자료구조 실습 01

Data Structures Lab 01

박유민
yumin0906@khu.ac.kr

Lab 01 : Unsorted List 구현 및 응용

◎ 목표:
- ☞ 배열을 이용한 Unsorted List를 구현 및 응용

◎ 내용:
- ☞ Array를 사용하여 Unsorted List를 구현
- ☞ Unsorted list의 내용을 순차적으로 검색할 수 있는 iterator 기능 구현
- ☞ 구현된 unsorted list를 이용하여 “실습과제 1” 에 정의된 응용 시스템을 구현

◎ 방법
- ☞ 예제에 정의된 Unsorted List와 이를 이용한 응용 프로그램 작성
- ☞ 첨부된 소스코드를 참조하여 예제를 완성
- ☞ 예제에 과제에서 제시한 기능들을 추가한다.
- ☞ 과제에서 구현한 프로그램을 수정하여 “실습과제 1”에 정의된 응용 시스템을 구현한다.

◎ 제출 내용
- ☞ 실습과제 1을 구현한 소스코드

예제: Array를 이용한 Unsorted List 구현

◎ 내용:

- ☞ 배열을 이용한 Unsorted List를 구현하고 이를 이용하여 학생의 신상기록을 관리하는 응용 프로그램을 작성한다.

◎ 방법:

- ☞ 배열을 이용한 Unsorted List Class 정의 및 구현
- ☞ 학생의 신상기록을 저장하기 위한 Item Class 정의 및 구현
- ☞ 신상기록을 키보드 입력, 화면 출력, 파일 입력, 파일 출력을 수행하는 구동 프로그램 작성.
- ☞ 예제 1의 해답은 실습자료와 함께 제공된다. 하지만 해답을 바로 보지 말고 스스로 프로그램을 작성하고 그 것을 검증하는 용도로 사용해야 한다. 그렇지 않고 바로 답부터 보면 실습에서 실력향상에 도움이 되지 않고 시험에서도 좋은 점수를 받을 수 없다

예제: Array를 이용한 Unsorted List 구현

◎ 다음과 같은 멤버변수와 멤버함수를 가지는 List class를 설계하고 구현한다.

☞ Domain:

- ItemType m_Array[MAXSIZE] // 레코드 배열
- int m_Length // 리스트에 저장된 레코드 수
- int m_CurPointer // current pointer

☞ Operations:

- void MakeEmpty() // 현재 레코드 모두 삭제
- int GetLength() // 현재 레코드 수 반환
- bool IsFull() // 모든 배열의 사용 여부
- int Add(ItemType data) // 새로운 레코드 추가
- void ResetList() // 레코드 포인터 초기화
- int GetNextItem(ItemType& data) // current pointer를 하나 증가시키고 끝이 아니면 record index를 리턴 끝이면 -1을 리턴

예제: Array를 이용한 Unsorted List 구현

```
#define MAXSIZE 5
class ArrayList
{
public:
        ArrayList();                    // default constructor
        ~ArrayList();                   // default destructor

        void MakeEmpty();               // Make list empty
        int GetLength();                // 레코드 수 반환
        bool IsFull();                  // 모든 배열의 사용 여부
        int Add(ItemType data);         // 새로운 데이터 추가
        void ResetList();               // 레코드 포인터(current pointer) 초기화
        int GetNextItem(ItemType& data);  // current pointer를 하나 증가시키고 끝이 아니면 record index를 리턴 끝이면 -1을 리턴
private:
        ItemType m_Array[MAXSIZE];      // 레코드 배열
        int m_Length;                   // 리스트에 저장된 레코드 수
        int m_CurPointer;               // current pointer
}
```

예제: List에 저장할 ItemType 정의 및 구현

◎ 학번, 이름, 주소로 구성되는 학생 기록을 저장할 ItemType Class를 정의하고 구현한다.

```
enum RelationType {LESS, GREATER, EQUAL};
class ItemType
{
public:
        ItemType();                                                    // default constructor
        ~ItemType();                                                   // default destructor

        int GetId();                                                   // 학생 ID 반환 함수
        string GetName();                                              // 학생 이름 반환 함수
        string GetAddress();                                           // 학생 주소 반환 함수
        void SetId(int inId);                                          // 학생 ID 저장 함수
        void SetName(string inName);                                   // 학생 이름 저장 함수
        void SetAddress(string inAddress);                             // 학생 주소 저장 함수
        void SetRecord(int inId, string inName, string inAddress);     // 학생 정보 저장 함수
        void DisplayIdOnScreen();                                      // 학생 ID 출력 함수
        void DisplayNameOnScreen();                                    // 학생 이름 출력 함수
        void DisplayAddressOnScreen();                                 // 학생 주소 출력 함수
        void DisplayRecordOnScreen();                                  // 학생 정보 출력 함수
```

