**Explain why the latter expression is more efficient?**

1. **Faster mathematics** 
   1. Simplifying expressions
   2. Faster calculation techniques ()
2. **Faster loops (Loop unrolling)**

**for (i=0; i<3; i++){ something(1);**

**something(i); V.S. something(2);**

**} something(3);**

**for (i=0; i<100; i++){**

**stuff(); for (i=0; i<100; i++){**

**} V.S. stuff();**

**for (i=0; i<100; i++){ morestuff();**

**morestuff(); }**

**}**

1. **Avoiding expression repetitions**

V.S.

**if ( (dataStructPointer->ExpensiveFunctionCall()) < 10 ) {  
 //some code here  
}  
else if ( (dataStructPointer->ExpensiveFunctionCal()) > 30 ) {  
    // some code here  
}**

**int temp = dataStructPointer>ExpensiveFunctionCall() ;  
if(temp < 10) {  
    // some code here  
}  
else if(temp > 30) {  
    // some code here  
}**

1. **Using assembly (How to optimize the assembly code? )**

**void main() {**

**int a, b, c,d;**

**a=-11;**

**b=3;**

**c=a/b;**

**d=a%b;**

**}**

**5: c=a/b;**

**00401036 8B 45 FC mov eax,dword ptr [ebp-4]**

**00401039 99 cdq**

**0040103A F7 7D F8 idiv eax,dword ptr [ebp-8]**

**0040103D 89 45 F4 mov dword ptr [ebp-0Ch], eax**

**6: d=a%b;**

**00401040 8B 45 FC mov eax,dword ptr [ebp-4]**

**00401043 99 cdq**

**00401044 F7 7D F8 idiv eax,dword ptr [ebp-8]**

**00401047 89 55 F0 mov dword ptr [ebp-10h], edx**

1. **Integers (If you know the range of that value)**

**// If you know the value will never be decimal**

**float b; V.S. int b;**

**// For decimals**

**double b; V.S. float b;**

1. **Pointer dereferencing**

**for( int i = 0; i < BigNum; i++ ) {  
  Inventory -> StuffToSell -> LowProfitStuff -> Count[i] = Value;   
}**

V.S.

**unsigned int \*InventoryCount =**

**Inventory -> StyffToSell -> LowProfitStuff -> Count;**

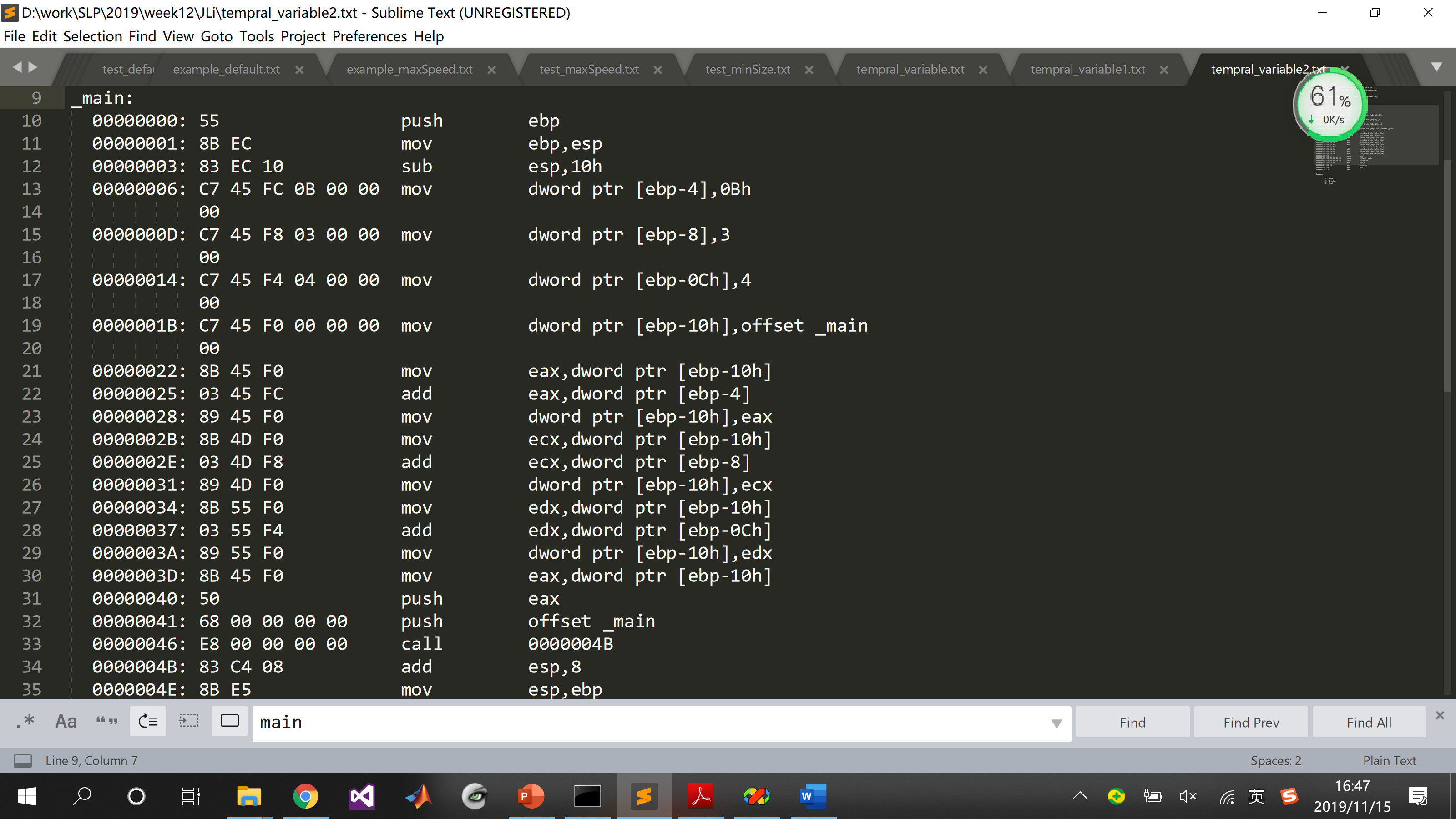
**for( int i = 0; i < BigNum; i++ ) {  
  InventoryCount[i] = Value;  
}**

