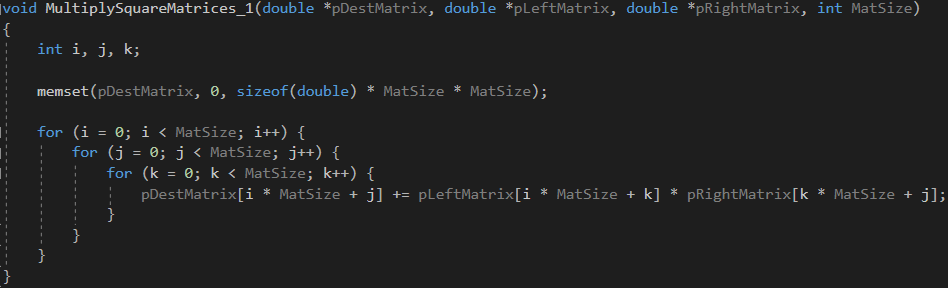
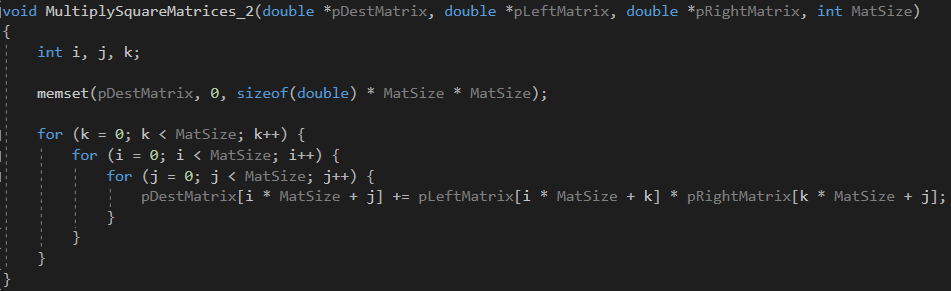
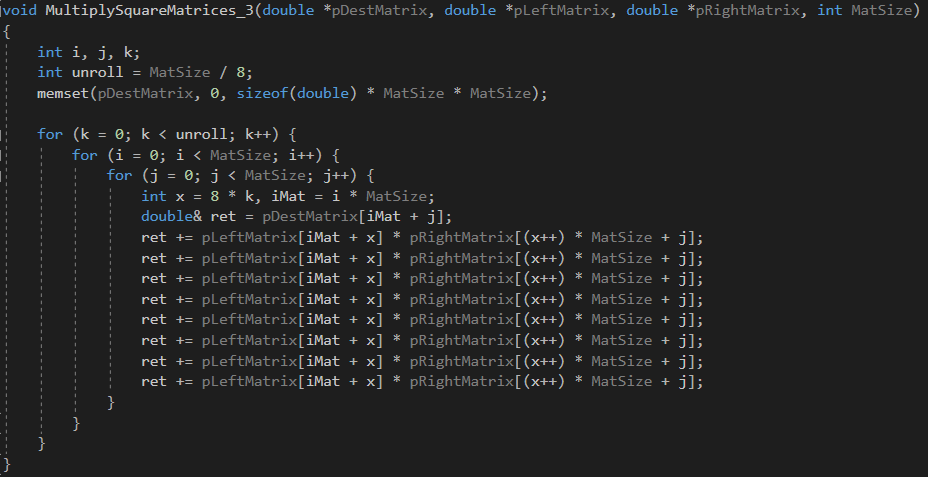
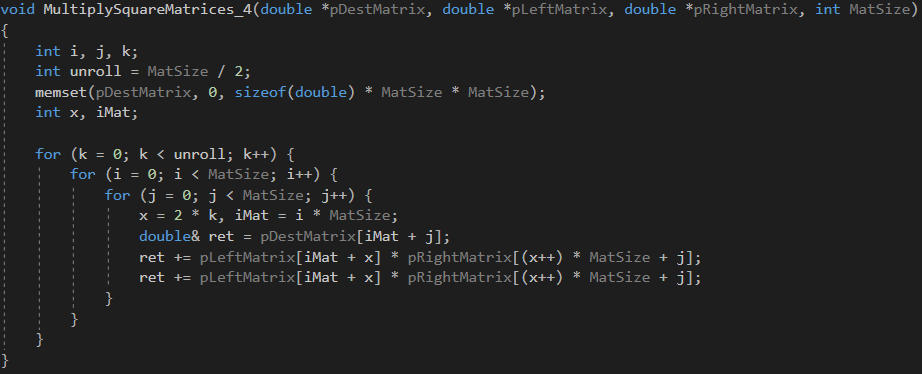
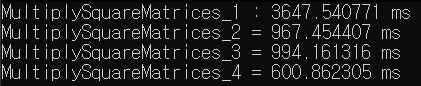
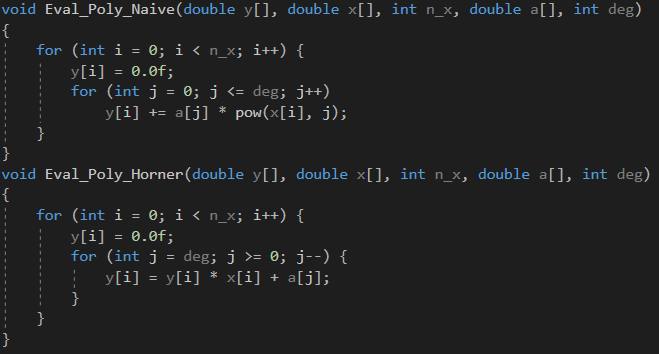
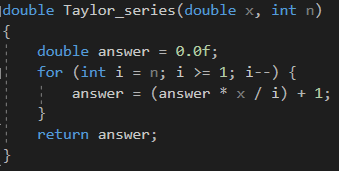
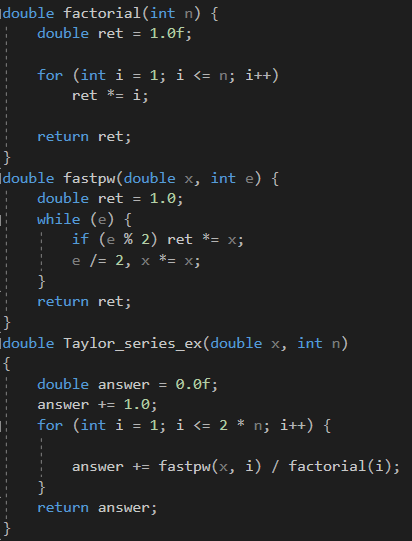
고급 소프트웨어실습 10주차 실습 보고서

