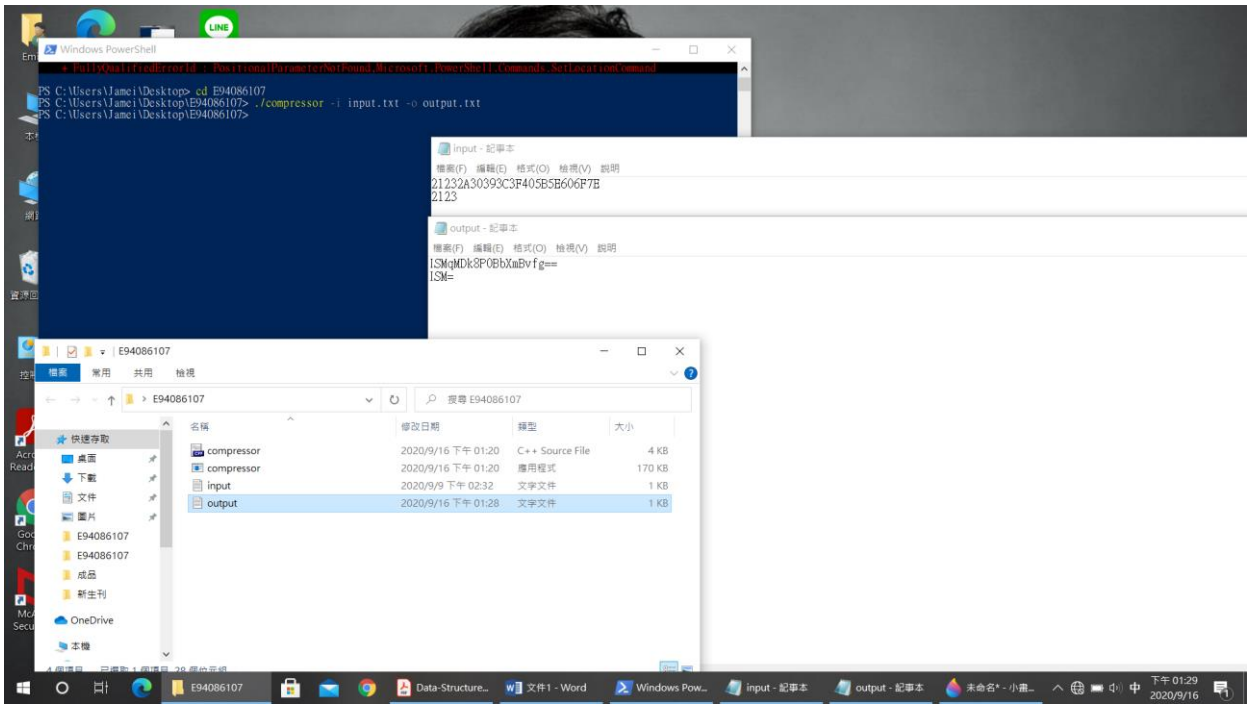
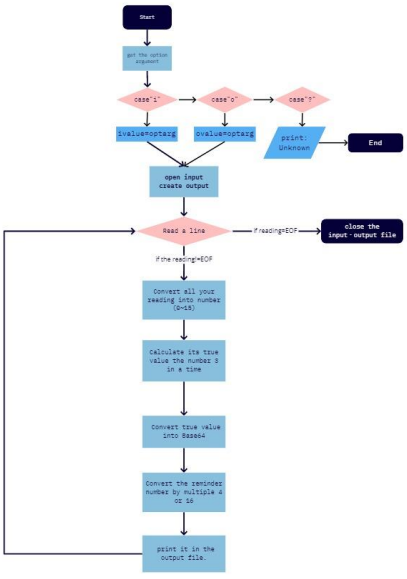
(1) result screenshot



(2) program architecture

1. Open and read the file once line a time.
2. Convert all your reading into number (0~15).
3. Calculate its true value the number 3 in a time. For example, if we have 245 in hexadecimal, in binary it is 0010 0100 0101, this true value is 581.
4. Convert true value into Base64. For example, if the true value is 581, in Base64 is “JF”.
(Method: 581 mod 64 first, and the answer would be the second word; then 581 divided 64, the answer would be the first world.)
5. Convert the reminder number by multiple 4 or 16. For example if the reminder number is 2, its binary is 0010, if you want to convert it into Base64, you need to add two 0 let it be 001000, and in Base64 it is “I”.
6. print it in the output file, if there had reminder you should add “=” or “==”.
7. Redo 1~6 until the end of the file.
8. Close the file.