1. **Avoiding temporary variables**

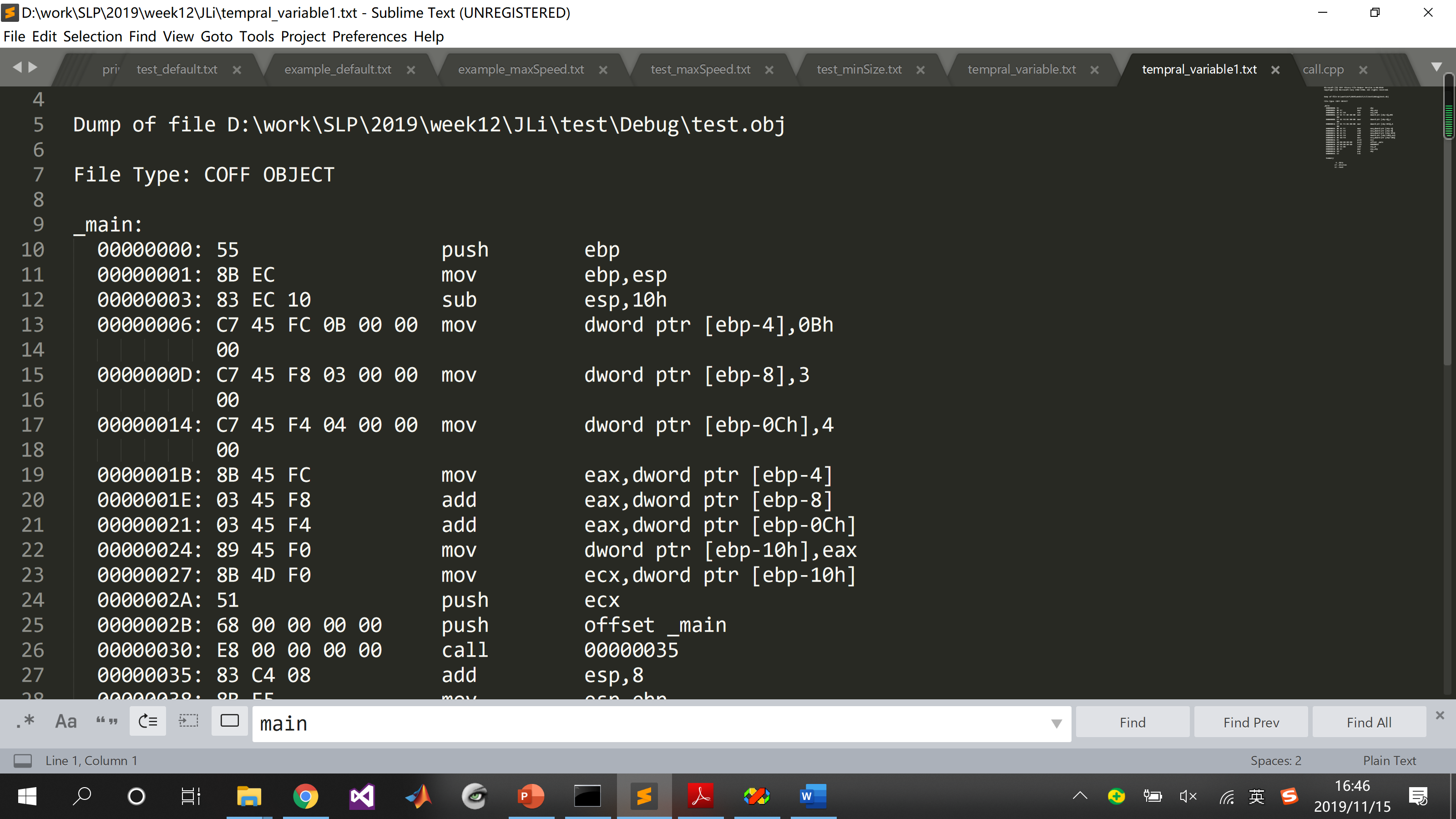
**a = a + a1 + a2 + a3;**

**a += a1;  
  a += a2;  
  a += a3;**

V.S.



V.S.



