**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 : 신승빈**

**<제출내역>**

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

|  |
| --- |
|  |

**< 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**

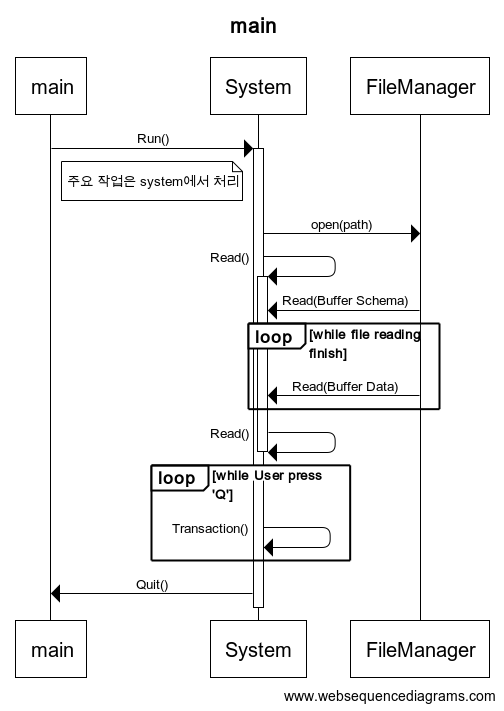
* **파일 입출력**

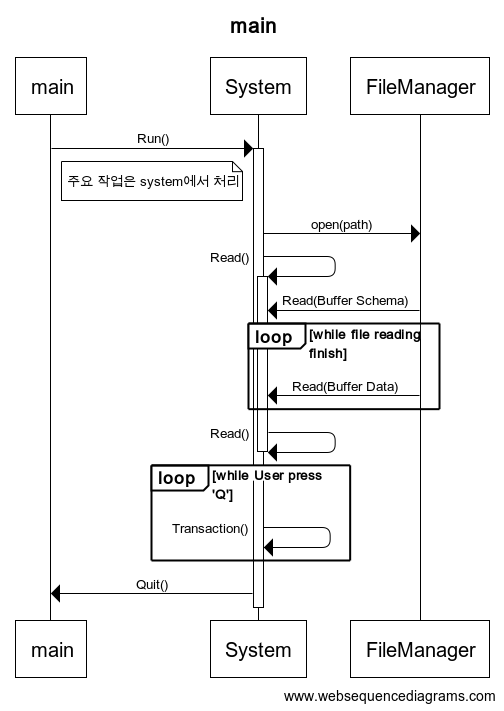
C에서는 두 가지의 파일 유형을 갖는다. 텍스트 파일과 바이너리 파일이다. 텍스트 파일은 말 그대로 우리가 읽을 수 있는 글자 그대로 파일에 저장된다. 텍스트 파일의 장점은 파일을 열었을 시 모든 사람이 읽을 수 있다는 장점이 있다. 하지만 텍스트 파일의 단점은 바이너리 파일보다 용량이 더 크며 외부에 노출되면 안되는 내용이 존재할 때 문제가 발생한다. 따라서 해당 프로젝트에서는 바이너리 형태로 파일을 저장하였으며 블록 단위 파일 입출력 함수인 fread(), fwrite()을 사용하여 구현하였다.

