PORTFOLIO

당첨 확률을 올린!
로또번호 추첨 프로그램

경북산업직업전문학교 서정빈
-통합 응용 SW 개발자(C#,JAVA)과정-

기획 의도

JAVA와 C#에 대해 배우면서 RANDOM클래스를 통해 난수를 추출하거나 확률계산이 가능하다는 것이 매우 흥미롭게 다가왔습니다.
초기에는 7개의 난수를 추출해 임의의 수를 뽑는데에만 그쳤으나,
과정이 진행될수록 여기에 조건문과 반복문을 활용해 당첨 확률을 더 높일 수 있겠다는 생각이 들어 이 프로그램을 기획하게 되었습니다.

목차

CONTENTS

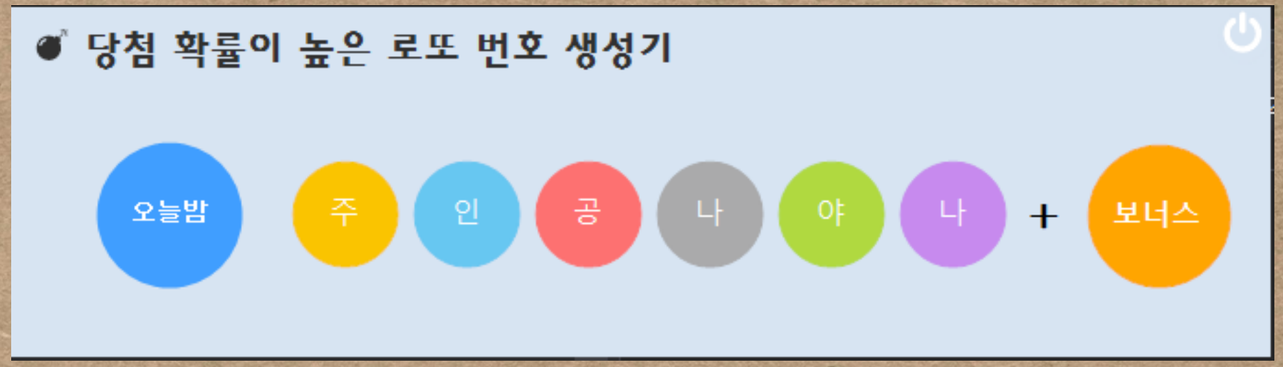
로또번호 생성기 조건 및 작동모습

OVERVIEW

좌측의 파란 버튼을 누르면 여섯개의 번호가 오름차순으로 정렬되고, 랜덤의 보너스 번호 하나가 추가로 출력됩니다.

이 프로그램은 단순히 임의의 숫자 7개를 출력하는 것이 아니라, 몇 가지의 법칙을 가지고 있습니다. 이 법칙을 간단하게 설명하고 어떤 코드로 짜여졌는지 살펴보겠습니다.

1. 첫번째 자리(1~34)/두번째자리(2~36)/세번째자리(3~39)를 벗어난 당첨번호는 없었다.
2. 지금까지 합계가 48보다 작거나, 238보다 큰 당첨 번호는 없었다
3. 고저차가 41-44일 경우가 확률이 가장 높다



첫번째 조건 구현 및 개발과정

랜덤 함수와 반복문을 생성해 난수를 추출하는 것 까지는 간단하다. 당첨 확률을 올리기 위해 각 각의 숫자에 범위를 지정하여, 그 범위 안에서만 숫자가 나올 수 있도록 구현했다.

배열과 조건문을 사용해서 각 번호에 출력될 숫자의 범위를 지정했으며, 해당 조건을 만족하지 못할 경우 이전 단계로 돌아갈 수 있도록 코드를 작성했다. 예를 들면 첫번째 숫자가 조건을 만족하고 두번째 숫자가 조건을 만족하지 못했을 경우, 첫번째에 출력된 숫자는 그대로 두고 두번째 숫자를 다시 출력할 수 있도록 i의 값을 지정했다.

```
NumBtn.Text = "당첨!";
int[] a = new int[7];
Random r = new Random();
int sum = 0;
```

```
while (true)
{
    for (int i = 0; i < 7; i++)
    {

        a[i] = r.Next(1, 46);
        sum += a[i];

        for (int j = 0; j < i; j++)
        {
            if (a[i] == a[j])
            {
                sum -= a[i];
                i--;
            }

            if (a[0] > 34)
            {
                Console.WriteLine("sum2" + sum + "a[0]" + a[0]);
                i = -1;
                sum = 0;
                break;
            }
            if (i == 1 && (a[1] < 2) || (a[1] > 36))
            {
                Console.WriteLine("sum2" + sum + "a[1]" + a[1]);
                i = 0;
                sum = 0;
                break;
            }
            if (i == 2 && (a[2] < 3) || (a[2] > 39))
            {
                Console.WriteLine("sum2" + sum + "a[2]" + a[2]);
                i = 1;
                sum = 0;
                break;
            }
```

두번째 조건 구현 및 개발과정

두번째 조건(출력된 숫자들의 합이 48보다 크고 238보다 작음)을 구현하던 중 따로 조건을 지정하지 않아도 해당 범위를 벗어나는 숫자가 나오기 힘들다는 것을 알게 되었다. 오히려 조건을 지정하는게 프로그램을 더 느리게 한다는 것을 알게 되었고, 코드 자체를 삭제하는 대신 조건을 좀 더 디테일하게 설정하기로 했다.

