응용 예제

내용

1. System함수, 화면 글자 출력
2. 컴퓨터 피아노
3. 행렬 계산
4. 미분 방정식
5. 게임1
6. 게임2

1. System 함수

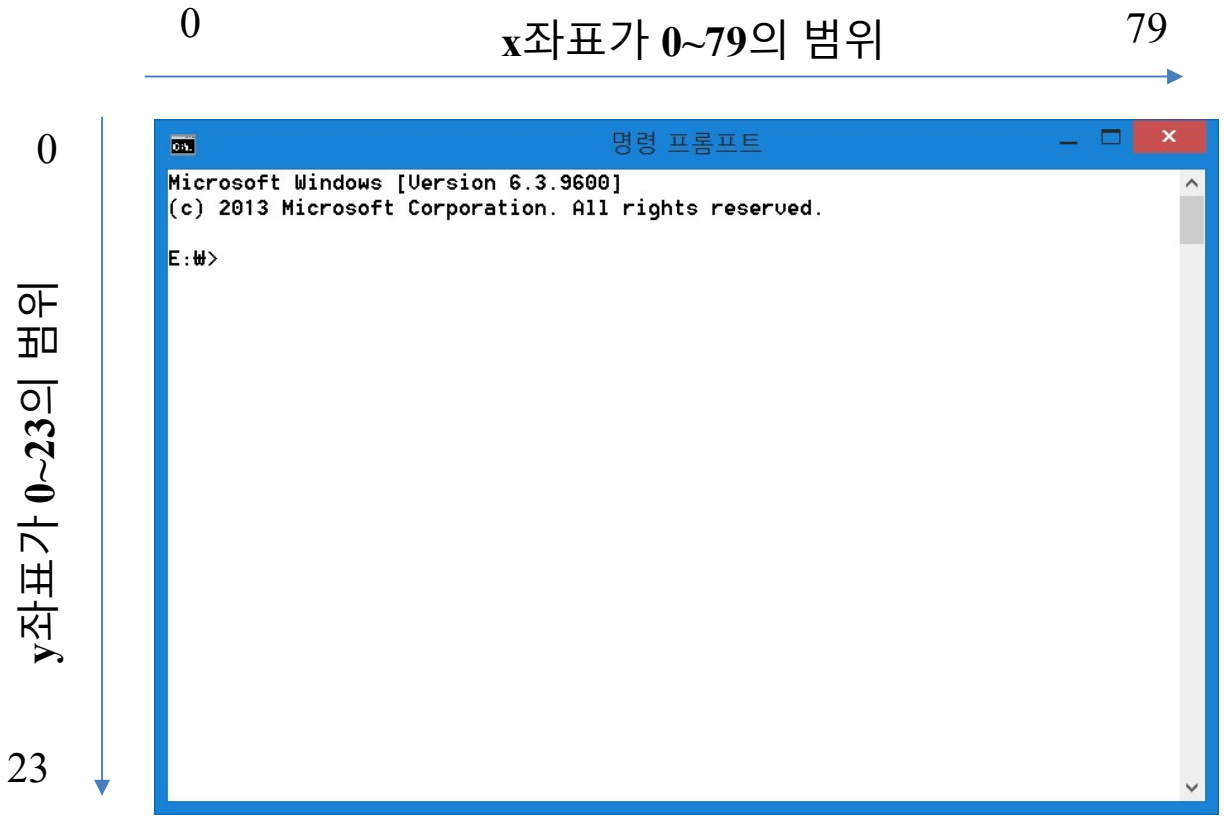
int system(const char *cmd); //시스템 명령 수행, stdlib.h에 정의

입력 매개 변수 리스트 : cmd 명령 문자열
반환 값 : 시스템 명령이 반환한 값

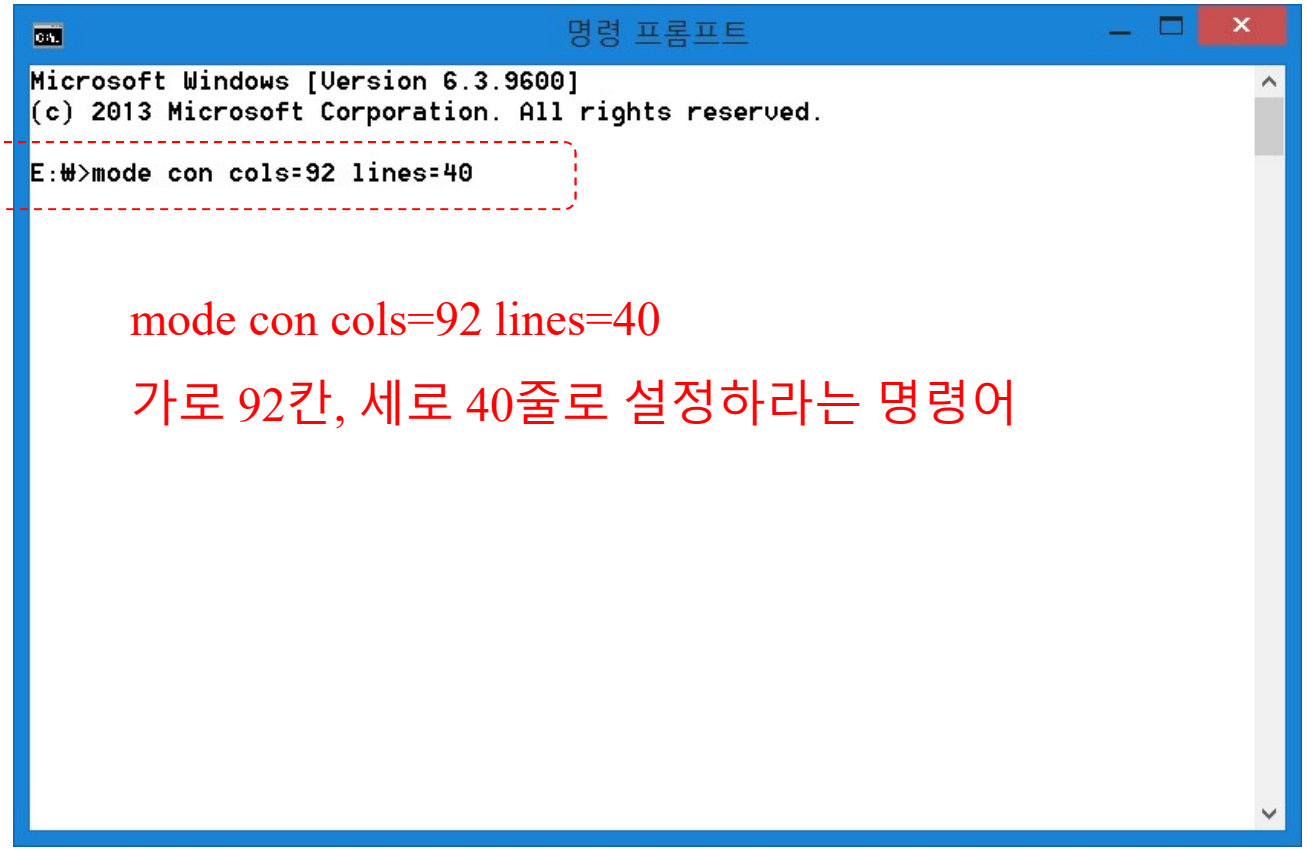
- system 함수는 명령어를 수행하는 함수입니다. 바꿔 말하면 프로그램을 실행하는 함수.
- system 함수는 명령을 수행하여 해당 프로세스가 종료하면 종료할 때의 값을 그대로 반환.

