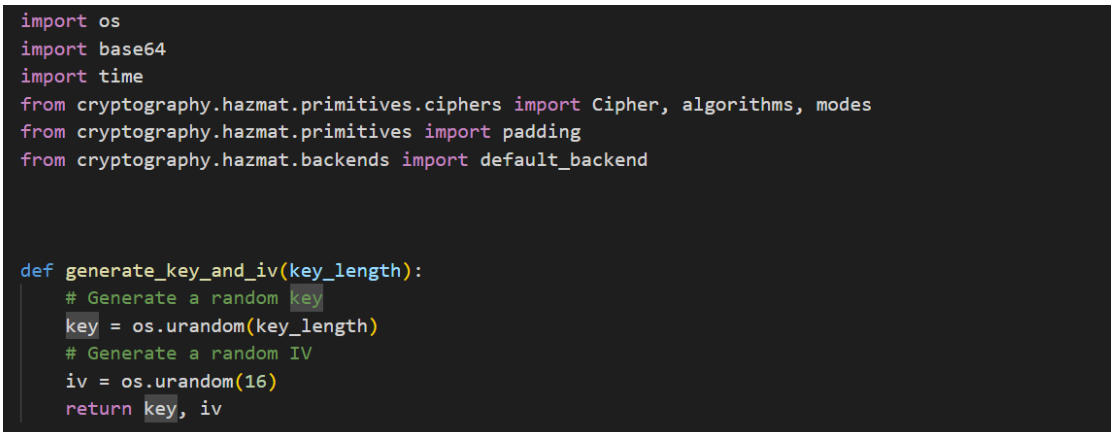
[ 作業一 ]

