Programming Tips and Tricks

자주 까먹는 자잘한 꼼수(?) 및 필수 지식들을 모아놓은 페이지입니다.

Tips, Algorithm

공통

한글 자동완성

https://support.microsoft.com/ko-kr/kb/2701840

Polymorphism(다형성)

Polymorphism

2차원배열 → 1차원배열

```
arr[(i * j.length) + j] //= arr[i][j]
```

단위 변환

경위도 변환

```
/* Language : C# */

   /* 도분초 → 천분율 */

        /// <summary>
   /// 도분초 → 천분율
        /// ("128:43:47" → "34:58.105")
        /// </summary>
        /// <param name="dms"></param>
        /// <returns></returns>
        public static string DMS2Permil(string dms)
        {
            return Degree2Permil(DMS2Degree(dms));
        }

        /// <summary>
```

```
    /// 도분초 → 천분율
         /// (128, 43, 47 → "34:58.105")
         /// </summary>
         /// <param name="deg"></param>
         /// <param name="min"></param>
         /// <param name="sec"></param>
         /// <returns></returns>
         public static string DMS2Permil(int deg, int min, int sec)
         {
             return Degree2Permil(DMS2Degree(deg, min, sec));
         }

         /// <summary>
    /// 도분초 → 천분율
         /// ("128:43:47" → {34, 58.105})
         /// </summary>
         /// <param name="dms"></param>
         /// <returns></returns>
         public static Tuple<int, double> DMS2Permil_toTuple(string dms)
         {
             return Degree2Permil_toTuple(DMS2Degree(dms));
         }

         /// <summary>
    /// 도분초 → 천분율
         /// (128, 43, 47 → {34, 58.105})
         /// </summary>
         /// <param name="deg"></param>
         /// <param name="min"></param>
         /// <param name="sec"></param>
         /// <returns></returns>
         public static Tuple<int, double> DMS2Permil_toTuple(int deg, int
min, int sec)
         {
             return Degree2Permil_toTuple(DMS2Degree(deg, min, sec));
         }


    /* 천분율 → 도분초 */

         /// <summary>
    /// 천분율 → 도분초 (string)
         /// ("34:58.105" → "34:58:06")
         /// </summary>
         /// <param name="permil"></param>
         /// <returns></returns>
         public static string Permil2DMS(string permil)
         {
             return Degree2DMS(Permil2Degree(permil));
         }
```

```
        /// <summary>
    /// 천분율 → 도분초 (string)
        /// (34, 58.105 → "34:58:06")
        /// </summary>
        /// <param name="deg"></param>
        /// <param name="sec"></param>
        /// <returns></returns>
        public static string Permil2DMS(int deg, double sec)
        {
            return Degree2DMS(Permil2Degree(deg, sec));
        }

        /// <summary>
    /// 천분율 → 도분초 (int[])
        /// ("34:58.105" → {34, 58, 06})
        /// </summary>
        /// <param name="permil"></param>
        /// <returns></returns>
        public static int[] Permil2DMS_toArray(string permil)
        {
            return Degree2DMS_toArray(Permil2Degree(permil));
        }

        /// <summary>
    /// 천분율 → 도분초 (int[])
        /// (34, 58.105 → {34, 58, 06})
        /// </summary>
        /// <param name="permil"></param>
        /// <returns></returns>
        public static int[] Permil2DMS_toArray(int deg, double sec)
        {
            return Degree2DMS_toArray(Permil2Degree(deg, sec));
        }


    /* 경위도 → 천분율 */

        /// <summary>
    /// 경위도 → 천분율 (string)
        /// (128.7297335 → 128:43.784)
        /// </summary>
        /// <param name="latLng"></param>
        /// <returns></returns>
        public static string Degree2Permil(double latLng)
        {
            return ((int)latLng).ToString("00") + ":" + ((latLng -
(int)latLng) * 60).ToString("00.000");
        }

        /// <summary>
    /// 경위도 → 천분율 (int, double)
```

```