20141574 임지환

1. matrix multiplication  
   1) Naïve implementation  
     
   위 함수는 두 행렬 의 곱셈의 결과인 행렬 에 대하여 를 하나씩 구하는 방법을 나타냈다. 는 다음과 같이 구할 수 있다.  
   는 모두 개념적으로는 2차원 배열이지만 1차원 index로의 접근을 하였다. 곱셈을 하는 과정에서 의 경우, 반복적인 의 증가는 메모리상에서의 인접한 위치로 접근하므로 spatial locality에 의해 접근 속도가 빠름이 보장되지만, 의 경우 반복적인 의 증가는 index가 MatSize만큼의 증가를 가져오므로, spatial locality가 적용되지 않는다.  
   2) Considering stride  
     
   값들을 확정적으로 하나씩 구한 후 다음 위치에 해당하는 값을 구하는 것보다, 계산을 하는 과정에서의 값 접근에 대한 응집성을 고려하면 속도 향상을 기대할 수 있다. 위 과정에서, 의 경우는 가장 안쪽의 for문에 영향을 받지 않고, 의 경우 가 증가함에 따라 메모리상에서의 인접한 위치로 접근하므로 spatial locality에 의해 접근 속도에서의 차이가 1)의 경우보다 크지 않다.  
   3) loop unrolling technique  
     
   loop unrolling을 활용함으로써 기대할 수 있는 부분은 연산 횟수의 감소이다. 보다 구체적으로 접근을 하면, 한번의 for문에서는 반복 조건을 위한 비교 연산, iterator의 증가 부분에 해당하는 덧셈연산 총 2번이 들어간다.  
   위 함수는 행렬의 크기가 2의 배수(=1024)이므로 8번을 unroll하여 풀어 쓴 결과이다.  
   직전의 함수의 경우, for문 자체에서 오는 연산 횟수는 대략 이고, 내부에서의 연산은 8번정도로, 총 번의 연산을 한다. 하지만 크기 8만큼의 loop unrolling을 하였을 경우 연산 횟수는 번 정도이다. 이 또한 코드의 가독성을 고려하여 어느정도 선까지만 줄인 것이고, 좀더 줄일 수 있다.  
   4) another case of loop unrolling technique  
     
   크기 2만큼의 loop unrolling을 했을 경우 가장 빠른 속도를 기대할 수 있었다.  
   4개 함수의 수행시간에 대한 결과는 다음과 같다.  
   
2. Polynomial Evaluation  
     
   이 문제의 핵심은 수학적으로 동일한 결과를 나타내는 식의 값을 어떠한 방법으로 연산하느냐에 따라 속도차이를 낼 수 있는지에 대한 것을 알아보기 위한 것이다.  
   두 코드를 비교하였을 때, y[i]를 구하는 과정에서 pow함수의 호출 유무가 차이가 나게 된다. pow(double \_Xx, int \_Yy) 함수의 경우, 그 결과값이 가 되며, 이를 구하는데 걸리는 시간복잡도는 임이 잘 알려져 있다. 따라서 Eval\_Poly\_Naive함수의 경우 최종 수행에 걸리는 시간복잡도는 이다.  
   반면, Eval\_Poly\_Horner의 경우 이를 풀어서 연산하였기 때문에 가 된다.  
     
   두 함수의 호출 결과는 다음과 같다.  
   
3. Numerical Approximation  
     
   주어진 문제에서의 테일러 급수 전개를 하다 보면 (answer \* x / i)의 값과 1의 절대값이 비슷하고 부호가 반대인 경우가 발생한다. 이 경우, 비슷한 수의 뺄셈이 발생하여 오차가 누적되게 된다.  
     
   horner’s method를 사용하지 않고 naive하게 테일러 급수를 전개한 값을 그대로 구하되, 항의 개수를 2배로 늘리니 그 오차를 줄일 수 있었다. 항의 개수가 늘어남으로써 이 문제가 어떤 이유로 해결이 되었는지는 알 수 없었다. 이는 비단 위에서 표현한 의 합들로의 결과를 구한 방법이 아닌, horner’s method로의 전개를 통해 구한 경우에도 전개횟수를 늘려서 오차를 피할 수 있었다.