로또 당첨확률 높이기

물리적 특성만을 가정한 당첨 확률 분석

빅데이터반 김혜민

진행 순서

1. 주제선택 이유 (3~5p)
2. 프로그램 순서도 (6p)
3. 프로그램 코드 (7~15p)
4. 프로그램 코드수행 결과 (16p)
5. 데이터의 그래프화 (16~18p)
6. 최종 결론 (19p)

독립시행이란,

매번 같은 조건에서 어떤 시행을 반복할 때, 각 시행의 결과가
다른 시행의 결과에 영향을 주지 않는 시행이다. [수학 백과, 대한수학회]

- 매주 로또 추첨은 **이전 추첨결과와 완전하게 무관하므로 독립시행**이다.
- '자동선택', '수동선택' 상관없이 **하나의 시행에서 당첨될 확률이 같으므로** 여러 번 뽑는다고 해도 뽑는 행위 자체는 독립시행이다.
- 한 회차에 다른 번호로 여러장을 사면 당첨 확률이 늘어나니까 종속시행이라 생각할수 있는데 종속시행의 정의는 시행의 '결과'가 다음 시행의 확률에 영향을 주는 경우를 말하는 것이다.
- **복권에서의 결과는 당첨번호가 발표됐을때 나오는 것**이니 한 회차의 당첨확률의 변화로는 독립시행이냐 종속시행이냐를 따질 수 없다.
- 더 쉽게 말하면 복권을 한장 샀을때는 8백만분의 1인 독립시행인 것이고 두장을 샀을땐 400만분의 1인 독립시행인 것이다.

>>그러므로 로또의 앞전의 당첨번호를 참고하여 당첨확률을 수학적 분석을 하는 것은 의미가 없다.

도박사의 오류

서로 영향을 끼치지 않는 일련의 확률적 사건들에서 상관관계를 찾아내려 하는 사고의 오류를 이야기한다.
아래와 같이 전제된 상황에서 각 사건 간의 상관관계를 찾아내려는 행위가 도박사의 오류에서 기인하는 행동이라고 볼 수 있다.

- 확률적 결과값을 갖는 어떠한 사건이
- 동일한 실행 조건 하에 (= 확률은 매번 같은 분포를 따른다.)
- 두 번 이상 일어나며 (= 반복 시행이다.)
- 사건의 발생이 다른 사건에 미치는 영향이 없다. (= 반복 시행의 확률은 서로 독립이다.)

쉽게 발생하는 오류의 예시)

- 동전을 던질 경우 (똑바로 서지 않는 한) 앞과 뒤 중 하나만 나올 수 있다. 즉, 동전을 던질 때의 확률은 1/2이다.
- 위에서 정의된 확률에 영향을 끼칠 수 있는 다른 외부 요인이 없는 한, 동전을 던지는 사건의 결과값은 언제나 1/2이다.

>>홀짝이나 이웃수, 숫자단위별 밀도 등으로 확률을 예측하려는 시도도 있으나 그것도 사고의 오류이다.

그럼 무엇이 실제 현실에서 확률에 영향을 미치는가?

로또번호를 수동으로 지정 : 구매자의 의도로 선택.

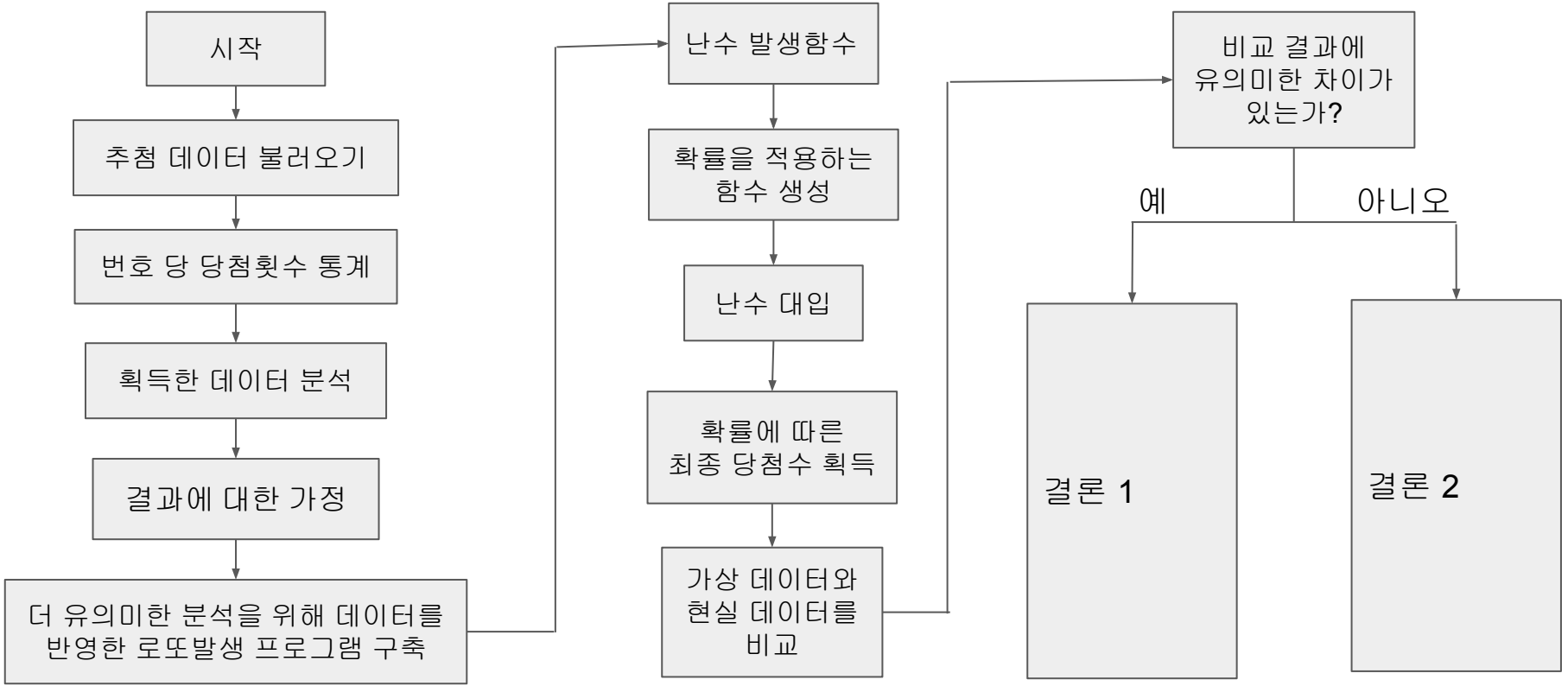
로또번호를 자동으로 지정 : 프로그램에 의한 난수 발생으로 선택.