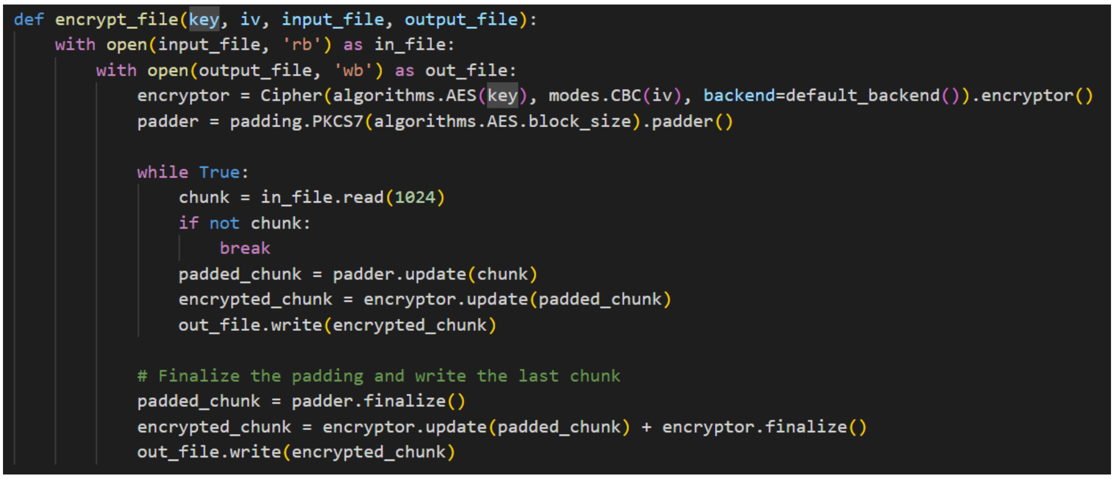
409410025 邱晨恩 資工三

## [ 1 ] 使用 AES-CBC mode 加密

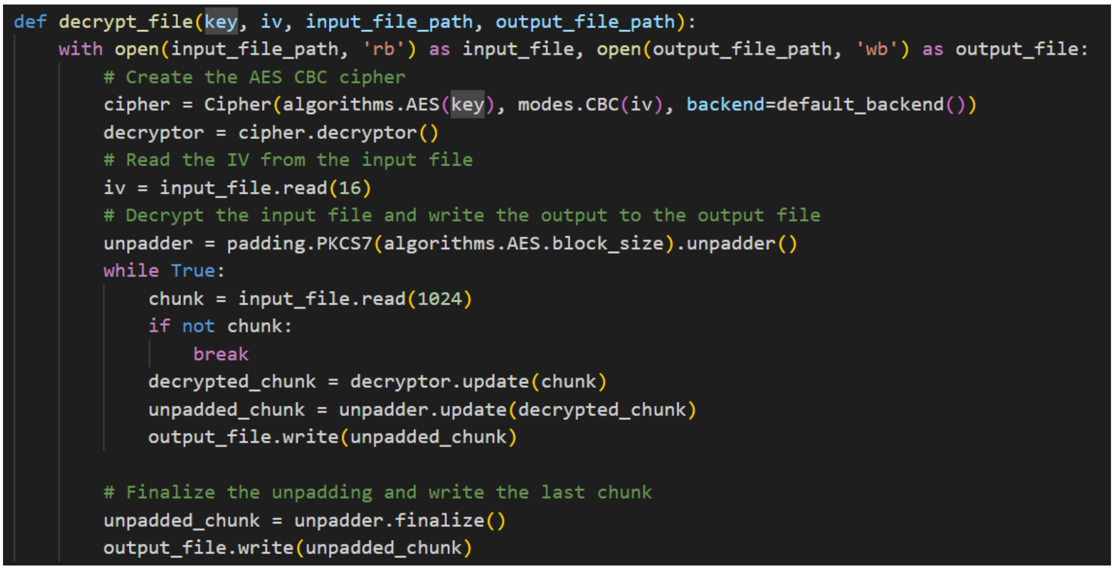
### ( a ) 原始程式碼與說明被加密的檔案大小



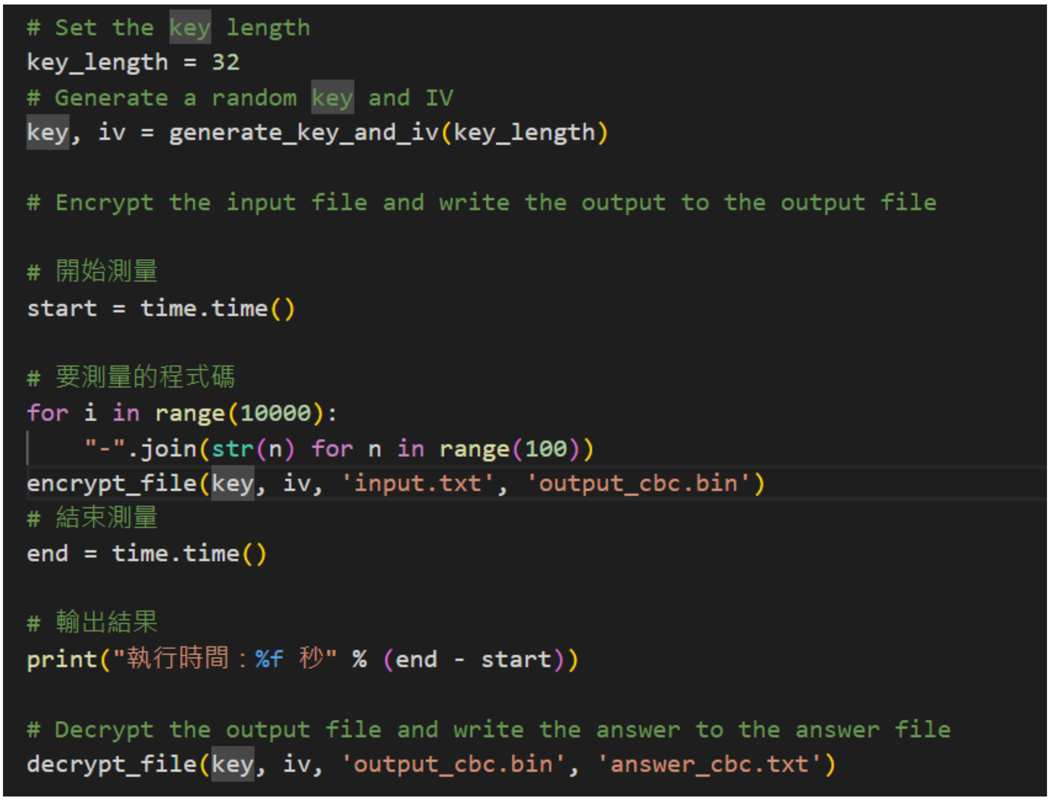
```python
import os
import base64
import time
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.primitives import padding
from cryptography.hazmat.backends import default_backend



def generate_key_and_iv(key_length):
    # Generate a random key
    key = os.urandom(key_length)
    # Generate a random IV
    iv = os.urandom(16)
    return key, iv
```



```python
def encrypt_file(key, iv, input_file, output_file):
    with open(input_file, 'rb') as in_file:
        with open(output_file, 'wb') as out_file:
            encryptor = Cipher(algorithms.AES(key), modes.CBC(iv), backend=default_backend()).encryptor()
            padder = padding.PKCS7(algorithms.AES.block_size).padder()

            while True:
                chunk = in_file.read(1024)
                if not chunk:
                    break
                padded_chunk = padder.update(chunk)
                encrypted_chunk = encryptor.update(padded_chunk)
                out_file.write(encrypted_chunk)

            # Finalize the padding and write the last chunk
            padded_chunk = padder.finalize()
            encrypted_chunk = encryptor.update(padded_chunk) + encryptor.finalize()
            out_file.write(encrypted_chunk)
```

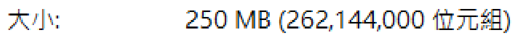


```python
def decrypt_file(key, iv, input_file_path, output_file_path):
    with open(input_file_path, 'rb') as input_file, open(output_file_path, 'wb') as output_file:
        # Create the AES CBC cipher
        cipher = Cipher(algorithms.AES(key), modes.CBC(iv), backend=default_backend())
        decryptor = cipher.decryptor()
        # Read the IV from the input file
        iv = input_file.read(16)
        # Decrypt the input file and write the output to the output file
        unpadder = padding.PKCS7(algorithms.AES.block_size).unpadder()
        while True:
            chunk = input_file.read(1024)
            if not chunk:
                break
            decrypted_chunk = decryptor.update(chunk)
            unpadded_chunk = unpadder.update(decrypted_chunk)
            output_file.write(unpadded_chunk)

        # Finalize the unpadding and write the last chunk
        unpadded_chunk = unpadder.finalize()
        output_file.write(unpadded_chunk)
```



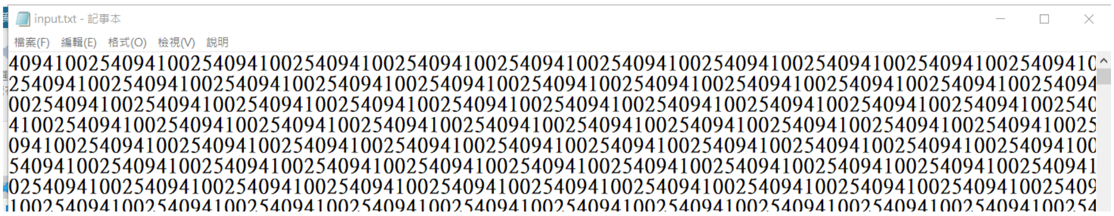
```
# Set the key length
key_length = 32
# Generate a random key and IV
key, iv = generate_key_and_iv(key_length)

# Encrypt the input file and write the output to the output file

# 開始測量
start = time.time()

# 要測量的程式碼
for i in range(10000):
    "-".join(str(n) for n in range(100))
encrypt_file(key, iv, 'input.txt', 'output_cbc.bin')
# 結束測量
end = time.time()

# 輸出結果
print("執行時間：%f 秒" % (end - start))

# Decrypt the output file and write the answer to the answer file
decrypt_file(key, iv, 'output_cbc.bin', 'answer_cbc.txt')
```

被加密的檔案 ：input.txt



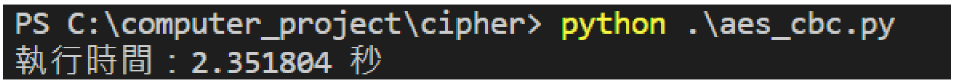
大小:　　　　250 MB (262,144,000 位元組)

內容 ： 我的學號



```
409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410
25409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025094
002540941002540941002540941002540941002540941002540941002540941002540941002540941002540941002540
410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025
094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100
540941002540941002540941002540941002540941002540941002540941002540941002540941002540941002540941
025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409
100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254
```

( b )



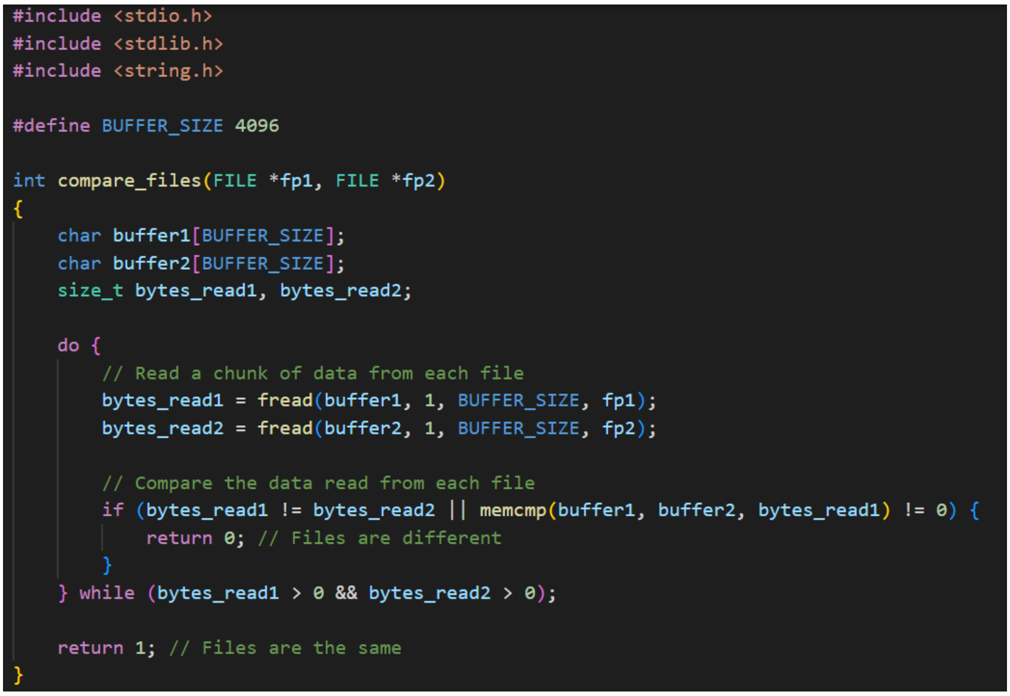
```
PS C:\computer_project\cipher> python .\aes_cbc.py
執行時間：2.351804 秒
```