        /// (128.7297335 → {128, 43.784})
        /// </summary>
        /// <param name="latLng"></param>
        /// <returns></returns>
        public static Tuple<int, double> Degree2Permil_toTuple(double
latLng)
        {
            return Tuple.Create<int, double>((int)latLng, Math.Round((latLng
- (int)latLng) * 60, 3));
        }


   /* 천분율 → 경위도 */

        /// <summary>
   /// 천분율 → 경위도 (double)
        /// ("128:43.784" → 128.729733333333)
        /// </summary>
        /// <param name="permil"></param>
        /// <returns></returns>
        public static double Permil2Degree(string permil)
        {
            int colonIndex = permil.IndexOf(':');

            int deg = int.Parse(permil.Substring(0, colonIndex));
            double min = double.Parse(permil.Substring(colonIndex + 1,
permil.Length - (colonIndex + 1)));

            return (double)deg + min / 60;
        }

        /// <summary>
   /// 천분율 → 경위도 (double)
        /// (128, 43.784 → 128.729733333333)
        /// </summary>
        /// <param name="deg"></param>
        /// <param name="min"></param>
        /// <returns></returns>
        public static double Permil2Degree(int deg, double min)
        {
            return (double)deg + min / 60;
        }


   /* 경위도 → 도분초 */

        /// <summary>
   /// 경위도 → 도분초 (string)
        /// (128.7297335 → 128:43:470)
        /// </summary>
```

```
        /// <param name="latLng"></param>
        /// <returns></returns>
        public static string Degree2DMS(double latLng)
        {
            int deg = (int)latLng;

            double decimalPart1 = latLng - deg;
            int min = (int)(decimalPart1 * 60);

            double decimalPart2 = (decimalPart1 * 60) - min;
            int sec = (int)(decimalPart2 * 60);

            return string.Format("{0:D02}:{1:D02}:{2:D02}", deg, min, sec);
        }

        /// <summary>
    /// 경위도 → 도분초 (int[])
        /// (128.7297335 → {128, 43, 47})
        /// </summary>
        /// <param name="latLng"></param>
        /// <returns></returns>
        public static int[] Degree2DMS_toArray(double latLng)
        {
            int deg = (int)latLng;

            double decimalPart1 = latLng - deg;
            int min = (int)(decimalPart1 * 60);

            double decimalPart2 = (decimalPart1 * 60) - min;
            int sec = (int)(decimalPart2 * 60);

            return new int[3] { deg, min, sec };
        }


    /* 도분초 → 경위도 */

        /// <summary>
    /// 도분초 → 경위도
        /// ("128:43:47" → 128.72972222222222)
        /// </summary>
        /// <param name="dms"></param>
        /// <returns></returns>
        public static double DMS2Degree(string dms)
        {
            int colonIndex1 = dms.IndexOf(':');
            int colonIndex2 = dms.LastIndexOf(':');

            string strDeg = dms.Substring(0, colonIndex1);
            string strMin = dms.Substring(colonIndex1 + 1, colonIndex2 -
(colonIndex1 + 1));
```

```
            string strSec = dms.Substring(colonIndex2 + 1, dms.Length -
(colonIndex2 + 1));

            return double.Parse(strDeg) + ((double)double.Parse(strMin) /
60) + ((double)double.Parse(strSec) / 3600);
        }

        /// <summary>
   /// 도분초 → 경위도
        /// (128, 43, 47 → 128.72972222222222)
        /// </summary>
        /// <param name="deg"></param>
        /// <param name="min"></param>
        /// <param name="sec"></param>
        /// <returns></returns>
        public static double DMS2Degree(int deg, int min, int sec)
        {
            return (double)deg + ((double)min / 60) + ((double)sec / 3600);
        }
   /* 신규 추가 */
        // <summary>
