랜덤 행렬을 만들어서 Inverse Matrix의 존재 여부를 확인하고 있으면 출력, 없으면 다른 랜덤 행렬을 만들어서 가역행렬 나올 때까지 실행시켜주는 코드이다. 먼저 전역 변수로 다음과 같이 설정을 했다.

```
#define Max_Row 128 //nxn 행렬에서 n값을 설정
#define Columns_language_type 8// 자료형 크기 (8 = unsigned char, 32 = unsigned int)

#define Success 0
#define False -1

#if Columns_language_type == 8
unsigned char Matrix_Original[Max_Row][Max_Row / Columns_language_type];
unsigned char Matrix_for_cpy[Max_Row][Max_Row / Columns_language_type];
unsigned char I[Max_Row][Max_Row / Columns_language_type];
#elif Columns_language_type == 32
unsigned int Matrix_Original[Max_Row][Max_Row / Columns_language_type];
unsigned int Matrix_for_cpy[Max_Row][Max_Row / Columns_language_type];
unsigned int I[Max_Row][Max_Row / Columns_language_type];
#endif
```

여기서 부터는 pseudo code 이다.

Making_Random_Matrix (랜덤 행렬 만드는 함수)

```
1.  Procedure Making_Random_Matrix()
2.  If Columns_language_type == 8 then
3.    For i ← 0 to Max_Row do
4.      For j ← 0 to Max_Row / Columns_language_type do
5.        Matrix_for_cpy[i][j] ← rand()
6.      Endfor
7.    Endfor
8.  Else
9.  For i ← 0 to Max_Row do
10.     For j ← 0 to Max_Row / Columns_language_type do
11.       Matrix_for_cpy[i][j] ← (rand() << 24) ^ (rand() << 16) ^ (rand() << 8) ^ rand()
```

```
12.      Endfor
13.    Endfor
14.  Endif
15.  End Procedure
```

기본적으로 헤더인 include<time.h>를 삽입 후 rand()함수를 이용한 코드이다.

Columns_language_type == 32일 때 11번 식이 나온 이유는 rand()함수는 최대 15비트까지만 생성하므로 32비트를 만들기 위해서 저 식을 사용했다.

Making_Identity_Matrix (항등 행렬 만드는 함수)

```
1.  Procedure Making_Identity_Matrix()
2.  For i ← 0 to Max_Row do
3.    For q ← 0 to Max_Row / Columns_language_type do
4.      I[i][q] ← 0x00
5.    End for
6.  End for
7.  For i ← 0 to Max_Row do
8.    q ← i / Columns_language_type
9.    r ← Columns_language_type - 1 - i + (Columns_language_type * q)
10.   I[i][q] ← 0x01 << r
11. End for
12. End Procedure
```

nxn항등 행렬을 만들어주는 코드이다. 2~6줄은 행렬 값들이 들어갈 I[][]를 0으로 초기화 시켜주고 8~10번 식을 이용하여 항등행렬을 만들어 준다.

Xor_Operation (두 행을 Xor 시켜주는 함수)

```
1.  Procedure Xor_Operation( Rows_King, Rows_other)
2.  For i ← 0 to Max_Row / Columns_language_type do
3.                                  Matrix_for_cpy[Rows_other][i]  ←
    Matrix_for_cpy[Rows_King][i]^Matrix_for_cpy[Rows_other][i]
```

```
4.    I[Rows_other][i] ← I[Rows_King][i] ^ I[Rows_other][i]
5.  End for
6.  End procedure
```

Swap (두 행을 바꿔주는 함수)

```
1.  Procedure Swap( i , j )
2.  For k ← 0 to Max_Row / Columns_language_type do
3.    temp ← Matrix_for_cpy[i][k]
4.    Matrix_for_cpy[i][k] ← Matrix_for_cpy[j][k]
5.    Matrix_for_cpy[j][k] ← temp
6.    temp ← I[i][k]
7.    I[i][k] ← I[j][k]
8.    I[j][k] ← temp
9.  End for
10. End procedure
```

Find_first_one (columns번째 행부터 current_row 번째열 중에 n자리에 1이 있는 행을 찾아서 리턴해주는 함수)

```
1.  Procedure Find_first_one( n, current_row, columns)
2.  i ← columns
3.  while ( Matrix_for_cpy[i][current_row] >> n ) & 0x01 != 0x01 do
4.    i ← i+1
5.    If i ≥ Max_Row then
6.      Return False
7.    End if
8.  End while
```

```
9.  Return i
10. End procedure
```

이 코드는 columns번째 행부터 current_row 번째 열 중에 n자리에 1이 있는 행을 찾아서 리턴 해주는 함수이다.

CopyMatrix (원래 행렬을 보존하기 위해 Matrix_Original에 복사 해준다. (나중에 가역행렬을 찾은 후 원래 행렬을 출력해주기 위해서 만듦)

```
1.  Procedure CopyMatrix()
2.  For i ← 0 to Max_Row do
3.    For j ← 0 to Max_Row / Columns_language_type do
4.      Matrix_Original[i][j] ← Matrix_for_cpy[i][j]
5.    End for
6.  End for
7.  End procedure
```

Gauss_Jordan_Elimination (가우스 조던 소거법을 이용하여 랜덤행렬의 역행렬을 구한다. 역행렬이 없을 경우 출력하지 않는다)

```
1.  procedure Gauss_Jordan_Elimination(*check)
2.  now_columns ← 1
3.  i ← Columns_language_type
4.  current_row ← 0
5.  While current_row != Max_Row / Columns_language_type do
6.    n ← n+1
7.    i ← i – 1
8.    position <- Find_first_one(i, current_row, now_columns)
9.    If position == False then
10.     *check ← False
11.     current_row ← Max_Row / Columns_language_type
```

```
12.  else
13.  if position != now_columns then
14.    Swap(position, now_columns);
15.  endif
16.  for l ← 0 to Max_Row do
17.    If l != now_columns then
18.      If (Matrix_for_cpy[l][current_row] >> i) & 0x01 == 0x01 then
19.        Xor_Operation(now_columns, l)
20.      end if
21.    end if
22.  end for
23.  If i == 0 then
24.    i ← Columns_language_type
25.    current_row ← current_row + 1
26.    *check ← Success
27.  end if
28. end while
29. If *check == Success then
30.  print_Matrix()
31.  sage_print()
32. end if
33. end procedure
```

이 함수는 가우스 조던 소거법을 구현한 함수이다. 여기서 봐야 될 부분은 9~12번 줄이다. Position 값이 False이면 즉 그 행의 값이 모두 0이면 check값에 0을 입력하고 11번 줄을 입력한다. 11번 줄의 역할은 while문을 벗어나게 해준다. 이렇게 하나의 행이 모두 0이면 while문을 종료시킬 수 있는 것이다. 이때 check = False이므로 29~32번 줄 또한 실행되지 않는다.

Main 함수

1. **Procedure** main()
2. srand((unsigned)time(NULL))
3. check ← False
4. **do**
5. Making_Identity_Matrix()
6. Making_Random_Matrix()
7. CopyMatrix()
8. Gauss_Jordan_Elimination(&check)
9. **While** check != Success
10. **End procedure**