250MB 是 25010241024=262,144,000 bytes。 處理時間為 2.3518 秒，因此平均每秒可處理 262,144,000 / 2.3518 = 111,302,656 bytes，約 111 MB。

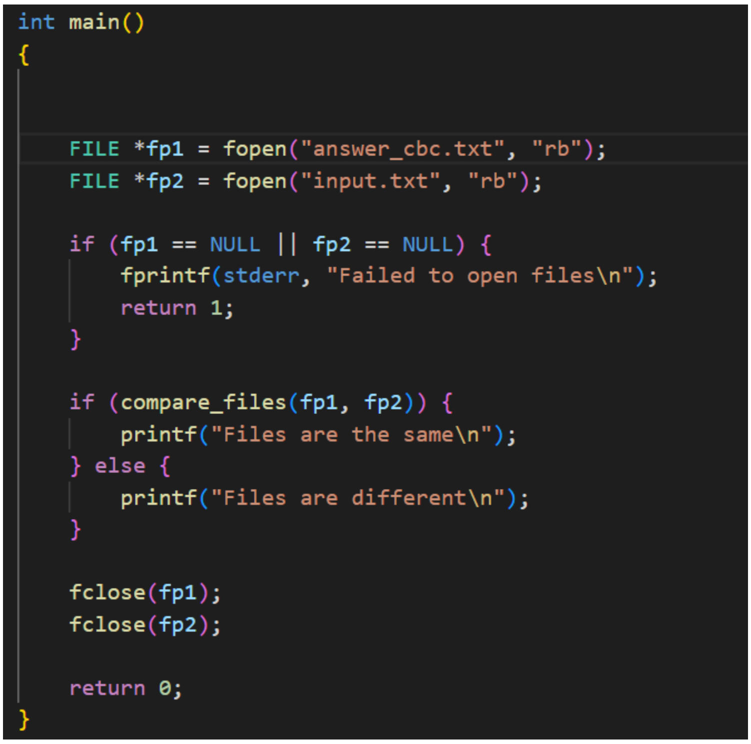
( c )

加密後的檔案為 answer_cbc.txt ，跑程式去比對 input.txt 是否相同 。

程式碼如下 ：

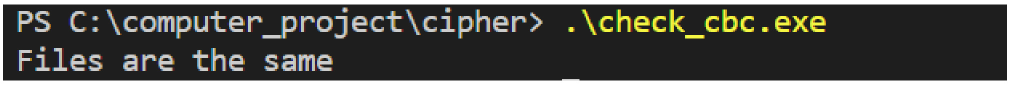


```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define BUFFER_SIZE 4096

int compare_files(FILE *fp1, FILE *fp2)
{
    char buffer1[BUFFER_SIZE];
    char buffer2[BUFFER_SIZE];
    size_t bytes_read1, bytes_read2;

    do {
        // Read a chunk of data from each file
        bytes_read1 = fread(buffer1, 1, BUFFER_SIZE, fp1);
        bytes_read2 = fread(buffer2, 1, BUFFER_SIZE, fp2);

        // Compare the data read from each file
        if (bytes_read1 != bytes_read2 || memcmp(buffer1, buffer2, bytes_read1) != 0) {
            return 0; // Files are different
        }
    } while (bytes_read1 > 0 && bytes_read2 > 0);

    return 1; // Files are the same
}
```

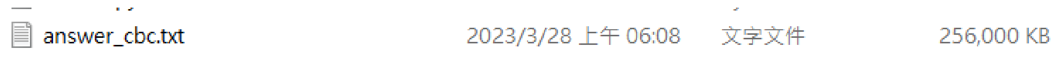


```c
int main()
{


    FILE *fp1 = fopen("answer_cbc.txt", "rb");
    FILE *fp2 = fopen("input.txt", "rb");

    if (fp1 == NULL || fp2 == NULL) {
        fprintf(stderr, "Failed to open files\n");
        return 1;
    }

    if (compare_files(fp1, fp2)) {
        printf("Files are the same\n");
    } else {
        printf("Files are different\n");
    }

    fclose(fp1);
    fclose(fp2);

    return 0;
}
```

執行結果 ：



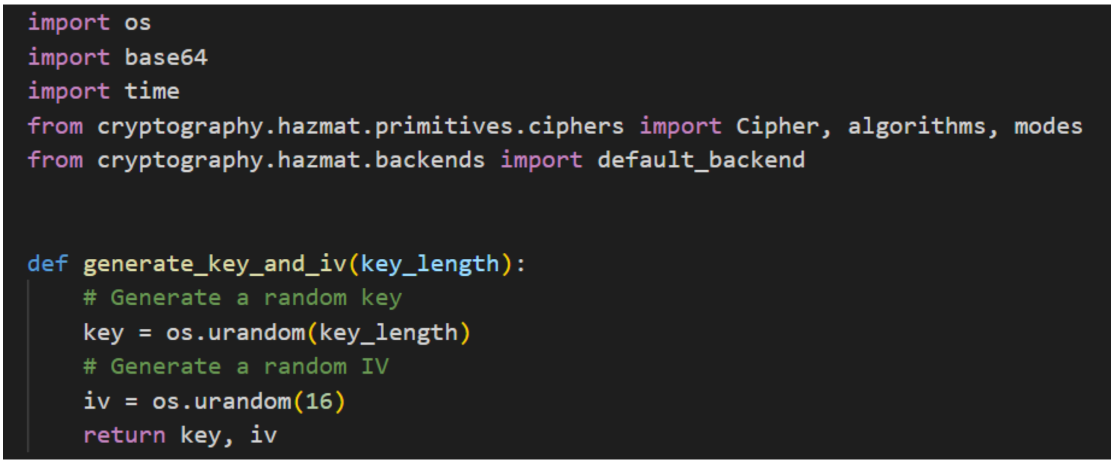
```
PS C:\computer_project\cipher> .\check_cbc.exe
Files are the same
```