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

void main()
{
  system("dir"); // dir 실행.
  system("cls"); // 화면 지우기
}
```

기본 콘솔창에서의 좌표값



콘솔창 가로, 세로 폭 변경



mode con cols=92 lines=40

가로 92칸, 세로 40줄로 설정하라는 명령어

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

void main()
{
    system("mode con cols=92 lines=40");
}
```

커서의 위치를 제어하는 함수 gotoxy의 사용방법

void gotoxy(int x, int y);

함수인자	int x	화면에서의 가로 위치를 지정(1~80)
	int y	화면에서의 세로 위치를 지정(1~24)

Util.h

```
#pragma once

void gotoxy(int x, int y);
```

Util.cpp

```
#include “util.h”
#include<windows.h>

void gotoxy(int x, int y)
{
   COORD Pos = { x-1, y-1 };
   SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), Pos);
}
```

gotoxy 예제 1


```
#include<stdio.h>
#include “util.h”

void main(void)
{
   gotoxy(2,4);
   printf("Hello");
   gotoxy(40, 20);
   printf("Hello");
   return 0;
}
```

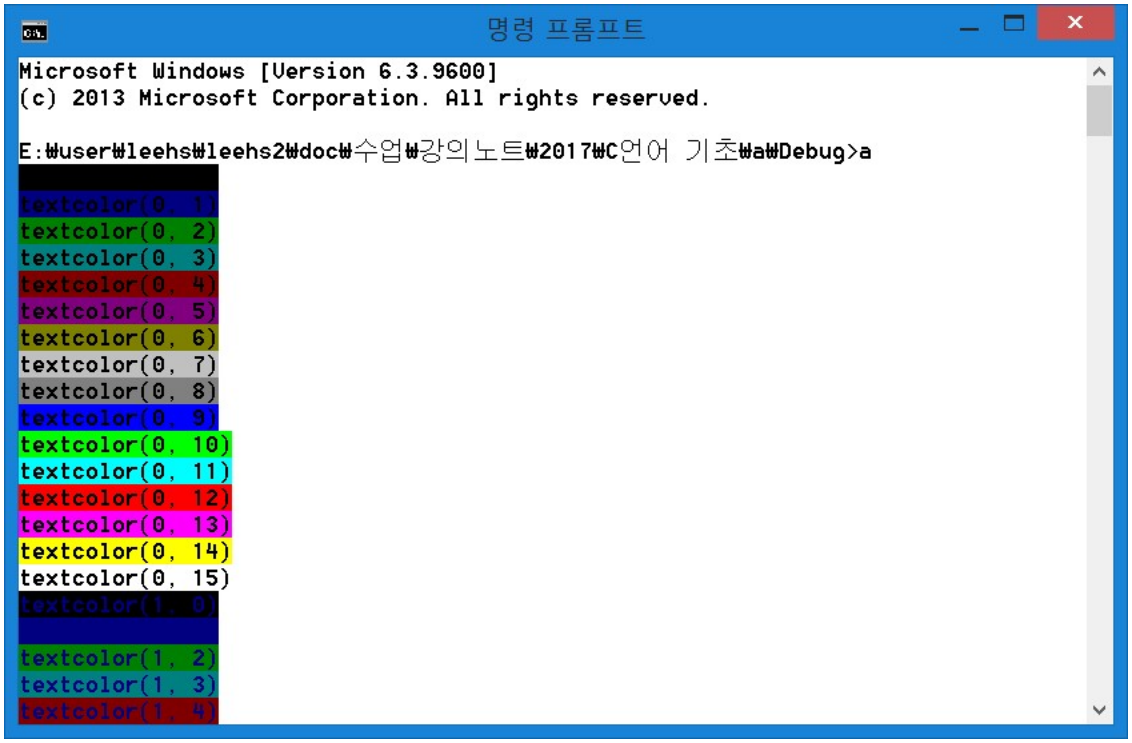
gotoxy 예제 2


```
#pragma once

#define UP 72
#define DOWN 80
#define LEFT 75
#define RIGHT 77

void gotoxy(int x, int y);
```

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include "util.h"

void main()
{
	int X = 44, Y = 22; //좌표값 초기화.
	int test(-1);
	system("mode con cols=92 lines=40");
	while (test!='q'){
		test = _getch(); //문자를 먼저 입력받습니다.
		switch (test){ //입력받은 문자에 따라서 스위치문을 구성합니다.
		case UP: gotoxy(X, Y); printf(" "); //기존에 출력된 문자는 지우고
			Y -= 1; //좌표를 이동시킨 뒤에
			gotoxy(X, Y);
			printf("●"); //새로운 위치에 문자를 출력해줍니다
			break; //이렇게 해서 마치 문자가 움직이는듯한
			//효과를 만들 수 있습니다.
		case DOWN: gotoxy(X, Y); printf(" ");
			Y += 1;
			gotoxy(X, Y);
			printf("●");
			break;
		case LEFT: gotoxy(X, Y);  printf(" ");
			X -= 1;
			gotoxy(X, Y);
			printf("●");
			break;
		case RIGHT: gotoxy(X, Y); printf(" ");
			X += 1;
			gotoxy(X, Y);
			printf("●");
			break;
		}
		gotoxy(80, 38);
		printf("%2d %2d", X, Y); //변수 X,Y는 좌표값이며
		//이를 출력시키는것으로 "좌표 측량"이 가능합니다!
	}
}
```

글자 색 바꾸기

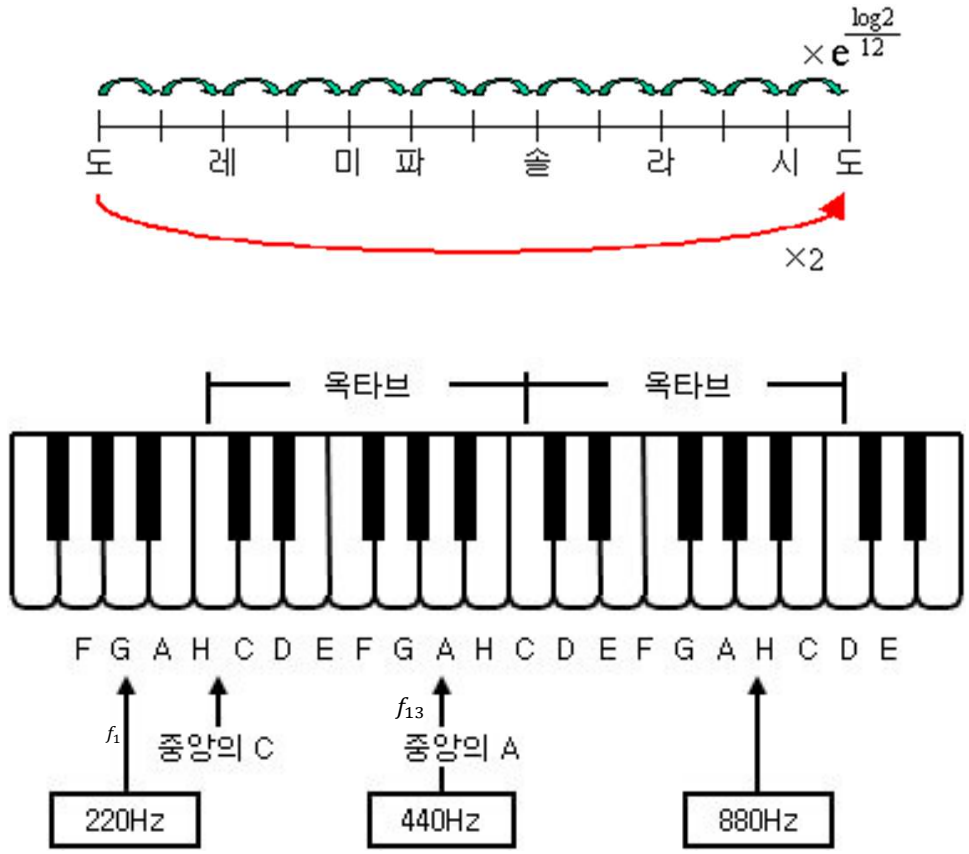

