컴퓨터보안 5 주차 DES 과제 / 2018920065 루안리치

프로그래밍 언어: C

소스코드: (실행결과 -> p.14)

```
//  main.c
//  DES_2018920065
//
//  Created by Luan Lichi on 2022/03/30.

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

int pt[64];
int key[64];
int ip_p[64];     // pt->IP
int C[17][28], D[17][28];
int k56[17][56];
int k48[17][48];
int L32[17][32];
int ER[17][48];
int K_ER[17][48];
int bin4[4];
int SB[17][32];
int PB[17][32];
int R32[17][32];
int MERGE[64]; // R[16]L[16]
int FINAL[64]; // merge->IIP
char CIPHER[16]; // hex
char Left[17][8];
char Right[17][8];
char RoundKey[17][12];

int IP[] =
{
    58, 50, 42, 34, 26, 18, 10, 2,    // initial permutation
    60, 52, 44, 36, 28, 20, 12, 4,
```

```
    62, 54, 46, 38, 30, 22, 14, 6,
    64, 56, 48, 40, 32, 24, 16, 8,
    57, 49, 41, 33, 25, 17,  9, 1,
    59, 51, 43, 35, 27, 19, 11, 3,
    61, 53, 45, 37, 29, 21, 13, 5,
    63, 55, 47, 39, 31, 23, 15, 7
};

int IIP[] =
{
    40, 8, 48, 16, 56, 24, 64, 32,   // inverse of initial permutation
    39, 7, 47, 15, 55, 23, 63, 31,
    38, 6, 46, 14, 54, 22, 62, 30,
    37, 5, 45, 13, 53, 21, 61, 29,
    36, 4, 44, 12, 52, 20, 60, 28,
    35, 3, 43, 11, 51, 19, 59, 27,
    34, 2, 42, 10, 50, 18, 58, 26,
    33, 1, 41,  9, 49, 17, 57, 25
};

int E[] =
{
    32,  1,  2,  3,  4,  5,       // expansion permutation
    4,   5,  6,  7,  8,  9,
    8,   9, 10, 11, 12, 13,
    12, 13, 14, 15, 16, 17,
    16, 17, 18, 19, 20, 21,
    20, 21, 22, 23, 24, 25,
    24, 25, 26, 27, 28, 29,
    28, 29, 30, 31, 32,  1
};

int S1[4][16] =
{
    14,  4, 13,  1,  2, 15, 11,  8,  3, 10,  6, 12,  5,  9,  0,  7,
    0,  15,  7,  4, 14,  2, 13,  1, 10,  6, 12, 11,  9,  5,  3,  8,
    4,   1, 14,  8, 13,  6,  2, 11, 15, 12,  9,  7,  3, 10,  5,  0,
    15, 12,  8,  2,  4,  9,  1,  7,  5, 11,  3, 14, 10,  0,  6, 13
```

```
};

int S2[4][16] =
{
    15,  1,  8, 14,  6, 11,  3,  4,  9,  7,  2, 13, 12,  0,  5, 10,
    3,  13,  4,  7, 15,  2,  8, 14, 12,  0,  1, 10,  6,  9, 11,  5,
    0,  14,  7, 11, 10,  4, 13,  1,  5,  8, 12,  6,  9,  3,  2, 15,
    13,  8, 10,  1,  3, 15,  4,  2, 11,  6,  7, 12,  0,  5, 14,  9
};

int S3[4][16] =
{
    10,  0,  9, 14,  6,  3, 15,  5,  1, 13, 12,  7, 11,  4,  2,  8,
    13,  7,  0,  9,  3,  4,  6, 10,  2,  8,  5, 14, 12, 11, 15,  1,
    13,  6,  4,  9,  8, 15,  3,  0, 11,  1,  2, 12,  5, 10, 14,  7,
    1,  10, 13,  0,  6,  9,  8,  7,  4, 15, 14,  3, 11,  5,  2, 12
};

int S4[4][16] =
{
    7,  13, 14,  3,  0,  6,  9, 10,  1,  2,  8,  5, 11, 12,  4, 15,
    13,  8, 11,  5,  6, 15,  0,  3,  4,  7,  2, 12,  1, 10, 14,  9,
    10,  6,  9,  0, 12, 11,  7, 13, 15,  1,  3, 14,  5,  2,  8,  4,
    3,  15,  0,  6, 10,  1, 13,  8,  9,  4,  5, 11, 12,  7,  2, 14
};

int S5[4][16] =
{
    2,  12,  4,  1,  7, 10, 11,  6,  8,  5,  3, 15, 13,  0, 14,  9,
    14, 11,  2, 12,  4,  7, 13,  1,  5,  0, 15, 10,  3,  9,  8,  6,