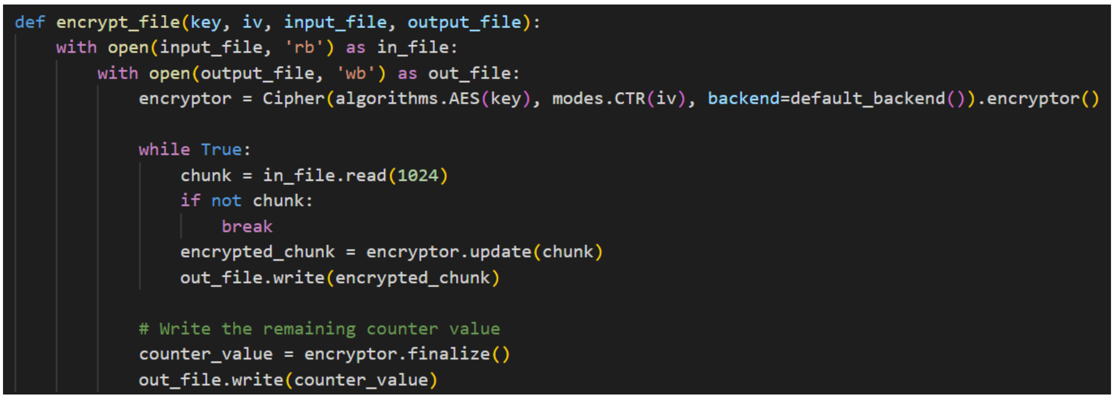
answer_cbc.txt 2023/3/28 上午 06:08 文字文件 256,000 KB

[ 2 ] 使用 AES-CTR mode (counter mode)加密

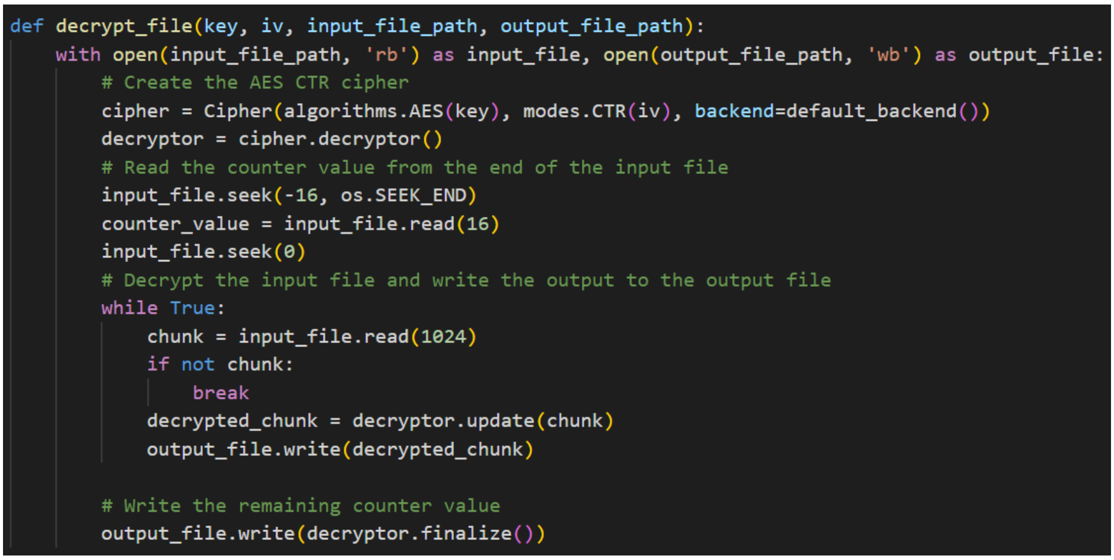
( a ) 原始程式碼與說明被加密的檔案大小



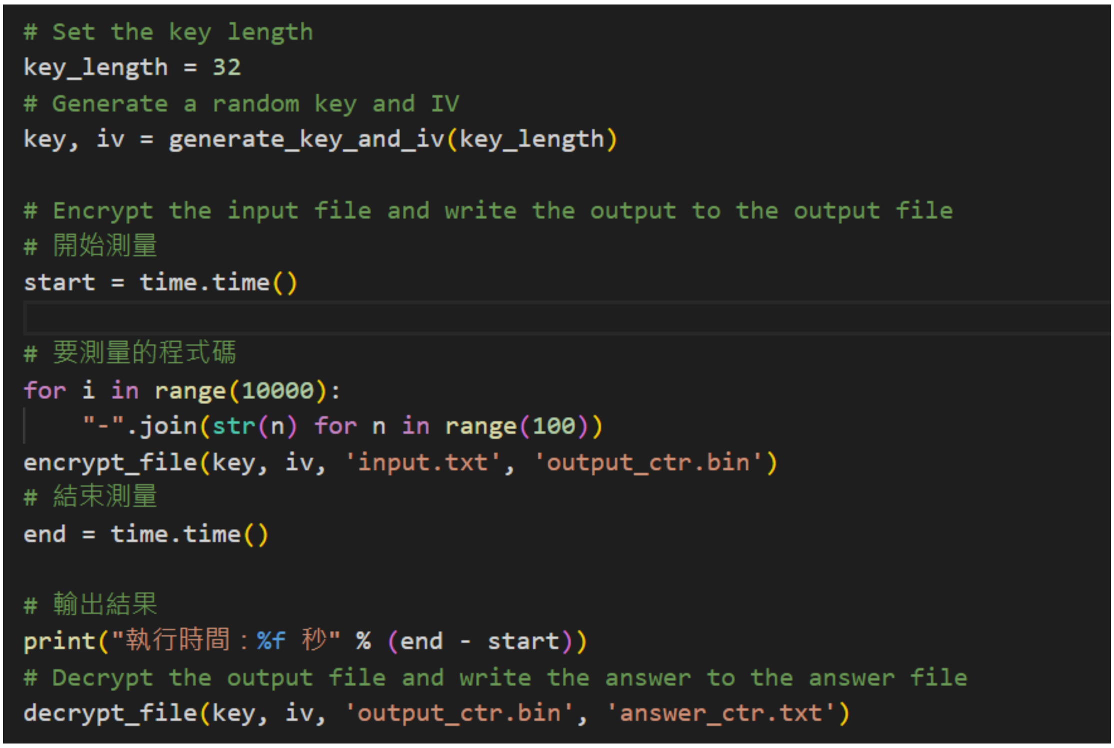
```python
import os
import base64
import time
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.backends import default_backend


def generate_key_and_iv(key_length):
    # Generate a random key
    key = os.urandom(key_length)
    # Generate a random IV
    iv = os.urandom(16)
    return key, iv
```



```python
def encrypt_file(key, iv, input_file, output_file):
    with open(input_file, 'rb') as in_file:
        with open(output_file, 'wb') as out_file:
            encryptor = Cipher(algorithms.AES(key), modes.CTR(iv), backend=default_backend()).encryptor()

            while True:
                chunk = in_file.read(1024)
                if not chunk:
                    break
                encrypted_chunk = encryptor.update(chunk)
                out_file.write(encrypted_chunk)

            # Write the remaining counter value
            counter_value = encryptor.finalize()
            out_file.write(counter_value)
```