```
void textcolor(int foreground, int background)
{
    int color = foreground + background * 16;
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), color);
}
```



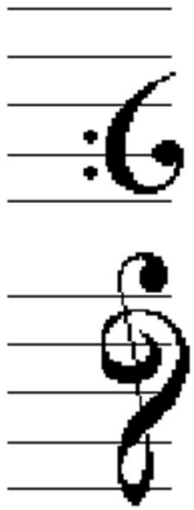
2. 컴퓨터 피아노 : 음계와 주파수

계명		평균율		순정율 (프롤레메우스 음율)		순정율 (피타고라스 음율)	
도	unison (prime)	1	1	1	1	1	1
도#	semitone	$(\sqrt[12]{2})^1$	1.05946	$\frac{16}{15}$	1.06667	$\frac{256}{243}$	1.0535
레	whole tone	$(\sqrt[12]{2})^2$	1.12246	$\frac{9}{8}$	1.125	$\frac{9}{8}$	1.125
레#	minor third	$(\sqrt[12]{2})^3$	1.18921	$\frac{6}{5}$	1.2	$\frac{32}{27}$	1.18519
미	major third	$(\sqrt[12]{2})^4$	1.25992	$\frac{5}{4}$	1.25	$\frac{81}{64}$	1.26563
파	fourth	$(\sqrt[12]{2})^5$	1.33484	$\frac{4}{3}$	1.33333	$\frac{4}{3}$	1.33333
파#	augmented fourth	$(\sqrt[12]{2})^6$	1.41421	$\frac{45}{32}$	1.40625	$\frac{729}{512}$	1.42383
솔 b	diminished fifth			$\frac{54}{45}$	1.42222	$\frac{1024}{729}$	1.40466
솔	fifth	$(\sqrt[12]{2})^7$	1.49831	$\frac{3}{2}$	1.5	$\frac{3}{2}$	1.5
솔#	minor sixth	$(\sqrt[12]{2})^8$	1.5874	$\frac{8}{5}$	1.6	$\frac{128}{81}$	1.58025
라	major sixth	$(\sqrt[12]{2})^9$	1.68179	$\frac{5}{3}$	1.66667	$\frac{27}{16}$	1.6875
라#	minor seventh	$(\sqrt[12]{2})^{10}$	1.7818	$\frac{16}{9}$	1.77778	$\frac{16}{9}$	1.77778
시	major seventh	$(\sqrt[12]{2})^{11}$	1.88775	$\frac{15}{8}$	1.875	$\frac{243}{128}$	1.89844
도	octave	2	2	2	2	2	2

$\frac{f_{13}}{f_1} = 2$



MIDI number	Note name	Frequency Hz	Period ms
21	A0	27.500	36.36
22		29.135	34.32
23	B0	30.868	32.40
24	C1	32.703	30.58
25		34.648	28.86
26	D1	36.708	27.24
27		38.891	25.71
28	E1	41.203	24.27
29	F1	43.654	22.91
30		46.249	21.62
31	G1	48.999	20.41
32		51.913	19.26
33	A1	55.000	18.18
34		58.270	17.16
35	B1	61.735	16.20
36	C2	65.406	15.29
37		69.296	14.29
38	D2	73.416	13.62
39		77.782	12.86
40	E2	82.407	12.13
41	F2	87.307	11.45
42		92.499	10.81
43	G2	97.999	10.20
44		103.83	9.631
45	A2	110.00	9.091
46		116.54	8.581
47	B2	123.47	8.099
48	C3	130.81	7.645
49		138.59	7.216
50	D3	146.83	6.811
51		155.56	6.428
52	E3	164.81	6.068
53	F3	174.61	5.727
54		185.00	5.405
55	G3	196.00	5.102
56		207.65	4.816
57	A3	220.00	4.545
58		233.08	4.290
59	B3	246.94	4.050
60	**C4**	**261.63**	**3.822**
61		277.18	3.608
62	D4	293.67	3.405
63		311.13	3.214
64	E4	329.63	3.034
65	F4	349.23	2.863
66		369.99	2.703
67	G4	392.00	2.551
68		415.30	2.408
69	**A4**	**440.00**	**2.273**
70		466.16	2.145
71	B4	493.88	2.025
72	C5	523.25	1.910
73		554.37	1.804
74	D5	587.33	1.703
75		622.25	1.607
76	E5	659.26	1.517
77	F5	698.46	1.432
78		739.99	1.351
79	G5	783.99	1.276
80		830.61	1.204
81	A5	880.00	1.136
82		932.33	1.073
83	B5	987.77	1.012
84	C6	1046.5	0.9556
85		1108.7	0.9020
86	D6	1174.7	0.8513
87		1244.5	0.8034
88	E6	1318.5	0.7584
89	F6	1396.9	0.7159
90		1480.0	0.6757
91	G6	1568.0	0.6378
92		1661.2	0.6020
93	A6	1760.0	0.5682
94		1864.7	0.5363
95	B6	1975.5	0.5062
96	C7	2093.0	0.4778
97		2217.5	0.4510
98	D7	2349.3	0.4257
99		2489.0	0.4018
100	E7	2637.0	0.3792
101	F7	2793.0	0.3580
102		2960.0	0.3378
103	G7	3136.0	0.3189
104		3322.4	0.3010
105	A7	3520.0	0.2841
106		3729.3	0.2681
107	B7	3951.1	0.2531
108	C8	4186.0	0.2389



컴퓨터로 소리 출력


```
#include <math.h>

int calc_frequency(int octave, int inx)
{
   double do_scale = 32.7032;
   double ratio = pow(2., 1 / 12.), temp;
   int i;
   temp = do_scale*pow(2., octave - 1);
   for (i = 0; i < inx; i++) {
      temp = (int)(temp + 0.5);
      temp *= ratio;
   }

   return (int)temp;
}
```

```
#include "util.h"
#include <windows.h>

void main()
{
   int index[] = { 0, 2, 4, 5, 7, 9, 11, 12 }, freq[8], i;

   for (i = 0; i < 8; i++) freq[i] = calc_frequency(4, index[i]);

   for (i = 0; i < 8; i++) Beep(freq[i], 500);

   Sleep(1000);

    for (i = 7; i >= 0; i--) Beep(freq[i], 500);
}
```

컴퓨터 피아노


```
#include <windows.h>

void practice_piano()
{
   int index[] = { 0, 2, 4, 5, 7, 9, 11, 12 }, freq[8], code, i;

   for (i = 0; i < 8; i++) freq[i] =  calc_frequency(4, index[i]);

  do {
      code = _getch();
      if ('1' <= code&&code <= '8') {
         code -= '0';
         Beep(freq[code], 300);
      }
  } while (code != 27);
}
```

```
#include "util.h"
#include <windows.h>
#include <stdio.h>

void main()
{
    printf("1부터 8까지 숫자 키를 누르면\n");
    printf("각 음의 소리가 출력됩니다.\n\n");
    printf("1:도 2:레 3:미 4:파 5:솔 6:라 7:시 8:도\n");
    printf("프로그램 종료는 ESC키\n");

    practice_piano();
}
```

3. 행렬 계산 (1)

Matrix.h

```
#pragma once

#define M          (3)
#define N          (3)

struct MATRIX
{
double m[M][N];
};

MATRIX Add(const MATRIX* a, const MATRIX* b);
MATRIX Subtract(const MATRIX* a, const MATRIX* b);
MATRIX Multiply(const MATRIX* a, const MATRIX* b);

void PrintMatrix(const MATRIX* a);
```

main.cpp

```
#include "matrix.h"
#include <stdio.h>

void main()
{
MATRIX a = { 1, 2, 3 , 4, 5, 6,  7, 8, 9 };
MATRIX b = { 2, 3, 4, 5, 6, 7, 8, 9, 1 };