    4,   2,  1, 11, 10, 13,  7,  8, 15,  9, 12,  5,  6,  3,  0, 14,
    11,  8, 12,  7,  1, 14,  2, 13,  6, 15,  0,  9, 10,  4,  5,  3
};

int S6[4][16] =
{
    12,  1, 10, 15,  9,  2,  6,  8,  0, 13,  3,  4, 14,  7,  5, 11,
    10, 15,  4,  2,  7, 12,  9,  5,  6,  1, 13, 14,  0, 11,  3,  8,
```

```
    9,  14, 15,  5,  2,  8, 12,  3,  7,  0,  4, 10,  1, 13, 11,  6,
    4,  3,  2, 12,  9,  5, 15, 10, 11, 14,  1,  7,  6,  0,  8, 13
};

int S7[4][16]=
{
    4,  11,  2, 14, 15,  0,  8, 13,  3, 12,  9,  7,  5, 10,  6,  1,
    13,  0, 11,  7,  4,  9,  1, 10, 14,  3,  5, 12,  2, 15,  8,  6,
    1,  4, 11, 13, 12,  3,  7, 14, 10, 15,  6,  8,  0,  5,  9,  2,
    6,  11, 13,  8,  1,  4, 10,  7,  9,  5,  0, 15, 14,  2,  3, 12
};

int S8[4][16]=
{
    13,  2,  8,  4,  6, 15, 11,  1, 10,  9,  3, 14,  5,  0, 12,  7,
    1,  15, 13,  8, 10,  3,  7,  4, 12,  5,  6, 11,  0, 14,  9,  2,
    7,  11,  4,  1,  9, 12, 14,  2,  0,  6, 10, 13, 15,  3,  5,  8,
    2,  1, 14,  7,  4, 10,  8, 13, 15, 12,  9,  0,  3,  5,  6, 11
};

int P[] =
{
    16,  7, 20, 21,        // p-box
    29, 12, 28, 17,
    1,  15, 23, 26,
    5,  18, 31, 10,
    2,  8, 24, 14,
    32, 27,  3,  9,
    19, 13, 30,  6,
    22, 11,  4, 25
};

int PC1[] =
{
    57, 49, 41, 33, 25, 17,  9,        // key permutation table
    1,  58, 50, 42, 34, 26, 18,
    10,  2, 59, 51, 43, 35, 27,
    19, 11,  3, 60, 52, 44, 36,
```

```
    63, 55, 47, 39, 31, 23, 15,
    7,  62, 54, 46, 38, 30, 22,
    14,  6, 61, 53, 45, 37, 29,
    21, 13,  5, 28, 20, 12,  4
};

int PC2[] =
{
    14, 17, 11, 24,  1,  5,          // compression permutation
    3,  28, 15,  6, 21, 10,
    23, 19, 12,  4, 26,  8,
    16,  7, 27, 20, 13,  2,
    41, 52, 31, 37, 47, 55,
    30, 40, 51, 45, 33, 48,
    44, 49, 39, 56, 34, 53,
    46, 42, 50, 36, 29, 32
};

int SHIFTS[] = { 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1 };
// num of key bits shifted per round

// change 4 bits binary to decimal
int Bin4toDec(int bin4[]){
    int i, d=0, x=8;
    for(i=0;i<4;i++){
        if(bin4[i]==1){
            d+=x;
        }
        x=x/2;
    }
    return d;
}

// change 'size' bits binary from 'k' to hex 'arr'
void BintoHex(int k[], int size, char arr[]){
    int i, j, d, c;
    for(i=0;i<size;i++){
        for(j=0;j<4;j++){
```

```
                bin4[j] = k[i*4+j];
            }
            d = Bin4toDec(bin4);
            if(d>9){
                c=d+55;
            }
            else{
                c=d+48;
            }
            arr[i] = c;
        }
}

// out[i] = in[permutation_table[i]]
void exchange(int in[], int per[], int out[], int size){
        int i, x;
        for(i=0;i<size;i++){
            x = per[i]-1;
            out[i] = in[x];
        }
}

// merge R[16]L[16] and use IIP, inverse of initial permutation.