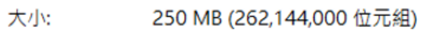


```python
def decrypt_file(key, iv, input_file_path, output_file_path):
    with open(input_file_path, 'rb') as input_file, open(output_file_path, 'wb') as output_file:
        # Create the AES CTR cipher
        cipher = Cipher(algorithms.AES(key), modes.CTR(iv), backend=default_backend())
        decryptor = cipher.decryptor()
        # Read the counter value from the end of the input file
        input_file.seek(-16, os.SEEK_END)
        counter_value = input_file.read(16)
        input_file.seek(0)
        # Decrypt the input file and write the output to the output file
        while True:
            chunk = input_file.read(1024)
            if not chunk:
                break
            decrypted_chunk = decryptor.update(chunk)
            output_file.write(decrypted_chunk)

        # Write the remaining counter value
        output_file.write(decryptor.finalize())
```



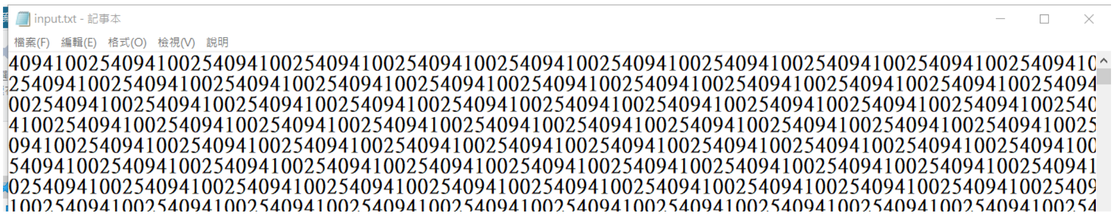
```
# Set the key length
key_length = 32
# Generate a random key and IV
key, iv = generate_key_and_iv(key_length)

# Encrypt the input file and write the output to the output file
# 開始測量
start = time.time()

# 要測量的程式碼
for i in range(10000):
    "-".join(str(n) for n in range(100))
encrypt_file(key, iv, 'input.txt', 'output_ctr.bin')
# 結束測量
end = time.time()

# 輸出結果
print("執行時間：%f 秒" % (end - start))
# Decrypt the output file and write the answer to the answer file
decrypt_file(key, iv, 'output_ctr.bin', 'answer_ctr.txt')
```

被加密的檔案: :input.txt



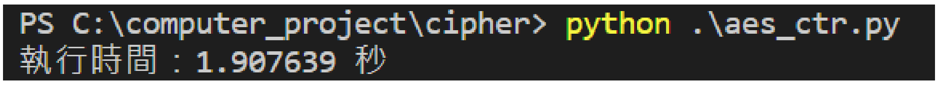
大小: 250 MB (262,144,000 位元組)

內容: : 我的學號



```
409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410
254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094
002540941002540941002540941002540941002540941002540941002540941002540941002540941002540941002540941002540941002540941002540
410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025
094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100
540941002540941002540941002540941002540941002540941002540941002540941002540941002540941002540941002540941002540941002540941
025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409
100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254
```

( b )



```
PS C:\computer_project\cipher> python .\aes_ctr.py
執行時間：1.907639 秒
```

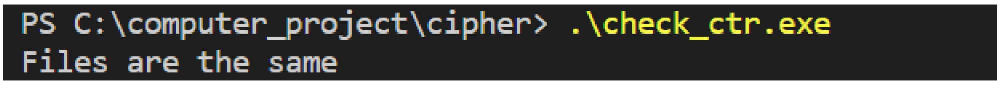
250MB 是 25010241024=262,144,000 bytes。 處理時間為 1.9076 秒，

因此平均每秒可處理 262,144,000 / 1.9076 = 137,422,930 bytes，約 131.1 MB。

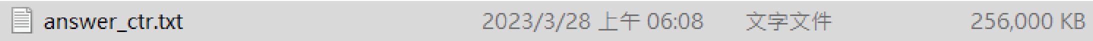
( c )