MATRIX c;
c = Add(&a, &b);
PrintMatrix(&c);

printf("--------------------------------------\n");

c = Subtract(&a, &b);
PrintMatrix(&c);

printf("--------------------------------------\n");
c = Multiply(&a, &b);
PrintMatrix(&c);
}
```

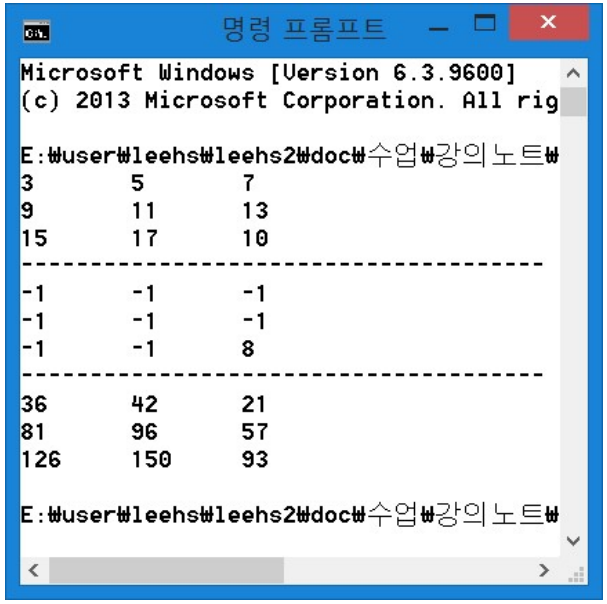
Implementation

matrix.cpp

$$\mathbf{C} = \mathbf{A} + \mathbf{B}$$
$$c_{i,j} = a_{i,j} + b_{i,j}$$

$$\mathbf{C} = \mathbf{A} - \mathbf{B}$$
$$c_{i,j} = a_{i,j} - b_{i,j}$$

$$\mathbf{C} = \mathbf{AB}$$
$$c_{i,j} = \sum_{k=1}^{N} a_{i,k} b_{k,j}$$




```
#include "matrix.h"
#include <stdio.h>

MATRIX Add(const MATRIX* a, const MATRIX* b)
{
    MATRIX ans;

    int i, j;
    for (i = 0; i < M; i++) for (j = 0; j < N; j++) ans.m[i][j] = a->m[i][j] + b->m[i][j];

     return ans;
}

MATRIX Subtract(const MATRIX* a, const MATRIX* b)
{
    MATRIX ans;

    int i, j;
    for (i = 0; i < M; i++) for (j = 0; j < N; j++) ans.m[i][j] = a->m[i][j] - b->m[i][j];

    return ans;
}

MATRIX Multiply(const MATRIX* a, const MATRIX* b)
{
    MATRIX ans;

    int i, j, k;
    for (i = 0; i < M; i++) for (j = 0; j < N; j++) {
        ans.m[i][j] = 0.;
        for (k = 0; k < M; k++) ans.m[i][j] += a->m[i][k] * b->m[k][j];
    }
    return ans;
}

void PrintMatrix(const MATRIX* a)
{
    int i, j;
    for (i = 0; i < M; i++) {
       for (j = 0; j < N; j++) printf("%g\t", a->m[i][j]);
        printf("\n");
    }
}
```

행렬 계산 (2)

멤버 함수 이용

```
#pragma once

#define M(3)
#define N (3)

struct MATRIX
{
    MATRIX Add(const MATRIX* b);
    MATRIX Subtract(const MATRIX* b);
    MATRIX Multiply(const MATRIX* b);

    void PrintMatrix();

    double m[M][N];
};
```

```
void main()
{
    MATRIX a = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    MATRIX b = { 2, 3, 4, 5, 6, 7, 8, 9, 1 };

    MATRIX c;
    c = a.Add(&b);
    c.PrintMatrix();

    printf("---------------------------------------\n");

    c = a.Subtract(&b);
    c.PrintMatrix();

    printf("---------------------------------------\n");
    c = a.Multiply(&b);
    c.PrintMatrix();
}
```

```
#include "matrix.h"
#include <stdio.h>

MATRIX MATRIX::Add(const MATRIX* b)
{
    MATRIX ans;

    int i, j;
    for (i = 0; i < M; i++) for (j = 0; j < N; j++) ans.m[i][j] = m[i][j] + b->m[i][j];

    return ans;
}

MATRIX MATRIX::Subtract(const MATRIX* b)
{
    MATRIX ans;

    int i, j;
    for (i = 0; i < M; i++) for (j = 0; j < N; j++) ans.m[i][j] = m[i][j] - b->m[i][j];

    return ans;
}

MATRIX MATRIX::Multiply(const MATRIX* b)
{
    MATRIX ans;

    int i, j, k;
    for (i = 0; i < M; i++) for (j = 0; j < N; j++) {
        ans.m[i][j] = 0.;
        for (k = 0; k < M; k++) ans.m[i][j] += m[i][k] * b->m[k][j];
    }
    return ans;
}

void MATRIX::PrintMatrix()
{
    int i, j;
    for (i = 0; i < M; i++) {
        for (j = 0; j < N; j++) printf("%g\t", m[i][j]);
        printf("\n");
    }
}
```

행렬 계산 (3)

멤버 함수 operator이용

```
#pragma once

#define M (3)
#define N (3)

struct MATRIX
{
    MATRIX operator+(const MATRIX* b);
    MATRIX operator-(const MATRIX* b);
    MATRIX operator*(const MATRIX* b);

    void PrintMatrix();

    double m[M][N];
};
```

```
void main()
{
    MATRIX a = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    MATRIX b = { 2, 3, 4, 5, 6, 7, 8, 9, 1 };

    MATRIX c;
//  c = a.operator+(&b);
    c = a+(&b);
    c.PrintMatrix();

    printf("---------------------------------------\n");

    c = a-(&b);
    c.PrintMatrix();

    printf("---------------------------------------\n");
    c = a*(&b);
    c.PrintMatrix();
}
```

```
#include "matrix.h"
#include <stdio.h>

MATRIX MATRIX::operator+(const MATRIX* b)
{
    MATRIX ans;

    int i, j;
    for (i = 0; i < M; i++) for (j = 0; j < N; j++) ans.m[i][j] = m[i][j] + b->m[i][j];

    return ans;
}

MATRIX MATRIX::operator-(const MATRIX* b)
{
    MATRIX ans;

    int i, j;
    for (i = 0; i < M; i++) for (j = 0; j < N; j++) ans.m[i][j] = m[i][j] - b->m[i][j];

    return ans;
}

MATRIX MATRIX::operator*(const MATRIX* b)
{
    MATRIX ans;

    int i, j, k;
    for (i = 0; i < M; i++) for (j = 0; j < N; j++) {
                    ans.m[i][j] = 0.;
                    for (k = 0; k < M; k++) ans.m[i][j] += m[i][k] * b->m[k][j];
    }
    return ans;
}

void MATRIX::PrintMatrix()
{
    int i, j;
    for (i = 0; i < M; i++) {
                    for (j = 0; j < N; j++) printf("%g\t", m[i][j]);
                    printf("\n");
    }
}
```

행렬 계산 (3)

참조형(&) 데이터 이용

```
#pragma once

#define M(3)
#define N (3)

struct MATRIX
{
    MATRIX operator+(const MATRIX& b);
    MATRIX operator-(const MATRIX& b);
    MATRIX operator*(const MATRIX& b);

    void PrintMatrix();

    double m[M][N];
};
```

```
void main()
{
    MATRIX a = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    MATRIX b = { 2, 3, 4, 5, 6, 7, 8, 9, 1 };

