제1강 난수와 시간

1. 난수 발생 만들기

■ **난수의 필요성**

- 게임은 항상 상대가 있어야 함. 여기서 상대라는 의미는 '적'이 될 수도 있고, '나 자신'이 될 수도 있고, '어떤 목표'가 될 수도 있다.
- 컴퓨터 판단의 기본이 되기 때문임.

■ **난수 생성 함수**

1) rand() 함수

- srand()로 인해 생성된 값을 바탕으로 난수를 생성하는 함수
- 생성되는 난수의 범위를 지정하는 방법

: ★ rand() % (마지막 값 - 시작 값 + 1) + 시작 값

ex. 100~200 난수를 얻고 싶다면? rand()%101+100

```cpp
#include<iostream>

using namespace std;

int main()
{
    int randnum; // 난수
    for (int i = 0; i < 10; ++i)
    {
        randnum = rand();
        cout << randnum << endl;
    }
```

```cpp
#include<iostream>
#include<ctime>
using namespace std;

int main()
{
    srand((unsigned int)time(NULL));
    int randnum; // 난수
    for (int i = 0; i < 10; ++i)
    {
        randnum = rand();
        cout << randnum << endl;
    }
```

2) srand() 함수

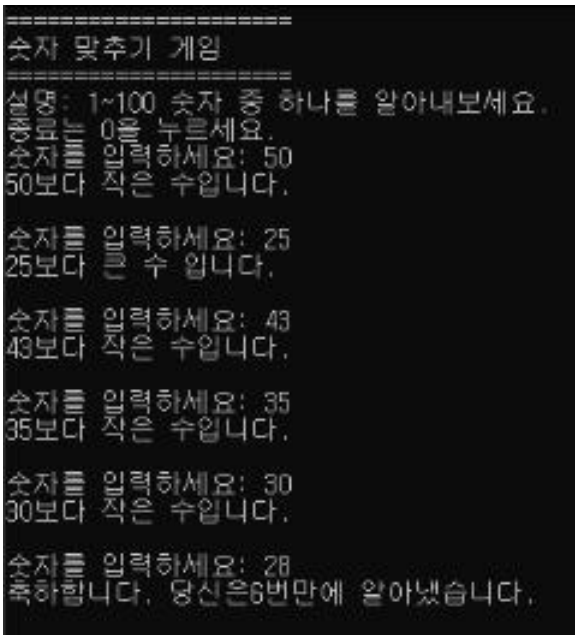
- 호출할 때 전달받는 인자를 기반으로 난수를 초기화하는 함수
- 정해진 순서가 안 나오도록 여러 개 만들어 매번 다른 난수표를 읽도록 만듦. 이 난수표를 선택하는 동작을 시드(Seed)라고 함.

=> ★기억할 것: srand((unsigned int)time(NULL));

■ **예제 게임**

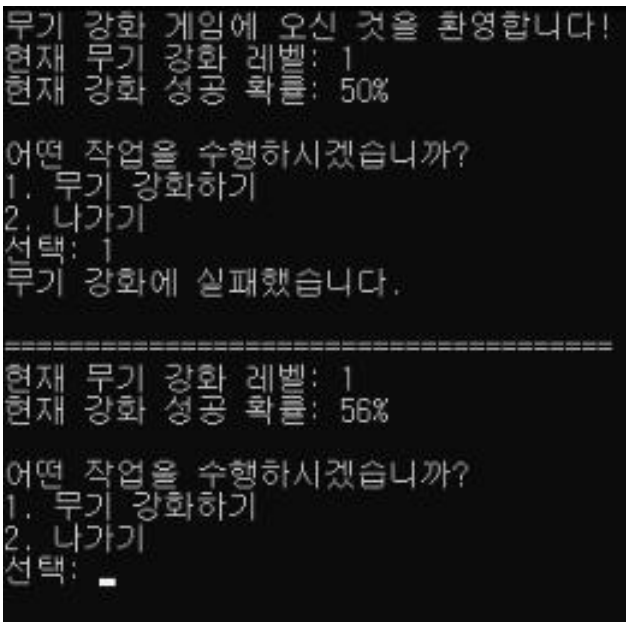
< ① 숫자 맞추기 게임(Up & down) >

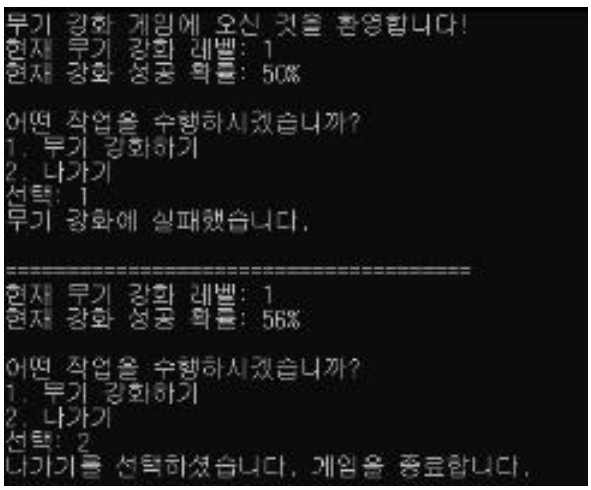
- 게임 설명: 컴퓨터가 1에서 100까지의 수를 고른다. 사용자가 예상 수를 선택하면, 컴퓨터가 '크다', '작다', '맞았다'를 알려 준다. 사용자가 선택한 수를 세어 몇 번만에 맞았는지를 알려 준다.
- 실행 예시



< ② 무기 강화 게임 >

- 게임설명
- 무기 강화 레벨은 1, 강화 성공 확률은 50%로 시작한다.
- 플레이어가 할 수 있는 것은 1로 무기를 강화하거나, 2로 나가는 것이다.
- 무기 강화에 성공한다면 다음 강화는 강화 성공 확률이 1~5% 감소하고, 실패한다면 다음 강화는 성공 확률이 5~10% 증가한다. (단, 1~5%와 5~10%는 랜덤하게 증감한다.)
- 2로 나간다면 게임이 종료된다.
- 실행 예시





< ③ 야구 게임 >

- 게임 설명: 1~9사이의 랜덤한 숫자를 3개 얻어온다. 단, 숫자는 중복되어서는 안된다. 3개의 숫자는 * * *의 형태로 출력되고, 이 3개의 숫자를 맞추는 게임이다. 사용자는 이 3개의 숫자를 맞출 때까지 계속해서 3개씩 숫자를 입력한다.

  만약, 맞춰야 할 숫자가 3 7 8 일 경우 사용자는 3개의 숫자를 입력한다.

  입력: 1 2 4를 입력했을 경우 1 2 4는 맞춰야할 숫자 중 아무것도 없으므로 Out을 출력한다.

  입력: 5 7 9를 입력했을 경우 7은 맞춰야 할 숫자 중 있고 위치도 같으므로 strike이다. 5 9는 없으므로 출력은 1strike 0ball을 출력한다.

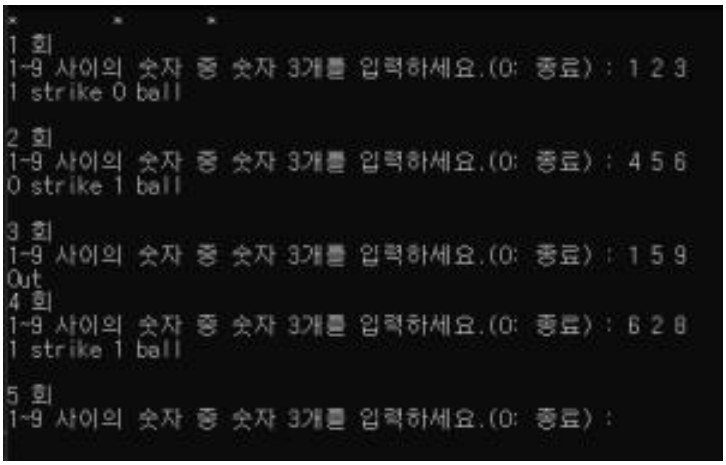
  입력: 3 8 6을 입력했을 경우 3은 1strike, 8은 숫자가 있지만 위치가 다르므로 1ball이므로 1strike 1ball이 출력된다.

이렇게 출력을 하고 입력을 받으면서 최종적으로 3개의 숫자를 자리까지 모두 일치하게 입력하면 게임이 종료된다.

※ 만약, 입력받은 숫자 3개 중 한 개라도 0이 있을 경우 게임을 종료함.

> ※ 여기서 알고 갈 것!
> \t: 특수 이스케이프 시퀀스로 탭 문자를 나타냄. 일정한 간격만큼 공백을 만들어줌.

- 실행 예시



- 2Player ver.

(초기값 세팅)

(실행 예시 화면)

-

> ※ 여기서 알고 갈 것!
> - 계속하려면 아무 키나 누르십시오...는 system("pause")를 활용하세요!
> - 화면을 지우는 함수는 system("cls")입니다.

■ **시간 관련 개념 (<time.h> 헤더)**

1) time()함수

- time(NULL): 1970년 1월 1일 0시(TimeZone: UTC) 이후부터 인자값까지 현재까지 흐른 초 수를 리턴해줌.
- srand 함수의 인자로 time(NULL)을 넘긴 후 rand 함수를 호출하면, 시간을 기준으로 초기화되는 난수를 생성할 수 있어 프로그램을 실행할 때마다 다른 난수 값을 얻을 수 있음.

2) localtime_s() 함수

- time함수 결과로 나온 초 값을 tm구조체 형식으로 리턴해줌. localtime_s(tm 구조체 타입 변수의 포인터, time_t 타입 변수의 포인터)

tm구조체 변수에 time_t 타입 변수의 정보를 저장함.

3) time_t 타입

- time 헤더에서 시간을 잘 다루기 위해서 만들어진 데이터 타입으로, 1970년 1월 1일 00시 00분부터 지금까지 초단위의 시간을 정수값으로 표현하는 데이터

4) struct tm 구조체

```
struct tm
{
    int tm_sec;   // seconds after the minute - [0, 60] including leap second
    int tm_min;   // minutes after the hour - [0, 59]
    int tm_hour;  // hours since midnight - [0, 23]
    int tm_mday;  // day of the month - [1, 31]
    int tm_mon;   // months since January - [0, 11]
    int tm_year;  // years since 1900
    int tm_wday;  // days since Sunday - [0, 6]
    int tm_yday;  // days since January 1 - [0, 365]
    int tm_isdst; // daylight savings time flag
};
```

※ 현재 시간 활용 예시

[main.cpp]

```
#include<iostream>
#include<time.h>
using namespace std;
int main(){
        time_t timer;
        struct tm t;
        timer = time(NULL);
        localtime_s(&t, &timer); // tm구조체 변수, time_t 타입 변수
        cout << t.tm_year + 1900 << "년 " << t.tm_mon + 1 << "월 " <<
        t.tm_mday << "일 " << t.tm_hour << "시 " << t.tm_min << "분 " << t.tm_sec << "초 ";
        if (t.tm_wday == 0) cout << "일요일";
        else if (t.tm_wday == 1) cout << "월요일";
        else if (t.tm_wday == 2) cout << "화요일";
        else if (t.tm_wday == 3)cout << "수요일";
        else if (t.tm_wday == 4)cout << "목요일";
        else if (t.tm_wday == 5)cout << "금요일";
        else cout << "토요일";
}
```