하지만 실제로 로또번호를 추첨하는 방식은 물리적인 방식이다.

- 기계 내에서 바람에 의해 공이 섞이면서 당첨통로로 나오는 방식임.
- 추첨구 : 약 4g, 지름 45mm의 폴리우레탄 재질의 색이 입혀진 공.
- 로또기계 : 비너스. 프랑스 윈티브사에서 구매한 기기. 3대를 보유.
- 공을 섞는 방식 : 73~80m/s속도의 바람을 공급하여 섞고 추첨함.
- 공 무게와 지름 검수방식은 상당히 철저하며, 무작위성을 지향한다.

>>그럼에도 불구하고 물리적인 방식이기에, 과연 이런 나열 된 물리적 특성들이 정말로 로또 당첨확률에 영향이 없는 지를 직접 분석하기로 했다.

어떻게 접근 할 것인가? 순서를 생각해보기



프로그램 코드

```c
#include <stdio.h>
#include <stdlib.h> //난수 생성함수 헤더파일
#include <string.h> //문자열 처리함수 헤더파일 (_strtok)
#include <time.h> // srand

#define MAX_DATA 7 //로또번호 6개 + 보너스번호 1개
#define SETNUM 1000

struct data //문자열을 정수변환 시 아스키코드 넣을 구조체. (0,1,2,end of character)
{
    char s[MAX_DATA][3];
};

struct LTNUM //중복 제거 후의 번호를 받을 구조체.
{
    int num[7];
};

struct NUMBERS //데이터를 파일에서 불러와서 입력받을 구조체. 주단위로 한 줄이 떨어져서 변수 7개.
{
    int first;
    int second;
    int third;
    int forth;
    int fifth;
    int sixth;
    int bonum; //보너스 넘버.
};

struct CHECKCNT //0부터 44까지 (1~45) 로또 중복넘버 카운트를 체크 할 구조체.
{
    int numbers[45];
};

struct CHECKCNT totalCnt; //CHECKCNT 구조체 형식으로 totalCnt 구조체 선언. 전역변수
```

```c
void remove_spaces(char* s) //아스키코드를 받은 배열 주소 찾아가서 공백을 없애는 함수.
{                           //아스키코드 세자리를 s에서 d로 넘긴다음 do-while 돌면서 다시 d로 넘긴 후 s로는 넘김X
    const char* d = s; //const >> 변수의 초기값 변하면X. 소중한 복사본! s에서 받은 코드를 d 배열을 주소로 넘김.
    do {                    //do { 반복 작업 할 코드 }
        while (*d == ' ') {
            ++d; //d 주소에 공백이 있으면 d를 증가시킨다. (기록X 넘어간다?) >>공백은 넘기고 다음을 기록
        }
    }
    while (*s++ = *d++); //while (조건) - s의 주소가 증가하면 d의 주소도 증가한다.(옆으로 이동)
}

void getfield(char* line, struct data* d, int end_idx) //문자열 자르는 함수.
//자를 문자열, 자른거 받을 구조체 주소, 배열사이즈(마지막 인덱스번호) 문자열 자르기는 포인터 이용. 콤마 기준
{
    int idx = 0; //인덱스 초기화

    char* context; //자르고 남은 애들 담는 임시장소
    char* token = strtok_s(line, ",", &context); //strtok_s : 문자열 자르는 함수. 콤마로 자름.

    do
    {
        strcpy_s(d->s[idx++], end_idx, token); //반복작업 할 코드. 인덱스 늘리면서 d->s로 토큰의 데이터를 인덱스 마지막번호까지 옮김 반복.
    }
    while (idx != end_idx && (token = strtok_s(NULL, ",", &context))); //조건 : 인덱스가 마지막 번호가 아닐 때 그리고 더 이상 자를 내용이 없을 때.
}

int getRandom(int max) //난수발생함수
{
    //_sleep(10);
    int result = rand() % max; //0~7182까지 랜덤하게 발생.
    return result; //반환값으로 넘겨준다.
}
void procYear(int year) //함수가 메인 아래에 있어서 미리 위에서 알려줌. 연도처리함수.
{
    char yearc[10]; //반복적으로 파일명을 불러오기 위해서 정수->문자열 변환필요. itoa 사용할거라 문자열 담을 배열 선언.
    itoa(year, yearc, 10); //정수를 문자로 변경. year가져와서 yearc에 변환해서 넣기. 10 크기로)
    printf("%s년도 로또 번호를 처리합니다.\n", yearc); //넣은 문자배열 출력.

    FILE* fs = NULL; //파일포인터 설정. fs(파일스트림) 초기화. csv 불러 올 준비.
    char* filename = "C:\\example\\lotto\\2003.csv"; //불러 올 파일주소를 변수로 지정.
    char filecp[250] = { 0, }; //문서에서 데이터를 불러와서 담을 버퍼. 가변경로버퍼. 파일명 숫자증가>> 가공해서 담음. 초기화시킴.
    sprintf_s(filecp, 250, "C:\\example\\lotto\\%s.csv", yearc); //출력값을 문자열에 저장하는 함수.
    //(출력값 저장할 문자열, 배열 사이즈, 붙일 문자열 내용(,치환할 값(뒤에 계속이어지게 추가됨)))
    char* accessMode = "r"; //파일 출력모드를 문자포인터로 변수 선언. (나중에 변경이 쉽도록?)
    printf("%S에서 데이터를 불러옵니다.\n", filecp); //버퍼에 받아놓은 문자열을 불러서 출력.
    fopen_s(&fs, filecp, accessMode); //해당 문서를 연다. (파일스트림주소, 데이터받은 버퍼, 출력모드>>r w 이런거)

    int maxlength = 0; //최대길이 변수 선언과 초기화
