C# 프로그래밍 프로젝트
최종 보고서

팀원
16011094 주이식
18010847 박상욱

파일 정렬 최적화 프로그램

보고서 개요

1. 프로그램 개발동기
2. 프로그램 개발계획
3. 프로그램 기능구현 방식 및 설명
4. 프로그램 소스 코드
5. 보완 및 개선점

프로그램 개발 동기

PC를 오래 사용하면 덤프 파일이 쌓이게 되면서 하드디스크의 용량이 모자라는 경우가 생깁니다.
용량을 많이 차지하는 프로그램을 삭제하는 것만으로는 용량 문제가 전부 해결되지 않기 때문에 사용자가 윈도우 파일 탐색기를 이용하여 용량이 큰 디렉토리를 확인하고 해당 디렉토리 내부 파일들의 크기를 일일이 비교해가면서 삭제해야하는 번거로움이 있습니다.
더불어 폴더와 파일을 검색하는데 시간이 많이 걸리고 용량을 확인해야할 경우 폴더의 속성 탭에 들어가야 하는 등 과정이 상당히 복잡한데 이 과정을 C# 라이브러리를 활용한 WinForm 프로그램을 개발하여 간편화시킬 수는 없을까라는 아이디어를 토대로 개발을 시작했습니다.
팀 프로젝트 예시처럼 C#을 공부한 것을 복습하는 기회로 삼고 이미 존재하는 프로그램들을 공부하는 방법도 있었지만 개발자로서 평소 불편함을 느끼던 것들을 해결하면서 프로그램을 만들게 된다면 더 의미가 있을거라 생각하여 실제로 사용이 가능한 파일 정렬 최적화 프로그램을 개발하게 됐습니다.

프로그램 개발계획

저희 팀은 깃허브를 이용하여 필요한 기능과 참고 문서 등 개발 진행 과정들을 기록했습니다.

날짜	주차	내용
4월 2주	1주차	팀 매칭
4월 3주	2주차	아이디어 기획 시작 나왔던 아이디어 : 파일 정렬 최적화 프로그램, 안테나와 시리얼통신을 기반으로 한 독립망 NTP 동기화 프로그램, 블랙잭 게임
4월 4주	3주차	파일 정렬 최적화 프로그램 구현 가능성 조사, 필요 기능 정리 및 제안서 작성 Winform FileInfo, DirectoryInfo 클래스로 구현 가능 필요한 기능 : 검색, 정렬, 필터링, 파일 표시, 선택, 즐겨찾기 폴더 추가, 카테고리 지정 기능
4월 5주	4주차	작성한 제안서를 바탕으로 깃허브에 문서 정리 시작
5월 1주	5주차	캡스톤 개발 및 시험기간으로 인해 개발 일시중지
5월 2주	6주차	깃허브 Readme.md에 팀원, 개발 기능, 참고 사이트 추가 개발에 필요한 FileInfo 클래스 메소드 조사, 폴더 추가 기능, 하위 디렉토리 내 모든 파일 명 검색 기능 개발
5월 3주	7주차	특정 파일로 이동하는 기능 개발, 바로가기 추가 기능, 버튼 / 이미지 동적으로 생성하여 버튼 클릭 시 해당 파일이 위치한 디렉토리로 이동하는 기능 개발, 폴더 정렬 기준 설정 (Compare_File 함수와 FileInfo 클래스의 메소드 사용)
5월 4주	8주차	작성한 제안서 깃허브에 추가, 새 윈도우 폼 열기 기능 조사, 즐겨찾기 화면 표시 기능 개발, 정렬 기능 개발, 파일 선택 후 카테고리 지정 버튼 클릭 시 디렉토리 이동 기능 추가
5월 5주	9주차	데이터베이스 영화 예매 프로그램 팀프로젝트 마감으로 인해 개발 일시중지
6월 1주	10주차	WinForm 종료시 이벤트 발생 기능 개발, 각종 오류 수정(빈칸 선택, 디렉토리 선택 창 취소, 화면 repaint() 오류 등)
6월 2주	11주차	프로그램 UI 보완 및 보고서 작성 후 제출

