**게임알고리즘기초과정**

**과 목 명 : 게임기초프로그래밍**

**능력단위 : C/C++ 프로그래밍**

**제출일자 : 2023년 07월 04일**

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

**작 성 자 : 김경연**

**<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 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**

내가 생성한 데이터를 프로그램이 종료 하기전에 저장하지 않는다면 내가 데이터를 생성한 의미를 잃어버린다. 이를 위해 C에서 제공하고 있는 File 포인터와 파일 입출력 함수인 fscanf()와 fprint()를 통해 데이터의 입력과 출력을 구현함으로써 내가 생성한 데이터를 저장할 수 있고, 프로그램이 종료되고 다시 프로그램을 시작하였을 때 그 데이터를 불러와 내가 작업 하던 일을 이어서 할 수 있게 하였다.

* **템플릿 연결리스트**

템플릿(Template)은 C++ 프로그래밍 언어의 한 기능으로, 함수와 클래스가 제네릭 형과 동작할 수 있게 도와준다. 함수나 클래스가 개별적으로 다시 작성하지 않고도 수많은 자료형에서 동작할 수 있게 한다. 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 Tail 객체 변수를 사용한 양방향 연결리스트로 구현하였으며 두 개의 노드 객체 변수를 이용하여 추가/삭제/수정/탐색/정렬/삽입/출력 등 용이하게 이용할 수 있다.

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

**텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명**

텍스트, 스크린샷, 도표, 평행이(가) 표시된 사진

자동 생성된 설명

[그림1-1 ,1-2] 시퀀스 다이어그램(Sequence Diagram)

**텍스트, 도표, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 도표, 평행, 스크린샷이(가) 표시된 사진

자동 생성된 설명**

[그림1-3 ,1-4] 시퀀스 다이어그램(Sequence Diagram)

**텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명**

[그림1-5 ,1-6] 시퀀스 다이어그램(Sequence Diagram)

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

[그림1-5 ,1-6] 시퀀스 다이어그램(Sequence Diagram)

텍스트, 도표, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

텍스트, 도표, 스크린샷, 평행이(가) 표시된 사진

자동 생성된 설명

[그림1-7 ,1-8] 시퀀스 다이어그램(Sequence Diagram)

**클래스 다이어그램(Class Diagram)**

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

[그림 1-9] 클래스 다이어그램(Class Diagram)

**2-1) 프로젝트에서 사용된 변수 역할**

|  |  |  |
| --- | --- | --- |
| **클래스명** | **변수** | **내용** |
| **NodeBox** | T\* nb\_Data | 노드에 저장할 데이터 변수 |
| NodeBox<T>\* nb\_Next | 다음 노드를 가리키기 위한 노드 포인터 변수 |
| NodeBox<T>\* nb\_Prev | 이전 노드를 가리키기 위한 노드 포인터 변수 |
| **KLinked**  **List** | NodeBox<T> head | 연결리스트의 첫번째 기준이 되는 시작점 노드 변수 |
| NodeBox<T> tail | 연결리스트의 마지막 기준이 되는 끝점 노드 변수 |
| int iCounter | 연결리스트의 저장된 노드의 개수 |
| **KStudent** | int m\_ID | ID값(유일함) |
| char m\_Name[20] | 학생 이름 |
| int m\_Kor | 국어 점수 |
| int m\_Eng | 영어 점수 |
| int m\_Math | 수학 점수 |
| int m\_Total | 국영수 합산 점수 |
| float m\_Average | 국영수 평균 점수 |
| **KPrint** | KLinkedList<T>\* KPrintList | KLinkedList객체를 가리키는 포인터 변수 |
| **KFind** | LinkedList<T>\* KFindList | KLinkedList객체를 가리키는 포인터 변수 |
| **KInsert** | LinkedList<T>\* KInsertList | KLinkedList객체를 가리키는 포인터 변수 |
| **KSort** | LinkedList<T> KSortList | KLinkedList객체를 복사한 객체 변수 |
| **KUpdate** | LinkedList<T>\* KUpdateList | KLinkedList객체를 가르기는 포인터 변수 |
| **FileIO** | FILE\* fp | 파일 입출력에 사용할 스트림 |
| **KStudent**  **Manager** | KLinkedList<T> KlinkedList | KLinkedList클래스의 객체 변수 |
| KPrint<T> Kprint | KPrint클래스의 객체 변수 |
| KInsert<T> Kinsert | KInsert클래스의 객체 변수 |
| KUpdate<T> Kupdate | KUpdate클래스의 객체 변수 |
| KFind<T> Kfind | KFind클래스의 객체 변수 |
| KSort<T> Ksort | KSort클래스의 객체 변수 |