加密後的檔案為 answer_ctr.txt ， 跑程式去比對 input.txt 是否相同 。

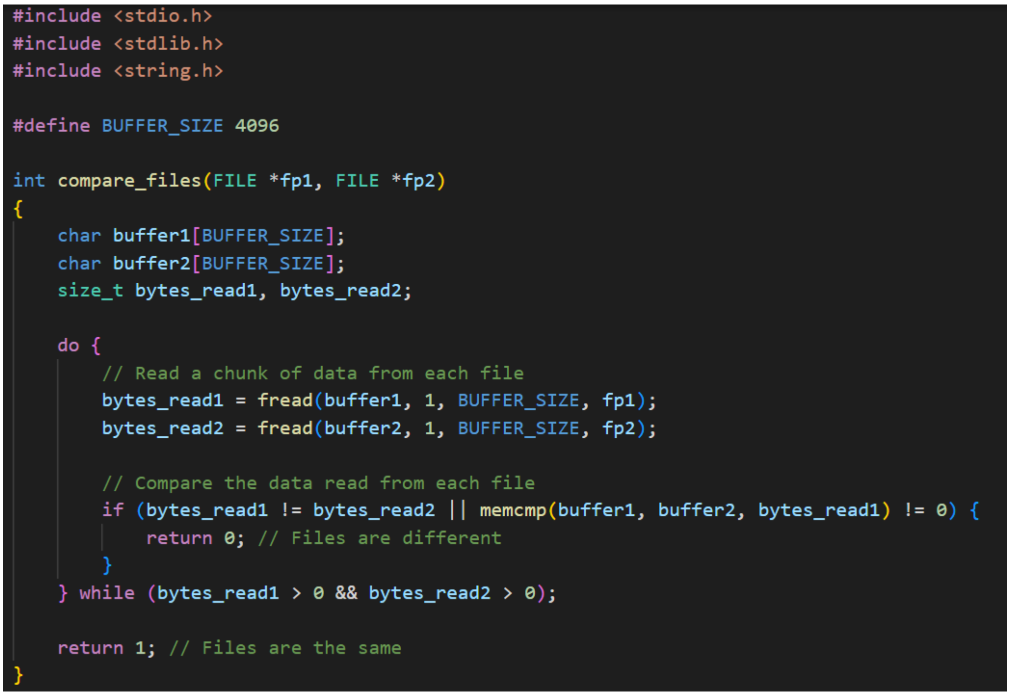
執行結果 ：



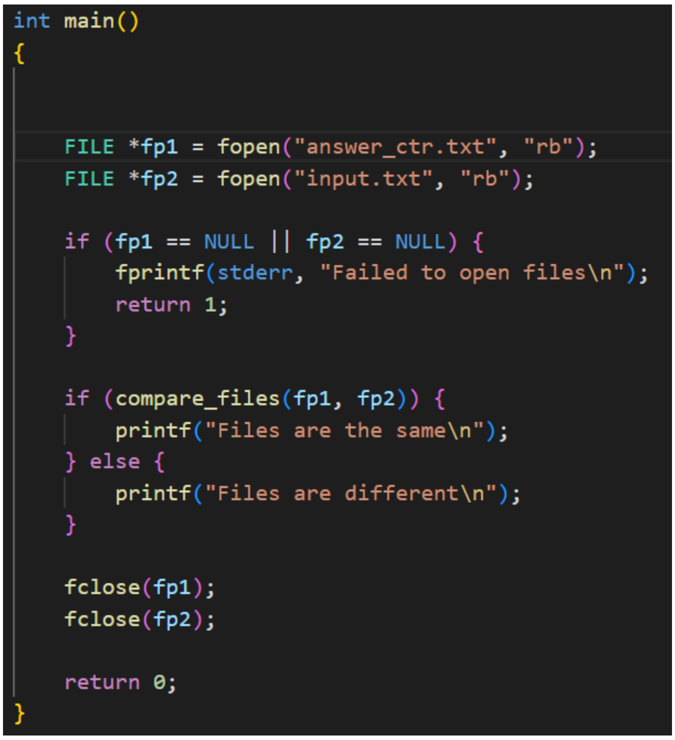
```
PS C:\computer_project\cipher> .\check_ctr.exe
Files are the same
```



answer_ctr.txt 2023/3/28 上午 06:08 文字文件 256,000 KB



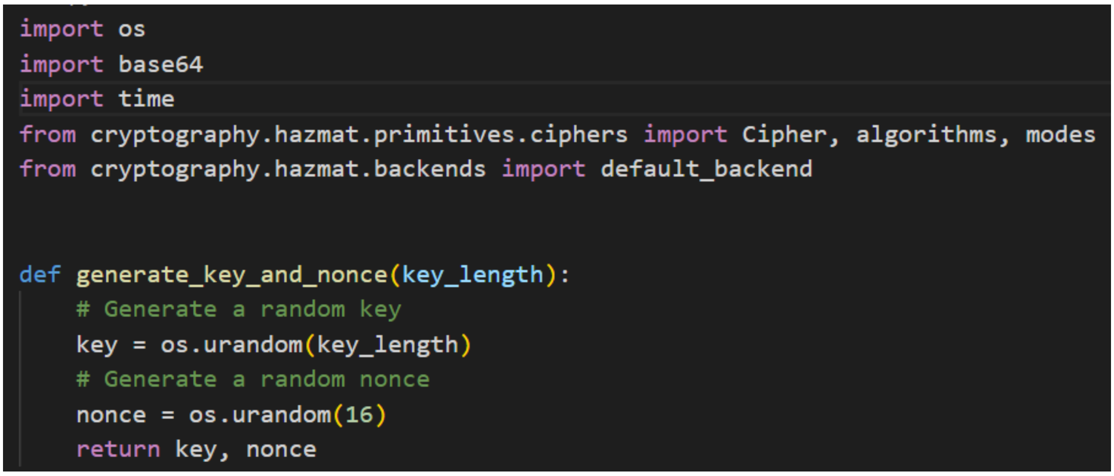
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define BUFFER_SIZE 4096

int compare_files(FILE *fp1, FILE *fp2)
{
    char buffer1[BUFFER_SIZE];
    char buffer2[BUFFER_SIZE];
    size_t bytes_read1, bytes_read2;

    do {
        // Read a chunk of data from each file
        bytes_read1 = fread(buffer1, 1, BUFFER_SIZE, fp1);
        bytes_read2 = fread(buffer2, 1, BUFFER_SIZE, fp2);

        // Compare the data read from each file
        if (bytes_read1 != bytes_read2 || memcmp(buffer1, buffer2, bytes_read1) != 0) {
            return 0; // Files are different
        }
    } while (bytes_read1 > 0 && bytes_read2 > 0);

    return 1; // Files are the same
}
```



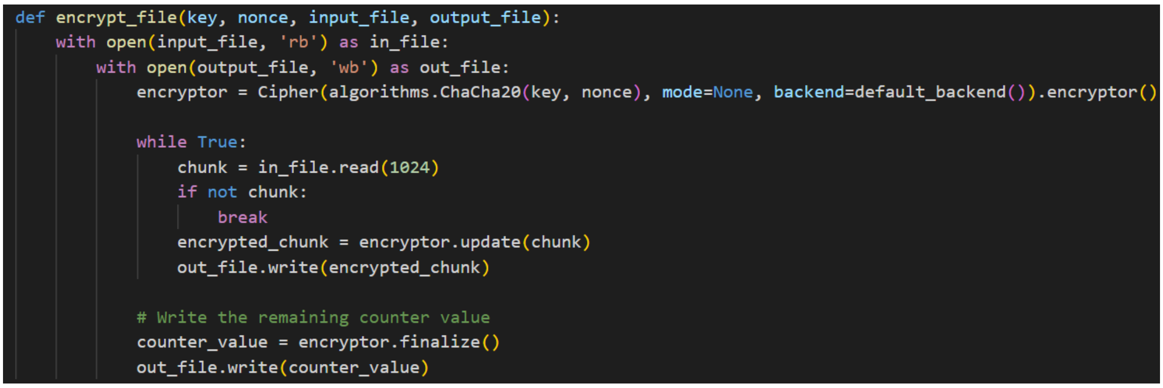
```c
int main()
{


    FILE *fp1 = fopen("answer_ctr.txt", "rb");
    FILE *fp2 = fopen("input.txt", "rb");

    if (fp1 == NULL || fp2 == NULL) {
        fprintf(stderr, "Failed to open files\n");
        return 1;
    }

    if (compare_files(fp1, fp2)) {
        printf("Files are the same\n");
    } else {
        printf("Files are different\n");
    }

    fclose(fp1);
    fclose(fp2);

    return 0;
}
```

## [ 3 ] 使用 ChaCha20 加密

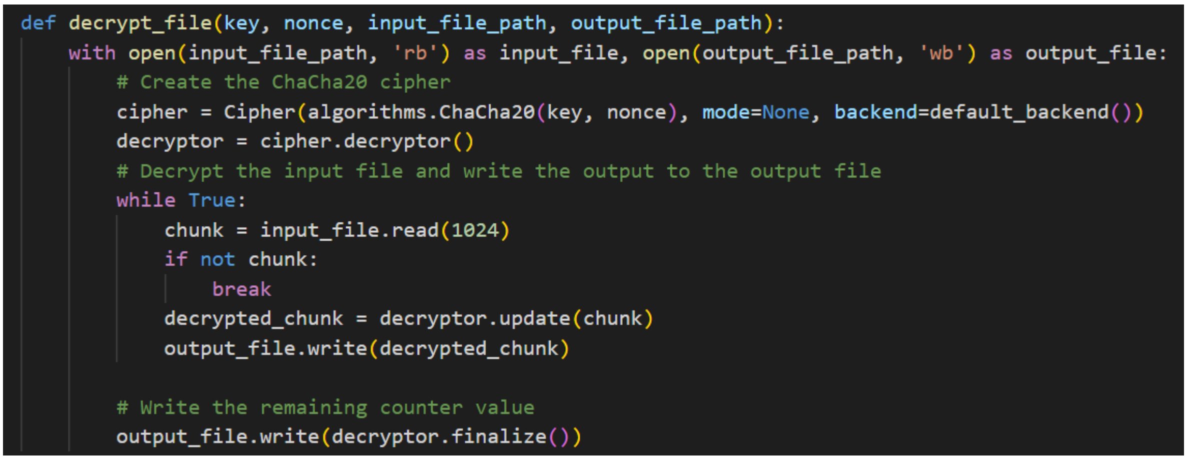
### ( a ) 原始程式碼與說明被加密的檔案大小