프로그램 기능구현 방식 및 설명

	기능 목록	기능 구현 방식 및 설명
기능 구현	디렉토리 검색	FloderBrowserDialog 클래스를 이용하여 Winform 프로그램에서 폴더 검색 버튼을 클릭 시 파일 탐색기 창이 뜨고 정렬을 원하는 최상위 폴더를 선택하면 string file 변수에 저장을 합니다. file에 저장한 최상위 디렉토리 이름을 가지고 Directory.getFiles(filename, filter, SearchOption.AllDirectories)를 이용하여 디렉토리 내에 있는 모든 파일이름들을 string array filenames에 저장합니다.
	정렬	사용자가 체크박스에 있는 정렬 기준들을 선택하면 선택한 순서대로 ListBox에 기준을 표시합니다. (오름/내림차순) (파일명, 확장자, 크기, 수정날짜 등) 정렬한 우선순위는 compare_File 메소드에서 listBox_sort.Items을 통해 사용자가 지정한 순서대로 비교를 하게 되며 CompareTo 메소드로 비교한 결과가 0인 경우(우선순위과 같은 경우) 재귀함수를 통해 사용자가 지정한 다음 정렬 우선순위를 비교하게 됩니다.
	필터링	필터링 기능으로는 제일 범용성 높은 파일명과 확장자 2가지를 사용했습니다. 각 라벨 옆에 있는 textBox에 와일드카드를 포함하여 폴더 검색을 하게 되면 Directory.getFiles(filename, filter, SearchOption.AllDirectories)에서 filter = 파일명.확장자 식으로 원하는 파일들을 검색할 수 있게 됩니다.
	파일 표시	filenames에 저장된 파일이름들을 사용자가 지정한 기준에 따라 정렬한 뒤 listbox_Sort_result 에 아이템으로 추가하여 정렬된 결과를 화면에 표시합니다.
	파일 선택	listbox_Sort_result 에 추가된 아이템들을 클릭하게 되면 System.Diagnostics.Process.Start("explorer.ex.","디렉토리명") 메소드를 이용하여 해당 파일이 위치한 디렉토리를 열어서 사용자에게 제공합니다.
	즐겨찾기 폴더 추가	프로그램을 시작할 때 DirectoryInfo 클래스의 Create 메소드를 이용하여 파일의 바로가기와 카테고리 폴더를 저장할 정럴된즐겨찾기 폴더를 생성합니다.
	카테고리 생성	사용자로부터 추가하고 싶은 카테고리 이름을 입력 받은 후 Create 메소드를 이용 해 폴더를 추가합니다.
	카테고리 지정	comboBox_category.SelectedItem에서 사용자가 선택한 카테고리 이름을 string folderPath에 저장하고 string name에 comboBox_file에서 선택한 파일 이름을 저장한 후에 이동할 카테고리 내부에 파일이 이미 존재하는 경우는 건너뛰고 없는 경우에는 바로가기 파일을 생성합니다. 모든 작업이 끝나면 comboBox_file과 listBox_Sort_result에서 선택한 아이템들을 제거해줍니다.
	카테고리 선택	comboBox_category에 추가된 아이템들을 클릭하게 되면 System.Diagnostics.Process.Start("explorer.ex.","디렉토리명") 메소드를 이용하여 해당 디렉토리를 파일 탐색기로 열어줍니다.
	카테고리 파일 보기	comboBox_category에서 아이템을 선택하고 카테고리 파일 보기 버튼을 누르면 listBox_Category_File에 디렉토리내 파일들을 아이템으로 추가해서 보여줍니다.
	즐겨찾기 화면 표시	GetDirectories 메소드를 이용하여 즐겨찾기 내부에 있는 카테고리 폴더 이름들을 comboBox_category의 아이템으로 추가합니다.
	파일 바로가기 추가	WshShell과 IWshShortcut 으로 바로가기를 만들어준 후에 StringBulider로 원본의 경로를 저장하여 바로가기와 원본 파일을 연결해줍니다.
	프로그램 종료	프로그램 종료버튼을 누르거나 오른쪽 위의 종료 버튼을 클릭하게 되면 FormClosing 메소드에서 카테고리를 지정하지 않은 파일들의 바로가기를 미분류 폴더에다 생성한 후에 파일을 종료합니다.