        /// Degree → 도분초 (단위 포함)
        /// (-122.333333 → 122:20:00 W)
        /// </summary>
        public static string Degree2DMS(double degree)
        {
            int deg = (int)Math.Abs(degree);

            double min_temp = (Math.Abs(degree) - deg) * 60;
            double min_decPart = min_temp - (int)min_temp;
            int min = min_decPart >= 0.999 ? (int)Math.Round(min_temp, 1,
MidpointRounding.AwayFromZero) : (int)min_temp; //소숫점이 0에 수렴하는가?

            double sec_temp = (Math.Abs(degree) - deg - ((double)min /
(double)60)) * 3600;
            double sec_decPart = sec_temp - (int)sec_temp;
            int sec = sec_decPart >= 0.999 ? (int)Math.Round(sec_temp, 1,
MidpointRounding.AwayFromZero) : (int)sec_temp; //소숫점이 0에 수렴하는가?



            string unit = string.Empty;
            string fmt = string.Empty;
            if (deg <= 90)
            {
                fmt = "2";
                unit = degree >= 0 ? " N" : " S";
            }
            else
            {
```

```
                fmt = "3";
                unit = degree >= 0 ? " E" : " W";
            }

            return string.Format("{0:D0" + fmt + "}:{1:D02}:{2:D2}", deg,
min, sec) + unit;
        }

        /// <summary>
        /// Degree → 천분율(단위 포함)
        /// (-122.333333 → 122:20.660 W)
        /// </summary>
        /// <param name="degree"></param>
        /// <returns></returns>
        public static string Degree2Permil(double degree)
        {
            string unit = string.Empty;
            string fmt = string.Empty;

            double absVal = Math.Abs(degree);

            if ((int)absVal <= 90)
            {
                fmt = "00";
                unit = degree >= 0 ? " N" : " S";
            }
            else
            {
                fmt = "000";
                unit = degree >= 0 ? " E" : " W";
            }

            return ((int)absVal).ToString(fmt) + ":" + ((absVal -
(int)absVal) * 60).ToString("00.000") + unit;
        }



        /// <summary>
   /// 도분초(단위포함) → Degree
        /// (122:20:00 W → -122.333333)
        /// </summary>
        /// <param name="degree"></param>
        /// <returns></returns>
        public static double DMS2Degree(string dms)
        {
            int unitIndex = dms.LastIndexOf(' ');
            string unit = dms.Substring(unitIndex + 1, 1).Trim();


            string[] dmsValue = dms.Substring(0, unitIndex).Split(':');
```

```
            double result = double.Parse(dmsValue[0]) +
(double)(double.Parse(dmsValue[1]) / (double)60) +
(double)(double.Parse(dmsValue[2]) / (double)3600);

            if (unit.ToUpper().Equals("W") || unit.ToUpper().Equals("S"))
                result = -result;

            return result;
        }

        /// <summary>
   /// 천분율(단위포함) → Degree
        /// (122:20.660 W → -122.333333)
        /// </summary>
        /// <param name="degree"></param>
        /// <returns></returns>
        public static double Permil2Degree(string permil)
        {
            int unitIndex = permil.LastIndexOf(' ');
            string unit = permil.Substring(unitIndex + 1, 1).Trim();


            string permilValue = permil.Substring(0, unitIndex);

            int colonIndex = permilValue.IndexOf(':');

            int deg = int.Parse(permilValue.Substring(0, colonIndex));
            double min = double.Parse(permilValue.Substring(colonIndex + 1,
permilValue.Length - (colonIndex + 1)));


            double result = (double)deg + min / 60;

            if (unit.ToUpper().Equals("W") || unit.ToUpper().Equals("S"))
                result = -result;

            return result;
        }
```