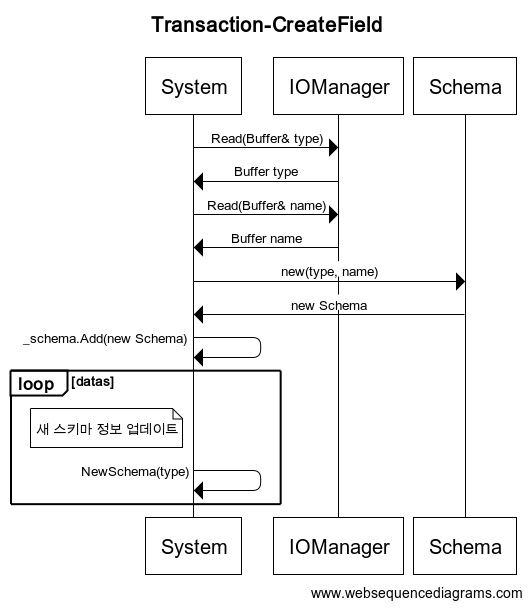
* **템플릿 연결리스트**

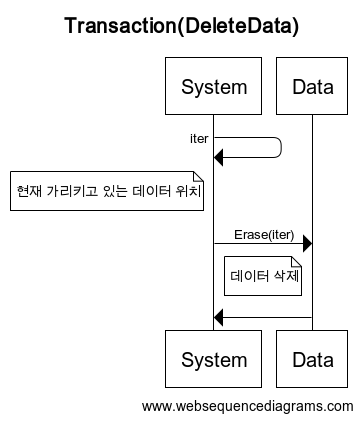
템플릿(Template)은 C++ 프로그래밍 언어의 한 기능으로, 함수와 클래스가 제네릭 형과 동작할 수 있게 도와 준다. 함수나 클래스가 개별적으로 다시 작성하지 않고도 수많은 자료형에서 동작할 수 있게한다. 따라서 템플릿 기능을 적용한다면 범용성을 높일 수 있다. 연결리스트 클래스를 템플릿화 하여 구축함으로써 다른 프로젝트에서 연결리스트가 필요할 경우 새로 구현할 필요 없이 구현한 연결리스트 스크립트를 적용하여 사용할 수 있다. 연결리스트는 Head와 Tail포인터를 사용한 단방향 연결리스트로 구현하였으며 두 개의 포인터 변수를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

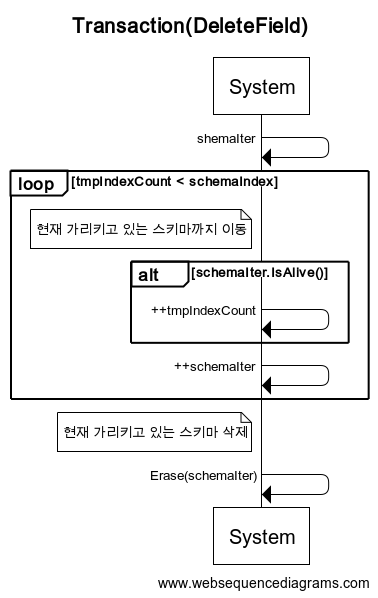
1. **프로젝트 설계 및 다이어그램**
2. **시퀀스 다이어그램(Sequence Diagram)**