    char line[1024]; //파일에서 텍스트 불러와서 한 줄씩 읽어서 담을 버퍼. 길이가 얼만지 몰라서 넉넉하게 잡고 한줄 옮기고 처리후 또 한줄 옮기고 처리.
    struct data d[60]; //해당 형태로 데이터를 담을 구조체 생성.(년단위)
    int count = 0; //회차 카운트 변수선언과 초기화
    while (fgets(line, 1024, fs)) //반복해서 1024 사이즈의 Line 문자배열에 파일스트림 버퍼를 가져온다.
    {
        remove_spaces(line); //받아 온 문자열에서 공백을 지우는 함수

        char* tmp = _strdup(line); //line 배열의 내용을 tmp에 동적할당. (길이를 모르므로)
        getfield(tmp, &d[count], MAX_DATA); //문자열 자르는 함수.(자를 문자열, 자른거 받을 구조체 주소, 배열사이즈)

        free(tmp); //동적할당 시에는 항상 사용 후 할당을 풀어줘야 함.
        maxlength++; //데이터가 추가되었으므로 최대길이를 늘린다.
        count++; //카운트를 늘린다.
    }
```

```c
    struct CHECKCNT checkCnt; //1~45까지 중복체크 할 구조체를 생성.
    memset(&checkCnt, 0, sizeof(struct CHECKCNT)); //구조체 내의 int 배열 내 값을 모두 0으로 초기화 함. 쓰레기값 때문에 ++연산불가.

    for (int i = 0; i < count; i++) //전 배열이 다 돌아갈 때 까지 반복
    {
        struct NUMBERS n; //struct NUMBERS 형식의 구조체 n 선언. i열의 0~6까지 데이터 대입
        n.first = atoi(d[i].s[0]);
        n.second = atoi(d[i].s[1]);
        n.third = atoi(d[i].s[2]);
        n.forth = atoi(d[i].s[3]);
        n.fifth = atoi(d[i].s[4]);
        n.sixth = atoi(d[i].s[5]);
        n.bonum = atoi(d[i].s[6]);

        checkCnt.numbers[n.first - 1]++; //년도별 중복 카운트. 해당자리에 중복되면 카운트가 올라간다.
        checkCnt.numbers[n.second - 1]++;
        checkCnt.numbers[n.third - 1]++;
        checkCnt.numbers[n.forth - 1]++;
        checkCnt.numbers[n.fifth - 1]++;
        checkCnt.numbers[n.sixth - 1]++;
        checkCnt.numbers[n.bonum - 1]++;

        totalCnt.numbers[n.first - 1]++; //전회차 중복 카운트. 해당자리에 중복되면 카운트가 올라간다.
        totalCnt.numbers[n.second - 1]++;
        totalCnt.numbers[n.third - 1]++;
        totalCnt.numbers[n.forth - 1]++;
        totalCnt.numbers[n.fifth - 1]++;
        totalCnt.numbers[n.sixth - 1]++;
        totalCnt.numbers[n.bonum - 1]++;

        printf("%02d회차 >> %02d %02d %02d %02d %02d %02d bonus %02d\n", //각 숫자마다 중복된 카운트를 보여줌.
            i + 1, n.first, n.second, n.third, n.forth, n.fifth, n.sixth, n.bonum);
    }
    printf("\n");
    printf("%d년도의 결과 \n", year);

    int ltNum[45] = {0,}; //로또 번호 정렬을 위한 로또 번호 배열 초기화
    for (int i = 0; i < 45; i++) //로또 넘버 전체를 +1 해주기 위함..(초기화해서)
    {
        ltNum[i] = i + 1;
    }

    printf("-------+-----------\n");
    printf("¦ 번호 ¦ 당첨횟수 ¦\n");
    printf("-------+-----------\n");
    for (int i = 0; i < 45; i++)
    {
        printf("¦  %02d  ¦    %02d    ¦\n", ltNum[i], checkCnt.numbers[i]); //02는 두자리로 정렬.
    }
    printf("-------+-----------\n");
    printf("\n");
    fclose(fs); //파일을 가져와서 열었으면 항상 다 쓰고 닫아줘야 함.
}
```

```c
int lottoNum(int num, int* counts, int* numbers, int length) ///??
{
    int result = 0;
    int offset = 0;
    for (int i = 0; i < length; i++)
    {
        if (offset <= num && num < offset + counts[i])
        {
            result = numbers[i];
        }
        offset += counts[i];
    }
    return result;
}

struct CHECKCNT lottoPick(int totalSetCnt, int* ltNum, int length) // totalSetCnt 가중치 합
{
    struct CHECKCNT recordCnt; //시뮬레이션 돌린 번호 당첨 횟수 기록. 중복카운트 기록.
    memset(&recordCnt, 0, sizeof(struct CHECKCNT)); //구조체 내의 int 배열 내 값을 모두 0으로 초기화 함. 쓰레기값 때문에 ++연산불가.