거리, 속도, 주파수 변환

```
   /* 거리 계산식 */

        /// <summary>
        /// Yard → Meter
        /// </summary>
```

```
        public static double YD2M(double yd)
        {
            return yd * 0.91440183;
        }

        /// <summary>
        /// Meter → Yard
        /// </summary>
        /// <param name="meter"></param>
        /// <returns></returns>
        public static double M2YD(double m)
        {
            return m / 0.91440183;
        }


        /// <summary>
        /// Kiloyard → Meter
        /// </summary>
        /// <param name="yard"></param>
        /// <returns></returns>
        public static double KYD2M(double yd)
        {
            return 1000.0 * YD2M(yd);
        }

        /// <summary>
        /// Meter → Kiloyard
        /// </summary>
        /// <param name="meter"></param>
        /// <returns></returns>
        public static double M2KYD(double m)
        {
            return M2YD(m) / 1000.0;
        }


        /// <summary>
        /// Kiloyard → Hectometer
        /// </summary>
        /// <param name="yard"></param>
        /// <returns></returns>
        public static double KYD2HM(double yd)
        {
            return 10.0 * YD2M(yd);
        }

        /// <summary>
        /// Hectometer → Kiloyard
        /// </summary>
        /// <param name="meter"></param>
```

```
        /// <returns></returns>
        public static double HM2KYD(double m)
        {
            return M2YD(m) / 10.0;
        }


        /// <summary>
        /// Kiloyard → Kilometer
        /// </summary>
        /// <param name="yard"></param>
        /// <returns></returns>
        public static double KYD2KM(double yd)
        {
            return YD2M(yd);
        }

        /// <summary>
        /// Kilometer → Kiloyard
        /// </summary>
        /// <param name="meter"></param>
        /// <returns></returns>
        public static double KM2KYD(double m)
        {
            return M2YD(m);
        }


        /// <summary>
        /// Kiloyard → Nautical Mile
        /// </summary>
        /// <param name="yard"></param>
        /// <returns></returns>
        public static double KYD2NM(double kyd)
        {
            return kyd * 0.493737;
        }

        /// <summary>
        /// Nautical Mile → Kiloyard
        /// </summary>
        /// <param name="meter"></param>
        /// <returns></returns>
        public static double NM2KYD(double nm)
        {
            return nm * 2.02537183;
        }


    /* 심도 계산식 */
```

```
        /// <summary>
        /// Meter → Feet
        /// </summary>
        /// <param name="meter"></param>
        /// <returns></returns>
        public static double M2FT(double m)
        {
            return m * 3.2808398950131233595800524934383;
        }

        /// <summary>
        /// Feet → Meter
        /// </summary>
        /// <param name="feet"></param>
        /// <returns></returns>
        public static double FT2M(double ft)
        {
            return ft * 0.3048;
        }


    /* 속도 계산식 */

        /// <summary>
        /// Knots → Meter Per Seconds
        /// </summary>
        /// <param name="knots"></param>
        /// <returns></returns>
        public static double KTS2MS(double kts)
        {
            return kts * 0.51444444444444444444444444444444;
        }

        /// <summary>
        /// Meter Per Seconds → Knots
        /// </summary>
        /// <param name="ms"></param>
        /// <returns></returns>
        public static double MS2KTS(double ms)
        {
            return ms * 1.943844492440604751619870410367 2;
        }


    /* 주파수 계산식 */

        /// <summary>
        /// Hertz → Revolutions Per Minute
        /// </summary>
        /// <param name="hertz"></param>
        /// <returns></returns>
```

```
        public static double HZ2RPM(double hz)
        {
            return hz * 60.0;
        }

        /// <summary>
        /// Revolutions Per Minute → Hertz
        /// </summary>
        /// <param name="rpm"></param>
        /// <returns></returns>
        public static double RPM2HZ(double rpm)
        {
            return rpm / 60.0;
        }

        public static double M2KM(double m)
        {
            return m / 1000;
        }

        public static double KM2M(double km)
        {
            return km * 1000;
        }

        public static double M2HM(double m)
        {
            return m / 100;
        }

        public static double HM2M(double hm)
        {
            return hm * 100;
        }

        public static double HM2KM(double hm)
        {
            return hm / 10;
        }

        public static double KM2HM(double km)
        {
            return km * 10;
        }
        /// <summary>
        /// Meter → Nautical Mile
        /// </summary>
        /// <param name="m"></param>
        /// <returns></returns>
        public static double M2NM(double m)
```

```
        {
            return KYD2NM(M2KYD(m));
        }
```