流程圖：



(3) program function

```
static char encoding_table[] = {'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H',
                                'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P',
                                'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X',
                                'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f',
                                'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n',
                                'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
                                'w', 'x', 'y', 'z', '0', '1', '2', '3',
                                '4', '5', '6', '7', '8', '9', '+', '/'};
```

encoding_table is the encoding table of base64

```
   char *ivalue=NULL;
   char *ovalue=NULL;
   int c;
```

parameter **ivalue** and **ovalue** are used to save the option argument which is caught, and **c** is the parameter to catch option argument.

```
            ivalue=optarg;
```

catch the argument which is after I and assing to parameter ivalue.

```
            case '?':
            printf("Unknown option: %c\n",(char)optopt);
            return 1;
```

return value: if it got incorrect option, talk to user and end the program.

```
      FILE *fp;
      FILE *fpo;
```

Parameter **fp** means the input file and **fpo** means the output file.

```
char origin[1000]={0};// the array use to save original file.
int hex[1000]={0};//the array use to save english word into integer.
```

Parameter:

origin-save the string which is be caught originally

hex-save the word in origin which is converted into integer

```
if(fp==NULL){
    printf("No file.\n");
    return 1;
}
```

Return value: if there didn't have anything in the file, talk to user and end the program.

```
while((fscanf(fp,"%s",origin))!=EOF)
```

read the file one line a time until the file's endding

```
        for(int i=0;i<strlen(origin);i++) {
    hex[i]=convert(origin[i]);
```

convert- a function, which change char into int, and change A~F into 10~15.

```
int convert(char a)
{
    int r;
        switch(a){
            case'0':
                r=0;
                break;
            case'1':
                r=1;
                break;
            case'2':
                r=2;
                break;
            case'3':
                r=3;
                break;
            case'4':
                r=4;
                break;
            case'5':
                r=5;
                break;
            case'6':
                r=6;
                break;
            case'7':
```

```
                r=7;
                break;
            case'8':
                r=8;
                break;
            case'9':
                r=9;
                break;
            case'A':
                r=10;
                break;
            case'B':
                r=11;
                break;
            case'C':
                r=12;
                break;
            case'D':
                r=13;
                break;
            case'E':
                r=14;
                break;
            case'F':
                r=15;
                break;
            default:
            printf("INCORRECT WORD.\n");
            return 1;
            }
    return r;
}
```

Parameter:

a- the origin char.

r- the word which should be return.

return value: the converted integer in usual case, if we find the word which is not in hex, print out and end the program.

```
    int SIZEorigin=(int)strlen(origin);
    int SIZEhex=SIZEorigin;
    int SIZEcon=(SIZEhex*2/3)+2;
    char convert[SIZEcon]={0};
    memset(convert,0,SIZEcon);
    int count=0;
    int re=0;
```

Parameter:

SIZEorigin- the size of array “origin”, since function strlen() return a size_t value, we should

convert it into integer than assign.
SIZEhex- the size of array “hex”, it should be the same as SIZEorigin.
SIZEcon- the size of array “convert”. With the compressor’s characteristic, it should be origin*2/3. However, we need to prepare two space incase of complementary bits.
convert- the array use to save the converted string.
memset(convert,0,SIZEcon)- function used to set convert, let all the element be 0 before the usage.
count- the number used to record how many conversions did we complete.
re- the number used to record where we are converting..

```
for(count=0;count<SIZEcon-2;count=count+2)//we get 3 number in once and convert to 2 number in base64
    {
        int real=0;//to calculate the sum of 3 number
            for (int i=0;i<3;i++)
        {
        real=real+hex[re+i]*pow(16,2-i);//add number from right to left
        }
        int    k=real%64;//convert then with 64,check the code by encoding table
        convert[count+1]=encoding_table[k];
        real=real/64;
        convert[count]=encoding_table[real];

        re=re+3;
    }
```

Parameter: **real-** used to save the sum of the 3 number.
We convert the number 3 in once, first we add them from right to left (multiple with 16^0 ,16 or 16^2, since this is the nearest one of real meaning.) . Then, we get the converted by the method of divided.

```
    if(SIZEorigin%3==1)
    {
        int real=hex[SIZEhex-1];
        real=real*4;
        convert[SIZEcon-2]=encoding_table[real];
        fprintf(fpo,"%s=\n",convert);
    }
    else if(SIZEorigin%3==2)
    {
        int real=hex[SIZEhex-2]*16+hex[SIZEhex-1];
        real=real*16;
        int    r=real%64;
        convert[SIZEcon-2]=encoding_table[r];
        real=real/64;
        convert[SIZEcon-3]=encoding_table[real];
        fprintf(fpo,"%s==\n",convert);
    }
```

```
        else fprintf(fpo,"%s\n",convert);
        }
        fclose(fp);
        fclose(fpo);
        return 0;
}
```

After we convert most of the number, there may have reminder. We should convert reminder by add 0 after their binary compression (which means multiple 4 or 16.) , we would also record how many 0 (2 or 4) we add when we print out. (it will have 1 or 2 “=” mark.)

Return value: return value is 0 since we finished the program successfully.

(4) how you design your program

First, encoding 3 number in once is the key in my program. In the beginning, I want to convert all the word into binary and convert them into decimal. However, I quickly find out the line is too long and I can’t save the true value successfully, so I change to this way after thinking. This could save some storage and it could also convert easily.

Second, my compressor could only transform one line in the begin, until I asked my classmate, he advised me to build a loop so that the compressor could be reused.

Last, I think my program is a little bit longer when I compare with my classmate, maybe some people could use the convertor inside C to convert hexadecimal, but I think the function convert first. My program might be unclean when I convert hexadecimal into Base64. Next time, I could try other way to test if it would be shorter or not.