'합계 100~200사이가 전체의 80%로 10번 추첨시 8번이나 이 구간에서 당첨번호 조합이 출현하고 있다'는 통계 결과를 찾았고, 해당 통계를 코드로 구현하기 위해 위와 마찬가지로 조건 문을 사용해 해당 조건을 만족하지 못할 경우 다시 처음으로 돌아가 숫자를 재추출하도록 설정했다.

```
    if (sum < 100 || sum > 200)
    {
        Console.WriteLine("sum:(sum < 100 || sum > 200)" + sum);
        sum = 0;
        continue;
    }
    else
        break;
}
Console.WriteLine("sum:" + sum);
```

세번째 조건 구현 및 개발과정

세번째 조건(고저차가 41~44의 숫자가 가장 당첨확률이 높다)을 만족시키기 위해 이번에도 조건문을 사용했다. 해당 조건을 만족시키지 못할 경우 goto 문을 사용해 랜덤 숫자를 출력하는 가장 윗부분으로 돌아갈 수 있도록 작성했다. 해당 조건을 만족하고 출력하는 과정에서 Refresh를 사용한 이유는 Form 제작 중 사용한 UI 버튼이 자동으로 바뀌지 않는 오류를 고치기 위해 사용한 일종의 '새로고침'이다.

```
if ((a[5] - a[0]) < 41 || (a[5] - a[0]) > 44)
{
    sum = 0;
    goto lotto;
}

for (int k = 0; k < 7; k++)
{
    buttonList[k].Text = (a[k].ToString());
    buttonList[k].Refresh();
}
```

```
private void NumBtn_Click(object sender, EventArgs e)
{
    Sunny.UI.UISymbolButton[] buttonList = new Sunny.UI.UISymbolButton[7];
    buttonList[0] = lotto_num1;
    buttonList[1] = lotto_num2;
    buttonList[2] = lotto_num3;
    buttonList[3] = lotto_num4;
    buttonList[4] = lotto_num5;
    buttonList[5] = lotto_num6;
    buttonList[6] = lottoBonusNum;

    NumBtn.Text = "당첨!";
    int[] a = new int[7];
    Random r = new Random();
    int sum = 0;

    //while (true)
    {
    lotto:
        while (true)
        {
            for (int i = 0; i < 7; i++)
            {

                a[i] = r.Next(1, 46);
                sum += a[i];

                for (int j = 0; j < i; j++)
```

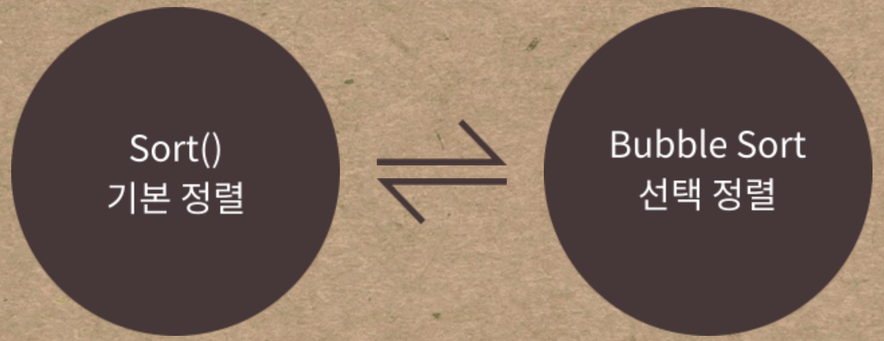
```
/*Array.Sort(a);*/
int temp = 0;
for (int i = 0; i < 6; i++)
{
    for (int j = i+1 ; j < 5; j++)
    {
        if(a[i] > a[j])
        {
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;
        }
    }

}
```

오류 개선

다음 과정은 출력된 난수를 순서대로 정렬하는 과정에 발생한 오류에 대한 수정 과정이다.

CONCEPT



Array.Sort();로 간단하게 숫자들을 오름차순으로 정렬할 수 있었다. 하지만 Sort()를 사용했을 경우 마지막 보너스 숫자까지 순차 정렬 되어, 보너스 번호는 상대적으로 큰 숫자 밖에 출력되지 않는 오류를 발견했다.

해당 오류를 개선하기 위해, Bubble Sort를 사용해서 정렬할 숫자들의 범위를 지정하고, 1번째에서 6번째 숫자까지만 인접한 숫자들끼리 비교해서 순차적으로 정렬할 수 있도록 코드를 수정했다.

논의 및 고찰

고저차 조건과 오름차순 정렬을 함께 사용하니, 자연스럽게 1번째 숫자는 1-3 사이의 수, 6번째 숫자는 43~ 45 사이의 숫자만 출력되는 오류를 발견했다. 당첨 확률이 높은 조건은 만족했을지 모르나, 확실히 부자연스러운 모습이었다. 정렬 조건을 만족하면서 조금 더 다양한 범위의 숫자를 출력할 수 있는 방법이 어떤 것이 있을까? 하고 생각하게 되었다.

또한 Nuget에서 UI를 사용하는 과정에서 발생한 여러 오류들이 아쉬움으로 남는다. 이번 프로그램에서는 Sunny Ui를 사용해 숫자가 출력되는 모습을 구현했는데, 정상적으로 코드를 작성했음에도 불구하고 숫자가 출력되는 버튼에 일일이 커서를 가져다대야만 숫자가 바뀌는 오류를 발견해 Refresh를 사용해 새로고침을 해주어야만 했다.

THANK YOU

감사합니다

M. 010.7271.6662 | E. jungbinibini@gmail.com