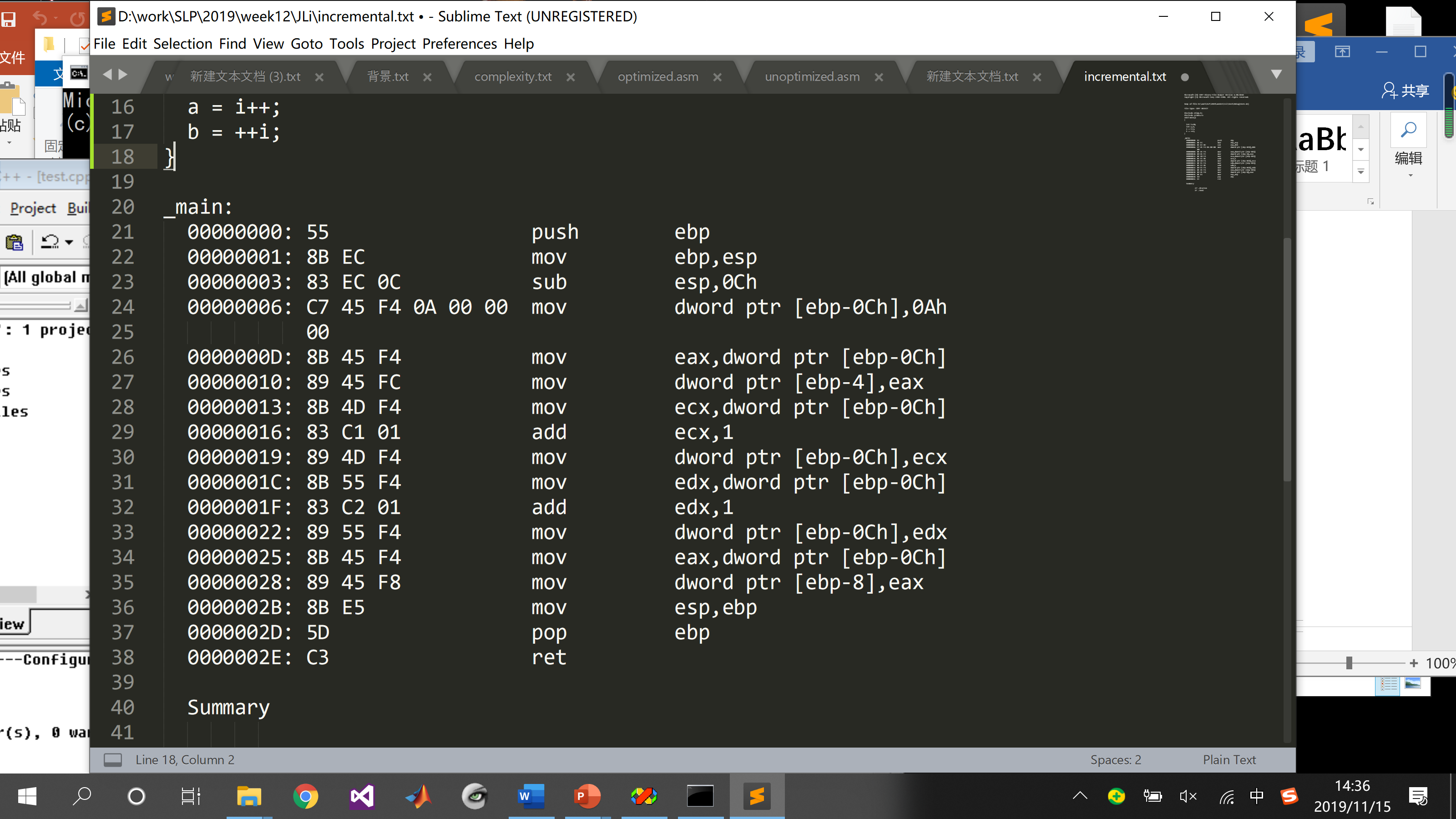
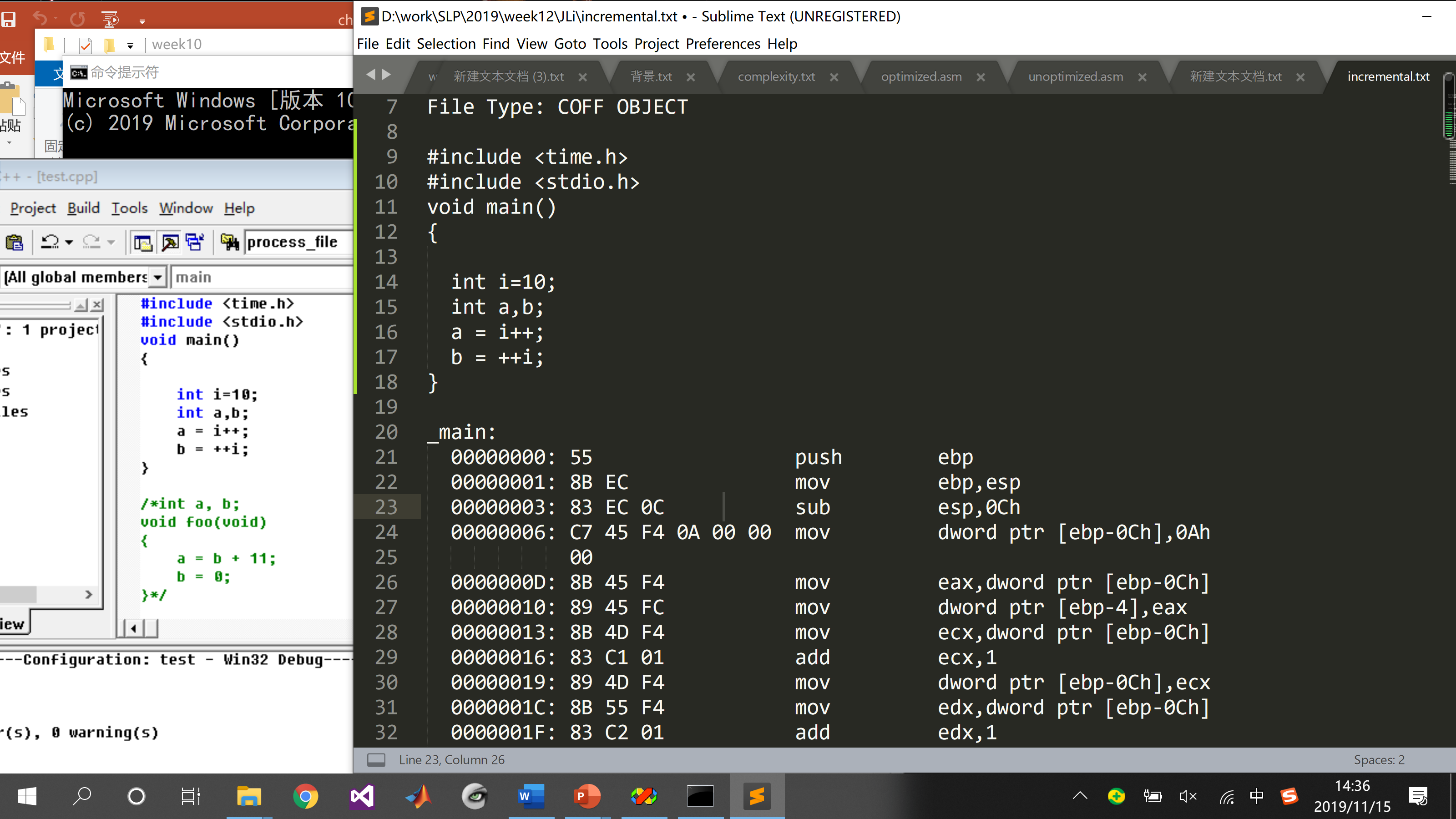
1. **String comparisons (How to improve efficiency?)**