예외 처리	**디렉토리 검색**	정렬할 폴더를 선택하지 않아서 file = “” 인 경우 정렬할 최상위 폴더를 선택하라는 팝업창 메시지를 표시해줍니다.
	박스 아이템 선택	필수적인 comboBox나 listBox의 아이템을 선택하지 않고 기능을 실행하려 하는 경우 “파일 / 카테고리를 선택해주세요” 라는 팝업창 메시지를 표시해줍니다.
	카테고리 추가	이미 존재하는 카테고리를 추가하려는 경우 “이미 존재하는 카테고리입니다” 라는 팝업창 메시지를 표시해주십니다.
	프로그램 종료 시	사용자가 실수로 종료버튼을 누르는 경우를 확인하기 위해 “종료하시겠습니까?” 라는 팝업창 메시지를 표시하여 한번 더 확인 받은 후에 종료를 합니다.

프로그램 소스 코드

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
using IWshRuntimeLibrary;

namespace File_Sorting_Optimization
{

    public partial class Form1 : Form
    {
        //Form 시작할 때 실행
        public Form1()
        {

            InitializeComponent();

            //Winform Form 제목 변경
            this.Text = "파일 정렬 최적화 프로그램";

            //ListBox에 정렬기준 추가
            CKList_Sort.Items.Add("파일명");
            CKList_Sort.Items.Add("파일크기");
            CKList_Sort.Items.Add("수정날짜");
            CKList_Sort.Items.Add("확장자");

            //오름차순 체크 상태로 변경
            checkBox_asc.Checked = true;

            //즐겨찾기 폴더가 없는 경우 생성
            string folderPath = "C:₩₩정렬된즐겨찾기";
            DirectoryInfo di = new DirectoryInfo(folderPath);

            if (di.Exists == false)
            {
                di.Create();
                folderPath += "₩₩미분류";
                DirectoryInfo temp = new DirectoryInfo(folderPath);
                temp.Create();
```

```
        }

        //즐겨찾기 폴더 내 하위 디렉토리 리스트에 아이템으로 저장
        DirectoryInfo[] category = di.GetDirectories();

        foreach (DirectoryInfo temp in category)
        {
            listBox_Category.Items.Add(temp.Name);
            comboBox_category.Items.Add(temp.Name);
        }
    }

    //폴더 선택 버튼
    private void Button_Select_Directory_Click(object sender, EventArgs e)
    {
        //공백일 경우 와일드카드 * 로 변경
        if (textBox_Filter_Filename.Text.ToString() == "") textBox_Filter_Filename.Text = "*";
        if (textBox_Filter_extension.Text.ToString() == "") textBox_Filter_extension.Text = "*";
        //제목이름 + 확장자 로 파일 검색

        string filter = textBox_Filter_Filename.Text.ToString() + "." + textBox_Filter_extension.Text.ToString();
        string file = "";
        string[] filenames = null;

        //파일탐색기 UI 열기
        FolderBrowserDialog OFD = new FolderBrowserDialog();
        try
        {
            if (OFD.ShowDialog() == DialogResult.OK)
            {
                //file에 선택한 디렉토리 이름 저장
                file = OFD.SelectedPath;
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("폴더를 선택해주세요");
            //MessageBox.Show(ex.Message);
        }

        try
        {
            try
            {
                //디렉토리 내 모든 파일 검색하여 저장
                filenames = Directory.GetFiles(file, filter, SearchOption.AllDirectories);
            }
```

```
        catch (Exception ex)
        {
            MessageBox.Show("폴더를 선택해주세요");
            //MessageBox.Show(ex.ToString());
            return;
        }