    MATRIX c;
//  c = a.operator+(&b);
    c = a+b;
    c.PrintMatrix();

    printf("---------------------------------------\n");

    c = a-b;
    c.PrintMatrix();

    printf("---------------------------------------\n");
    c = a*b;
    c.PrintMatrix();
}
```

```
#include "matrix.h"
#include <stdio.h>

MATRIX MATRIX::operator+(const MATRIX& b)
{
    MATRIX ans;

    int i, j;
    for (i = 0; i < M; i++) for (j = 0; j < N; j++) ans.m[i][j] = m[i][j] + b.m[i][j];

    return ans;
}

MATRIX MATRIX::operator-(const MATRIX& b)
{
    MATRIX ans;

    int i, j;
    for (i = 0; i < M; i++) for (j = 0; j < N; j++) ans.m[i][j] = m[i][j] - b.m[i][j];

    return ans;
}

MATRIX MATRIX::operator*(const MATRIX& b)
{
    MATRIX ans;

    int i, j, k;
    for (i = 0; i < M; i++) for (j = 0; j < N; j++) {
        ans.m[i][j] = 0.;
        for (k = 0; k < M; k++) ans.m[i][j] += m[i][k] * b.m[k][j];
    }
    return ans;
}

void MATRIX::PrintMatrix()
{
    int i, j;
    for (i = 0; i < M; i++) {
        for (j = 0; j < N; j++) printf("%g\t", m[i][j]);
        printf("\n");
    }
}
```

행렬 : C++ 이용

mat.cpp

```
#include "mat.h"
#include <stdio.h>

void main()
{
    MATRIX a(3, 3), b(3, 3);
    a(0, 0) = 1; a(0, 1) = 2; a(0, 2) = 3;
    a(1, 0) = 4; a(1, 1) = 5; a(1, 2) = 6;
    a(2, 0) = 7; a(2, 1) = 8; a(2, 2) = 9;

    b(0, 0) = 2; b(0, 1) = 3; b(0, 2) = 4;
    b(1, 0) = 5; b(1, 1) = 6; b(1, 2) = 7;
    b(2, 0) = 8; b(2, 1) = 9; b(2, 2) = 1;

    MATRIX c(3, 3);

    c = a + b;
    c.print();
    printf("-------------------------------------\n");

    c = a - b;
    c.print();
    printf("-------------------------------------\n");

    c = a*b;
    c.print();
}
```

```
#pragma once

struct MATRIX {
	double *m;
	int row, col;

	MATRIX(){
		m = 0;
		row = col = -1;
	}

	MATRIX(const int p, const int q)
	{
		SetSize(p, q);
	}

	MATRIX(const MATRIX& a);

	~MATRIX()
	{
		delete []m;
	}

	void SetSize(const int p, const int q);
	double& operator()(const int& p, const int& q);

	MATRIX& operator=(const MATRIX& a);

	MATRIX operator+(const MATRIX& a) const;
	MATRIX operator+() const;
	MATRIX operator-(const MATRIX& a) const;
	MATRIX operator-() const;
	MATRIX operator*(const MATRIX& a) const;

	MATRIX operator*(const double& a) const;

	void print();
};
```

4. 미분 방정식

$$y' = f(x, y), \quad y(x_0) = y_0$$

$$x_1 = x_0 + h, \quad x_2 = x_0 + 2h, \quad x_3 = x_0 + 3h$$

Taylor series : $y(x+h) = y(x) + hy'(x) + \frac{h^2}{2}y''(x) + \cdots$

$$y(x+h) \cong y(x) + hy'(x) = y(x) + hf(x, y)$$

Euler method :

$$y_{n+1} = y_n + hf(x_n, y_n)$$
$$x_{n+1} = x_n + h$$

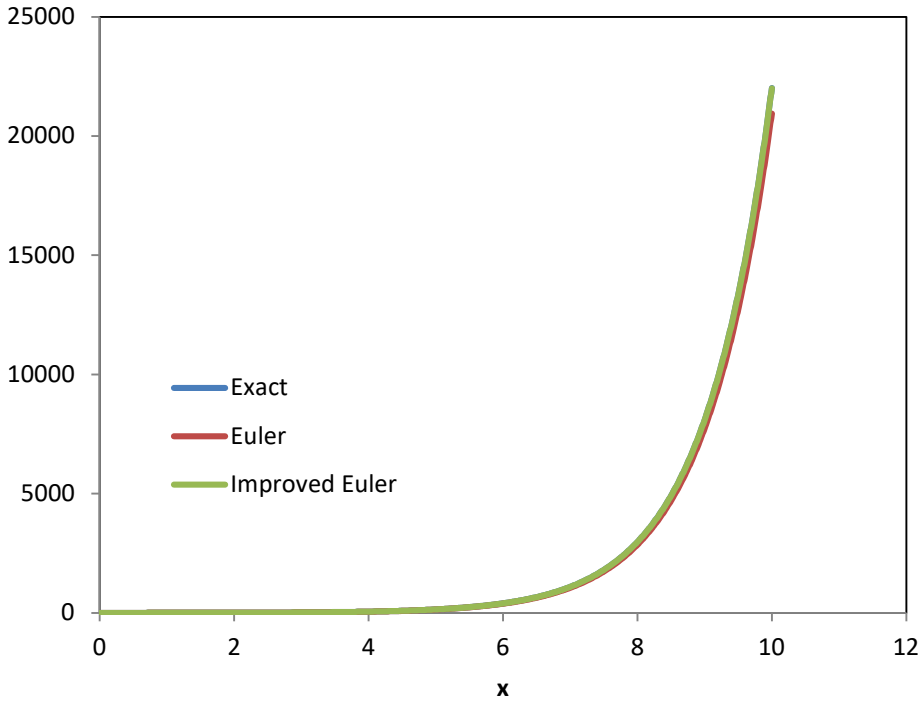
Improved Euler Method (Heun's Method)

$$x_{n+1} = x_n + h$$

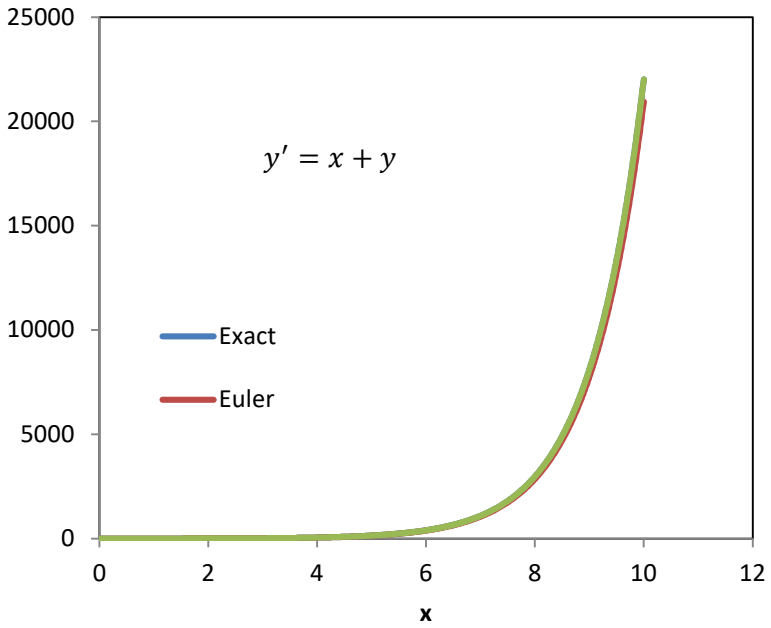
$$k_1 = hf(x_n, y_n)$$

$$k_2 = hf(x_{n+1}, y_n + k_1)$$

$$y_{n+1} = y_n + \frac{1}{2}(k_1 + k_2)$$



1차 미분 방정식