**// compare each element and break out the loop as soon as the different element is found**

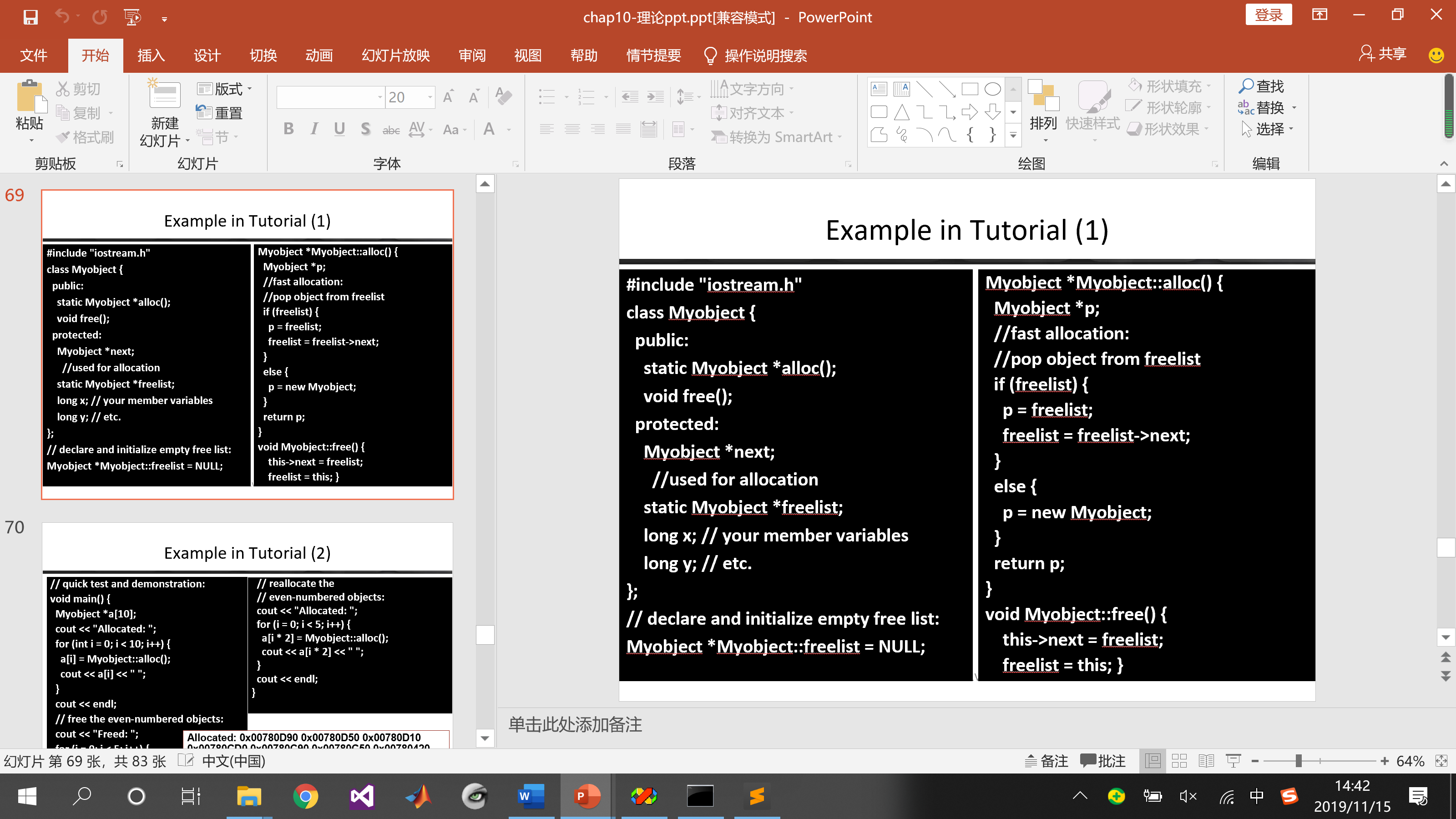
**unsigned char checksum = 0;  
char str[20];  
int len = strlen(str);  
for(int i = 0; i < len; i++) {  
  checksum += str[i];  
}**

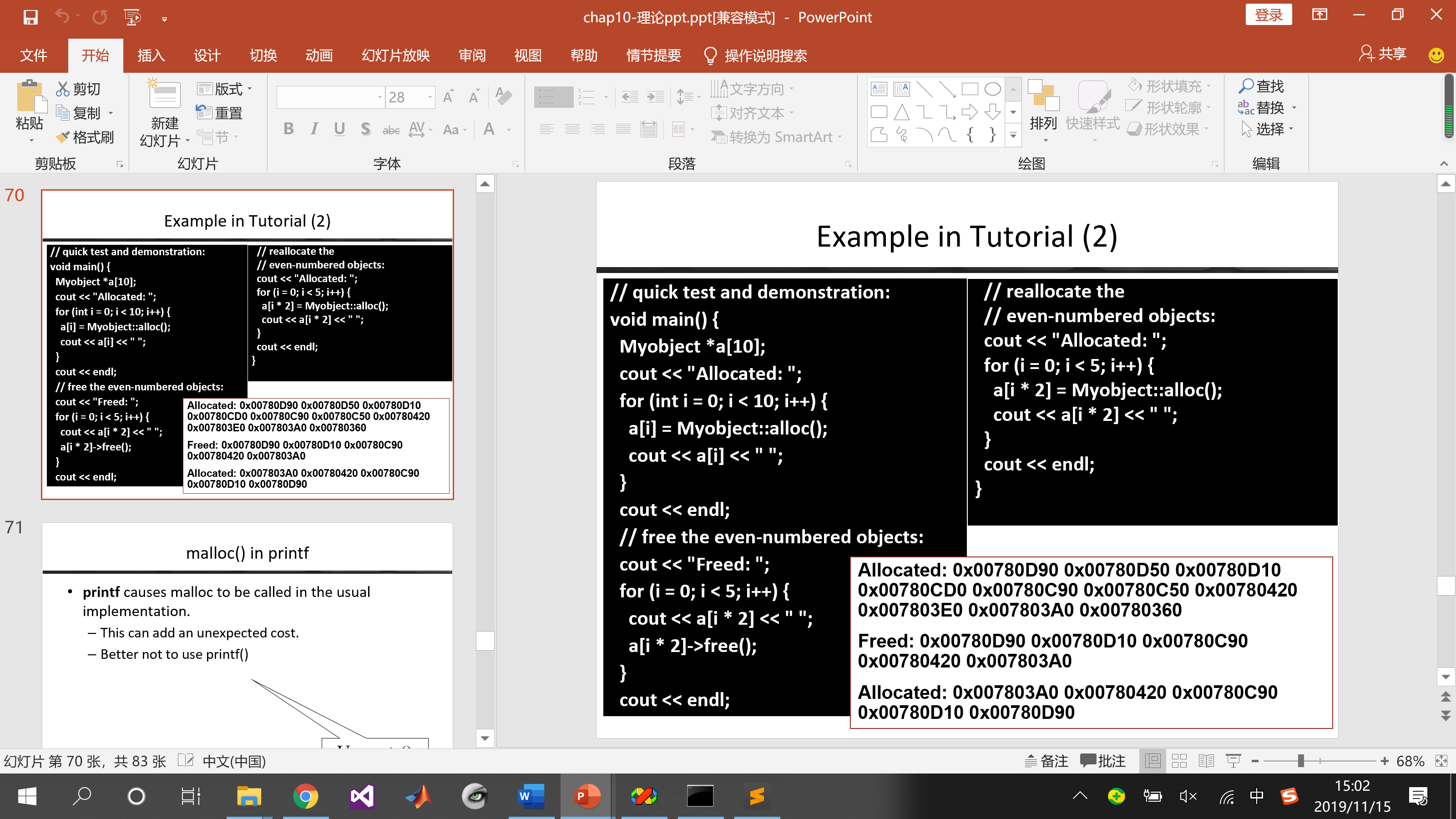
1. **Prefix v.s. Postfix (Increment / Decrement)**

i++ V.S. ++i



1. **Call by value .V.S Call by reference (Cons and Pros)**
   1. Passing by value **void f (Type x)**
   2. Passing by address **void f (Type\* x)**
   3. Passing by reference-to-const (常引用) **void f (const Type &x)**
2. **Deal with malloc (or new in C++) V.S. List when the objects are allocated often (Explain why)**





1. **A[4][4] V.S A[16]**
2. **100/2 V.S. 100\*0.5**
3. **Human readable ASCII V.S. Machine readable binary**