        //우선순위에 따라 정렬 (compare_File 함수 이용)
        for(int i = 0; i < filenames.Length-1; i++)
        {
            //정렬 우선순위가 높다면 서로 교환 (오름차순)
            if (compare_File(filenames[i], filenames[i + 1], 0) > 0 && checkBox_asc.Checked)
            {
                string temp = filenames[i];
                filenames[i] = filenames[i + 1];
                filenames[i + 1] = temp;
            }
            //정렬 우선순위가 높다면 서로 교환 (내림차순)
            else if (compare_File(filenames[i], filenames[i + 1], 0) < 0 && !checkBox_asc.Checked)
            {
                string temp = filenames[i];
                filenames[i] = filenames[i + 1];
                filenames[i + 1] = temp;
            }
            //우선순위가 같다면 순서 바꾸지 않음
            else
            {
                continue;
            }
        }

        //기존에 ListBox와 ComboBox에 들어있는 파일들 제거
        listBox_Sort_result.Items.Clear();
        comboBox_file.Items.Clear();

        //ListBox와 ComboBox에 정렬된 파일 순서대로 저장
        for (int i = 0; i < filenames.Length; i++)
        {
            listBox_Sort_result.Items.Add(filenames[i].ToString());
            comboBox_file.Items.Add(filenames[i].ToString());
        }

    }
    catch (IOException ex)
    {
        MessageBox.Show(ex.Message);
    }
```

```
}

//사용자 정의 정렬 기준
private void CKList_Sort_SelectedIndexChanged(object sender, EventArgs e)
{
    //선택한 item이 체크 됐다면
    if (CKList_Sort.GetItemChecked(CKList_Sort.SelectedIndex))
    {
        //listBox에 포함되어있지 않은 item이라면 추가
        if (!listBox_sort.Items.Contains(CKList_Sort.SelectedItem.ToString()))
        {
            listBox_sort.Items.Add(CKList_Sort.SelectedItem);
        }

    }
    //선택한 item이 체크 되어있지 않다면
    else
    {
        //listBox에 포함되어있다면 list에서 item 제거
        if (listBox_sort.Items.Contains(CKList_Sort.SelectedItem.ToString()))
        {
            listBox_sort.Items.Remove(CKList_Sort.SelectedItem);
        }
    }
}

//정렬 함수
private int compare_File(string file1, string file2,int count)
{
    //모든 정렬 기준을 비교했다면 SWAP X
    if (count == listBox_sort.Items.Count) return 0;

    //파일의 정보 저장
    var info1 = new FileInfo(file1);
    var info2 = new FileInfo(file2);

    //비교 타입 문자열 저장
    string type = listBox_sort.Items[count].ToString();

    //파일명을 선택한 경우
    if (type == "파일명")
    {
        //파일이름이 같은 경우, 다음에 선택된 정렬기준으로 비교
        if (info1.Name.ToString().Split('.')[0].CompareTo(info2.Name.ToString().Split('.')[0]) == 0)
        {
            return compare_File(file1, file2, count + 1);
        }
```

```
        //다른 경우 비교결과 리턴
        else
        {
            return info1.Name.ToString().Split('.')[0].CompareTo(info2.Name.ToString().Split('.')[0]);
        }
    }
    //파일크기를 선택한 경우
    else if (type == "파일크기")
    {
        //length에다 파일의 길이 저장
        long length1 = new System.IO.FileInfo(file1).Length;
        long length2 = new System.IO.FileInfo(file2).Length;

        //파일크기가 같은 경우 다음에 선택된 정렬기준으로 비교
        if (length1 == length2)
        {
            return compare_File(file1, file2, count + 1);
        }
        //다른 경우 비교결과 리턴
        else
        {
            return length1.CompareTo(length2);
        }

    }
    //수정날짜를 선택한 경우
    else if (type == "수정날짜")
    {
        //수정날자가 같은경우 다음에 선택된 정렬 기준으로 비교
        if (info1.LastWriteTime.CompareTo(info2.LastWriteTime)==0)
        {
            return compare_File(file1, file2, count + 1);
        }
        //다른 경우 비교결과 리턴
        else
        {
            return info1.LastWriteTime.CompareTo(info2.LastWriteTime);
        }
    }
    //확장자를 선택한 경우
    else if (type == "확장자")
    {
        //확장자가 같은 경우 다음에 선택된 정렬 기준으로 비교
        if (info1.ToString().Split('.')[1].CompareTo(info2.ToString().Split('.')[1]) == 0)
        {
            return compare_File(file1, file2, count + 1);
        }
```