```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <math.h>

// 아래 함수 f1()는 y = e^x - x - 1 를 1차 미분한 함수다.
double f1(double x, double y)
{
    // y'(x) = x + y
    return x + y;
}

void main()
{
    FILE *fp = fopen("exact.txt", "wt");

    const double h = 0.01;
    double x, y(0.);
    for (x = 0; x < 10; x += h) {
        y = exp(x) - x - 1.;
        fprintf(fp, "%g\t%g \n", x, y);
    }
    fclose(fp);

    // Euler's method
    fp = fopen("Euler.txt", "wt");

    y = 0.;
    for (x = 0; x < 10; x += h) {
        fprintf(fp, "%g\t%g\n", x, y);
        y = y+h*f1(x, y);
    }
    fclose(fp);

    // Improved Euler's method
    fp = fopen("ImproveEuler.txt", "wt");
    y = 0.;
    for (x = 0; x < 10; x += h) {
        fprintf(fp, "%g\t%g\n", x, y);
        double ys_1 = y + h*f1(x, y);
        y = y + 0.5*h*(f1(x, y) + f1(x + h, ys_1));
    }
    fclose(fp);
}
```

2차 미분 방정식

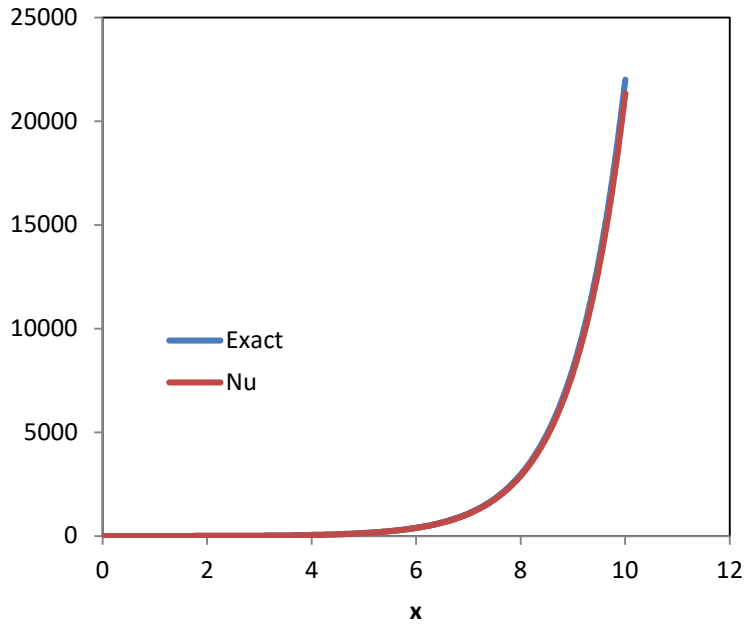
$$y'' = f(x, y, y'), \quad y(x_0) = y_0, \quad y'(x_0) = y_0'$$

$$y(x+h) = y(x) + hy'(x) + \frac{h^2}{2}y''(x) + \cdots$$

$$y'(x+h) = y'(x) + hy''(x) + \frac{h^2}{2}y'''(x) + \cdots$$

$$y(x+h) \approx y(x) + hy'(x) + \frac{h^2}{2}y''(x)$$

$$y'(x+h) \approx y'(x) + hy''(x)$$

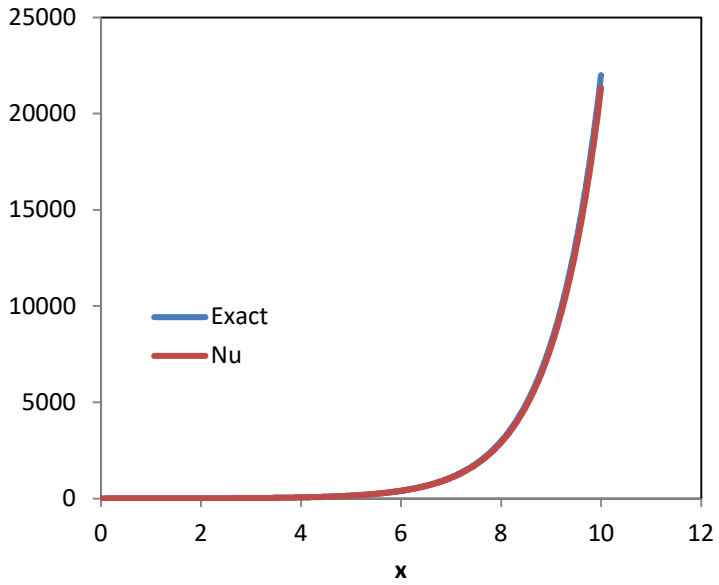


$$y_n'' = f(x_n, y_n, y_n')$$

$$y_{n+1} = y_n + hy_n' + \frac{h^2}{2}y''$$

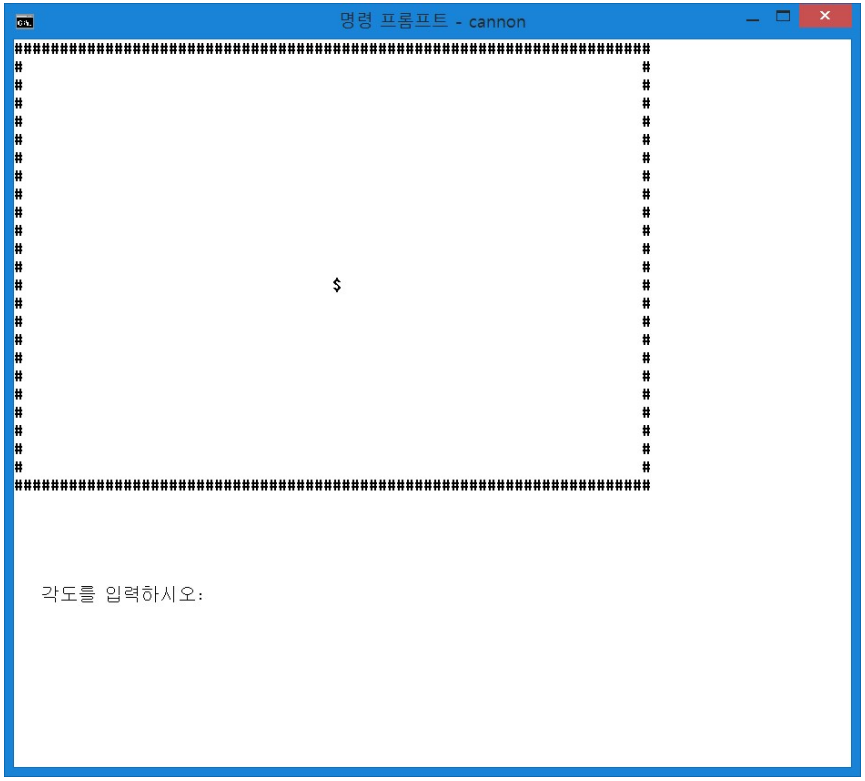
$$y_{n+1}' = y_n' + hy''$$

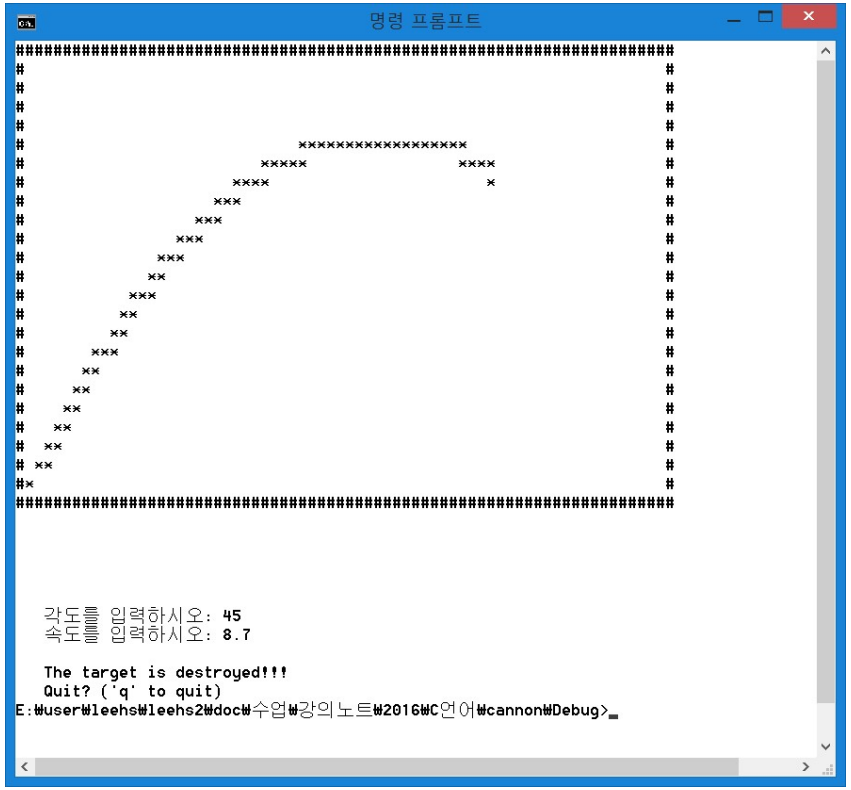
2차 미분 방정식




```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
#include <math.h>

// 수치 계산을 통해 초기값이 있는 2차 미분 방정식 풀기

// 아래 함수 f2()는 y = e^x - x - 1 를 2차 미분한 함수다.
double f2(double x, double y, double yp)
{
    // y''(x) = (x + y' + 2)/2
    return (x +yp +2.)/2.;
}

void main()
{
    const double h = 0.01;

    FILE *fp = fopen("exact.txt", "wt");

    double x, y(0.);
    for (x = 0; x < 10; x += h) {
      y = exp(x) - x - 1.;
      fprintf(fp, "%g\t%g \n", x, y);
    }

    fclose(fp);

    // Euler's method
    fp = fopen("general.txt", "wt");

    y = 0.;
    double yp(0.), ypp;
    for (x = 0; x < 10; x += h) {
      fprintf(fp, "%g\t%g\n", x, y);

      ypp = f2(x, y, yp);
      y = y + h*yp+h*h/2.*ypp;
      yp = yp + h*ypp;
    }
    fclose(fp);
}
```

5. Cannon ball 게임





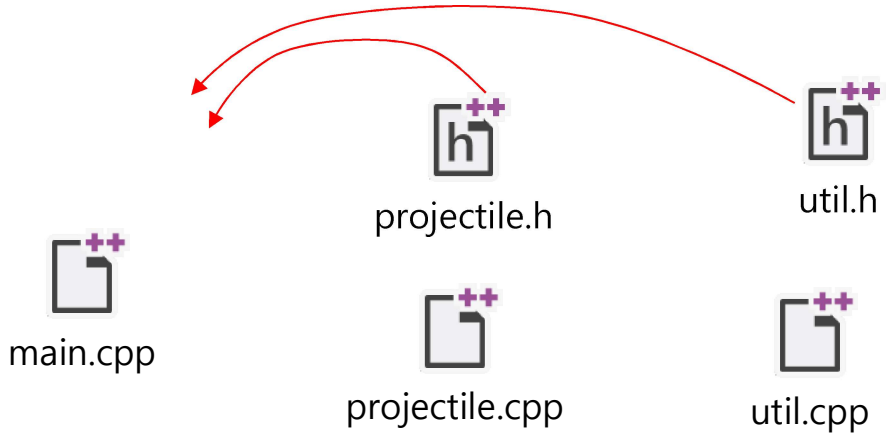
$$\mathbf{F} = m\mathbf{a} = m\frac{d^2\mathbf{r}}{dt^2} = -m\mathbf{g}$$

$$\rightarrow \quad \frac{d^2\mathbf{r}}{dt^2} = -\mathbf{g}$$

$$\mathbf{r}_{n+1} = \mathbf{r}_n + \Delta t \cdot \frac{d\mathbf{r}}{dt} + \frac{(\Delta t)^2}{2} \cdot (-\mathbf{g})$$

$$\left.\frac{d\mathbf{r}}{dt}\right|_{n+1} = \left.\frac{d\mathbf{r}}{dt}\right|_{n} + \Delta t \cdot (-\mathbf{g})$$

소스 파일

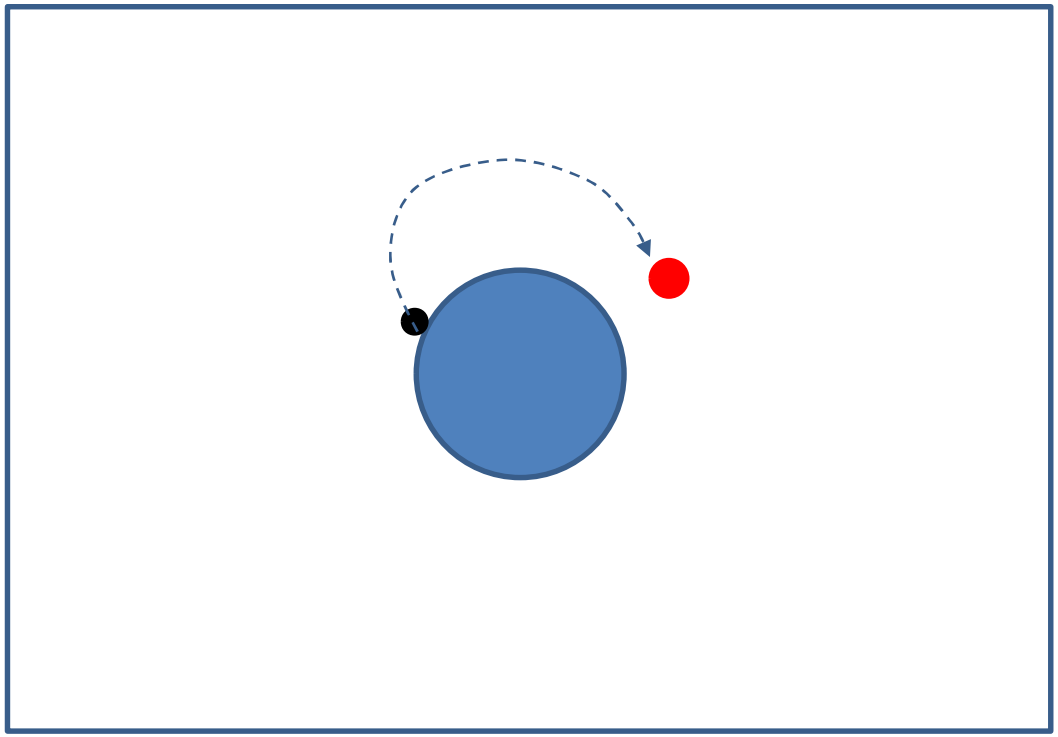


Projectile (포탄) 및
boundary(경계 및 목표물)

```
void gotoxy(int x, int y);
void textcolor(int foreground, int background);

double RAD(const double& x);
double DEG(const double& x);
```

과제

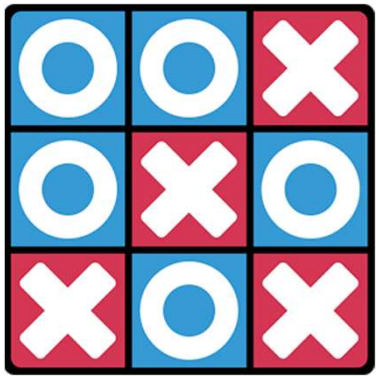
- 지구 표면에서 포탄을 발사하여 목표물을 맞추는 게임이 되도록 수정
- 포탄은 지구 중심을 향한 중력 가속도의 영향을 받게 함.
- G, M등의 상수는 임의로 지정.
- 기타 게임의 재미를 향상시킬 수 있는 요소 추가하면 보너스 점수.



$$\mathbf{F} = m\mathbf{a} = m\frac{d^2\mathbf{r}}{dt^2} = -\frac{GmM}{r^2}\hat{r} = -\frac{GmM}{r^3}\mathbf{r}$$

$$\frac{d^2\mathbf{r}}{dt^2} = -\frac{GM}{r^3}\mathbf{r} \quad \Rightarrow \quad \begin{cases} \dfrac{d^2x}{dt^2} = -\dfrac{GM}{r^3}x \\ \dfrac{d^2y}{dt^2} = -\dfrac{GM}{r^3}y \end{cases}$$

6. Tic Tac Toe 게임



ttt.cpp

```
while (state ==CONTINUE) {

    Human move

    Computer move

    Check state

}
```

```
void print_board();
int human_move();
int computer_move();
int check_result(int side);
int dfs_search(int side);
```

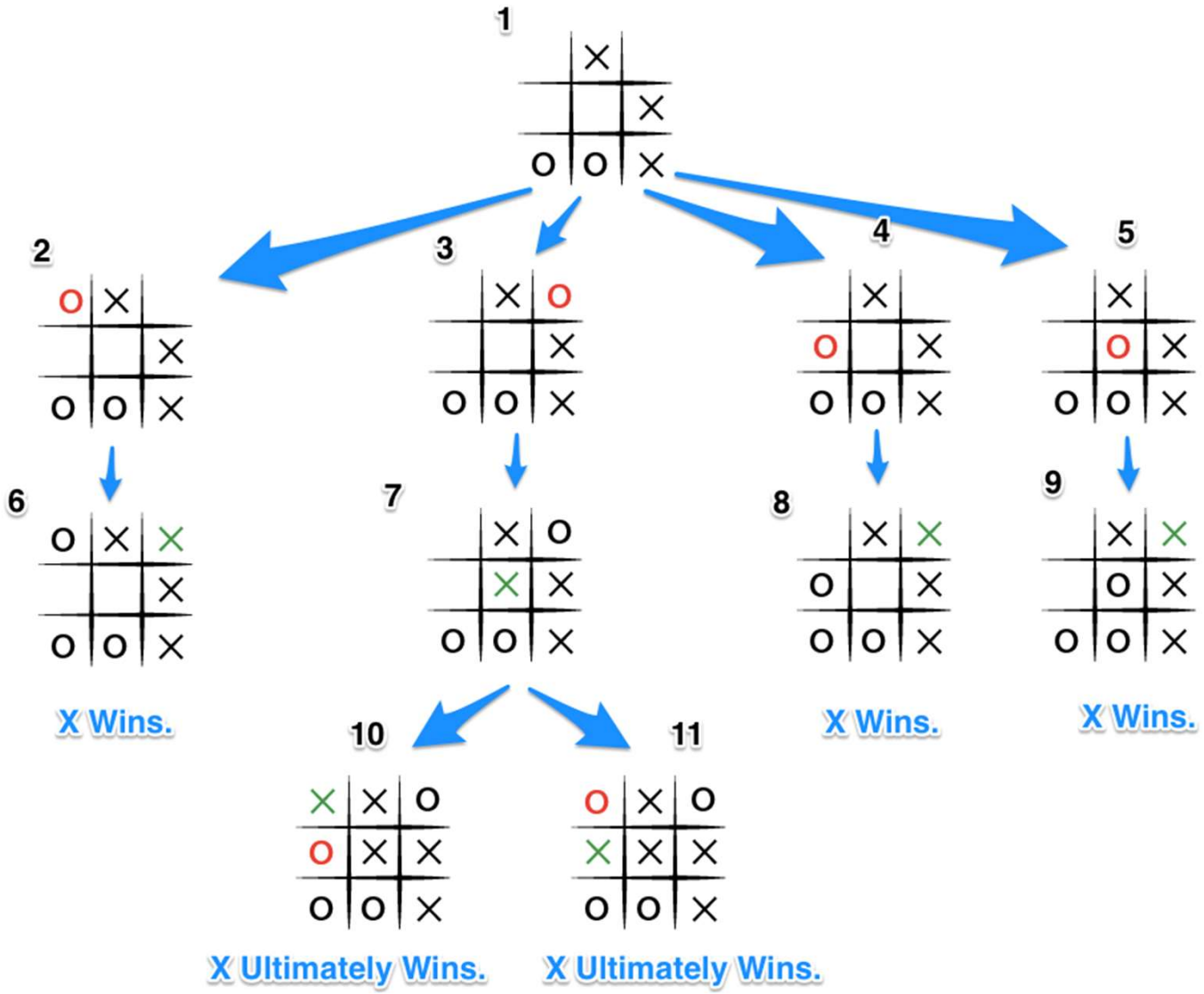
Computer move()

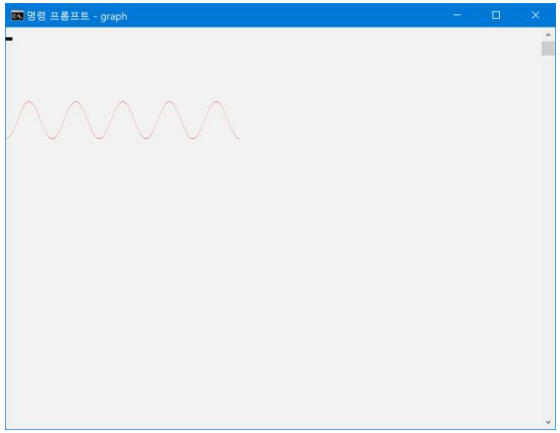
```
int computer_move()
{
	int best_move;	// best move so far
	int best_score = -100;	// best score so far
	int score;// current score
	int i;

	for (i = 0; i < 9; ++i) {
	    if (pos[i] == EMPTY) {	// if a legal move can be made
	      pos[i] = COMPUTER;	// mark the move
	      score = - dfs_search(HUMAN);

	      pos[i] = EMPTY;	// take back the move

	      if (score > best_score) {
	            best_score = score;
	            best_move = i;
	      }
	    }
	}

	printf("Computer's move: %d\n", best_move);
	return best_move;	// return the best move found
}
```

DFS search()

```
int dfs_search(int player)
{
    int best_score = -100;
    int score;
    int result;
    int i;

    result = check_result(player);
    if (result != CONTINUE) return result;	// return the result

    for (i = 0; i < 9; ++i) {
        if (pos[i] == EMPTY) {
            pos[i] = player;
            score = -dfs_search(CHANGE_PLAYER(player));
            pos[i] = EMPTY;

            if (score > best_score)
                best_score = score;
        }
    }

    return best_score;	// return the best score
}
```

Depth first search algorithm






```
#include <Windows.h>
#include <iostream>
#include <cmath>

using namespace std;

#define PI   3.14

void main()
{
   system("cls");

   HWND myConsole = GetConsoleWindow();
   HDC mdc = GetDC(myConsole);

   COLORREF COLOR = RGB(255, 0, 0);

   int x = 0, y;

   for (double x = 0.; x < PI*10.; x += 0.05) {
      y = (int)(250. + 50.*cos(x));
      SetPixel(mdc, int(x*10), y, COLOR);
   }

   ReleaseDC(myConsole, mdc);
   cin.ignore();
}
```