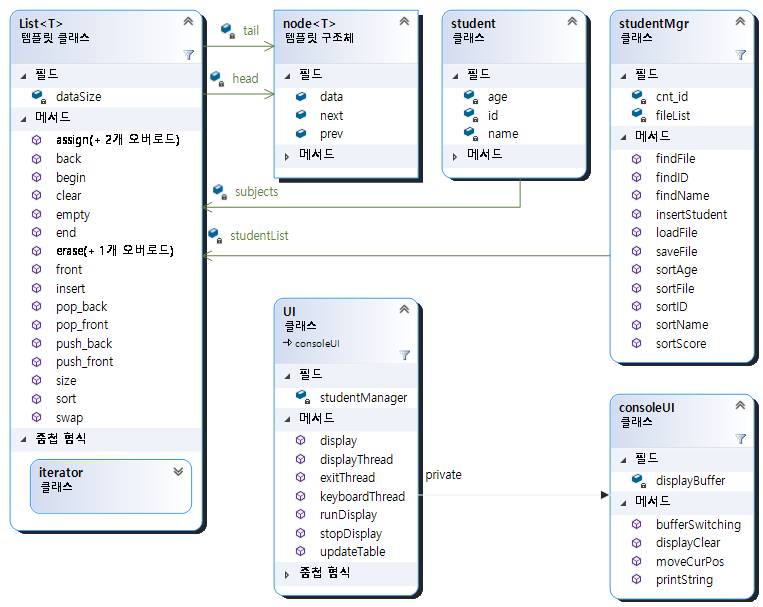
* 1. **Save File**

****

* 1. **Load File**

****

1. **클래스 다이어그램(Class Diagram)**



[그림 1-2] 클래스 다이어그램(Class Diagram)

**2-1) 프로젝트에서 사용된 변수 역할**

node의 변수

|  |  |
| --- | --- |
| 이름 | 설명 |
| T data | 데이터 변수 |
| Node<T>\* prev | 이전 노드를 위한 포인터 변수 |
| Node<T>\* next | 다음 노드를 위한 포인터 변수 |

List의 변수

|  |  |
| --- | --- |
| 이름 | 설명 |
| Node<T>\* head | List의 처음을 가리키는 포인터 변수. |
| Node<T>\* tail | List의 끝을 가리키는 포인터 변수. |
| size\_t dataSize | List에 저장된 노드의 개수 |

.

List의 함수

|  |  |
| --- | --- |
| 이름 | 설명 |
| void assign(int \_count, T& \_data = 0) | \_count: 생성 개수.  \_data: 데이터. (default = 0)  생성 개수만큼 데이터 참조 생성. |
| void assign(List& \_list) | \_list: 참조 리스트.  참조 리스트 복사 후 생성. |
| void assign(iterator \_first, iterator \_last) | \_first: 시작 위치 반복자.  \_end: 마지막 위치 반복자.  시작 위치부터 마지막 위치까지의 데이터 복사 후 생성. |
| T& back() | 마지막 요소 반환. |
| iterator begin() | 첫 요소 위치의 반복자 반환. |
| const iterator cbegin() | 첫 요소 위치의 const 반복자 반환. |
| iterator end() | 마지막 요소 다음 위치의 반복자 반환. |
| const iterator cend() | 마지막 요소 다음 위치의 const 반복자 반환. |
| void clear() | 리스트 내의 노드 전부 삭제. |
| void erase(iterator \_iter) | \_iter: 위치 반복자.  반복자 위치의 요소 삭제. |
| void erase(iterator \_first, iterator \_last) | \_first: 시작 위치의 반복자.  \_last: 마지막 위치의 반복자.  시작 위치부터 마지막 위치까지의 요소 삭제. |
| T& front() | 첫 요소 반환. |
| void insert(iterator \_iter, T& \_data) | \_iter: 위치 반복자.  \_data: 데이터.  반복자 위치에 데이터 삽입. |
| void pop\_back() | 마지막 요소 삭제. |
| void pop\_front() | 첫 요소 삭제. |
| void push\_back(T& \_data) | \_data: 데이터.  리스트 맨 뒤에 데이터 삽입. |
| void push\_front(T& \_data) | \_data: 데이터.  리스트 맨 앞에 데이터 삽입. |
| void swap(iterator \_iter1, iterator \_iter2) | \_iter1: 첫 번째 반복자.  \_iter2: 두 번째 반복자.  첫 번째 반복자 위치의 요소와 두 번째 반복자 위치의 요소 교환. |
| void sort(iterator \_first, iterator \_last, std::function<bool(T&, T&)> \_comparisonFunc) | \_first: 시작 위치 반복자.  \_last: 마지막 위치 반복자.  \_comparisonFunc: 비교 함수  시작 위치부터 마지막 위치까지의 데이터 정렬. |
| bool empty() | 리스트가 비어있는지 확인 후 반환. |
| size\_t size() | 리스트의 요소 개수 반환. |

student의 변수

|  |  |
| --- | --- |
| 이름 | 설명 |
| int id | 고유 ID 변수. |
| std::string name | 이름 변수. |
| int age | 나이 변수. |
| List<subject> subjects | 과목과 점수를 저장하는 리스트. |

studentMgr의 변수

|  |  |
| --- | --- |
| 이름 | 설명 |
| int cnt\_id | 고유 ID 부여 횟수. |
| std::vector<std::string> fileList | 파일 이름 저장 컨테이너. |
| List<student> studentList | 학생 정보를 저장하는 리스트. |

studentMgr의 함수

|  |  |
| --- | --- |
| 이름 | 설명 |
| bool findFile(std::string \_extension = ".csv"); | \_extension: 확장자.  현재 위치에서 입력 받은 확장자의 파일 탐색. |
| bool saveFile(std::string \_name) | \_name: 파일 이름.  파일 생성 후 학생 정보 저장. (csv 포맷) |
| bool loadFile(std::string \_name) | \_name: 파일 이름.  파일에서 학생 정보 읽어 온 후 리스트에 삽입. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**2-2) 프로젝트에서 사용된 함수 역할**

[표 1-2] 프로젝트의 클래스 함수

1. **최종 결과 및 추가 내용**
2. **범용성/유연성/확장성/간결성 고려하여 추가된 내용**
   1. **현재 프로젝트 설계(구현 내용)**
3. 현재 연결리스트 클래스는 어떤 프로그램에서도 사용할 수 있도록 템플릿화 하여 어떤 클래스도 객체로 가질 수 있음
4. 파일을 읽거나 쓸 때 블록단위로 저장하여 학생 한명의 데이터를 한번에 처리 가능
5. 파일 입출력 시 데이터가 노출이 되지 않도록 바이너리 파일로 저장
   1. **향후 개발 내용**
6. 현재 연결리스트에는 추가, 전체 삭제, 전체 출력, 저장, 불러오기 기능만 있지만 추후 학생 한명 찾기, 특정 학생 데이터 삭제 등의 기능 추가 예정
7. 단방향 연결리스트를 양방향 연결리스트로 구축하여 연결리스트의 관리를 좀 더 쉽도록 구현
8. 프로그램이 실행 중에 OS와 충돌이 나 갑작스럽게 셧다운이 될 수 있으므로 타이머를 추가하여 특정 시간마다 자동으로 저장하는 기능을 개발
9. **최종 결과**

텍스트, 전자기기, 스크린샷이(가) 표시된 사진

자동 생성된 설명