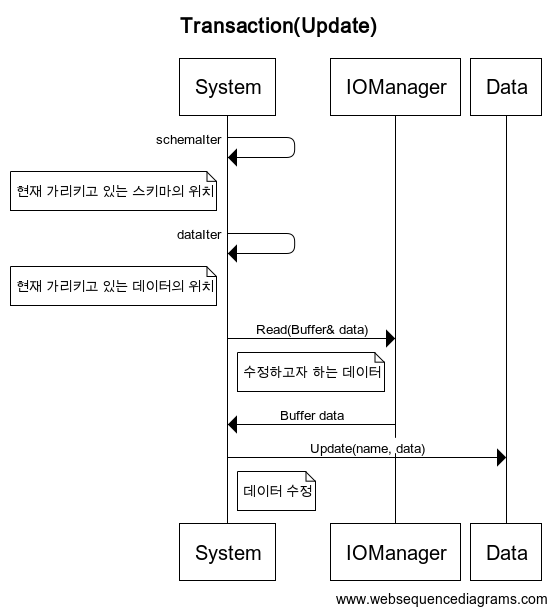
****

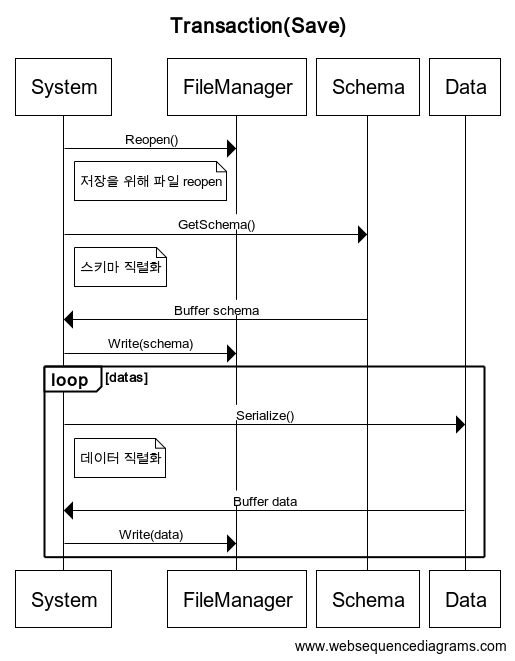
****

****

****

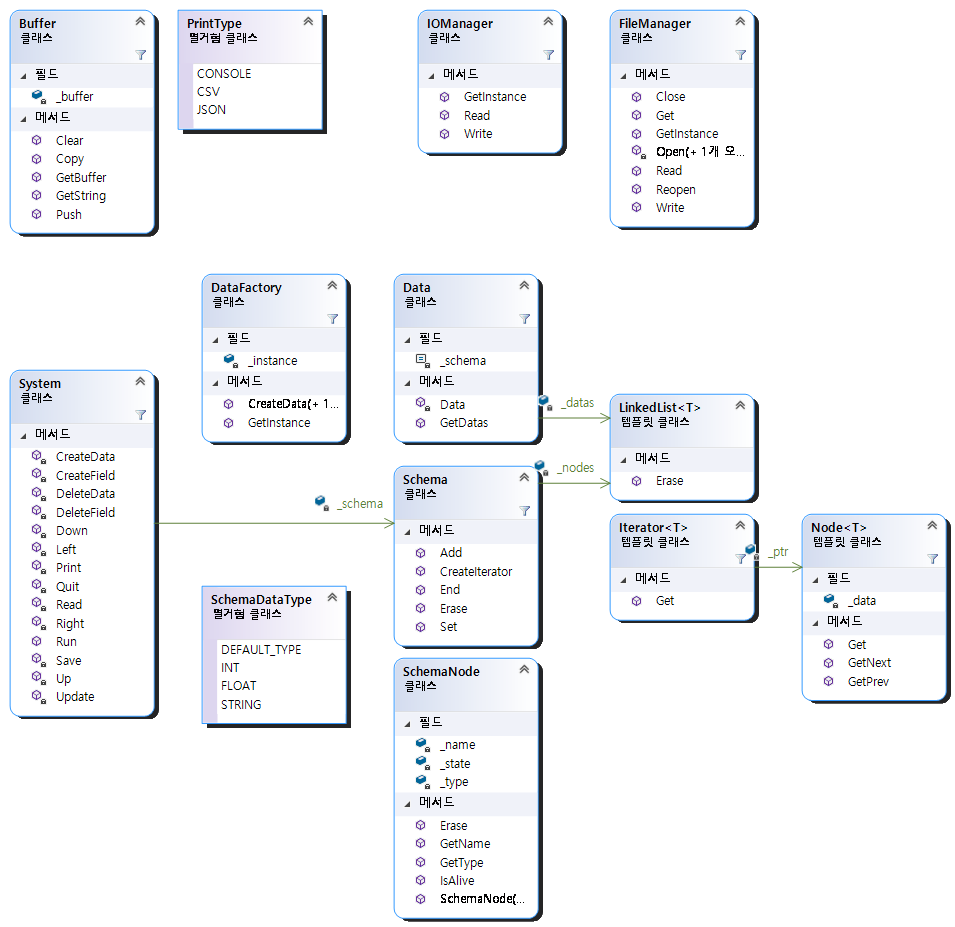
****

****

****

[그림 1-1] 시퀀스 다이어그램(Sequence Diagram)

1. **클래스 다이어그램(Class Diagram)**

****

[그림 1-2] 클래스 다이어그램(Class Diagram)