각도, 반지름 ↔ X, Y 좌표

```
        public Tuple<double, double> DegreeRadius2XY(double degree, double
radius)
        {
            double radian = Degree2Radian(degree);
            return Tuple.Create(radius * Math.Cos(radian), radius *
Math.Sin(radian));
        }

        public Tuple<double, double> XY2DegreeRadius(double x, double y)
        {
            double degree = Math.Atan2(y, x) * (180 / 3.1415926535897932);
            double radious = Math.Sqrt(Math.Pow(x, 2) + Math.Pow(y, 2));
            return Tuple.Create(degree, radious);
        }

        public double Degree2Radian(double degree)
        {
            return 3.1415926535897932 / (180 / degree);
        }
```

재귀(Recursion)

Recursion

C# 하부 디렉토리 파일까지 탐색 (현 프로젝트 기준)

```
        public static List<String> GetFiles(string extension)
        {
            List<String> result = new List<String>();

            try
            {
                foreach (string d in
Directory.GetDirectories(AppDomain.CurrentDomain.BaseDirectory))
                {
                    foreach (string f in Directory.GetFiles(d, "*." +
extension))
```

```
                        result.Add(f);

                    GetFiles(d);
                }
            }
            catch (System.Exception)
            {
                return null;
            }

            return result;
        }
```

정렬 알고리즘(Sort Algorithm)

합병 정렬(Merge Sort)

합병 정렬(Merge Sort)

Language & FrameWork

Java

양력/음력 날짜 계산

calendarlib.zip

- http://site.icu-project.org/download
- https://github.com/JodaOrg/joda-time/releases

```
import org.joda.time.DateTime;

import com.ibm.icu.util.ChineseCalendar;

public class LunarDateTime {
    private static int yearCorrectionValue = 2637;

    private LunarDateTime() {
    }
    /**
     *
     * @return 오늘 음력날짜
     */
    public static DateTime now() {
```

```
        ChineseCalendar cc = new ChineseCalendar();
        int year = cc.get(ChineseCalendar.EXTENDED_YEAR) -
yearCorrectionValue;
        int monthOfYear = cc.get(ChineseCalendar.MONTH) + 1;
        int dayOfMonth = cc.get(ChineseCalendar.DAY_OF_MONTH);
        int[] hms = getNowHMS();
        return new DateTime(year, monthOfYear, dayOfMonth, hms[0], hms[1],
hms[2]);
    }
    public static DateTime solarToLunar(int solarYear, int solarMonthOfYear,
int solarDayOfMonth) {
        int[] hms = getNowHMS();
        DateTime dt = new DateTime(solarYear, solarMonthOfYear,
solarDayOfMonth, hms[0], hms[1], hms[2]);

        ChineseCalendar cc = new ChineseCalendar();
        cc.setTimeInMillis(dt.getMillis());
        int year = cc.get(ChineseCalendar.EXTENDED_YEAR) -
yearCorrectionValue;
        int monthOfYear = cc.get(ChineseCalendar.MONTH) + 1;
        int dayOfMonth = cc.get(ChineseCalendar.DAY_OF_MONTH);
        return new DateTime(year, monthOfYear, dayOfMonth, hms[0], hms[1],
hms[2]);
    }
    public static DateTime lunar2Solar(int lunarYear, int lunarMonthOfYear,
int lunarDayOfMonth) {
        ChineseCalendar cc = new ChineseCalendar();
        cc.set(ChineseCalendar.EXTENDED_YEAR, lunarYear +
yearCorrectionValue);
        cc.set(ChineseCalendar.MONTH, lunarMonthOfYear - 1);
        cc.set(ChineseCalendar.DAY_OF_MONTH, lunarDayOfMonth);

        int[] hms = getNowHMS();
        cc.set(ChineseCalendar.HOUR_OF_DAY, hms[0]);
        cc.set(ChineseCalendar.MINUTE, hms[0]);
        cc.set(ChineseCalendar.SECOND, hms[0]);
        return new DateTime(cc.getTimeInMillis());
    }
    private static int[] getNowHMS() {
        DateTime now = DateTime.now();
        return new int[] {now.getHourOfDay(), now.getMinuteOfHour(),
now.getSecondOfMinute()};
    }
}
```