```
            //다른 경우 비교결과 리턴
            else
            {
                return info1.ToString().Split('.')[1].CompareTo(info2.ToString().Split('.')[1]);
            }
        }
        //정렬 우선순위가 없는 경우 0리턴
        else
        {
            return 0;
        }
    }

    //파일 선택 Method
    private void listBox_Sort_result_SelectedIndexChanged(object sender, EventArgs e)
    {
        //정렬된 파일을 선택하지않고 ListBox를 클릭한 경우
        if(listBox_Sort_result.SelectedItem is null)
        {
            return;
        }

        try
        {
            //리스트에서 선택한 아이템의 파일 정보를 info에 저장
            var info = new FileInfo(listBox_Sort_result.SelectedItem.ToString());
            //dir에 해당 파일의 디렉토리 주소를 저장
            string dir = $@"{info.DirectoryName}";
            //해당 위치로 파일탐색기 열기
            System.Diagnostics.Process.Start("explorer.exe", dir);

            //카테고리 선택 후에 선택된 파일들 체킹 해제
            listBox_Sort_result.SelectedItems.Clear();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }

    }

    //즐겨찾기내 카테고리 폴더 추가
    private void button_add_category_Click(object sender, EventArgs e)
    {
        //추가할 카테고리 이름 저장
        string folderPath = category.Text;
        DirectoryInfo di = new DirectoryInfo("C:₩₩정렬된즐겨찾기₩₩"+folderPath);
```

```
    //해당 카테고리가 존재하지 않는다면
    if (di.Exists == false)
    {
        //카테고리 폴더 추가 및 리스트와 콤보박스에 아이템 추가
        di.Create();
        listBox_Category.Items.Add(category.Text);
        comboBox_category.Items.Add(category.Text);
    }
    //이미 있다면 에러메세지 호출
    else
    {
        MessageBox.Show("이미 존재하는 카테고리입니다.");
    }

}

//파일 카테고리 지정
private void button_add_file_Click(object sender, EventArgs e)
{
    //파일을 선택하지 않고 선택 버튼을 누른 경우 에레메세지 호출
    if (comboBox_file.SelectedItem == null)
    {
        MessageBox.Show("파일을 선택해주세요.");
    }
    else
    {

        string folderPath = "C:₩₩정렬된즐겨찾기₩₩" + comboBox_category.SelectedItem.ToString();
        //Linkstr에 바로가기 파일 이름 저장
        string name = Path.GetFileName(comboBox_file.SelectedItem.ToString());
        string Linkstr = $"{name} 바로가기.lnk";
        string LinkFileName = folderPath.ToString() + $@"₩{Linkstr}";
        FileInfo LinkFile = new FileInfo(LinkFileName);

        //바로가기 파일이 이미 있다면 건너뜀
        if (LinkFile.Exists)
        {

        }

        else
        {
            //바로가기 생성
            WshShell wsh = new WshShell();
            IWshShortcut Link = wsh.CreateShortcut(LinkFile.FullName);

            //원본파일 경로
```

```
                StringBuilder SB = new StringBuilder();

                //바로가기 저장
                SB.Append(@comboBox_file.SelectedItem.ToString());
                Link.TargetPath = SB.ToString();
                Link.Save();
            }

            listBox_Sort_result.Items.Remove(comboBox_file.SelectedItem);
            comboBox_file.Items.Remove(comboBox_file.SelectedItem);
        }
    }

    //카테고리 ITEM 클릭 이벤트
    private void listBox_Category_SelectedIndexChanged(object sender, EventArgs e)
    {
        //빈 아이템을 선택시 예외처리
        if (listBox_Category.SelectedItem == null)
        {
            return;
        }
        try
        {
            //카테고리 클릭시 해당 카테고리의 디렉토리 창 열기
            System.Diagnostics.Process.Start("explorer.exe", "C:₩₩정렬된즐겨찾기₩₩" +
listBox_Category.SelectedItem.ToString());
            listBox_Category.SelectedItems.Clear(); //카테고리 선택 후에 선택된 파일들 초기화
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }

    }

    //윈폼 종료시 이벤트 설정
    private void Form1_FormClosing(object sender, FormClosingEventArgs e)
    {
        //사용자에게 종료 여부 확인
        if (MessageBox.Show("종료하시겠습니까?", "종료", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            //즐겨찾기 내에서 남아있는 모든 파일을 미분류로 옮겨줌
            for (int i = 0; i < listBox_Sort_result.Items.Count; i++)
            {
                string folderPath = "C:₩₩정렬된즐겨찾기₩₩미분류";
                //Linkstr에 바로가기 파일 이름 저장
                string name = Path.GetFileName(listBox_Sort_result.Items[i].ToString());
```

```
                    string Linkstr = $"{name} 바로가기.lnk";
                    string LinkFileName = folderPath.ToString() + $@"₩{Linkstr}";
                    FileInfo LinkFile = new FileInfo(LinkFileName);

                    //바로가기 파일이 이미 있다면 건너뜀
                    if (LinkFile.Exists)
                    {
                        continue;
                    }

                    else
                    {
                        //바로가기 생성
                        WshShell wsh = new WshShell();
                        IWshShortcut Link = wsh.CreateShortcut(LinkFile.FullName);

                        //원본파일 경로
                        StringBuilder SB = new StringBuilder();

                        //바로가기 저장
                        SB.Append(@listBox_Sort_result.Items[i].ToString());
                        Link.TargetPath = SB.ToString();
                        Link.Save();
                    }
                }
            }
            else
            {
                e.Cancel = true;
                return;
            }
        }

        private void button_Exit_Click(object sender, EventArgs e)
        {

            //FormClosing() 실행
            Application.Exit();

        }
    }
}
```

프로그램 보완 및 개선점

정렬 기능

: MicroSoft에서 제공하는 FileInfo 클래스에 대한 문서를 살펴보면 CreationTime, IsReadOnly, LastAccessTime, 등 정렬을 할 때 쓸 수 있는 여러 가지 정보들을 뽑아올 수 있는데 이를 활용하여 사용자가 지정할 수 있는 정렬 기준을 더 늘릴 계획입니다. 그리고 Winform 프로그램에서는 정렬된 순서로 보이지만 실제 윈도우 파일 탐색기에서는 이름순으로 정렬이 되어있어서 이 부분을 해결하여 실제 폴더에서도 사용자가 지정한 우선순위 기준으로 정렬이 될 수 있도록 개선할 예정입니다.

필터링 기능

: 파일 이름과 확장자 뿐만 아니라 파일 크기, 수정 날짜등 정렬 기능에서 쓸 수 있는 조건들을 필터링에서도 쓸 수 있게 적용할 계획입니다.

카테고리 지정 기능

: 실제 프로그램을 테스트 해보니 디렉토리 내 파일들이 적은 경우에는 하나씩 옮기는 것이 괜찮지만 파일의 개수가 100개를 넘어가고 한 번에 여러 개의 파일들을 카테고리 지정하고 싶을 때에는 현재의 방식이 불편하다는 것을 느꼈습니다. 새롭게 개선할 버전에서는 기존 ListBox가 아닌 CheckBox 로 구성하여 카테고리 지정을 원하는 파일들을 체크표시로 선택하게 하고 옮기고 싶은 카테고리로 한번에 옮기도록 바꿀 예정입니다.

파일 바로가기 추가 기능

: 파일을 실제로 이동시켜야 할지 바로가기만 추가해야 할지 의견차이가 있었는데 아직 테스트 버전이어서 실제 파일 / 폴더를 삭제, 이동, 복사 하도록 기능을 추가하기에는 오류가 발생하면 복원이 안될 수 있어서 실제 제작 버전에서는 위의 기능들을 보완, 수정할 계획입니다.