**2-1) 프로젝트에서 사용된 변수 역할**

|  |  |  |
| --- | --- | --- |
| **클래스명** | **변수** | **내용** |
| **Node** | T\* data | 노드에 저장할 데이터 변수 |
| Node<T>\* next | 다음 노드를 가리키기 위한 노드 포인터 변수 |
| **Linked**  **List** | Node<T>\* m\_pHead | 연결리스트의 첫번째 노드를 가리키는 노드 포인터 변수 |
| Node<T>\* m\_pTail | 연결리스트의 마지막 노드를 가리키는 노드 포인터 변수 |
| int m\_iNodeCount | 연결리스트의 저장된 노드의 개수 |
| **Student** | int m\_iIndex | 학생 번호 |
| char m\_csName[64] | 학생 이름 |
| int m\_iKor | 국어 점수 |
| int m\_iEng | 영어 점수 |
| int m\_iMath | 수학 점수 |
| int m\_iTotal | 국영수 합산 점수 |
| float m\_fAverage | 국영수 평균 점수 |
| char m\_szBuffer[1000] | 파일 입출력에 사용될 버퍼 |
| int m\_iBufferSize | 버퍼에 저장된 데이터의 크기 |
| Int m\_iCurrentPosition | 버퍼가 사용된 위치 |
| **Student**  **Manager** | LinkedList<Student> m\_StudentList | 학생 데이터를 저장할 연결리스트 |
| FileIO m\_FileIO | 학생 정보를 파일에 담기 위한 FileIO 클래스 |
| **FileIO** | FILE\* m\_pStream | 파일 입출력에 사용할 스트림 |

[표 1-1] 프로젝트의 클래스 변수

**2-2) 프로젝트에서 사용된 함수 역할**

|  |  |  |
| --- | --- | --- |
| **클래스명** | **함수** | **내용** |
| **Node** | void SetData(T\* data) | 노드의 데이터를 저장하는 함수 |
| void SetNext(Node<T>\* nextNode) | 현재 노드의 노드 포인터에 다음 노드를 저장하는 함수 |
| T\* GetData() | 노드의 데이터를 반환하는 함수 |
| Node<T>\* GetNext() | 현재 노드와 연결된 노드를 반환하는 함수 |
| **LinkedList** | void AddNode(T\* data) | 연결리스트에 노드를 추가하는 함수 |
| void DeleteAll(T\* data) | 연결리스트에 저장된 모든 노드를 삭제하는 함수 |
| Node<T>\* GetHead() | 연결리스트의 첫 번째 노드를 반환하는 함수 |
| Node<T>\* GetTail() | 연결리스트의 마지막 노드를 반환하는 함수 |
| int GetNodeCount() | 연결리스트에 저장된 노드의 수를 반환하는 함수 |
| **Student** | void SetData(int iIndex=-1) | 학생의 모든 데이터를 입력받고 계산 |
| void Show() | 학생의 데이터를 출력하는 함수 |
| void CheckSubjectScore(ESubject subject, int& data) | 예외 처리 함수로 국어/영어/수학 점수를 잘못 입력(0>점수 && 100<점수)하였을 때 다시 입력받도록 해주는 함수 |
| friend std::ostream& operator<<(std::ostream& os, Student& data) | 학생 정보를 간단하게 출력하기 위해 출력 할 때 사용하는 연산자 재정의 |
| **Student**  **Manager** | void Create() | 새로 추가되는 학생을 생성하고 데이터를 채운 후 연결리스트에 추가하는 함수 |
| void Delete() | 연결리스트의 모든 노드를 제거하는 함수 |
| void Show() | 저장된 모든 학생 정보를 출력하는 함수 |
| bool Save() | 저장된 모든 학생 정보를 파일로 저장하는 함수 |
| bool Load() | 이전에 저장되었던 학생 정보가 들어있는 파일을 읽어온 후 연결리스트에 추가하는 함수 |
| **FileIO** | bool CreateFile(const char\* pFileName) | 파일 이름을 매개변수로하여 해당 파일을 “쓰기용”으로 여는 함수 |
| bool OpenFile(const char\* pFileName) | 파일 이름을 매개변수로하여 해당 파일을 “읽기용”으로 여는 함수 |
| void CloseFile() | 파일을 닫는 함수 |
| size\_t Write(void const\* pData, int iSize) | 데이터를 블록단위로 입력받아 파일에 데이터를 작성하는 함수 |
| size\_t Read(void\* pData, int iSize) | 데이터를 블록단위로 입력받아 파일에 데이터를 읽어오는 함수 |
| FILE\* GetStream() | 파일스트림을 반환하는 함수 |

[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