C

포인터와 구조체 기초 개념 예제

```
#include <stdio.h>
#include <string.h>
#include "stdlib.h"

struct student
{
    char name[20+1];
    int year;
    double score;
};

typedef struct MyStruct
{
    char str[10];
} hyuk_struct;

void main()
{
  /* 구조체 1*/
    //struct student st_lee = {"Lee",2008,95.4};
    struct student st_lee = {0};
    strcpy(st_lee.name, "HYUK");
    st_lee.year=2008;
    st_lee.score=99.7;

    printf("%s %d %f\n",st_lee.name,st_lee.year,st_lee.score);

  /*구조체 2 - 타입 정의*/
    hyuk_struct asdf = {"Lucas"};

    printf("%s\n",asdf.str);

    printf("\n\n");

  /* 포인터 */
    float val1 = 5.23f;
    float* ptr1 = NULL;
    ptr1 = &val1; // 포인터이기에 주소값만 대입 가능!

    printf("%d\n",ptr1); // val1 주소값 출력
    printf("%f\n",val1);


    printf("\n");

    char val2 = 'A';
    char* ptr2 = &val2;
    *ptr2 = 'X'; // 포인터로 메모리 직접 접근하여 값 바꿈
    val2='Y'; // 변수에 그냥 대입
```

```
    printf("%d\n",ptr2);
    printf("%c\n",val2); // 결과적으로 Y 출력

    printf("\n");

  /* 동적 메모리 할당 */
    int *ptr3 = NULL;
    int mem_size = 100;
    ptr3=(int*)malloc(sizeof(int)*mem_size);

    if(ptr3!=NULL)
    {
        memset(ptr3, 0, sizeof(int)*mem_size); // 메모리의 쓰레기값을 모두 0으로 초기화
        printf("정수 입력 : ");
        scanf("%d", ptr3); // ptr3가 포인터 변수이므로 인자값도 "&"가 아닌 "%"!!
        printf("%d \n", *ptr3); // 주소값이 아닌 실제 참조값을 출력해야하기에

        free(ptr3); // 해제
    }
    else
        printf("ptr3==NULL\n");

    printf("\n");

  /* 포인터의 포인터 */ //2차원 이상의 다차원 배열에서 쓰인다.
    int iVal = 777;
    int *iPtr = &iVal;
    int **iPPtr = &iPtr;

    printf("%d %d \n",*iPtr, iPtr); // iVal, iVal 주소값
    printf("%d %d %d \n",**iPPtr, *iPPtr, iPPtr); // iVal, iVal 주소값, iVal 주소의 주소값

    printf("\n");

  /* 동적 다차원 배열 */ //2차원배열은 "1차원 배열들의 배열"
    int row=3, col=4;

    int **iPPtrArr=NULL;
    iPPtrArr=(int**)malloc(sizeof(int*)*row); // 가로(행) 수만큼 메모리 할당(확보)

    for(int i=0 ; i<row ; i++) // row수만큼 끊어가며 메모리 할당 → 2차원 배열 만들기
    {
        iPPtrArr[i] = (int*) malloc(sizeof(int)*col); // 인덱스마다 메모리 할당
        memset(iPPtrArr[i], 0, sizeof(int)*col); // 쓰레기값을 0으로 변환
    }

    *(*(iPPtrArr+2)+3) = 111; // iPPtrArr[2][3] = 777
    printf("%d %d %d\n", **iPPtrArr, *(*(iPPtrArr+1)+1), *(*(iPPtrArr+2)+3)
```

```
); // 0, 0, 111

    free(iPPtrArr);

    printf("\n");

  /* 구조체 포인터 */
    hyuk_struct qwerty;
    hyuk_struct *qwertyPtr=NULL;
    qwertyPtr=&qwerty;

    strcpy((*qwertyPtr).str, "XXX");
    printf("%s\n",qwertyPtr->str); // 위와 같은 표현
}
```

동적 배열 할당

```
//1차원
int *input;
int staticSize = 할당크기;
input = (int*)malloc(sizeof(int)*staticSize);

//2차원
int **input;
int staticSizeOfROW = 할당크기;
int staticSizeOfColumn = 할당크기;
input = (int(*)[])malloc (sizeof(int*)*staticSizeOfROW);  //행
input[0] = (int*)malloc(sizeof(int*)*staticSizeOfColumn); //열
```

Android

Parcelable

- Intent를 이용하여 **Reference Type**의 데이터를 넘길 때 사용.