```python
import os
import base64
import time
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.backends import default_backend


def generate_key_and_nonce(key_length):
    # Generate a random key
    key = os.urandom(key_length)
    # Generate a random nonce
    nonce = os.urandom(16)
    return key, nonce
```



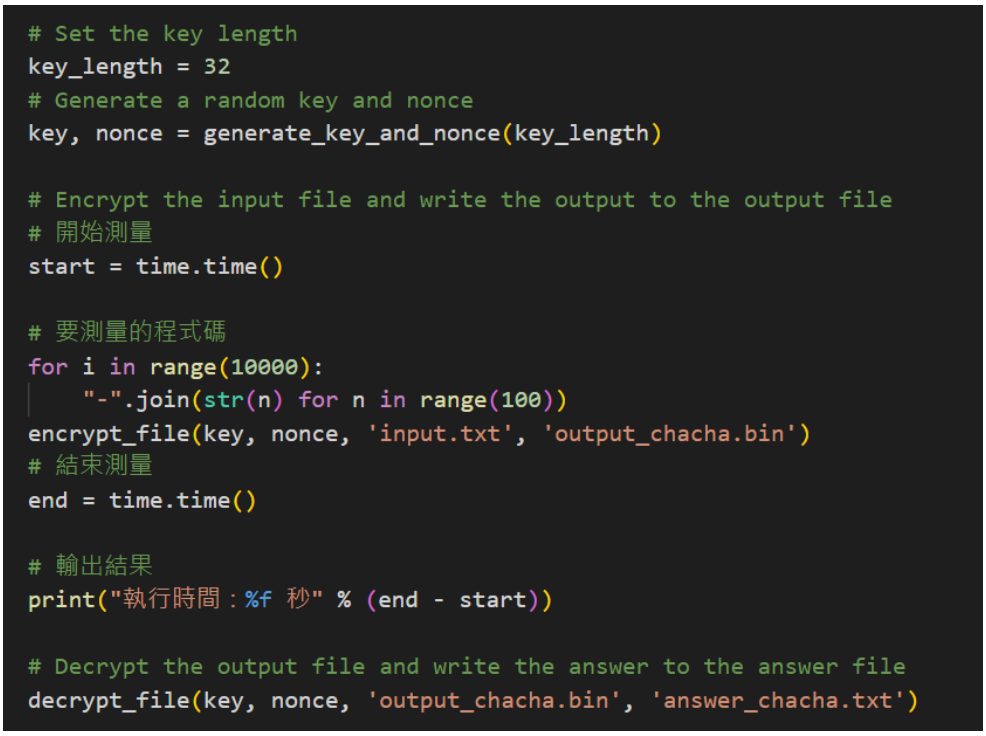
```python
def encrypt_file(key, nonce, input_file, output_file):
    with open(input_file, 'rb') as in_file:
        with open(output_file, 'wb') as out_file:
            encryptor = Cipher(algorithms.ChaCha20(key, nonce), mode=None, backend=default_backend()).encryptor()

            while True:
                chunk = in_file.read(1024)
                if not chunk:
                    break
                encrypted_chunk = encryptor.update(chunk)
                out_file.write(encrypted_chunk)

            # Write the remaining counter value
            counter_value = encryptor.finalize()
            out_file.write(counter_value)
```



```python
def decrypt_file(key, nonce, input_file_path, output_file_path):
    with open(input_file_path, 'rb') as input_file, open(output_file_path, 'wb') as output_file:
        # Create the ChaCha20 cipher
        cipher = Cipher(algorithms.ChaCha20(key, nonce), mode=None, backend=default_backend())
        decryptor = cipher.decryptor()
        # Decrypt the input file and write the output to the output file
        while True:
            chunk = input_file.read(1024)
            if not chunk:
                break
            decrypted_chunk = decryptor.update(chunk)
            output_file.write(decrypted_chunk)

        # Write the remaining counter value
        output_file.write(decryptor.finalize())
```

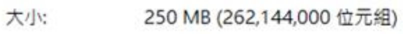
[ 3 ]  使用 ChaCha20 加密

( a )  原始程式碼與說明被加密的檔案大小



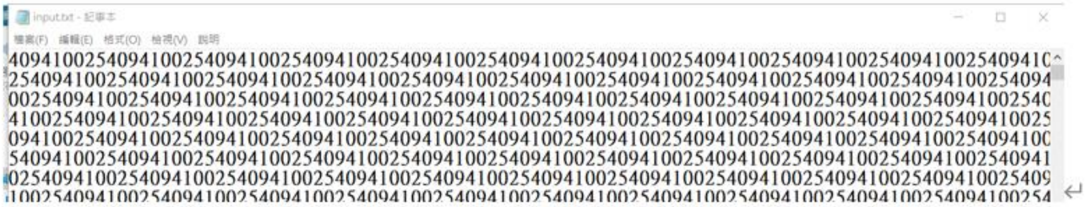
```
# Set the key length
key_length = 32
# Generate a random key and nonce
key, nonce = generate_key_and_nonce(key_length)

# Encrypt the input file and write the output to the output file
# 開始測量
start = time.time()

# 要測量的程式碼
for i in range(10000):
    "-".join(str(n) for n in range(100))
encrypt_file(key, nonce, 'input.txt', 'output_chacha.bin')
# 結束測量
end = time.time()

# 輸出結果
print("執行時間：%f 秒" % (end - start))

# Decrypt the output file and write the answer to the answer file
decrypt_file(key, nonce, 'output_chacha.bin', 'answer_chacha.txt')
```

被加密的檔案 : input.txt



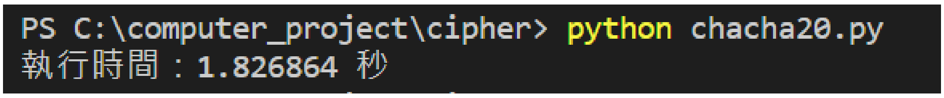
大小:    250 MB (262,144,000 位元組)

內容 : 我的學號