예제: List에 저장할 ItemType 정의 및 구현

```
        void SetIdFromKB();                                  // 키보드로 학생 ID 입력 함수
        void SetNameFromKB();                                // 키보드로 학생 이름 입력 함수
        void SetAddressFromKB();                             // 키보드로 학생 주소 입력 함수
        void SetRecordFromKB();                              // 키보드로 학생 정보 입력 함수
        int ReadDataFromFile(ifstream& fin);                 // 학생 정보를 파일에서 읽는 함수
        int WriteDataToFile(ofstream& fout);                 // 학생 정보를 파일로 출력하는 함수
        RelationType CompareByID(const ItemType &data);      // primary key (ID)를 기준으로 학생 정보를 비교하는 함수

Private:
        int m_Id;                                            // 학생 ID
        string m_sName;                                      // 학생 이름 저장 변수
        string m_sAddress;                                   // 학생 주소 저장 변수
}
```

예제: 신상관리 시스템을 구동하기 위한 Application Class 정의 및 구현

◎ 다음과 같이 기능을 수행하는 구동 프로그램 작성

- ☞ Add item: 키보드로부터 한 사람의 기록을 읽어 list에 저장
- ☞ Print all on screen: list에 저장된 모든 기록을 화면에 출력
- ☞ Make empty list: list 비우기
- ☞ Get from file : 파일에 있는 모든 신상기록을 list에 loading
- ☞ Put to file : list에 있는 모든 신상기록을 파일로 출력

```
--- ID – Command -----
    1 : Add item
    2 : Print all on screen
    3 : Make empty list
    4 : Get from file
    5 : Put to file
    0 : Quit

Choose a Command --> _
```

예제: Application Class 정의 및 구현

```
class Application
{
public:
    void Run();
    int GetCommand();
    int AddItem();
    void DisplayAllItem();
    int OpenInFile(char *fileName);
    int OpenOutFile(char *fileName);
    int ReadDataFromFile();
    int WriteDataToFile();

private:
    ifstream m_InFile;///< input file descriptor.
    ofstream m_OutFile;///< output file descriptor.
    ArrayList m_List;///< item list.
    int m_Command;///< current command number.
};
```

예제: Application Class 정의 및 구현

```
void Application::Run()
{
while(1) {
    m_Command = GetCommand();
    switch(m_Command)
   {
       case 1:// read a record and add to list.
            AddItem();              break;
       case 2:// display all the records on screen.
            DisplayAllItem();   break;
       case 3:// make empty list.
            m_List.MakeEmpty();    break;
       case 4:// load list data from a file.
           ReadDataFromFile();    break;
       case 5:// save list data into a file.
           WriteDataToFile();    break;
      case 0:
          return;
     default:
        cout << "\tIllegal selection...\n";   break;
 }}}
```

```
// Display command on screen and get a input from
keyboard.
int Application::GetCommand()
{
        int command;
        cout << endl << endl;
        cout << "\t---ID -- Command ----- " << endl;
        cout << "\t   1 : Add item" << endl;
        cout << "\t   2 : Print all on screen" << endl;
        cout << "\t   3 : Make empty list" << endl;
        cout << "\t   4 : Get from file" << endl;
        cout << "\t   5 : Put to file " << endl;
        cout << "\t   0 : Quit" << endl;

        cout << endl << "\t Choose a Command--> ";
        cin >> command;
        cout << endl;

        return command;
}
```

실습 : 예제에 새로운 기능 추가

◎ 예제에서 구현한 List Class에 다음 멤버 함수를 추가한다.

- ☞ bool IsEmpty(); // 배열이 비었는지 여부
- ☞ int Get(ItemType& data);
  - ➢ // Primary key를 기준으로 데이터를 검색하고 해당 데이터를 가져옴
- ☞ int Delete(ItemType data); // 기존 레코드 삭제
- ☞ int Replace(ItemType data); // 입력된 data와 Primary key와 동일한 기록을 찾아서 List의 해당 기록을 입력된 data로 치환한다.

◎ *** List Class에서는 itemType class의 내용을 절대, 절대, 절대 몰라야 한다. 즉 List class는 저장될 내용과 무관하게 정의 되어야 한다. 이를 어기면 아주 큰 감점을 받게 된다.

실습 : 예제에 새로운 기능 추가

◎ List Class에 추가된 멤버함수를 테스트할 수 있는 기능을 application class에 추가한다.

☞ RetrieveStudent()

➢ // 키보드로부터 학생 ID를 받아서 List에서 학생의 기록을 찾아서 화면에 출력

☞ DeleteStudent()

➢ // 키보드로부터 학생 ID를 받아서 List에서 해당 학생을 삭제

☞ Replace()

➢키보드로부터 학생 기록을 받은 다음에 list에서 같은 학생의 기록을 찾아서 새로 입력된 기록으로 치환