http://arsviator.tistory.com/169

Image 압축 및 실제 경로 얻기

```
public class ImageUtil {

    private static final int MAX_IMAGE_SIZE = 1280;
```

```
    private static final int MAX_IMAGE_LENGTH = 720 * 1280;

    /**
     * Image Uri → Compressed byte array
  *(반복문으로 인하여 별도의 Thread 필요)
     * @param context
     * @param uri
     * @return
     * @throws Exception
     */
    public static byte[] getCompressedByteArray(Context context, Uri uri)
throws Exception {
        BitmapFactory.Options options = new BitmapFactory.Options();

        options.inSampleSize = calculateInSampleSize(options, 800, 800);
        options.inJustDecodeBounds = false;
        options.inPreferredConfig= Bitmap.Config.ARGB_8888;

        BufferedInputStream bin = new
BufferedInputStream(context.getContentResolver().openInputStream(uri));
        Bitmap bitmap = BitmapFactory.decodeStream(bin, null, options);

        // resize (지정한 크기보다 작은 경우만)
        if(bitmap.getWidth() * bitmap.getHeight() > MAX_IMAGE_SIZE * 2)
            bitmap = resizeBitmap(bitmap, MAX_IMAGE_SIZE);

        int compressQuality = 100;
        int streamLength;

        byte[] bitmapdata;

        String realPath = getRealImagePath(context, uri);
        String extension = realPath.substring(realPath.lastIndexOf('.') +
1);
        Bitmap.CompressFormat format;
        switch (extension.toLowerCase()) {
            case "jpg" :
            case "jpeg" :
                format = Bitmap.CompressFormat.JPEG;
                break;

            case "png" :
            case "gif" :
                format = Bitmap.CompressFormat.PNG;
                break;

            case "webp" :
                format = Bitmap.CompressFormat.WEBP;
                break;

            default:
```

```
                throw new Exception("Unsupported format!");
        }

        do {
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            bitmap.compress(format, compressQuality, baos);
            bitmapdata = baos.toByteArray();
            streamLength = bitmapdata.length;
            compressQuality -= 5;
        } while (streamLength >= MAX_IMAGE_LENGTH);

        Log.d("compressBitmap", "Quality: " + compressQuality);
        Log.d("compressBitmap", "Size: " + streamLength/1024+" kb");

        return bitmapdata;
    }

    private static int calculateInSampleSize(BitmapFactory.Options options,
int reqWidth, int reqHeight) {
        final int height = options.outHeight;
        final int width = options.outWidth;
        int inSampleSize = 1;

        if (height > reqHeight || width > reqWidth) {

            final int halfHeight = height / 2;
            final int halfWidth = width / 2;

            // Calculate the largest inSampleSize value that is a power of 2
and keeps both
            // height and width larger than the requested height and width.
            while ((halfHeight / inSampleSize) > reqHeight && (halfWidth /
inSampleSize) > reqWidth) {
                inSampleSize *= 2;
            }
        }
        Log.d("inSampleSize", String.valueOf(inSampleSize));
        return inSampleSize;
    }

    private static Bitmap resizeBitmap(Bitmap bitmap, int maxSize) {
        float ratio = Math.min((float) maxSize / bitmap.getWidth(), (float)
maxSize / bitmap.getHeight());
        int width = Math.round(ratio * bitmap.getWidth());
        int height = Math.round(ratio * bitmap.getHeight());
        return Bitmap.createScaledBitmap(bitmap, width, height, false);
    }

    /**
     * Uri를 이용해 실제 저장소 경로 얻기
```

```
     * @param context
     * @param uri
     * @return
     */
    public static String getRealImagePath(Context context, Uri uri) {
        Cursor cursor = context.getContentResolver().query(uri, null, null,
null, null);
        cursor.moveToFirst();
        int index =
cursor.getColumnIndex(MediaStore.Images.ImageColumns.DATA);
        return cursor.getString(index);
    }

    /**
     * [0] : fileName
     * [1] : extension
     * @param filePath
     * @return
     */
    public static String[] getFileNameAndExtension(String filePath) {
        return new String[]
{filePath.substring(filePath.lastIndexOf(File.pathSeparator) + 1),
filePath.substring(filePath.lastIndexOf('.') + 1)};
    }
}
```

From:
https://wiki.ledyx.xyz/ - **Ledyx WIKI**

Permanent link:
https://wiki.ledyx.xyz/doku.php?id=programming_tips_and_tricks&rev=1579014733

Last update: **2026/10/24 07:03**