    struct LTNUM pickNum; //랜덤번호 넣을 구조체(배열) ltNum 선언
    memset(&pickNum, 0, sizeof(struct LTNUM)); //배열 사이즈만큼 ltNum 첫위치부터 0 으로 초기화

    int wCnt = 0; //?? 중복카운트 돌아가는 횟수 카운트?
    int simulationCnt = SETNUM; //시뮬레이션 돌리는 횟수
    while (wCnt < simulationCnt)
    {
    Refick:
        for (int i = 0; i < 7; i++) //0~6번까지 7회 반복하며 번호를 받는다.
        {
            int r = getRandom(totalSetCnt - 1); //확률 범위 값
            int ltNm = lottoNum(r, &totalCnt.numbers, ltNum, length); //랜덤 값으로 부터 로또 번호 역산
            //printf(" %d -> result : %d \n", r, ltNm); 지저분해서 출력안하기로 함
            //printf(" %d->%d,", r, ltNm);
            //printf(" %d /", r);

            //1회째에는 수행할 필요 없음 (첫번째 뽑는거니 겹치는게 없음)
            for (int j = 0; j < i; j++)
            {
                if (pickNum.num[j] == ltNm) //새로 뽑은 값이 이미 뽑은 값 중에 있다면
                {
                    //printf("중복발생");
                    goto Refick; //중단
                }
            }
            //}
            pickNum.num[i] = ltNm;
        }

        printf("\n%d, %d, %d, %d, %d, %d, %d\n",
            pickNum.num[0], pickNum.num[1], pickNum.num[2], pickNum.num[3], pickNum.num[4], pickNum.num[5], pickNum.num[6]); //1주치 로또번호 추첨 완성
        }

        printf("\n%d, %d, %d, %d, %d, %d, %d\n",
            pickNum.num[0], pickNum.num[1], pickNum.num[2], pickNum.num[3], pickNum.num[4], pickNum.num[5], pickNum.num[6]); //1주치 로또번호 추첨 완성

        //당첨번호 카운트 기록
        recordCnt.numbers[pickNum.num[0] - 1]++; //년도별 중복 카운트
        recordCnt.numbers[pickNum.num[1] - 1]++;
        recordCnt.numbers[pickNum.num[2] - 1]++;
        recordCnt.numbers[pickNum.num[3] - 1]++;
        recordCnt.numbers[pickNum.num[4] - 1]++;
        recordCnt.numbers[pickNum.num[5] - 1]++;
        recordCnt.numbers[pickNum.num[6] - 1]++;

        wCnt++;
    }

    return recordCnt;
}
```

```c
void Simulpick()
{
    int ltNum[45] = { 0, }; //로또 번호 정렬을 위한 로또 번호 배열 초기화
    for (int i = 0; i < 45; i++)
    {
        ltNum[i] = i + 1;
    }
    //통계결과를 바탕으로 확률 정의
    int totalSetCnt = 0; //2003~2021까지의 모든 추첨번호 갯수의 합. 토탈카운트
    int length = sizeof(totalCnt.numbers) / sizeof(totalCnt.numbers[0]); //추첨번호마다 차지하는 비율.
    //int array 므로 전체 사이즈 나누기 타입사이즈를 하면 배열의 갯수가 나온다
    for (int i = 0; i < length; i++) //토탈카운트 반복계산.
    {
        totalSetCnt += totalCnt.numbers[i];
    }
    printf("total Size %d\n", totalSetCnt); //전체 당첨 수 갯수 총합을 출력
    //랜덤하게 처리
    srand(time(NULL)); //이걸 적어주어야 난수가 생성된다.

    //로또번호 픽 시뮬레이션
    struct CHECKCNT resultCnt = lottoPick(totalSetCnt, &ltNum, length); //전체통계의 당첨번호 순서를 그대로 받아옴..왜?

    printf("-------+----------\n");
    printf("¦ 번호 ¦ 당첨횟수 ¦\n");
    printf("-------+----------\n");
    for (int i = 0; i < 45; i++)
    {
        printf("¦  %02d  ¦   %d    ¦\n", ltNum[i], resultCnt.numbers[i]); //넘버가 1부터 시작해야 하므로 +1. 02는 두자리로 정렬.
    }
    printf("-------+----------\n");
    printf("\n");
}
```

```c
struct CHECKCNT lottoPickEx() // totalSetCnt 가중치 합
{
    struct CHECKCNT recordCnt; //시뮬레이션 돌린 번호 당첨 횟수 기록. 중복카운트 기록.
    memset(&recordCnt, 0, sizeof(struct CHECKCNT)); //구조체 내의 int 배열 내 값을 모두 0으로 초기화 함. 쓰레기값 때문에 ++연산불가.


    struct LTNUM pickNum; //랜덤번호 넣을 구조체(배열) ltNum 선언
    memset(&pickNum, 0, sizeof(struct LTNUM)); //배열 사이즈만큼 ltNum 첫위치부터 0 으로 초기화

    int wCnt = 0; //?? 중복카운트 돌아가는 횟수 카운트?
    int simulationCnt = SETNUM; //시뮬레이션 돌리는 횟수
    while (wCnt < simulationCnt)
    {
        for (int i = 0; i < 7; i++) //0~6번까지 7회 반복하며 번호를 받는다.
        {
            int r = getRandom(45) + 1; //확률 범위 값
            //if (i > 0) //1회째에는 수행할 필요 없음 (첫번째 뽑는거니 겹치는게 없음)
            //{
            pickNum.num[i] = r;

            for (int j = 0; j < i; j++)
            {
                if (pickNum.num[j] == r) //새로 뽑은 값이 이미 뽑은 값 중에 있다면
                {
                    //printf("중복발생");
                    i--;
                    break;
                }
            }
            //}

        }

        printf("%d >> %02d|%02d|%02d|%02d|%02d|%02d|%02d|\n", wCnt,
            pickNum.num[0], pickNum.num[1], pickNum.num[2], pickNum.num[3], pickNum.num[4], pickNum.num[5], pickNum.num[6]); //1주치 로또번호 추첨 완성

        //당첨번호 카운트 기록
        recordCnt.numbers[pickNum.num[0] - 1]++; //년도별 중복 카운트
        recordCnt.numbers[pickNum.num[1] - 1]++;
        recordCnt.numbers[pickNum.num[2] - 1]++;
        recordCnt.numbers[pickNum.num[3] - 1]++;
        recordCnt.numbers[pickNum.num[4] - 1]++;
        recordCnt.numbers[pickNum.num[5] - 1]++;
        recordCnt.numbers[pickNum.num[6] - 1]++;

        wCnt++;
    }

    return recordCnt;
}
```

```c
void NolmalPick()
{
    int ltNum[45] = { 0, }; //로또 번호 정렬을 위한 로또 번호 배열 초기화
    for (int i = 0; i < 45; i++)
    {
        ltNum[i] = i + 1;
    }

    //랜덤하게 처리
    srand(time(NULL)); //이걸 적어주어야 난수가 생성된다.

    //로또번호 픽 시뮬레이션
    struct CHECKCNT resultCnt = lottoPickEx(); //전체통계의 당첨번호 순서를 그대로 받아옴..왜?

    //bubbleSortWithNum(&resultCnt, &ltNum, length);

    printf("-------+----------\n");
    printf("¦ 번호 ¦ 당첨횟수 ¦\n");
    printf("-------+----------\n");
    for (int i = 0; i < 45; i++)
    {
        printf("¦  %02d  ¦    %02d    ¦\n", ltNum[i], resultCnt.numbers[i]); //넘버가 1부터 시작해야 하므로 +1. 02는 두자리로 정렬.
    }
    printf("-------+----------\n");
    printf("\n");
}
```

```c
int main()
{
    memset(&totalCnt, 0, sizeof(struct CHECKCNT)); //전체 증복카운트 구조체의 배열 내 값 모두 초기화.
                                                   //초기화 할 구조체, 초기화 값, 구조체 사이즈.
    int startYear = 2003; //시작년도 설정
    for (int i = 0; i < 20; i++) //20회 돌린다.
    {
        procYear(startYear); //시작년도부터 연도처리 함수로 처리한다.
        startYear++; //함수 한 번 돌리고 연도 증가시킴.
    }

    int ltNum[45] = { 0, }; //로또 번호 정렬을 위한 로또 번호 배열 초기화
    for (int i = 0; i < 45; i++)
    {
        ltNum[i] = i + 1;
    }

    printf("전체 결과\n"); //전체회자의 각 당첨수의 중복통계 표시. 자료가 많아서 반복문으로.
    printf("-------+----------\n");
    printf("¦ 번호 ¦ 당첨횟수 ¦\n");
    printf("-------+----------\n");
    for (int i = 0; i < 45; i++)
    {
        printf("¦  %02d  ¦    %d    ¦\n",ltNum[i], totalCnt.numbers[i]); //넘버가 1부터 시작해야 하므로 +1. 02는 두자리로 정렬.
    }
    printf("-------+----------\n");
    printf("\n");
    //Simulpick(); //가중치 적용 현실로또 랜덤함수
    NolmalPick(); //가중치 없는 이상적인 랜덤함수
    return 0;
}
```

코드 수행 결과

2003년도 로또 번호를 처리합니다.

01회차 >> 10 14 30 31 33 37 bonus 19
02회차 >> 17 21 31 37 40 44 bonus 07
03회차 >> 01 08 21 27 36 39 bonus 37
04회차 >> 07 08 14 32 33 39 bonus 42
05회차 >> 02 04 15 16 20 29 bonus 01
06회차 >> 02 03 11 16 26 44 bonus 35
07회차 >> 02 10 12 15 22 44 bonus 01
08회차 >> 04 07 16 19 33 40 bonus 30
09회차 >> 06 10 18 26 37 38 bonus 03
10회차 >> 14 17 26 31 36 45 bonus 27
11회차 >> 08 13 15 23 31 38 bonus 39
12회차 >> 01 10 20 27 33 35 bonus 17
13회차 >> 03 11 21 30 38 45 bonus 39
14회차 >> 06 31 35 38 39 44 bonus 01
15회차 >> 17 18 19 21 23 32 bonus 01
16회차 >> 13 20 23 35 38 43 bonus 34
17회차 >> 07 13 18 19 25 26 bonus 06
18회차 >> 06 07 13 15 21 43 bonus 08
19회차 >> 16 17 22 30 37 43 bonus 36
20회차 >> 07 27 30 33 35 37 bonus 42
21회차 >> 01 10 23 26 28 40 bonus 31
22회차 >> 02 03 11 26 37 43 bonus 39
23회차 >> 09 26 35 37 40 42 bonus 02
24회차 >> 04 07 32 33 40 41 bonus 09
25회차 >> 06 14 19 25 34 44 bonus 11
26회차 >> 07 09 18 23 28 35 bonus 32
27회차 >> 08 17 20 35 36 44 bonus 04
28회차 >> 01 05 13 34 39 40 bonus 11
29회차 >> 09 18 23 25 35 37 bonus 01
30회차 >> 01 20 26 28 37 43 bonus 27
31회차 >> 04 05 07 18 20 25 bonus 31
32회차 >> 02 04 21 26 43 44 bonus 16
33회차 >> 07 08 27 29 36 43 bonus 06
34회차 >> 05 13 17 18 33 42 bonus 44
35회차 >> 04 05 06 08 17 39 bonus 25
36회차 >> 06 12 17 18 31 32 bonus 21
37회차 >> 10 14 18 20 23 30 bonus 41
38회차 >> 06 30 38 39 40 43 bonus 26
39회차 >> 03 12 13 19 32 35 bonus 29
40회차 >> 03 04 09 17 32 37 bonus 01
41회차 >> 06 07 24 37 38 40 bonus 33
42회차 >> 03 04 16 30 31 37 bonus 13
43회차 >> 02 06 12 31 33 40 bonus 15
44회차 >> 22 23 25 37 38 42 bonus 26
45회차 >> 02 11 21 25 39 45 bonus 44
46회차 >> 01 07 36 37 41 42 bonus 14
47회차 >> 09 25 30 33 41 44 bonus 06
48회차 >> 02 04 16 17 36 39 bonus 14
49회차 >> 08 19 25 34 37 39 bonus 09
50회차 >> 02 09 16 25 26 40 bonus 42
51회차 >> 14 15 26 27 40 42 bonus 34
52회차 >> 16 24 29 40 41 42 bonus 03
53회차 >> 14 27 30 31 40 42 bonus 02
54회차 >> 11 16 19 21 27 31 bonus 30
55회차 >> 09 13 21 25 32 42 bonus 02
56회차 >> 10 23 29 33 37 40 bonus 16

2003년도의 결과

번호	당첨횟수
01	12
02	12
03	08
04	10
05	04
06	12
07	12
08	08
09	09
10	07
11	07
12	04
13	09
14	09
15	06
16	11
17	11
18	09
19	08
20	07
21	10
22	03
23	09
24	02
25	11
26	13
27	09
28	03
29	05
30	11
31	12
32	08
33	11
34	05
35	10
36	07
37	17
38	08
39	12
40	14
41	05
42	11
43	08
44	10
45	03

전체 결과

번호	당첨횟수
01	174
02	163
03	161
04	168
05	153
06	157
07	159
08	155
09	134
10	164
11	161
12	171
13	173
14	165
15	159
16	160
17	173
18	170
19	153
20	168
21	160
22	131
23	141
24	163
25	151
26	164
27	174
28	144
29	143
30	151
31	161
32	143
33	170
34	178
35	156
36	156
37	163
38	163
39	168
40	163
41	143
42	157
43	180
44	158
45	160

(왼쪽>>>오른쪽)

csv파일에서
데이터를 불러와서
번호순대로 당첨된
횟수를 기록한다.

전체파일에 대한
각 숫자별 횟수를
기록 한 후에는
전체 회차에 대한
당첨회수도 표로
만들어서 출력한다.

total Size 7182

17, 31, 20, 18, 2, 11, 37

28, 23, 37, 9, 17, 5, 43

34, 19, 27, 41, 35, 20, 12

17, 20, 14, 1, 4, 28, 6

42, 14, 23, 8, 28, 18, 12

35, 7, 34, 23, 17, 8, 14

4, 24, 15, 9, 16, 29, 39

2, 8, 20, 12, 7, 4, 9

9, 11, 44, 41, 27, 15, 22

41, 23, 43, 18, 22, 8, 34

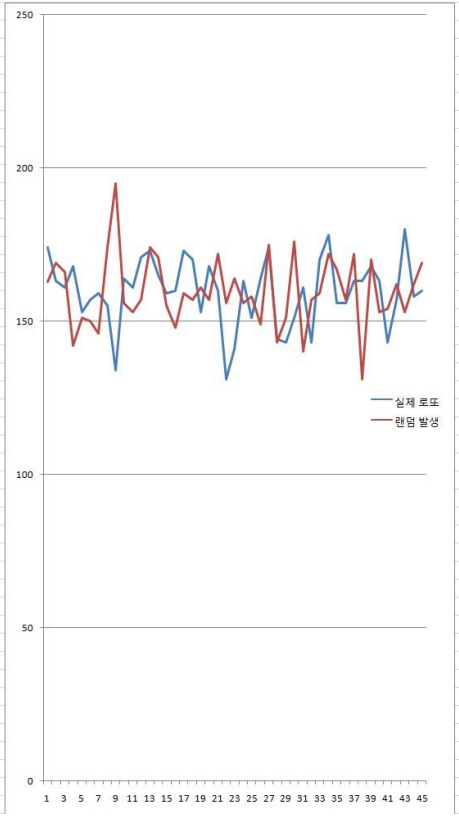
번호	당첨횟수
01	01
02	02
03	00
04	03
05	01
06	01
07	02
08	04
09	04
10	00
11	02
12	03
13	00
14	03
15	02
16	01
17	04
18	03
19	01
20	04
21	00
22	02
23	04
24	01
25	00
26	00
27	02
28	03
29	01
30	00
31	01
32	00
33	00
34	03
35	02
36	00
37	02
38	00
39	01
40	00
41	03
42	01
43	02
44	01
45	00

그 후에는 실제
당첨된 각 수의
확률을 적용한
함수를
이용하여
가상의
로또추첨
시뮬레이션을
10회 시행했다.

시행결과 역시
번호순대로
도표로 만들어
출력했다.

수행결과 그래프 (1) : 실제 로또와 랜덤 로또 통계 비교

	실제 로또	랜덤 발생	1026회 시행
1	174	163	
2	163	169	
3	161	166	
4	168	142	
5	153	151	
6	157	150	
7	159	146	
8	155	174	
9	134	195	
10	164	156	
11	161	153	
12	171	157	
13	173	174	
14	165	171	
15	159	155	
16	160	148	
17	173	159	
18	170	157	
19	153	161	
20	168	157	
21	160	172	
22	131	156	
23	141	164	
24	163	156	
25	151	158	
26	164	149	
27	174	175	
28	144	143	
29	143	151	
30	151	176	
31	161	140	
32	143	157	
33	170	159	
34	178	172	
35	156	167	
36	156	157	
37	163	172	
38	163	131	
39	168	170	
40	163	153	
41	143	154	
42	157	162	
43	180	153	
44	158	162	
45	160	169	



실제 로또란?

- 1회~1026회까지의 실제 로또 당첨번호마다 당첨빈도를 계산하여 그 수치를 확률범위로 삼음.
- 확률이 높을 수록 그 범위가 높아서 당첨수로 뽑히는 빈도가 높아짐. 이는 다른 수에 비해서 가중치를 받는다고 가정함.
- 이 가중치는 각 번호의 공에 따라서 물리적 가중치가 다르다고 가정하였음. (공이 매번 바뀐대도 치우침이 발생)

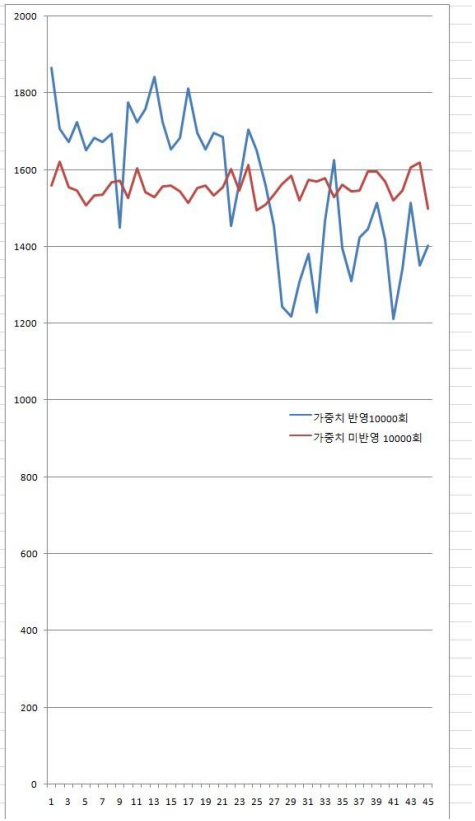
랜덤 로또란?

- 아무런 제한요인이 없는 난수발생으로만 당첨번호 결정함.
- 완전한 무작위 추첨이기 때문에 외부요인의 개입이 없음.
- 실제 로또와 같은 횟수인 1026회로 시행하여 비교하였음.

>> 두 가지 모두 규칙이 없는 당첨패턴을 보였음.
이 결과에서는 실제 로또가 랜덤 로또와 같이 특정한 패턴이 없으므로 물리적인 변인들이 잘 통제되고 있는 것으로 보여짐.

수행결과 그래프 (2) : 확률 가중치 유무에 따른 통계 비교

가중치 반영10000회	가중치 미반영 10000회
1865	1559
1706	1621
1672	1555
1723	1546
1650	1507
1683	1534
1673	1536
1694	1567
1449	1571
1775	1528
1723	1603
1758	1543
1841	1529
1723	1556
1652	1560
1682	1544
1812	1514
1695	1552
1652	1559
1695	1534
1685	1555
1455	1602
1566	1546
1705	1612
1651	1495
1558	1511
1453	1536
1245	1564
1219	1584
1309	1521
1381	1573
1229	1570
1467	1578
1626	1530
1396	1562
1311	1545
1423	1546
1446	1596
1514	1596
1418	1569
1212	1521
1342	1546
1513	1606
1351	1619
1402	1499



가중치 반영 시뮬레이션? (실제 로또)

- 수행결과(1)의 실제 로또와 동일한 확률 가중치로 수행 횟수를 10000회로 늘림.
- 확률 가중치를 많이 받은 공이 계속 많이 당첨되는 결과를 예상하였으나 이와 다르게 그래프의 쏠림이 발생하였음.
- 추첨횟수가 무한대로 늘어났을 경우까지 예상할 수는 없으나 10000회 시점에서는 번호마다 확률 차이를 보임.

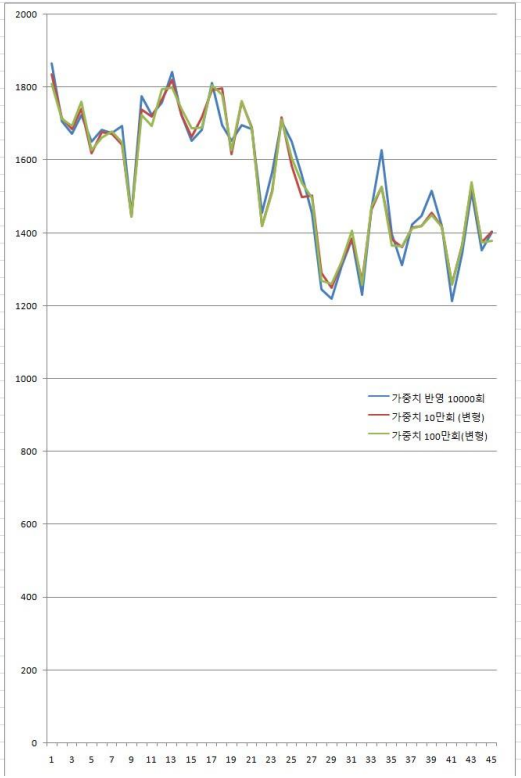
가중치 미반영 시뮬레이션? (랜덤 로또)

- 확률 가중치를 가정하지 않은 순수한 난수발생에 의한 시뮬레이션으로 수행결과(1)의 랜덤 로또와 동일함.
- 추첨횟수가 적을 때에는 불규칙한 패턴을 보였으나 회차가 늘어나니 오히려 더 균등한 확률 패턴을 보임.
- 추첨횟수가 무한대로 늘어났을 경우를 예상 할 수는 없으나 10000회 시점에서 번호간 확률이 서로 유사해 짐.