```
409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410C
254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094
002540941002540941002540941002540941002540941002540941002540941002540941002540941002540941002540941002540941002540C
410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025
094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100
540941002540941002540941002540941002540941002540941002540941002540941002540941002540941002540941002540941002540941
025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409410025409
100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254094100254
```

( b )



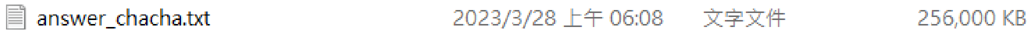
```
PS C:\computer_project\cipher> python chacha20.py
執行時間：1.826864 秒
```

250MB 是 25010241024=262,144,000 bytes。 處理時間為 1.8268 秒，

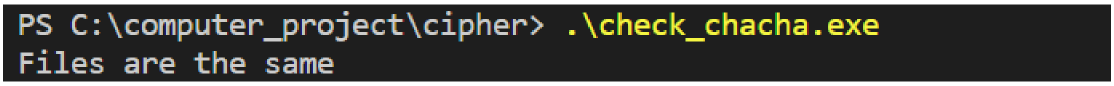
因此平均每秒可處理 262,144,000 / 1.8268 = 136,730,147 bytes 約為 130.3 MB。

（c） 比較解密後的檔案與原始檔案，證明實作正確

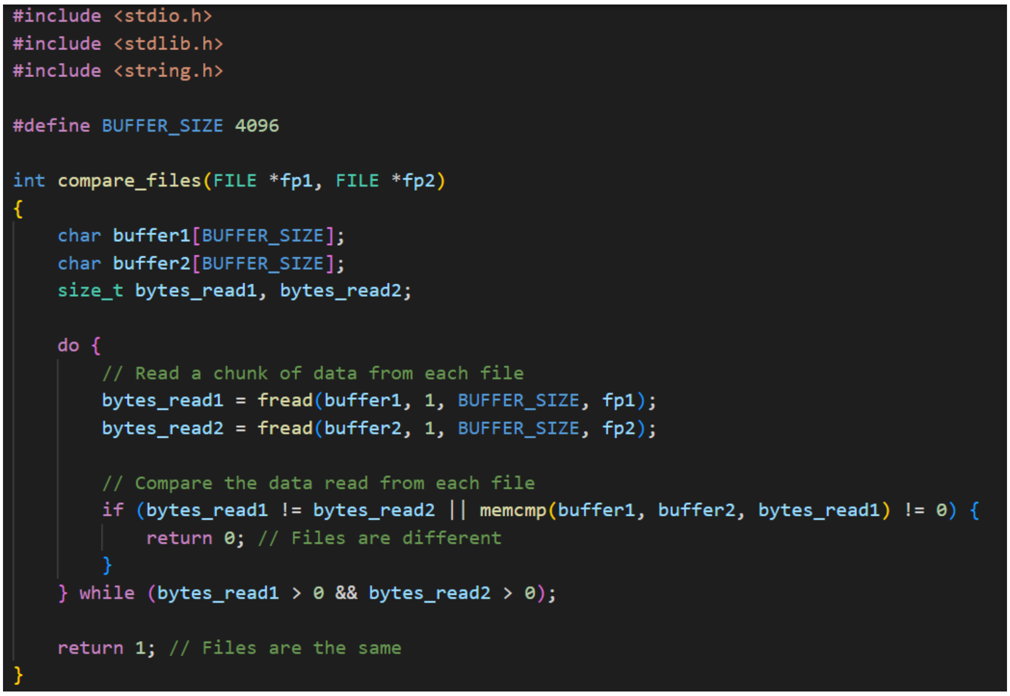
跑程式比對 answer_chacha20.txt 和 input.txt 是否一樣，若一樣則為正確



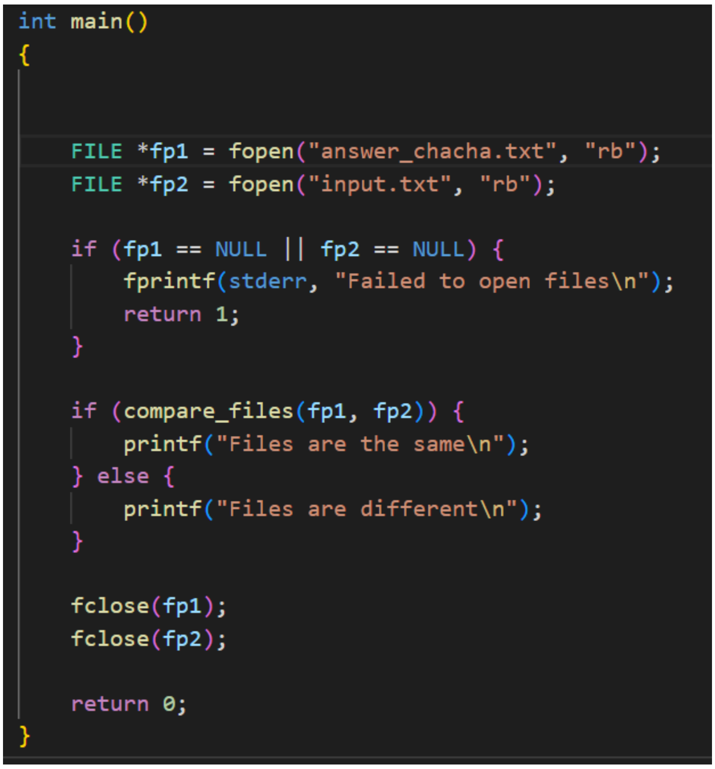
answer_chacha.txt 2023/3/28 上午 06:08 文字文件 256,000 KB



```
PS C:\computer_project\cipher> .\check_chacha.exe
Files are the same
```



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

#define BUFFER_SIZE 4096

int compare_files(FILE *fp1, FILE *fp2)
{
    char buffer1[BUFFER_SIZE];
    char buffer2[BUFFER_SIZE];
    size_t bytes_read1, bytes_read2;

    do {
        // Read a chunk of data from each file
        bytes_read1 = fread(buffer1, 1, BUFFER_SIZE, fp1);
        bytes_read2 = fread(buffer2, 1, BUFFER_SIZE, fp2);

        // Compare the data read from each file
        if (bytes_read1 != bytes_read2 || memcmp(buffer1, buffer2, bytes_read1) != 0) {
            return 0; // Files are different
        }
    } while (bytes_read1 > 0 && bytes_read2 > 0);

    return 1; // Files are the same
}
```



```c
int main()
{


    FILE *fp1 = fopen("answer_chacha.txt", "rb");
    FILE *fp2 = fopen("input.txt", "rb");

    if (fp1 == NULL || fp2 == NULL) {
        fprintf(stderr, "Failed to open files\n");
        return 1;
    }

    if (compare_files(fp1, fp2)) {
        printf("Files are the same\n");
    } else {
        printf("Files are different\n");
    }

    fclose(fp1);
    fclose(fp2);

    return 0;
}
```