void merge(){
        int i, j;
        for(i=0;i<32;i++){
            MERGE[i] = R32[16][i];
        }
        for(i=32;i<64;i++){
            MERGE[i] = L32[16][i-32];
        }

        exchange(MERGE, IIP, FINAL, 64);

        printf("\n");
        BintoHex(FINAL, 16, CIPHER);
}
```

```
// 5. R[n] = L[n-1]+f(R[n-1], K[n])
void R_XorCalculator(int k[], int r[], int num){
	int i;
	for(i=0;i<32;i++){
		if(k[i]==r[i]){
			R32[num][i] = 0;
		}
		else{
			R32[num][i] = 1;
		}
	}
}

// 4. P-box
void PBox(int num){
	exchange(SB[num], P, PB[num], 32);
}

// change decimal to 4 bits binary
void dectobin4(int dec){
	int x=8, i=0;
	while(x>0){
		if(dec>=x){
			bin4[i] = 1;
			dec=dec-x;
		}
		else{
			bin4[i] = 0;
		}
		i++;
		x=x/2;
	}
}

// 3. S-box, 6*8 -> 4*8
void SBox(int num){
	int i, j, row, col, x, d;
	for(i=0;i<8;i++){
```

```
            x = i*6;
            row = K_ER[num][x]*2 + K_ER[num][x+5];
            col = K_ER[num][x+1]*8 + K_ER[num][x+2]*4 + K_ER[num][x+3]*2 + K_ER[num][x+4];
            if(i==0){
                d = S1[row][col];
            }
            else if (i==1){
                d = S2[row][col];
            }
            else if (i==2){
                d = S3[row][col];
            }
            else if (i==3){
                d = S4[row][col];
            }
            else if (i==4){
                d = S5[row][col];
            }
            else if (i==5){
                d = S6[row][col];
            }
            else if (i==6){
                d = S7[row][col];
            }
            else{
                d = S8[row][col];
            }

            dectobin4(d);

            for(j=0;j<4;j++){
                SB[num][i*4+j] = bin4[j];
            }
        }
}

// 2. k48[n]+E(R[n-1])
void K_ER_XorCalculator(int k[], int r[], int num){
```

```
        int i;
        for(i=0;i<48;i++){
            if(k[i]==r[i]){
                K_ER[num][i] = 0;
            }
            else{
                K_ER[num][i] = 1;
            }
        }
}

// 1. E(R[n-1])
void ERtool(int i){
        exchange(R32[i-1], E, ER[i-1], 48);
}

// L[n] = R[n-1]
void Lntool(int i){
        for(int j=0;j<32;j++){
            L32[i][j]=R32[i-1][j]; // Ln = R(n-1)
        }
}

// L[n] = R[n-1]
// R[n] = L[n-1]+f(R[n-1], K[n]), (5 steps)
void ExpansionPermutation(){
        int i;
        for(i=1;i<17;i++){
            Lntool(i);    // Li
            ERtool(i);    // ER(i-1)
            K_ER_XorCalculator(k48[i], ER[i-1], i); // K_ER[i]
            SBox(i);        // SB[i]
            PBox(i);        // PB[i]
            R_XorCalculator(L32[i-1], PB[i], i);

            BintoHex(L32[i], 8, Left[i]);
            BintoHex(R32[i], 8, Right[i]);
            BintoHex(k48[i], 12, RoundKey[i]);
```

```
    }
}

// 56 bits key[n] -> PC-2 -> 48 bits key[n]
void KeyCompression(){
    for(int i=0;i<17;i++){
        exchange(k56[i], PC2, k48[i], 48);
    }
}

// C[n], D[n] -> shift -> C[n+1], D[n+1] -> merge to 56 bits key[n+1]
void KeyPermutation(){
    int cnt, i, j, x, tempc, tempd;
    for(i=1;i<17;i++){
        cnt = SHIFTS[i-1];
        x = i-1;