>> 수행횟수가 늘어나자 실제 로또와 랜덤 로또의 통계 그래프가 차이를 보이기 시작함. 매우 큰 차이는 아니지만 이것으로 실제로 로또가 추첨 될 때에는 어떤 요인이든지 확률에 작용하는 요인이 있다는 것을 의심 해 볼 수 있음.

수행결과 그래프(3) : 수행회차 증가 따른 통계결과 비교

가중치 반영 10000회	가중치 10만회 (변형)	가중치 100만회(변형)
1865	1836	1811
1706	1717	1715
1672	1686	1695
1723	1742	1761
1650	1620	1627
1683	1677	1662
1673	1673	1681
1694	1643	1649
1449	1445	1447
1775	1739	1725
1723	1720	1695
1758	1767	1796
1841	1821	1801
1723	1724	1741
1652	1664	1687
1682	1718	1690
1812	1792	1806
1695	1798	1781
1652	1617	1629
1695	1761	1762
1685	1689	1687
1455	1421	1419
1566	1516	1511
1705	1718	1712
1651	1585	1608
1558	1499	1537
1453	1503	1497
1245	1290	1270
1219	1250	1261
1309	1322	1323
1381	1385	1407
1229	1270	1259
1467	1465	1474
1626	1528	1528
1396	1384	1367
1311	1361	1365
1423	1416	1413
1446	1420	1419
1514	1457	1450
1418	1418	1418
1212	1260	1259
1342	1368	1367
1513	1534	1540
1351	1376	1375
1402	1404	1378



가중치반영 10만회	가중치반영 100만회
18362	181050
17169	171454
16859	169531
17421	176075
16204	162731
16771	166183
16725	168057
16425	164872
14448	144651
17394	172500
17200	169484
17669	179566
18214	180069
17238	174126
16638	168714
17175	168995
17924	180585
17984	178065
16167	162871
17612	176226
16893	168692
14208	141947
15159	151124
17183	171150
15854	160767
14993	153704
15032	149707
12901	126995
12503	126085
13218	132281
13845	140661
12704	125902
14651	147360
15280	152770
13835	136698
13613	136483
14163	141303
14199	141917
14567	144999
14183	141777
12600	125912
13680	136664
15342	153959
13756	137521
14039	137817

수행횟수 별 로또 확률의 변화?

- 확률가중치를 반영한 현실 로또 시뮬레이션을 10000회, 10만회, 100만회 수행하여 결과값을 얻음.
- 10만회와 100만회의 결과값은 그래프 비교의 용이함을 위하여 각각 나눈 후 소수점 이하는 버림.
- 세 데이터를 비교하니 약간의 빈도 변화가 발생하는 구간도 있었으나 전반적으로는 거의 비슷한 그래프 양상을 보임.

>> 추첨회차가 어느정도 늘어나도 그래프의 형태가 완만하지 않고 한 쪽으로 쏠리는 것이 유지된다는 것을 알 수 있음.

>>이는 앞서 수행결과(2)에서 볼 수 있던 랜덤 로또와는 다른 결과를 보여주는 것임.

최종 결론

1. 실제 로또 추첨 시에 5세트의 공을 랜덤하게 사용하거나 차이를 검사하는 등의 노력에도 불구하고 각 번호 간의 추첨 확률은 달랐다. 그 근거는 추첨 빈도의 차이이다.

2. 이 빈도의 차이값을 확률에 반영하여 번호마다 확률 가중치가 다른 로또 시뮬레이션을 구축하여 수행하였을 경우에 역시나 번호 간에 당첨빈도 차이가 발생하며, 시뮬레이션 수행회차가 늘어 날수록 더욱 그 패턴이 고착화되었다.

3. 이와는 다르게 외부의 요인을 배제하고 난수를 발생시켜 로또번호를 추첨하는 로또 시뮬레이션은 회차가 늘어날 수록 오히려 들쭉날쭉한 패턴은 사라지고 번호마다 확률이 서로 균일해지는 그래프 패턴이 발생했다.

4. 두 시뮬레이션을 비교하였을 때, 현실에서 발생하는 로또 추첨은 컴퓨터 프로그램에서가 아닌 기계와 공, 바람, 중력 등의 여러가지 물리적인 환경에서 추첨을 시행하므로 아무리 표준화, 무작위화 등으로 변인을 조절한다고 해도 완전히 외부요인을 배제한 랜덤 시뮬레이션과는 다른 결과를 나타냈다.

5. 많은 비약으로 시작한 시뮬레이션이지만 본 프로그램을 구현하여 시행 해 본 결과, 로또 1등 당첨확률을 올리려면 **매 회차마다 구매 직전까지의 번호들을 모두 파악 한 후 그 중 가장 많이 나오는 번호 6개를 구매하는 것**이 당첨 확률을 높이는 방법 중의 하나가 될 수 있다는 결론을 낼 수