[표 1-1] 프로젝트의 클래스 변수

**2-2) 프로젝트에서 사용된 함수 역할**

|  |  |  |
| --- | --- | --- |
| **클래스명** | **함수** | **내용** |
| **KLinkedList** | NodeBox<T>\* NewNode() | 연결리스트에 노드를 생성해 반환하는 함수 |
| void PushHead(NodeBox<T>\* newNode) | 생성 한 노드를 Head의 Next 노드에 연결해주는 함수 |
| void PushTail(NodeBox<T>\* newNode) | 생성 한 노드를 Tail의 Prev 노드에 연결해주는 함수 |
| void DeleteNode(int nodeNum) | Head를 기준으로 몇번째에 있는 노드를 삭제하는 함수 |
| void DeleteAllNode() | 연결리스트에 저장된 모든 노드를 삭제하는 함수 |
| **KStudent** | void ValueSetting() | 학생의 모든 데이터를 입력받고 계산해주는 함수 |
| void Print() | 학생의 데이터를 출력하는 함수 |
| void TotalAvgCompute() | 국어/영어/수학의 총점, 평균을 계산 해주는 함수 |
| void Read() | fscanf를 이용해 데이터 입력 |
| void Save() | fprint를 이용해 데이터 출력 |
| **Student**  **Manager** | KStudent\*<T>\* NewStudentData() | 학생이라는 데이터를 동적할당해서 값을 세팅하고 반한해주는 함수 |
| void run() | 학생 관리프로그램을 실행 시켜주는 함수 |
| void SaveFile () | 저장된 모든 학생 정보를 파일로 저장하는 함수 |
| void LoadFile() | 이전에 저장되었던 학생 정보가 들어있는 파일을 읽어온 후 연결리스트에 추가하는 함수 |
| **FileIO** | bool CreateFile(const char\* pFileName, const char\* filemode) | 파일 이름을 매개변수로하여 해당 파일을 “쓰기용”으로 여는 함수 |
| void CloseFile() | 파일을 닫는 함수 |
| **KPrint** | void PrintFront() | Head의 Next노드부터 시작하여 전체 노드를 출력해주는 함수 |
| void PrintBack() | Tail의 Prev노드부터 시작하여 전체 노드를 출력해주는 함수 |
| void PrintFront(NodeBox<T>\* startNode) | Head의 Next노드부터 시작하여 지정된 노드까지 출력해주는 함수 |
| void PrintBack(NodeBox<T>\* startNode) | Tail의 Prev노드부터 시작하여 지정된 노드까지 출력해주는 함수 |
| **KFind** | NodeBox<T>\* find(int id) | id(유일한 값)를 기준으로 노드를 찾아서 반환 해주는 함수 |
| NodeBox<T>\* find(const char\* name) | 이름을 기준으로 노드를 찾아서 반환해주는 함수 |
| **KInsert** | void PushInsert(NodeBox<T>\* newNode, int nodeNum) | Head를 기준으로 몇번째 노드에 새로운 노드를 삽입(연결)해주는 함수 |
| **KSort** | void AscendSortNode() | 버블정렬을 이용하여 총점을 기준으로 오른차순 해주는 함수 |
| void DescendSortNode() | 버블정렬을 이용하여 총점을 기준으로 내림차순 해주는 함수 |
| **KUpdate** | void UpdateNode(int id) | id(유일한 값)를 기준으로 노드를 찾아서 데이터를 수정해주는 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 있다.
4. 연결리스트 클래스의 기능은 최소한으로 구성하고 추가적으로 필요한 기능들은 다른 클래스에서 한 클래스당 하나의 기능을 할 수 있게 구현하였다.
5. 따로 구현한 KSort클래스의 경우 정렬을 실행해도 원본은 유지하고 정렬된 데이터는 따로 저장할 수 있도록 구현하였다.
6. FileIO를 사용하여 데이터를 저장하고 필요시에 불러올 수 있다.
   1. **향후 개발 내용**
7. 현재 예외처리가 단순하고 체계화되어 있지 않습니다. Exception클래스를 활용하거나 예외처리기능을 직접 클래스로 구현하여 예외처리를 체계화 할 필요가 있어 보인다.
8. 기능을 분리하여 클래스를 구현하는 것도 좋지만 너무 연결리스트에 특화된 기능으로 분리하여 연결리스트 외에 다른 작업을 할 때에는 재사용 활용을 할 수 없다. 정렬, 탐색와 같은 기능은 연결리스트 외에 다른 작업에서도 재사용 할 수 있게 연결리스트와의 관계를 완전히 불리하는 작업이 필요해 보인다.
9. 클래스의 선언과 기능작업을 분리하여 헤더파일하고 .cpp로 따로 나눌 필요가 있어 보인다.
10. 반복적으로 여러 번 사용하는 로직들은 함수화 필요가 있어 보인다.
11. **최종 결과**

**2-1) 학생 생성**

텍스트, 전자제품, 스크린샷, 소프트웨어이(가) 표시된 사진

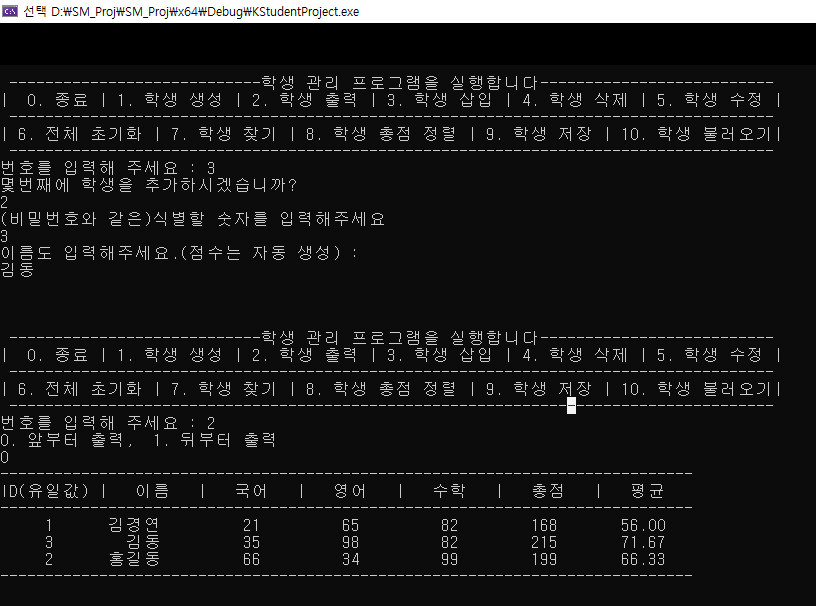
자동 생성된 설명

**2-2) 학생 출력**

텍스트, 스크린샷, 디스플레이, 번호이(가) 표시된 사진

자동 생성된 설명

**2-3) 학생 삽입**

****

**2-4) 학생 삭제**

텍스트, 스크린샷, 디스플레이, 번호이(가) 표시된 사진

자동 생성된 설명

**2-5) 학생 수정**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**2-6) 전체 초기화**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**2-7) 학생 찾기**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**2-8) 학생 총점 정렬**

텍스트, 스크린샷, 디스플레이, 번호이(가) 표시된 사진

자동 생성된 설명

**2-9) 학생 저장, 불러오기**

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명