        while(cnt!=0){
            tempc = C[x][0];
            tempd = D[x][0];
            for(j=0;j<27;j++){
                C[i][j] = C[x][j+1];
                D[i][j] = D[x][j+1];
                k56[i][j] = C[i][j];
                k56[i][j+28] = D[i][j];
            }
            C[i][j] = tempc;
            D[i][j] = tempd;
            k56[i][j] = C[i][j];
            k56[i][j+28] = D[i][j];
            cnt--;
            x++;
        }
    }
}

// 64 bits initial key -> PC-1 -> 56 bits key
void InitialKeySelection(){
    exchange(key, PC1, k56[0], 56);
```

```
        for(int i=0;i<56;i++){
            if(i<28){
                C[0][i] = k56[0][i];
            }
            else{
                D[0][i-28] = k56[0][i];
            }
        }
}

// initial Plaintext -> IP -> IP(P)
void InitialPermution(){
        exchange(pt, IP, ip_p, 64);
        for(int i=0;i<64;i++){
            if(i<32){
                L32[0][i] = ip_p[i];
            }
            else{
                R32[0][i-32] = ip_p[i];
            }
        }
        BintoHex(L32[0], 8, Left[0]);
        BintoHex(R32[0], 8, Right[0]);
}

// change char to decimal
int chartodec(char arr){
        int dec, x;
        x = (int)arr;
        if(x>64){
            x = x-55;
        }
        else{
            x = x-48;
        }
        dec = x;

        return dec;
```

```
}

// change char to 64 bits binary
void char16tobinary64(char arr[], int target[]){
	int dec[16], x, temp, i, j;

	for(i=0;i<16;i++){
		x=8;
		j=0;
		dec[i] = chartodec(arr[i]);
		temp=dec[i];
		while(x>0){
			if(temp>=x){
				target[i*4+j] = 1;
				temp = temp-x;
			}
			else{
				target[i*4+j] = 0;
			}
			j++;
			x = x/2;
		}
	}
}

// print array
void printarray(int arr[], int size, int space){
	for(int i=0;i<size;i++){
		if(i!=0 && i%(space*8)==0){
			printf("\n");
		}
		else if(i!=0 && i%space==0){
			printf(" ");
		}
		printf("%d",arr[i]);
	}
	printf("\n");
}
```

```
// print cipher
void printcipher(char arr[], int size){
	printf("[");
	for(int i=0;i<size;i++){
		printf("%c", arr[i]);
	}
	printf("]");
}

int main(int argc, const char * argv[]) {
	char Plaintext[] = "0123456789ABCDEF";
	char Keytext[] = "133457799BBCDFF1";
	printf("2018920065 루안리치 컴퓨터보안 5주차 DES\n");
	printf("Plaintext: 0123456789ABCDEF | Key: 133457799BBCDFF1\n");

	char16tobinary64(Plaintext, pt);
	char16tobinary64(Keytext, key);

	InitialPermution();
	InitialKeySelection();
	KeyPermutation();
	KeyCompression();
	ExpansionPermutation();

	merge();


	printf("------------------------------------------------\n");
	printf(" round		left		right		round key \n");
	for(int i=1;i<17;i++){
		printf(" %d	", i);
		if(i<10){
			printf(" ");
		}
		printcipher(Left[i], 8);
		printf("	");
		printcipher(Right[i], 8);
```

```
        printf("     ");
        printcipher(RoundKey[i], 12);
        printf("\n");
    }
    printf("------------------------------------------------\n");

    printf("Cipher:    ");
    printcipher(CIPHER, 16);
    printf("\n");
    printarray(FINAL, 64, 8);
    printf("\n");
    printf("\n");

    return 0;
}
```

실행결과: