

Họ và tên: NGUYỄN TRÍ TRUNG

MSSV: 1712847

Khoa: Công nghệ thông tin

Lớp: 17CTT7

Mail: [tritrung232@gmail.com](mailto:tritrung232@gmail.com)

Tên học phần: Kĩ thuật lập trình

Nội dung : đồ án 2- Học kì 2 -Năm học: 2017-2018

**1.Tiêu chí đồ án và mức độ hoàn thành:** gồm 3 tiêu chí

* Thống kê số phép gán hoàn thành
* Thống kê số phép so sánh hoàn thành
* Thời gian sắp xếp hoàn thành
* **2.Báo cáo đồ án:**
* **2.1 Mô tả:**

**Để đánh giá một thuật toán ta cần dựa vào thời gian,đối với mỗi máy ( hay mỗi lần chạy) sẽ cho ra thời gian khác nhau .Những vấn đề đó một phần là do trật tự các số của mảng và số phần tử.Nếu trong trường hợp tốt nhất về trật tự sắp xếp nhanh hơn,và ngược lại.Không những thế ,với mỗi loại máy sẽ có tốc độ xử lý riêng.Từ đó ta có thể lập biểu đồ đánh giá với N(100->1000000).Với cùng N ta sẽ so sánh từng Sort.**

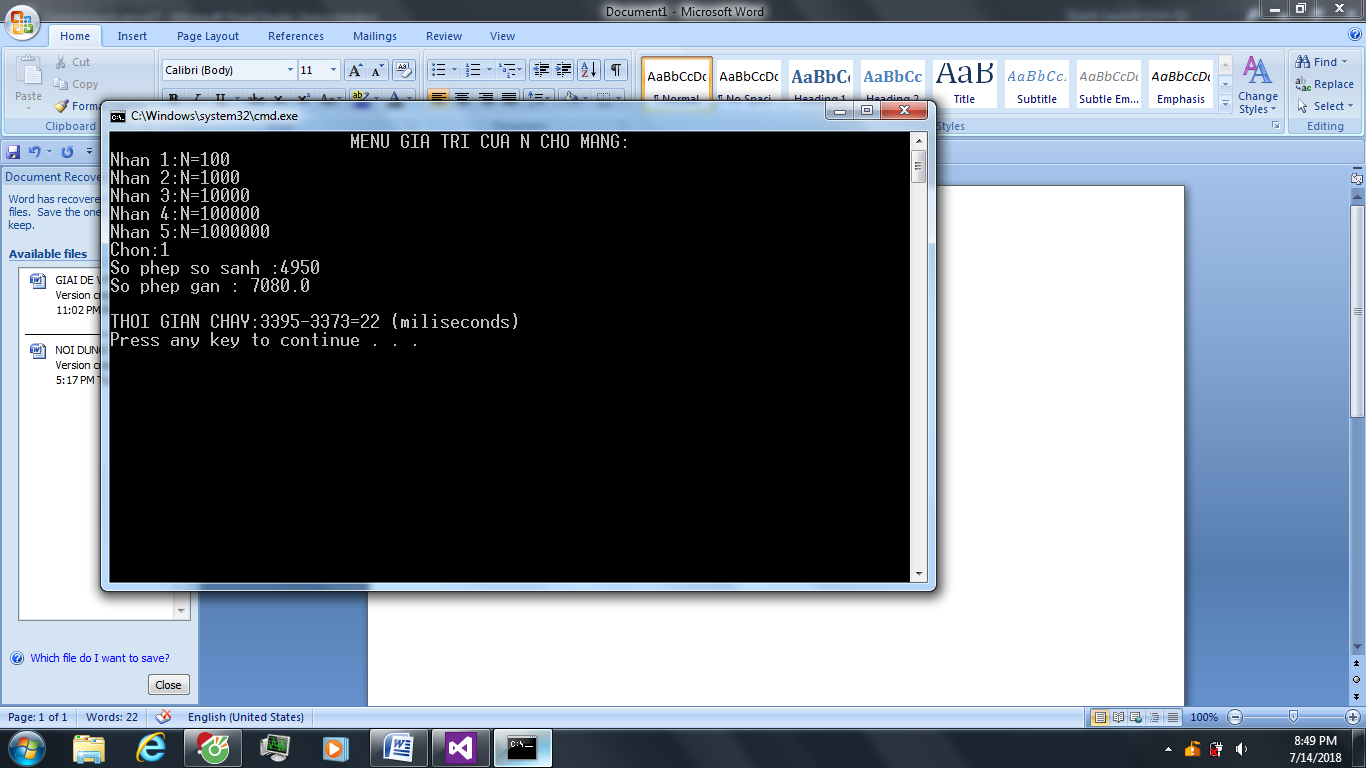
**2.2.Chi tiết**

BÁO CÁO ĐỒ ÁN 2

**Kết qủa các lần chạy**

**Với N=100**

BUBBLE SORT

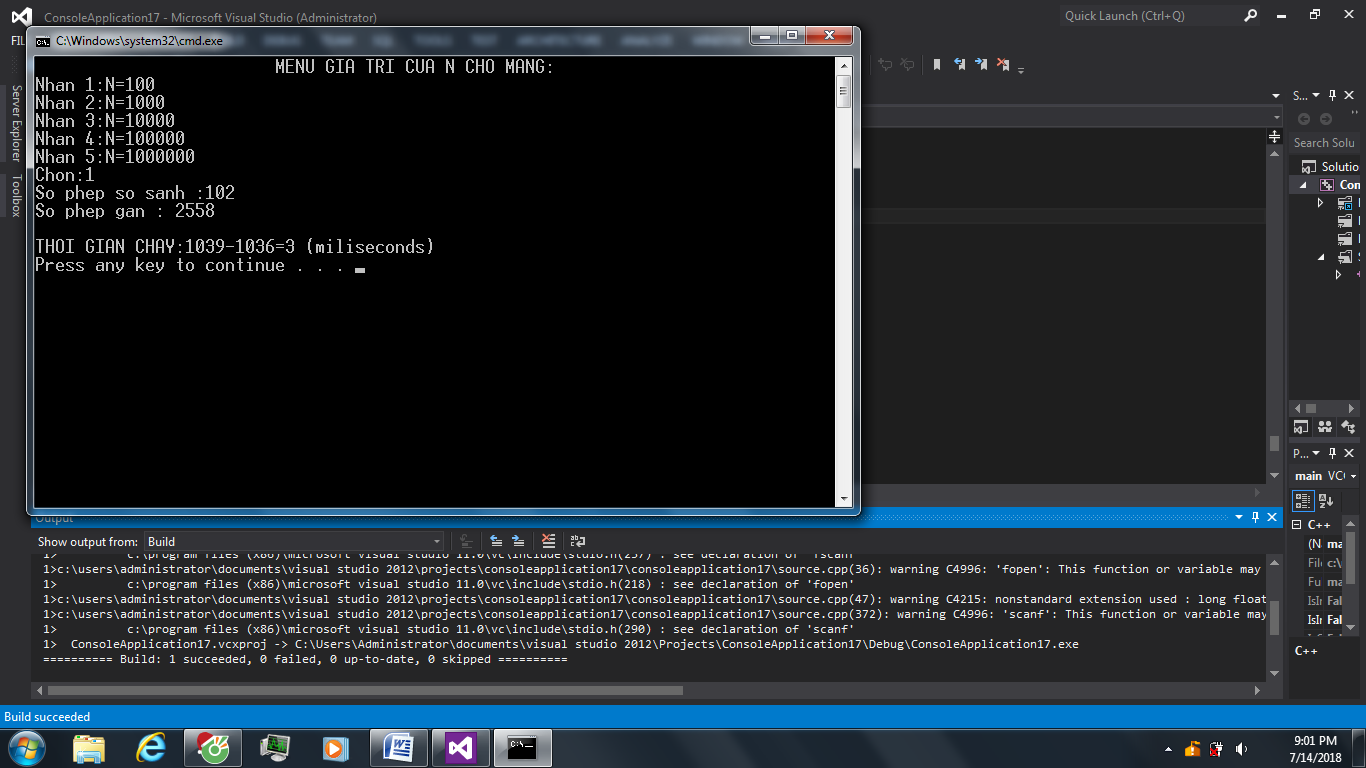


Số phép so sánh:4950

Số phép gán:7080

Thời gian chạy:22ms

INSERTION SORT

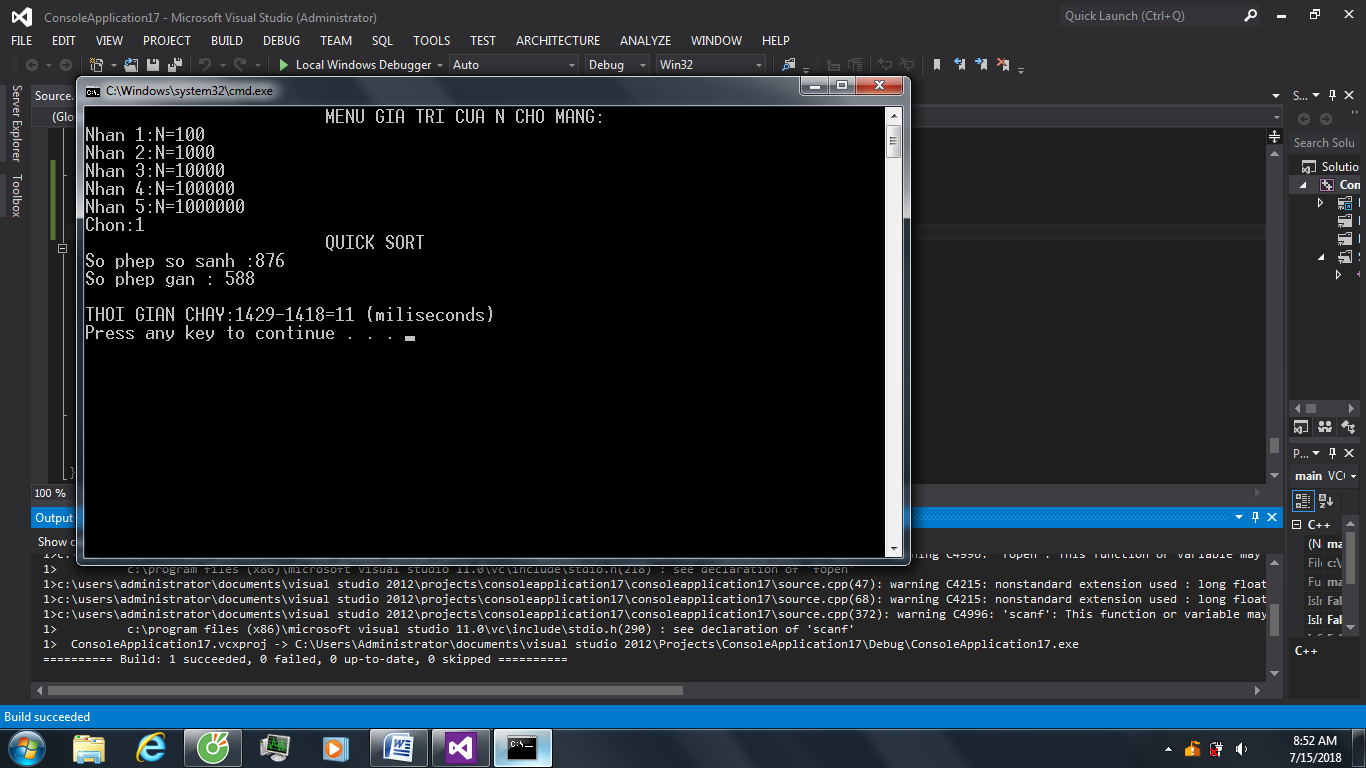


Số phép so sánh:102

Số phép gán:2558

Thời gian chạy:3ms

QUICK SORT

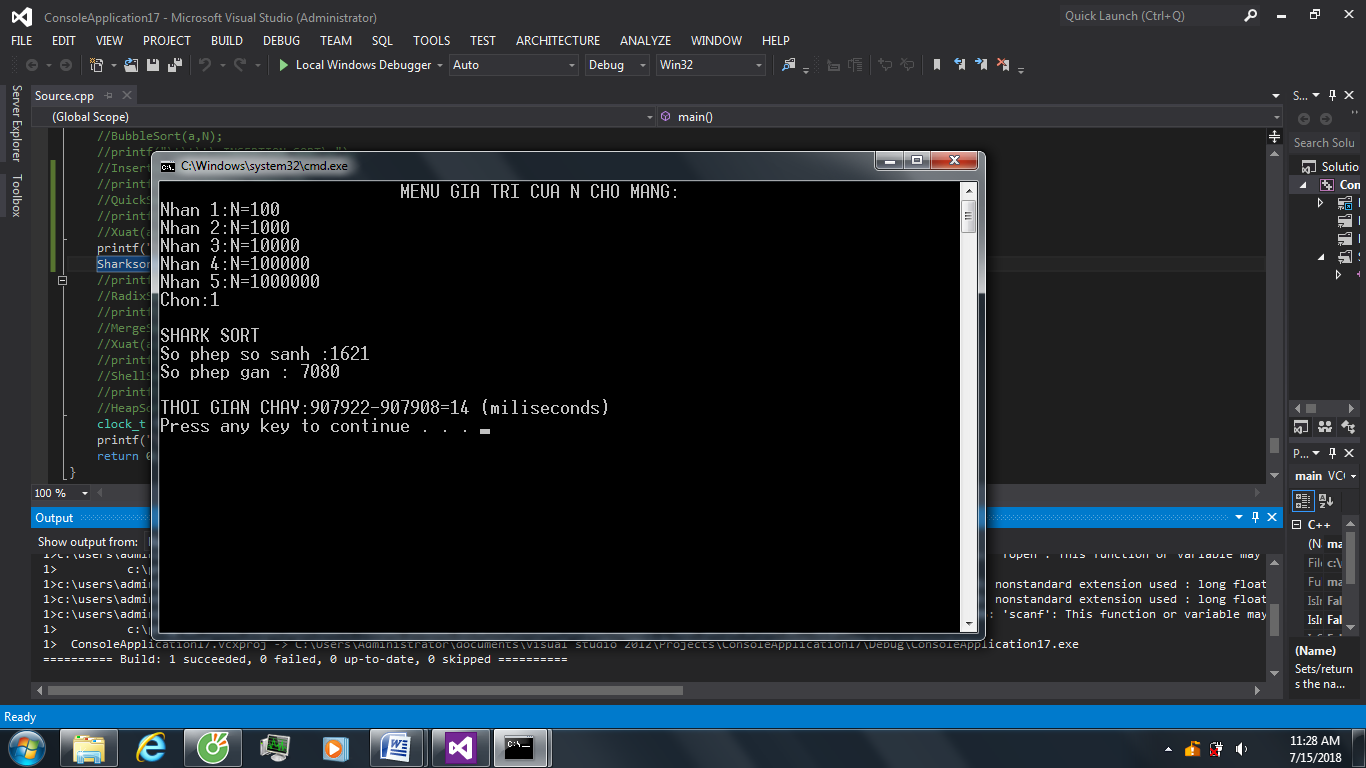


Số phép so sánh:876

Số phép gán:588

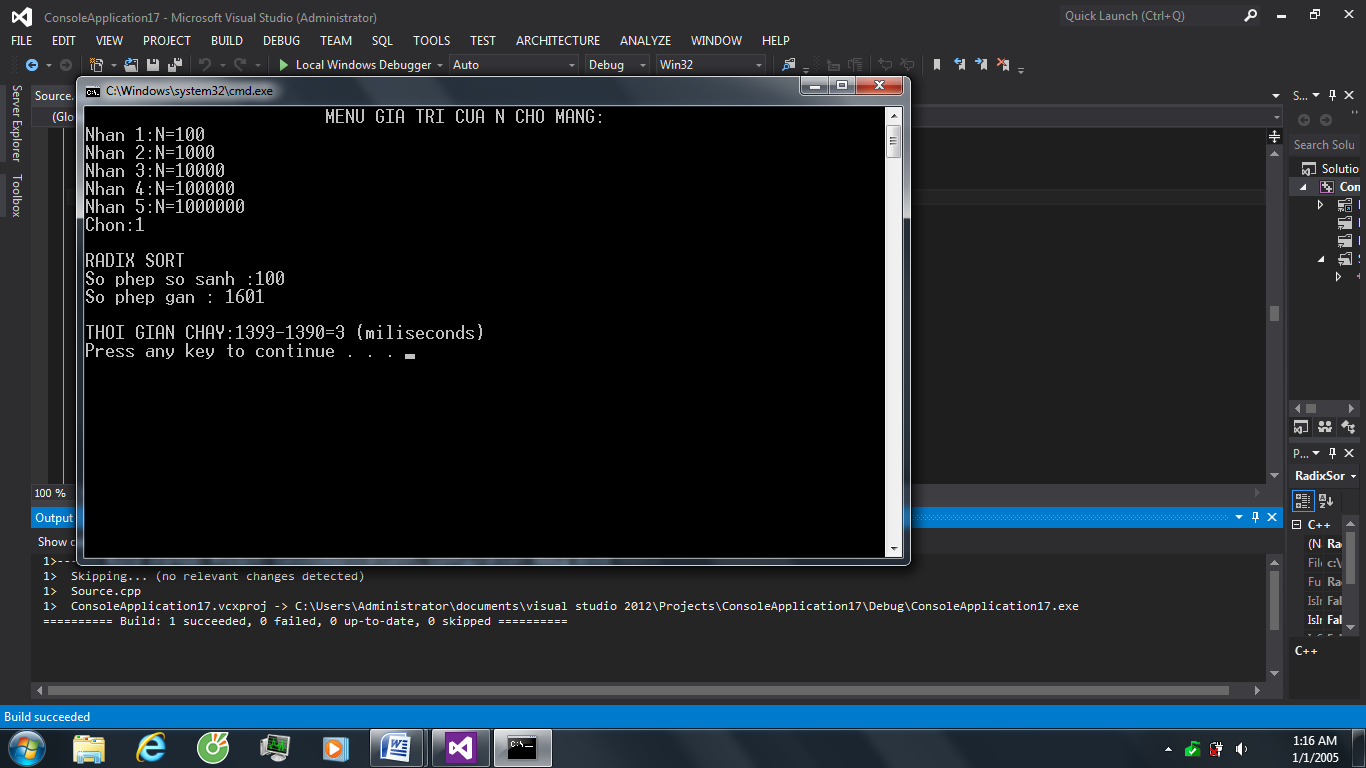
Thời gian chạy:11ms

**SHARK SORT**

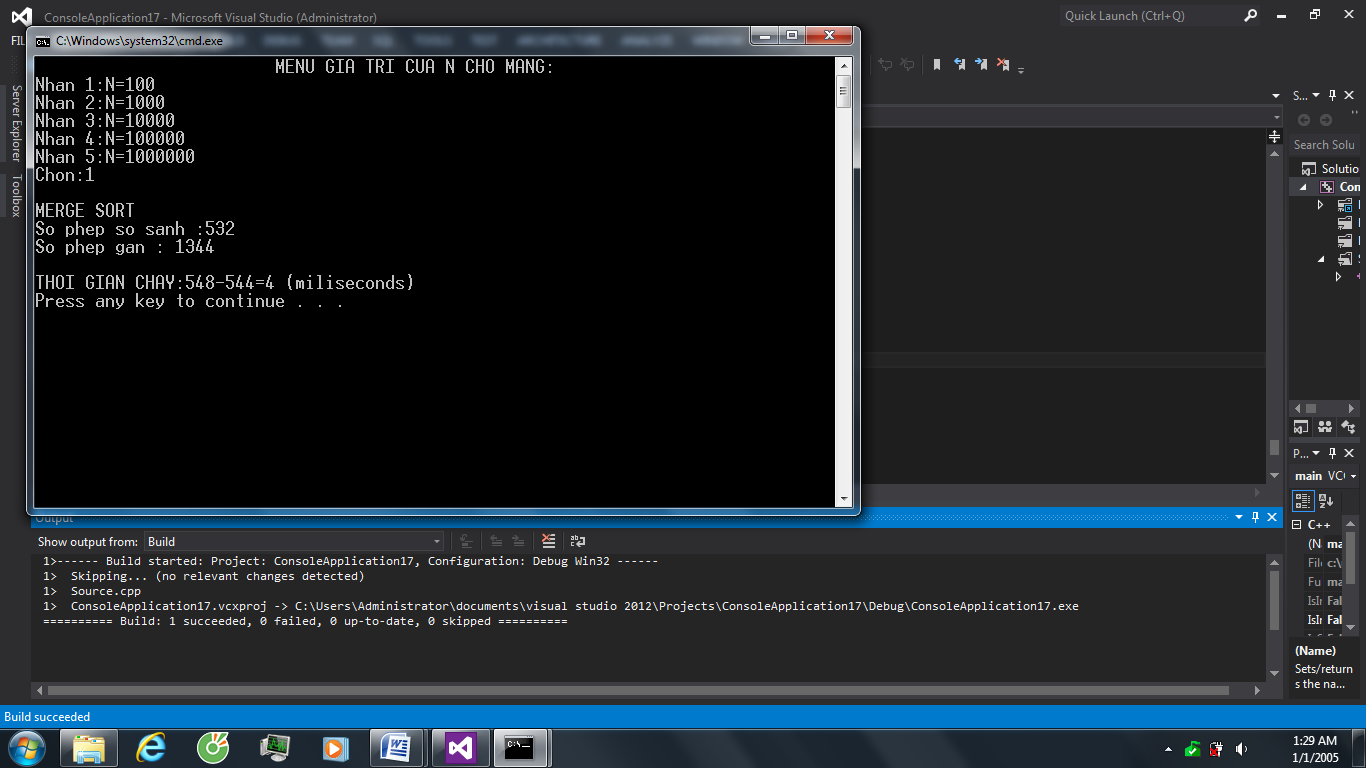


162170804

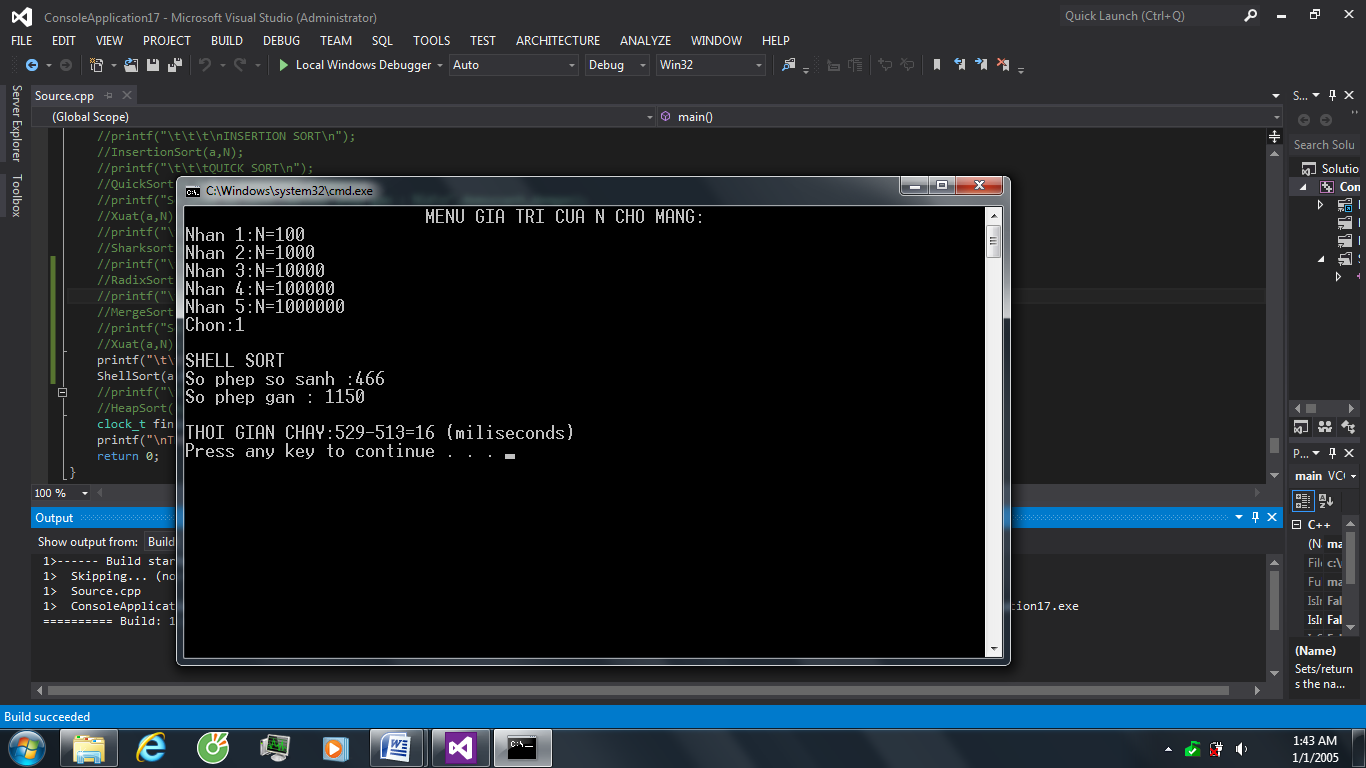
**RADIX SORT**

****

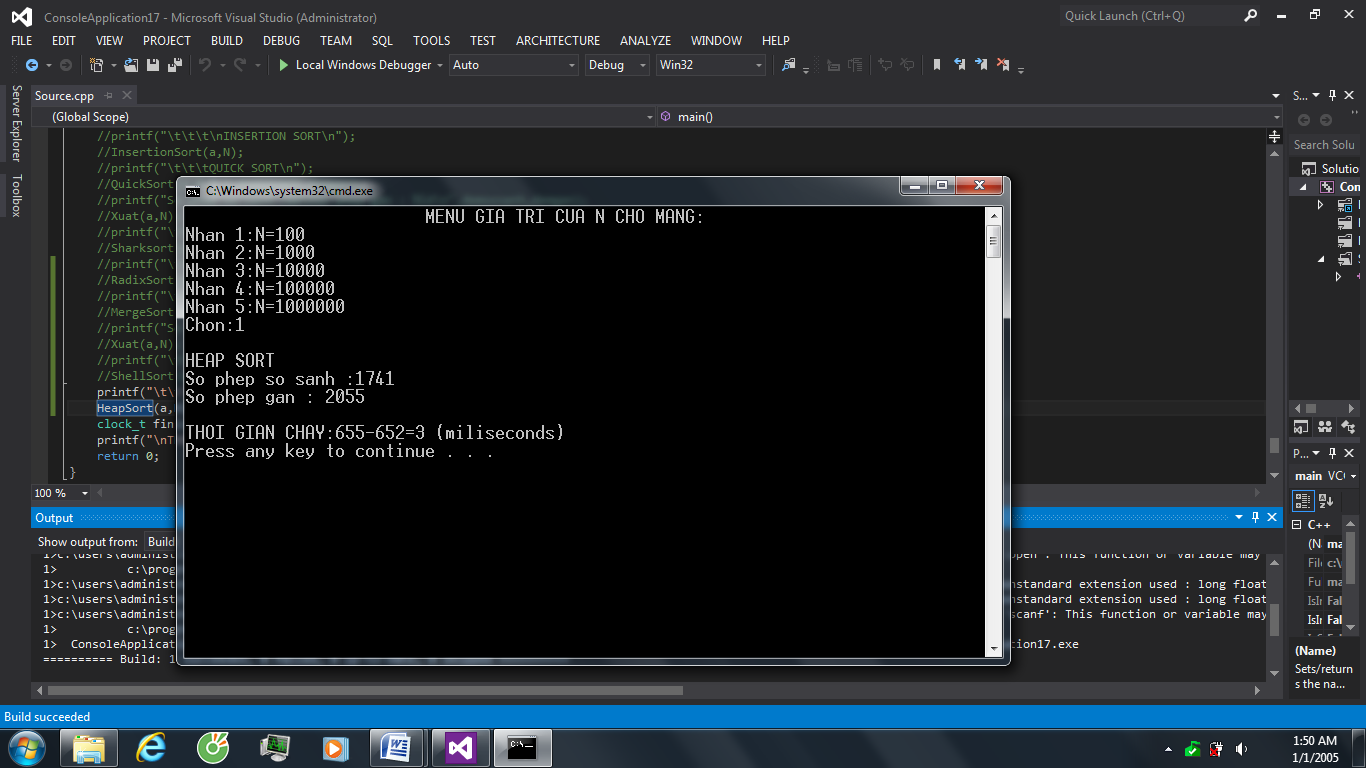
**MERGE SORT**

****

**SHELL SORT**

****

**HEAP SORT**

****

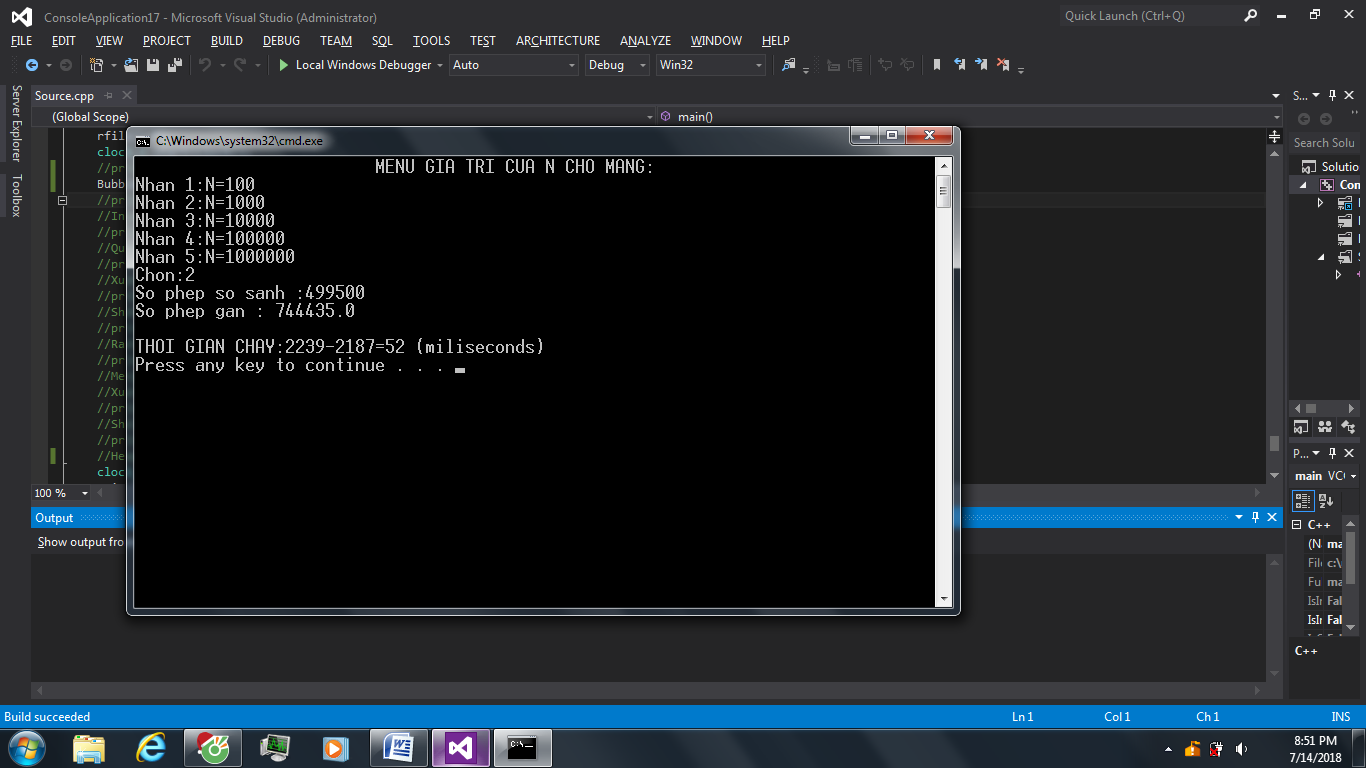
**Nhận xét:với việc sắp xếp N nhỏ thì thời gian sẽ không chênh lệch quá nhiều,không làm rõ được thuật toán nào có tốc độ sắp xếp nhanh**

**BẢNG SỐ LIỆU THÔNG KÊ MẢNG 100 PHẦN TỬ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Thuật toán | Số phép gán | Số phép so sánh | Thời gian(ms) |
| 1 | InsertionSort | 2558 | 102 | 3 |
| 2 | BubbleSort | 7080 | 4950 | 22 |
| 3 | RadixSort | 1601 | 100 | 3 |
| 4 | ShellSort | 1150 | 466 | 16 |
| 5 | MergeSort | 1344 | 532 | 4 |
| 6 | QuickSort | 588 | 876 | 11 |
| 7 | HeapSort | 2055 | 1741 | 3 |
| 8 | Sharksort | 7080 | 1621 | 4 |

**Với N=1000**

BUBBLE SORT

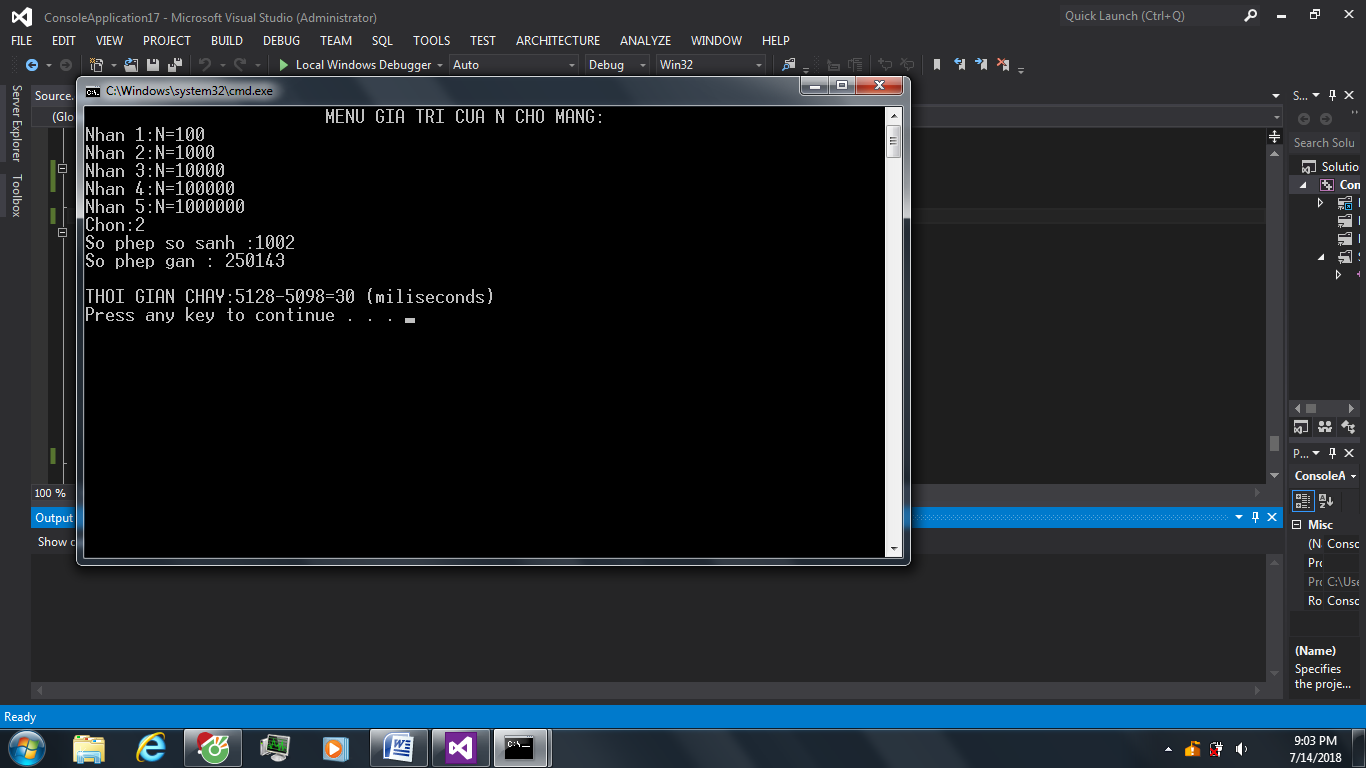


Số phép so sánh :499500

Số phép gán:744435

Thời gian chạy:52ms

INSERTION SORT

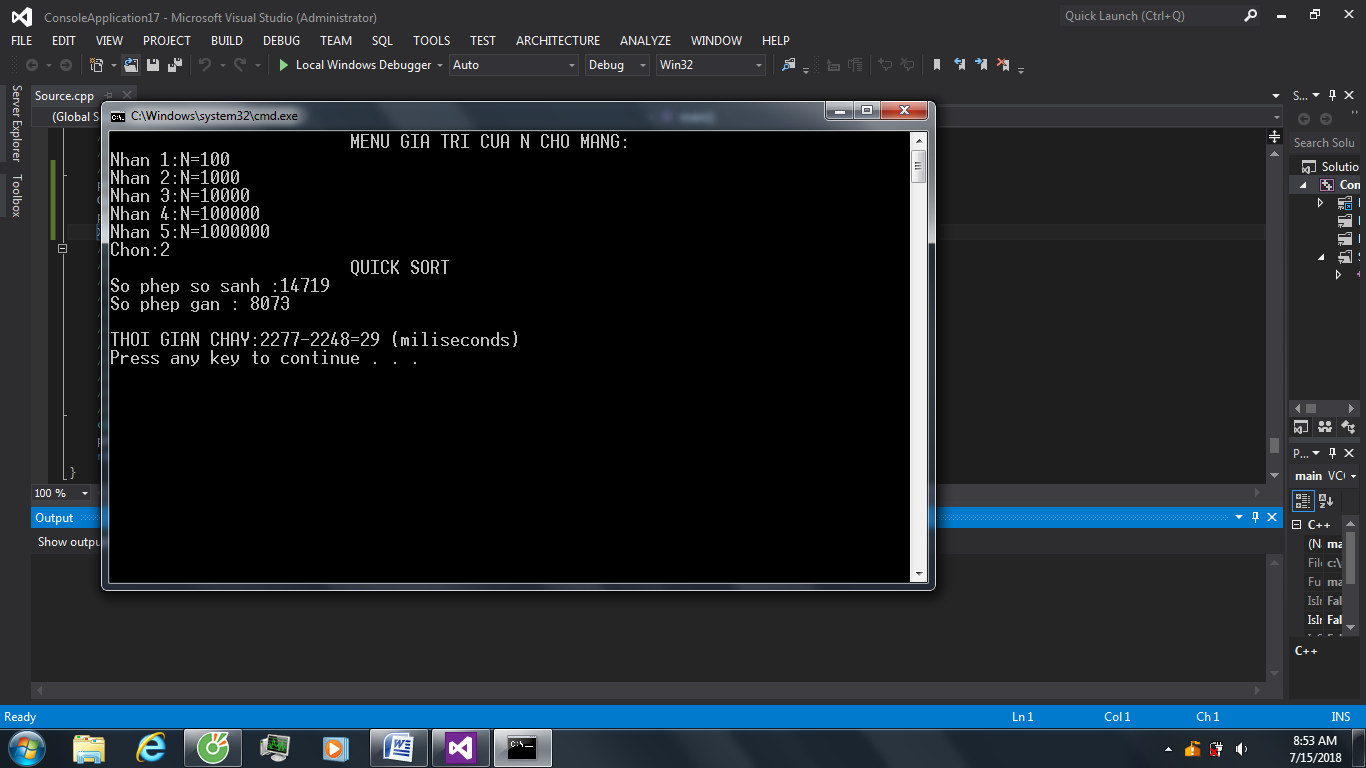


Số phép so sánh :1002

Số phép gán:250143

Thời gian chạy:30ms

QUICK SORT

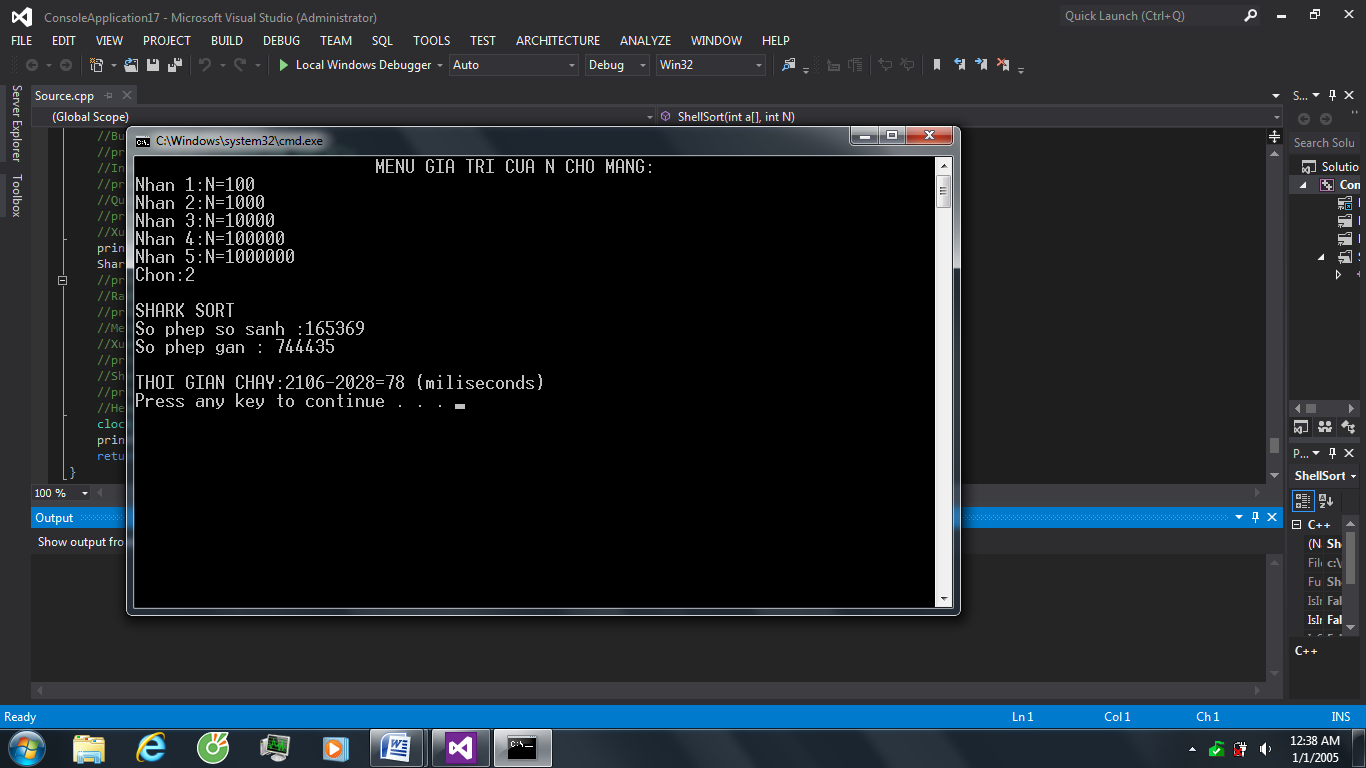


Số phép so sánh :14719

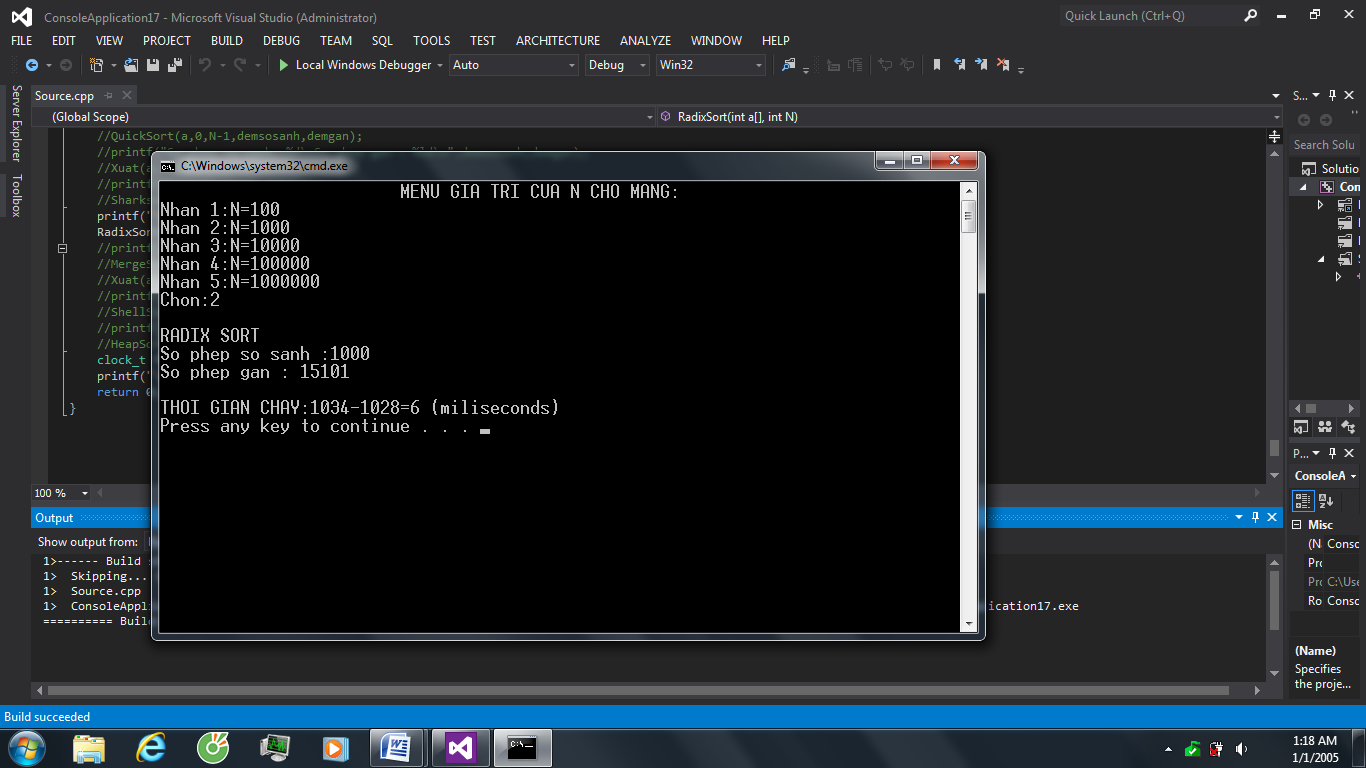
Số phép gán:8073

Thời gian chạy:29ms

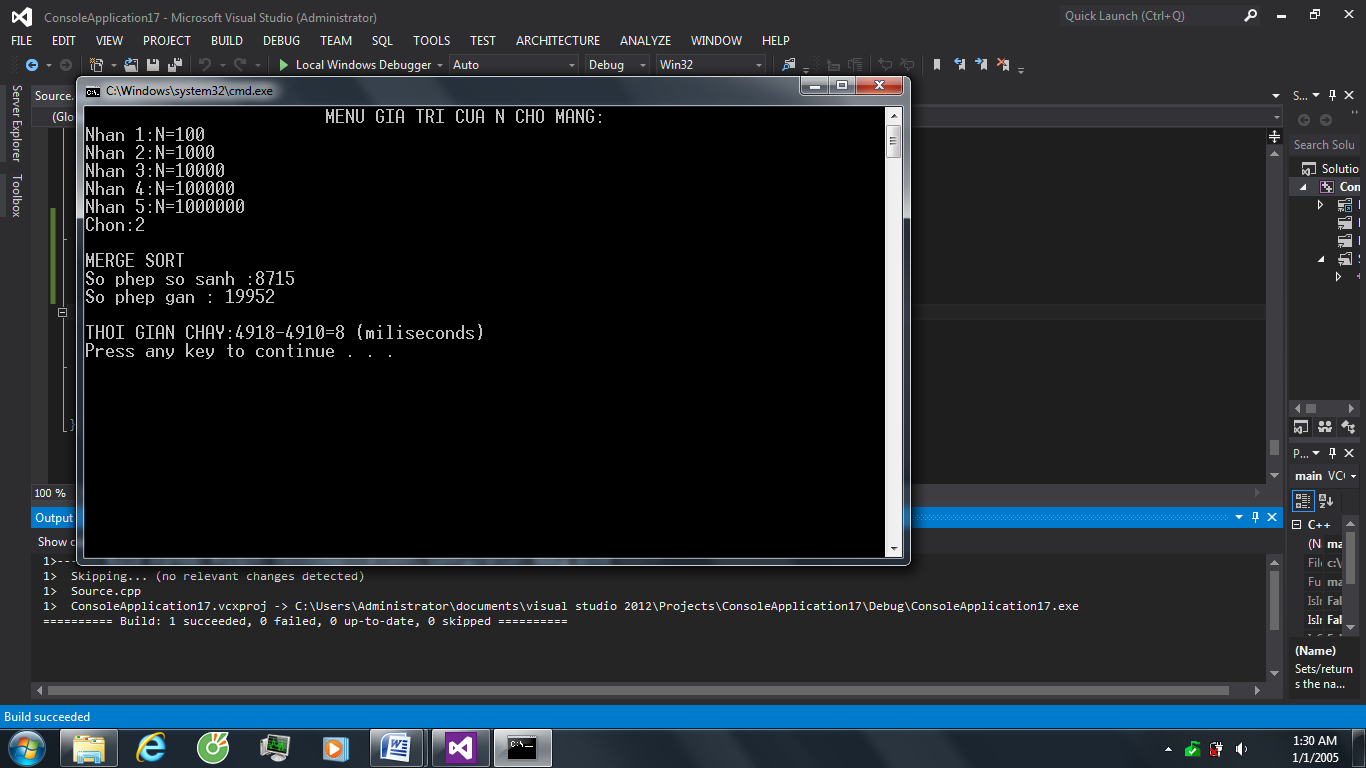
SHARK SORT



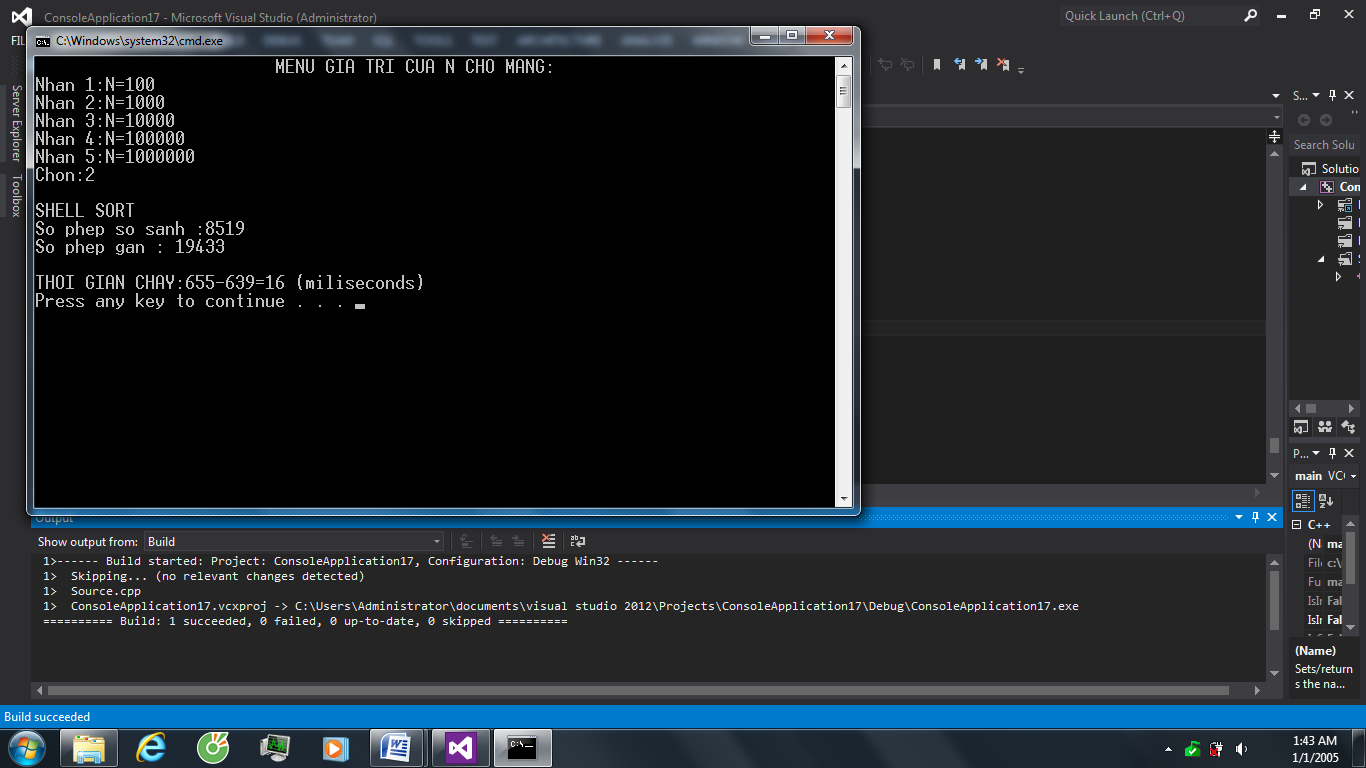
**RADIX SORT**

****

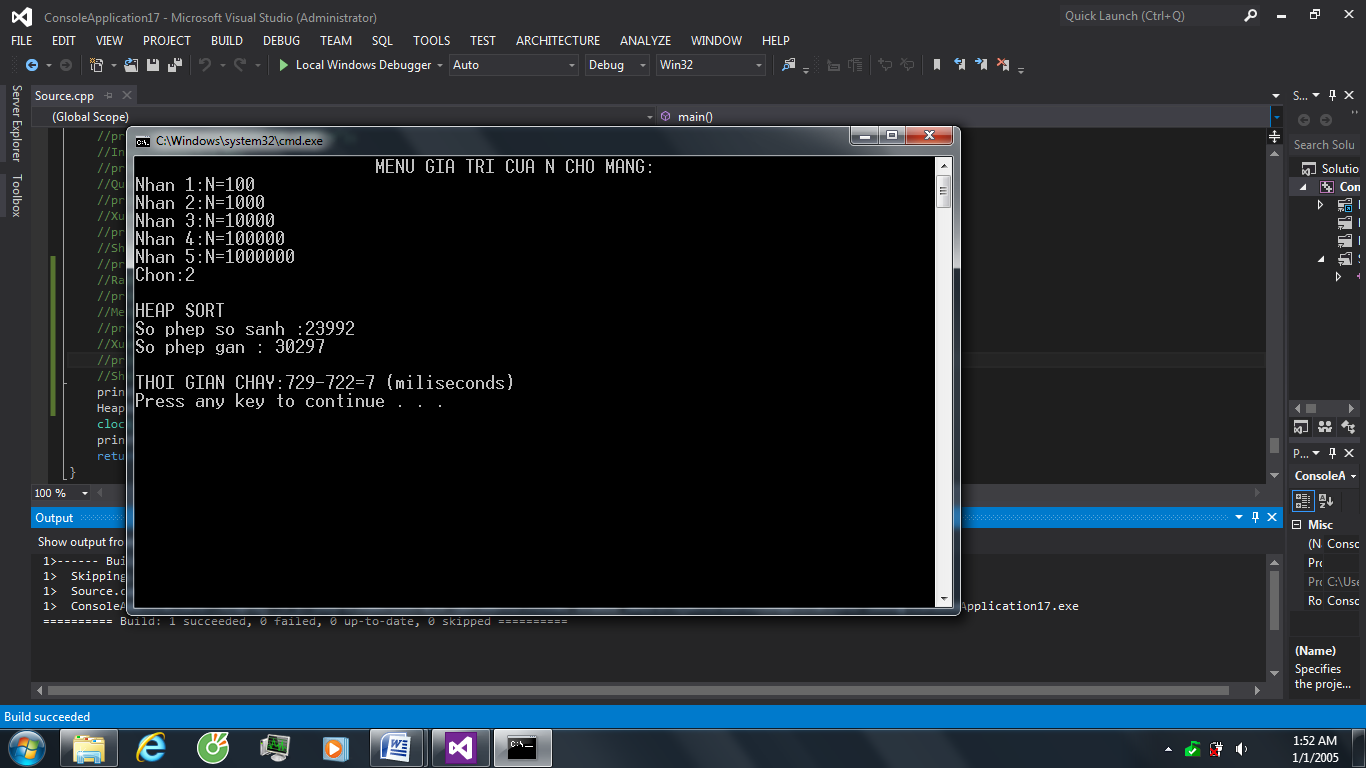
**MERGE SORT**

****

**SHELL SORT**

****

**HEAP SORT**

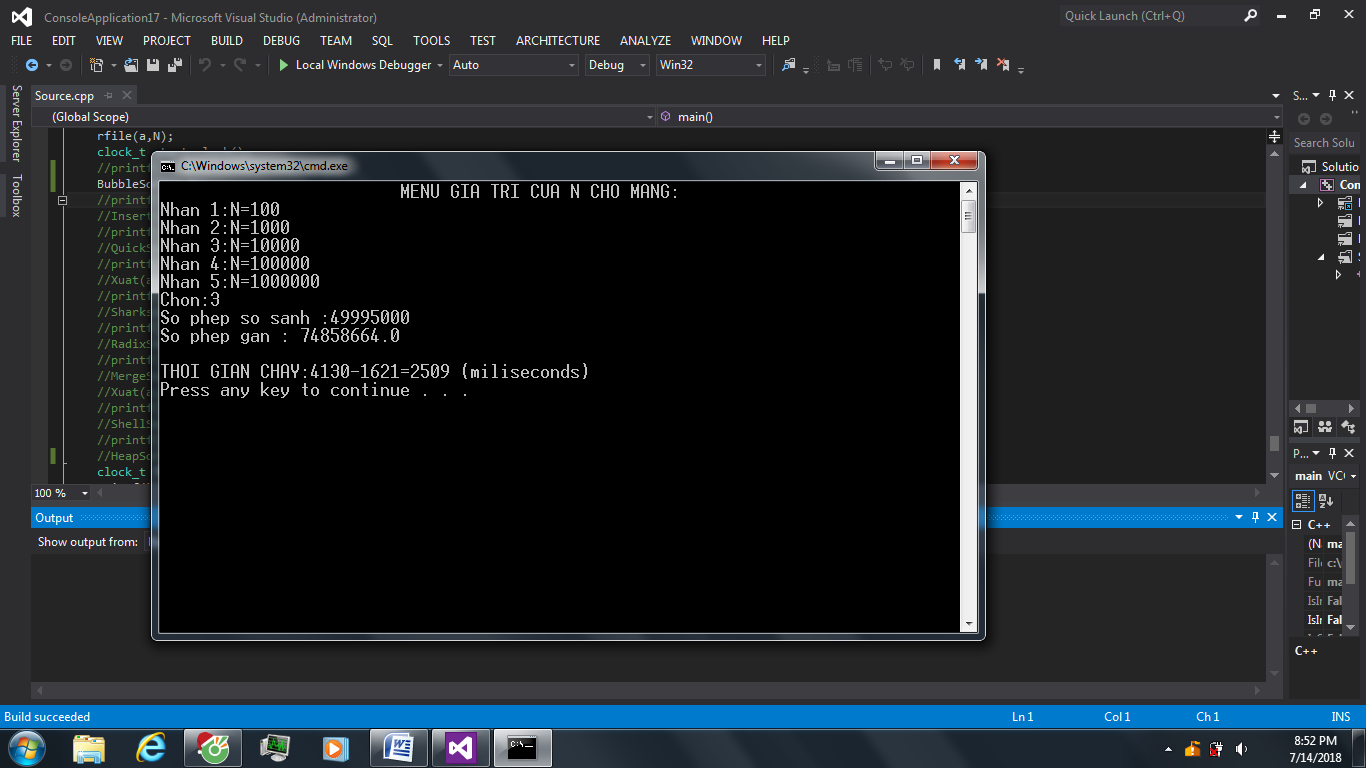
****

**BẢNG SỐ LIỆU THÔNG KÊ MẢNG 1000 PHẦN TỬ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Thuật toán | Số phép gán | Số phép so sánh | Thời gian(ms) |
| 1 | InsertionSort | 250143 | 1002 | 30 |
| 2 | BubbleSort | 744435 | 499500 | 52 |
| 3 | RadixSort | 15101 | 1000 | 6 |
| 4 | ShellSort | 19433 | 8519 | 16 |
| 5 | MergeSort | 19952 | 8715 | 8 |
| 6 | QuickSort | 8073 | 14719 | 29 |
| 7 | HeapSort | 30297 | 23992 | 7 |
| 8 | Sharksort | 74435 | 165369 | 78 |

**Với N=10000**

BUBBLE SORT

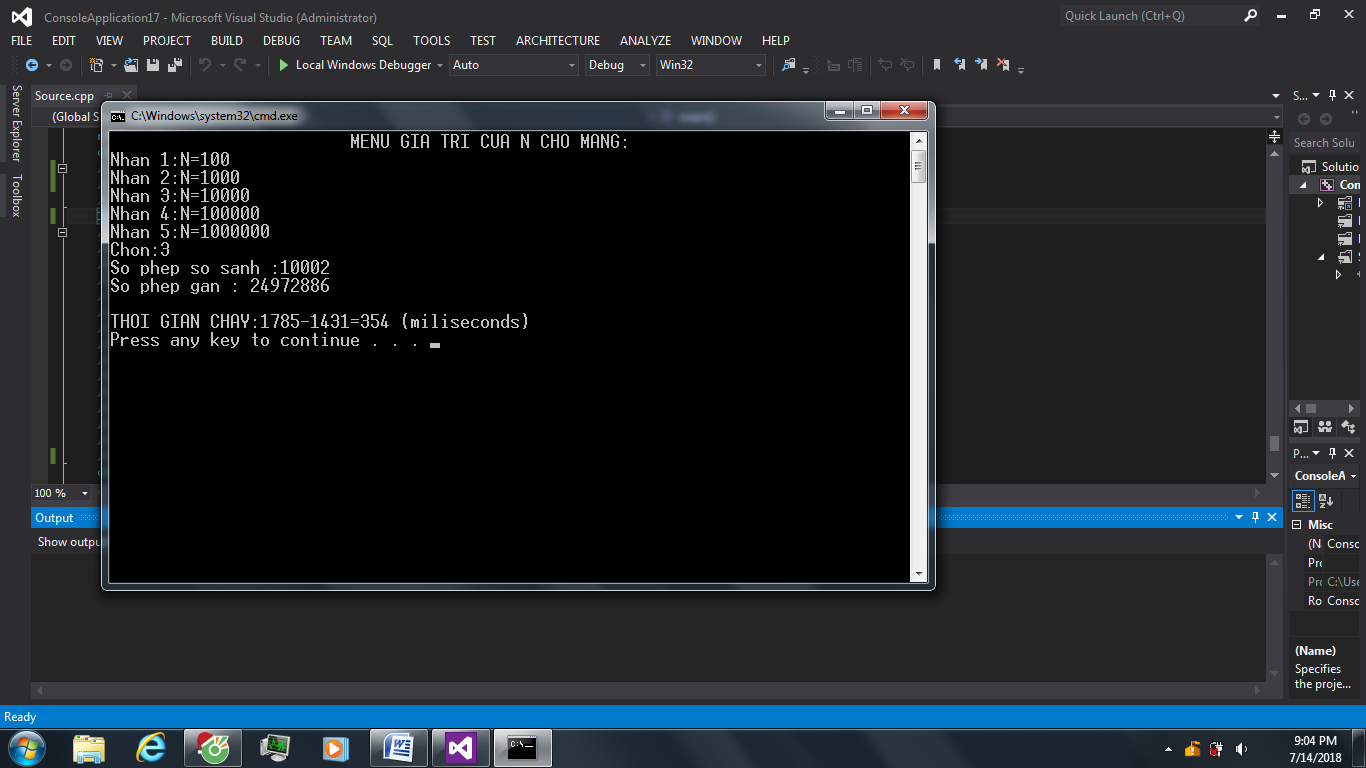


Số phép so sánh :49995000

Số phép gán:74858664

Thời gian chạy:2509ms

INSERTION SORT

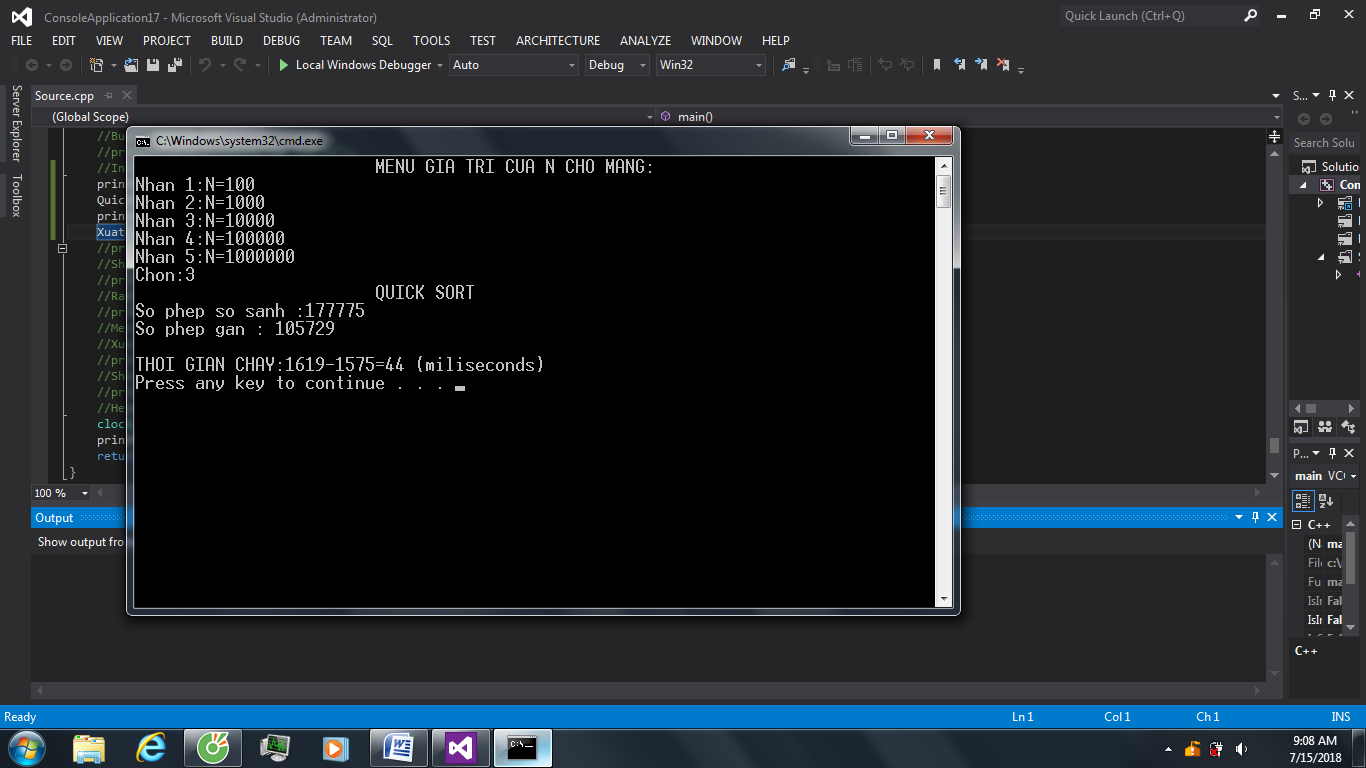


Số phép so sánh :10002

Số phép gán:24972886

Thời gian chạy:354ms

QUICK SORT

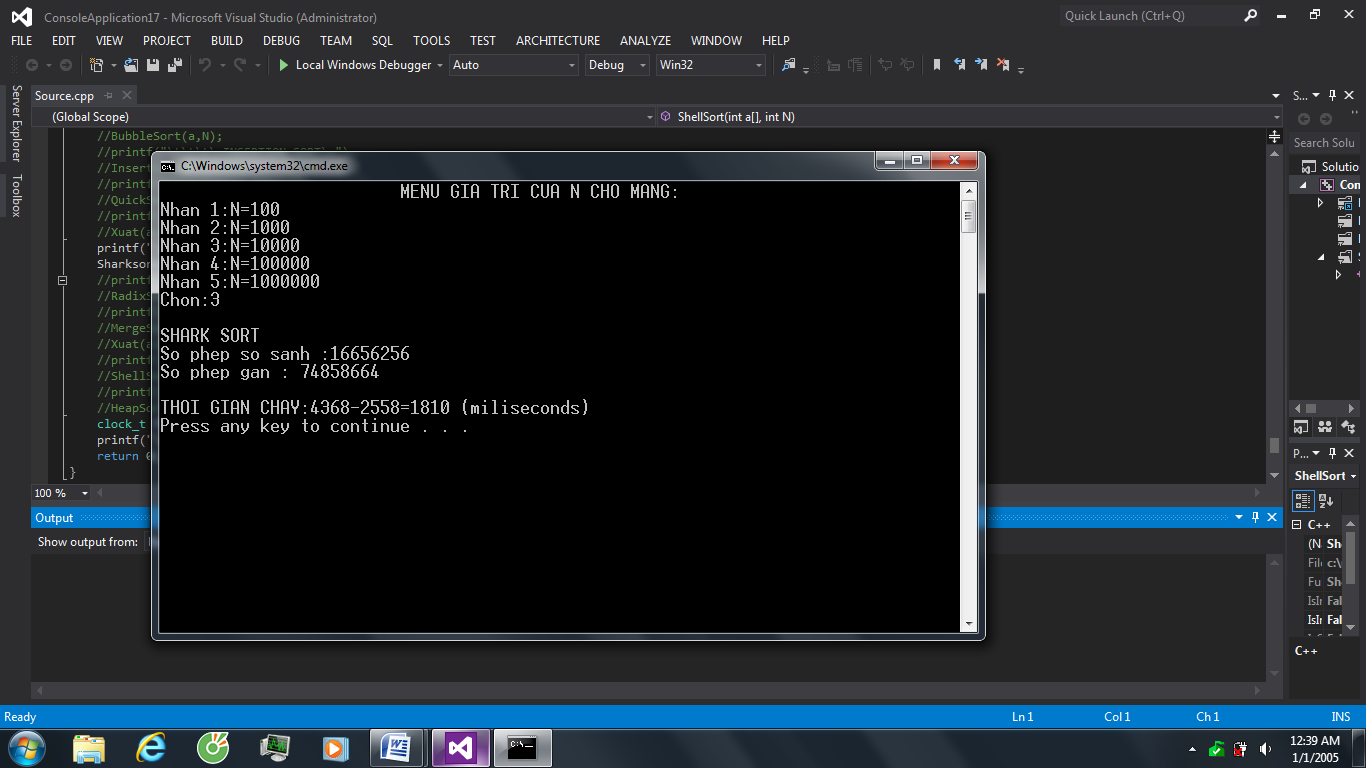


Số phép so sánh :177775

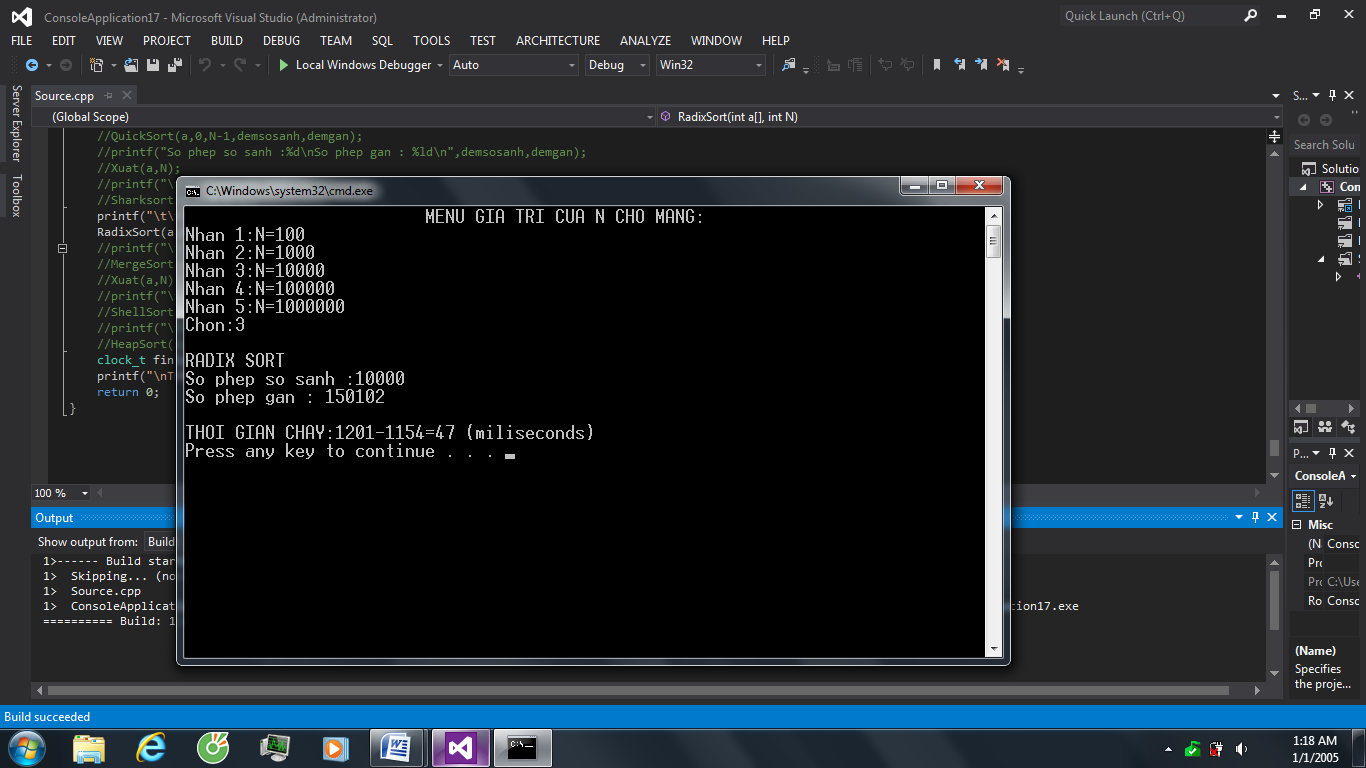
Số phép gán:105729

Thời gian chạy:44ms

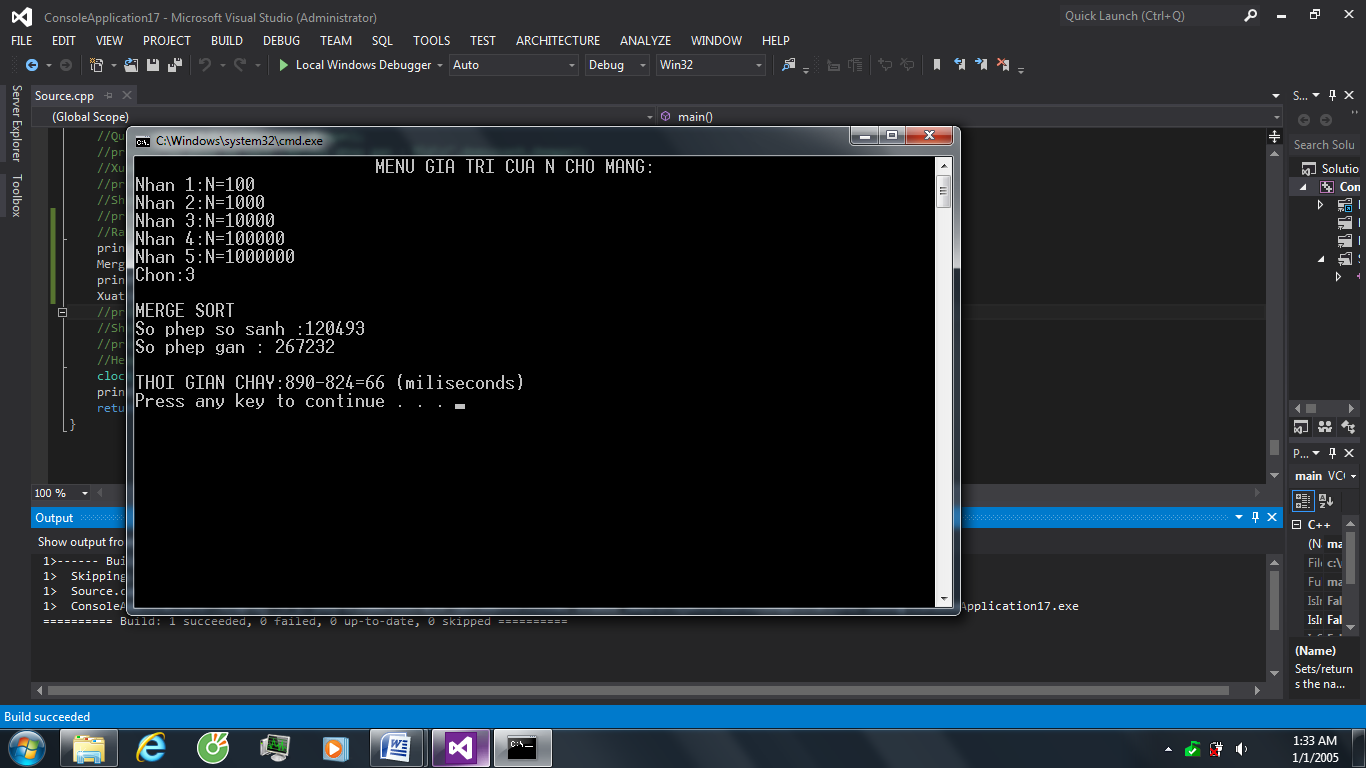
SHARK SORT



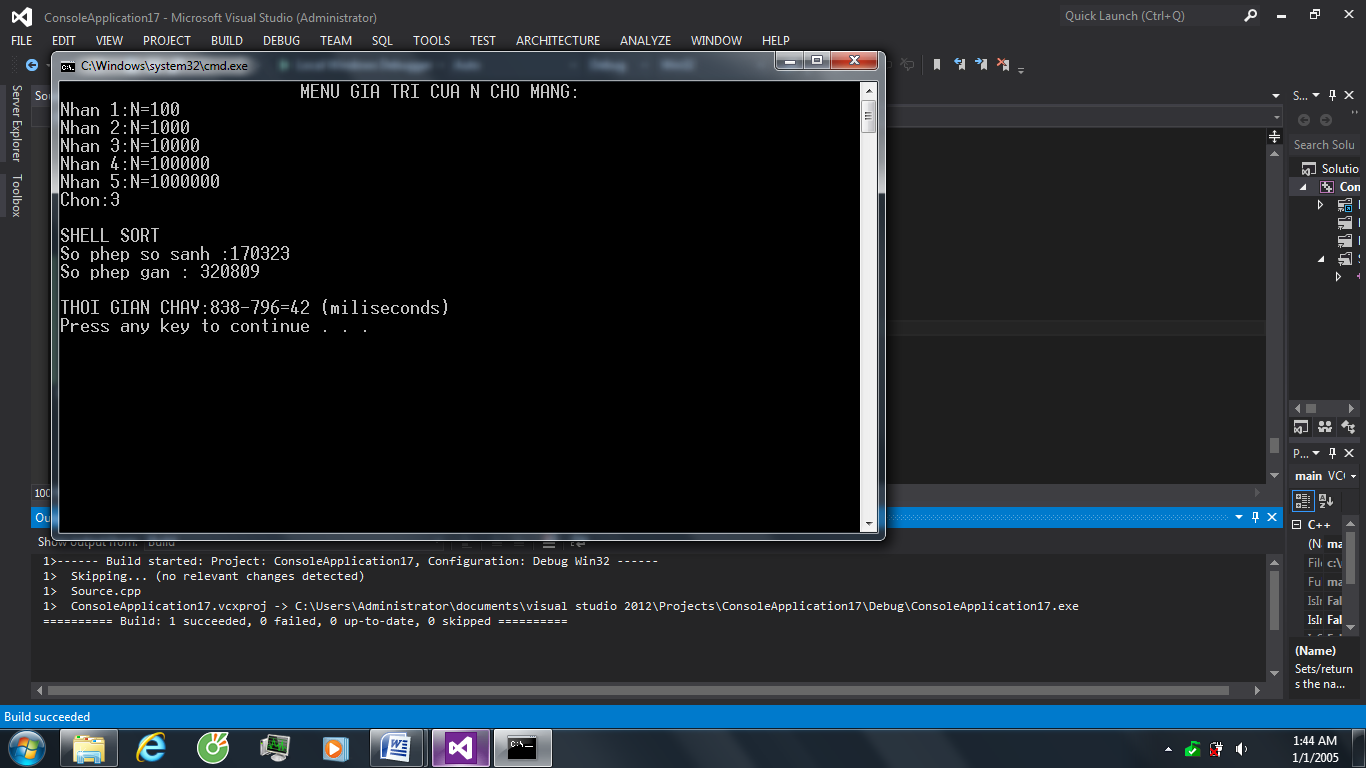
**RADIX SORT**

****

**MERGE SORT**

****

**SHELL SORT**

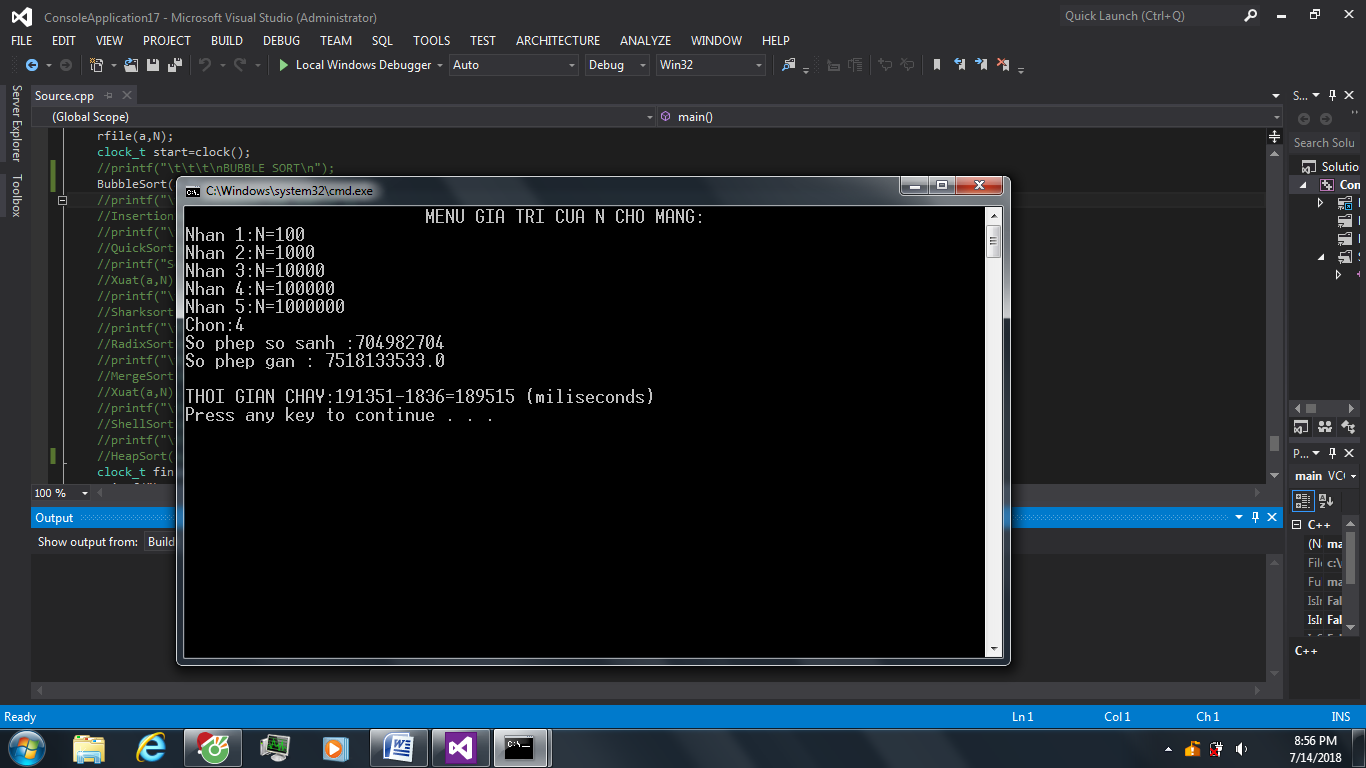
****

**BẢNG SỐ LIỆU THÔNG KÊ MẢNG 10000 PHẦN TỬ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Thuật toán | Số phép gán | Số phép so sánh | Thời gian(ms) |
| 1 | InsertionSort | 10002 | 24972886 | 354 |
| 2 | BubbleSort | 49995000 | 74858664 | 2509 |
| 3 | RadixSort | 10000 | 150102 | 47 |
| 4 | ShellSort | 320809 | 170323 | 42 |
| 5 | MergeSort | 267232 | 120493 | 66 |
| 6 | QuickSort | 177775 | 105729 | 44 |
| 7 | HeapSort | LỖI | LỖI | LỖI |
| 8 | Sharksort | 16656256 | 74858664 | 1810 |

**Với N=100000**

BUBBLE SORT

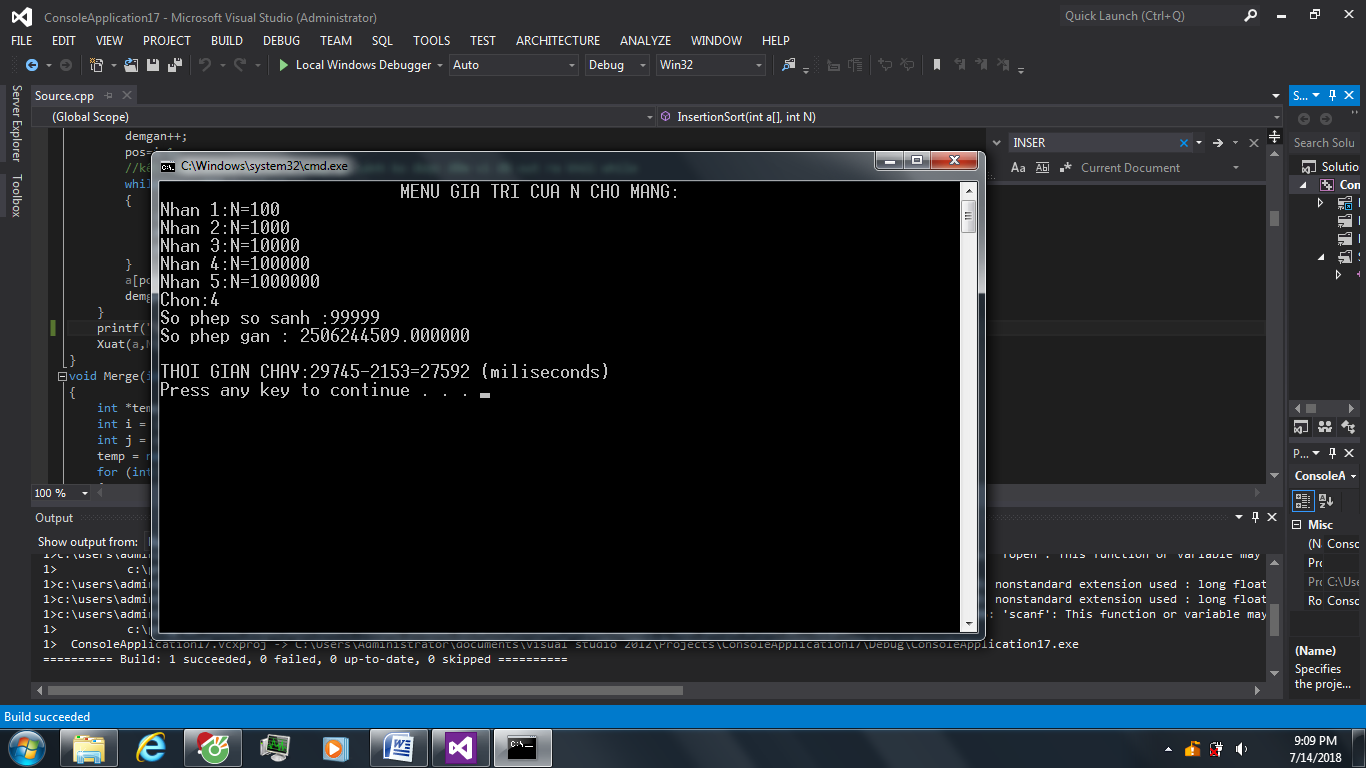


Số phép so sánh :704982704

Số phép gán:7518133533

Thời gian chạy:189515ms

INSERTION SORT

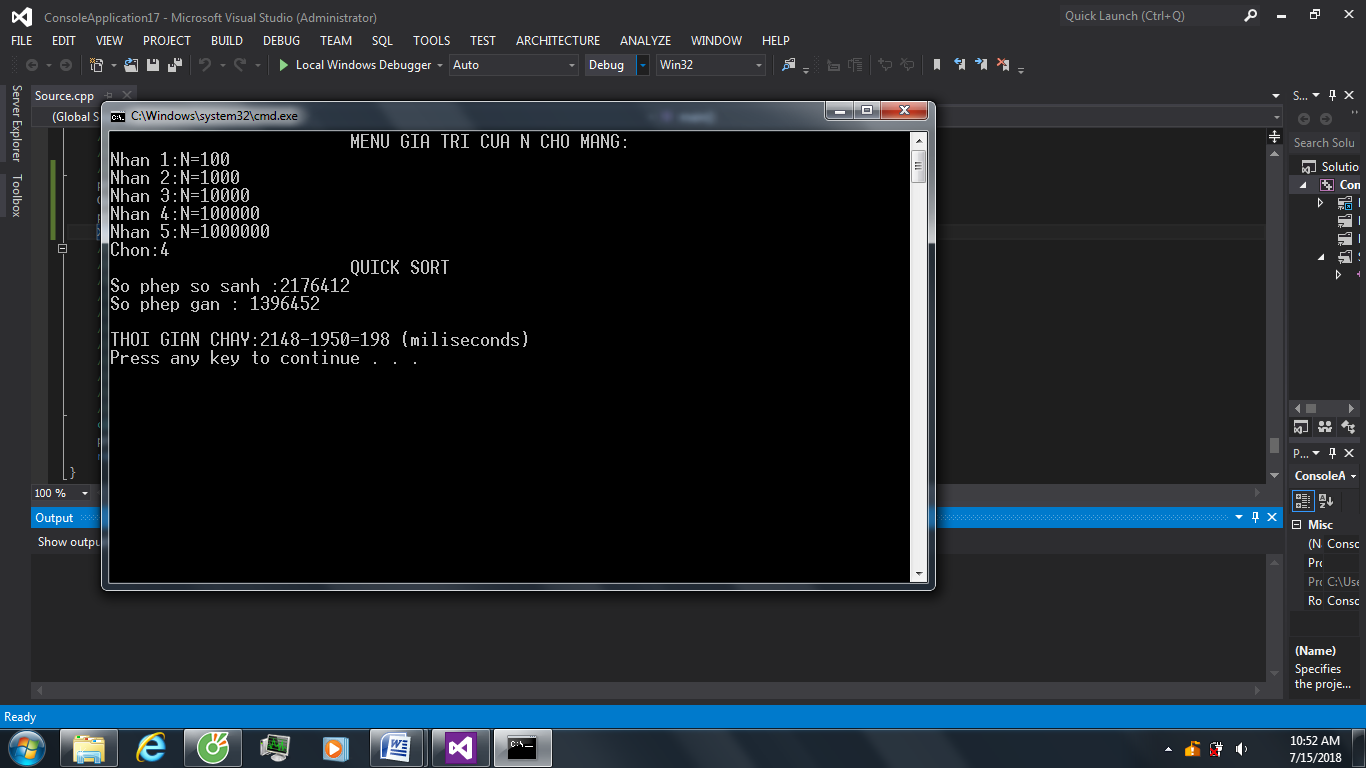


Số phép so sánh :99999

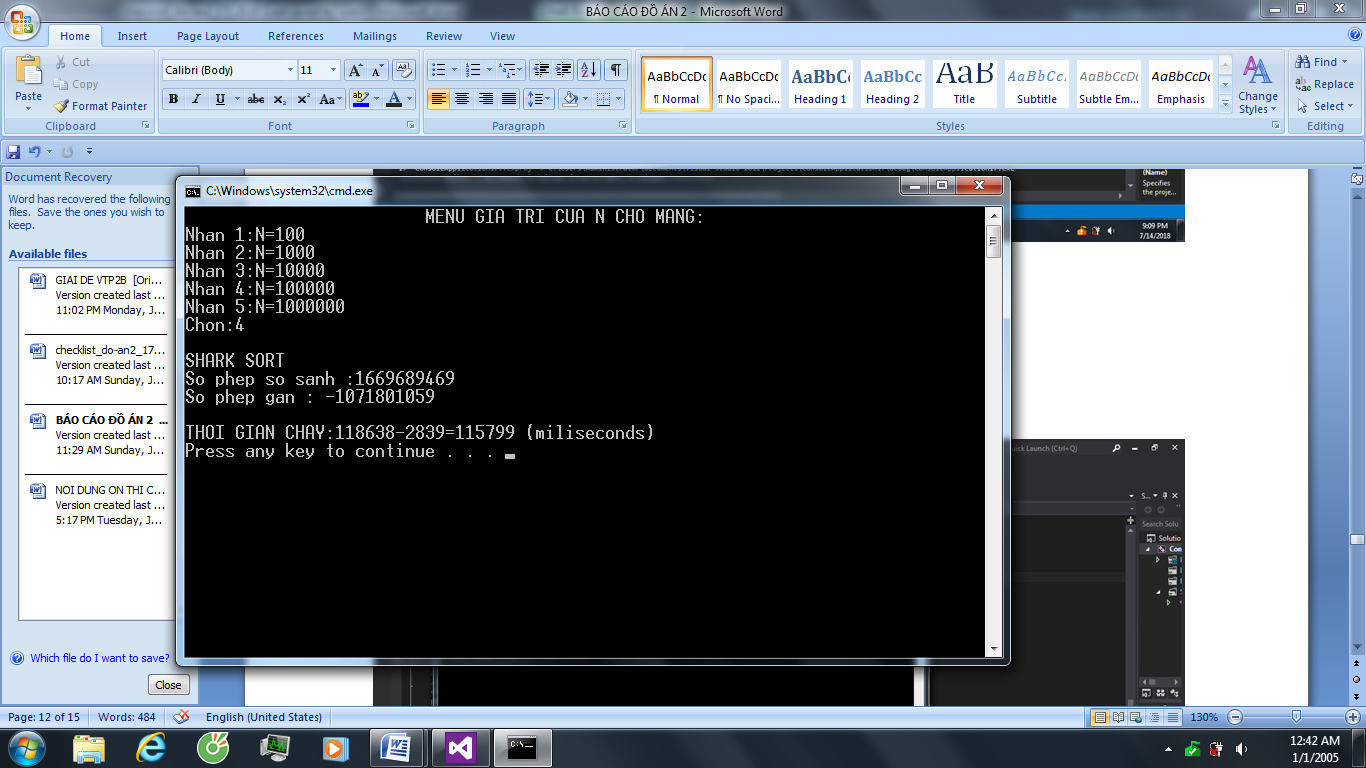
Số phép gán:2506244509

Thời gian chạy:27592ms

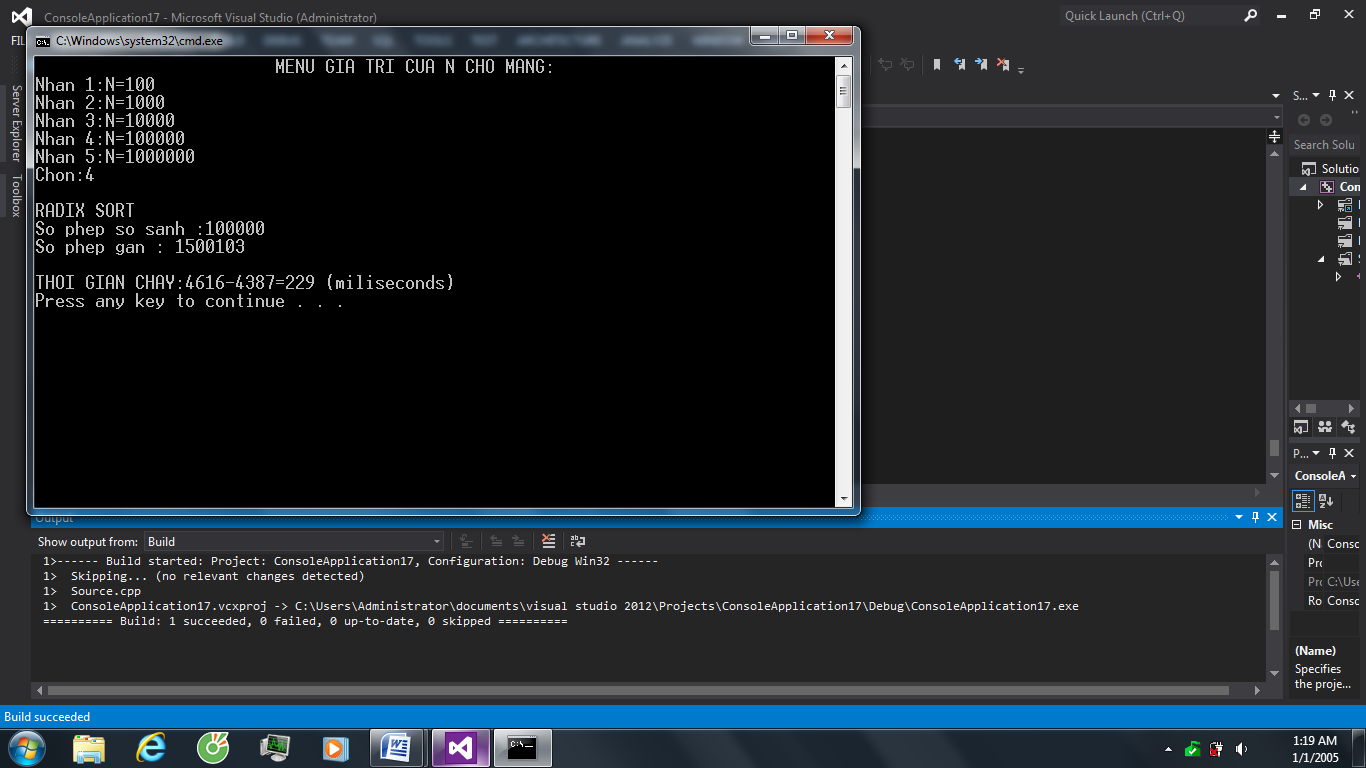
QUICK SORT



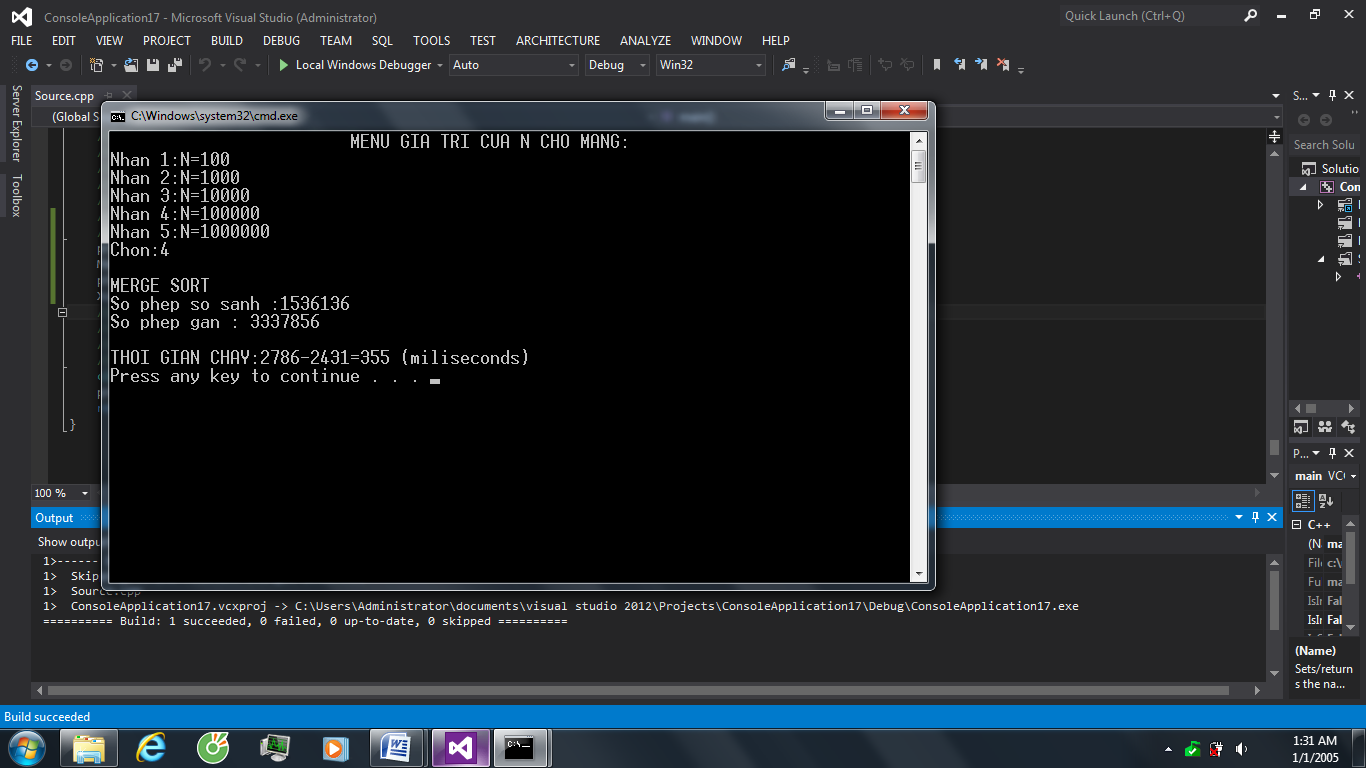
SHARK SORT



**RADIX SORT**

****

**MERGE SORT**

****

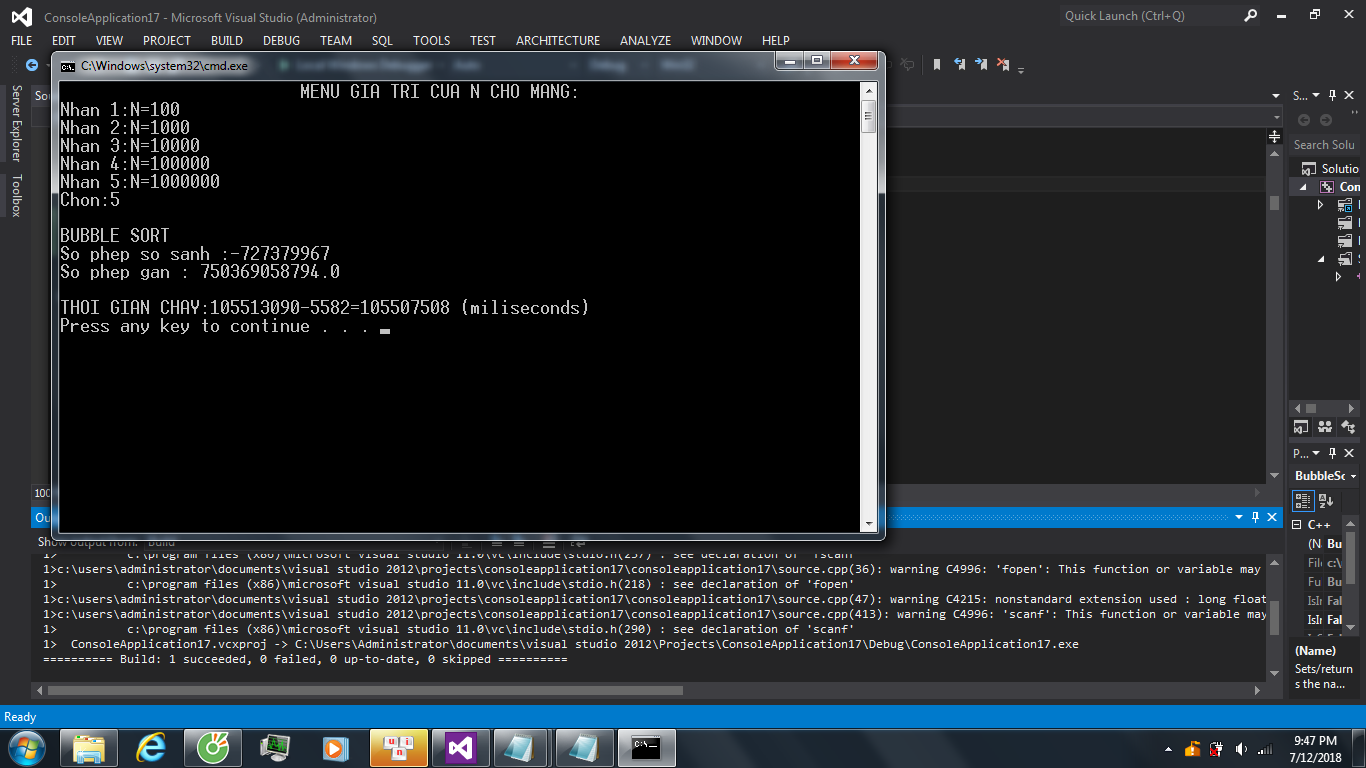
**Bảng báo cáo thống kê:**

**BẢNG SỐ LIỆU THỐNG KÊ MẢNG GỒM 100000 PHẦN TỬ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Thuật toán | Số phép gán | Số phép so sánh | Thời gian(ms) |
| 1 | InsertionSort | 2506244509 | 99999 | 27592 |
| 2 | BubbleSort | 7518133533 | 704982704 | 189515 |
| 3 | RadixSort | 1500103 | 100000 | 229 |
| 4 | ShellSort | LỖI | LỖI | LỖI |
| 5 | MergeSort | 3337856 | 1536136 | 355 |
| 6 | QuickSort | 1396452 | 2176412 | 198 |
| 7 | HeapSort | LỖI | LỖI | LỖI |
| 8 | Sharksort | Lỗi | 1669689469 | 115799 |

**Với N=1000000**

BUBBLE SORT

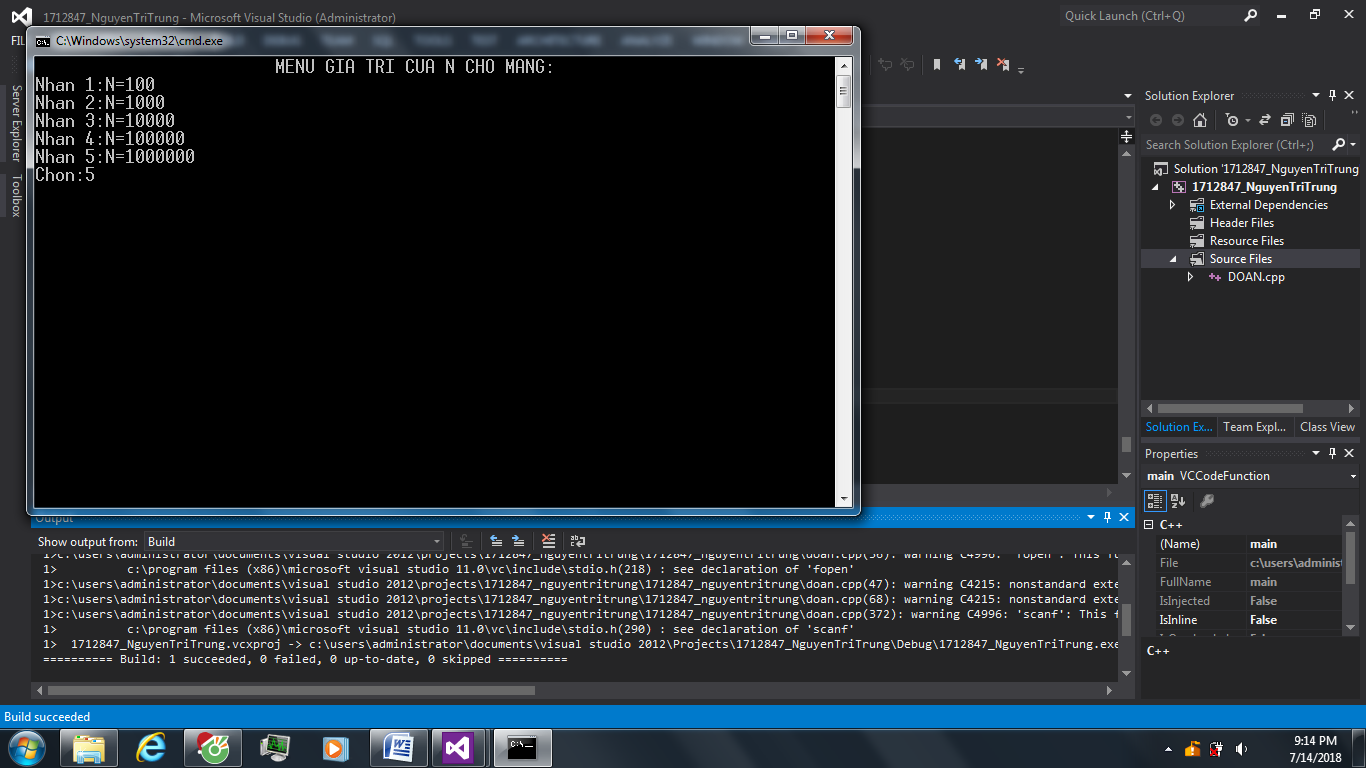


Số phép so sánh :lỗi do số quá lớn (nếu khai báo kiểu long float sẽ được số cụ thể như phép gán)

Số phép gán:750369058794

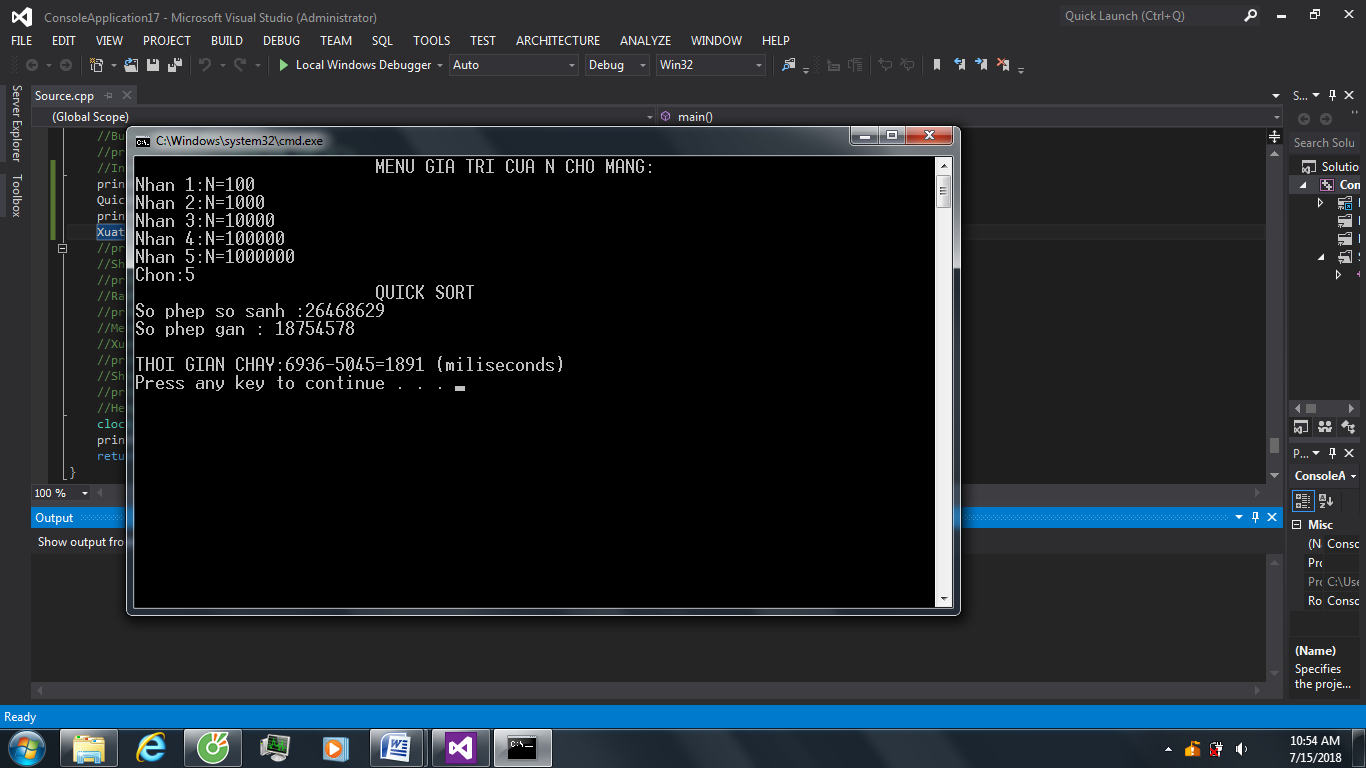
Thời gian chạy:105507508ms

INSERTION SORT



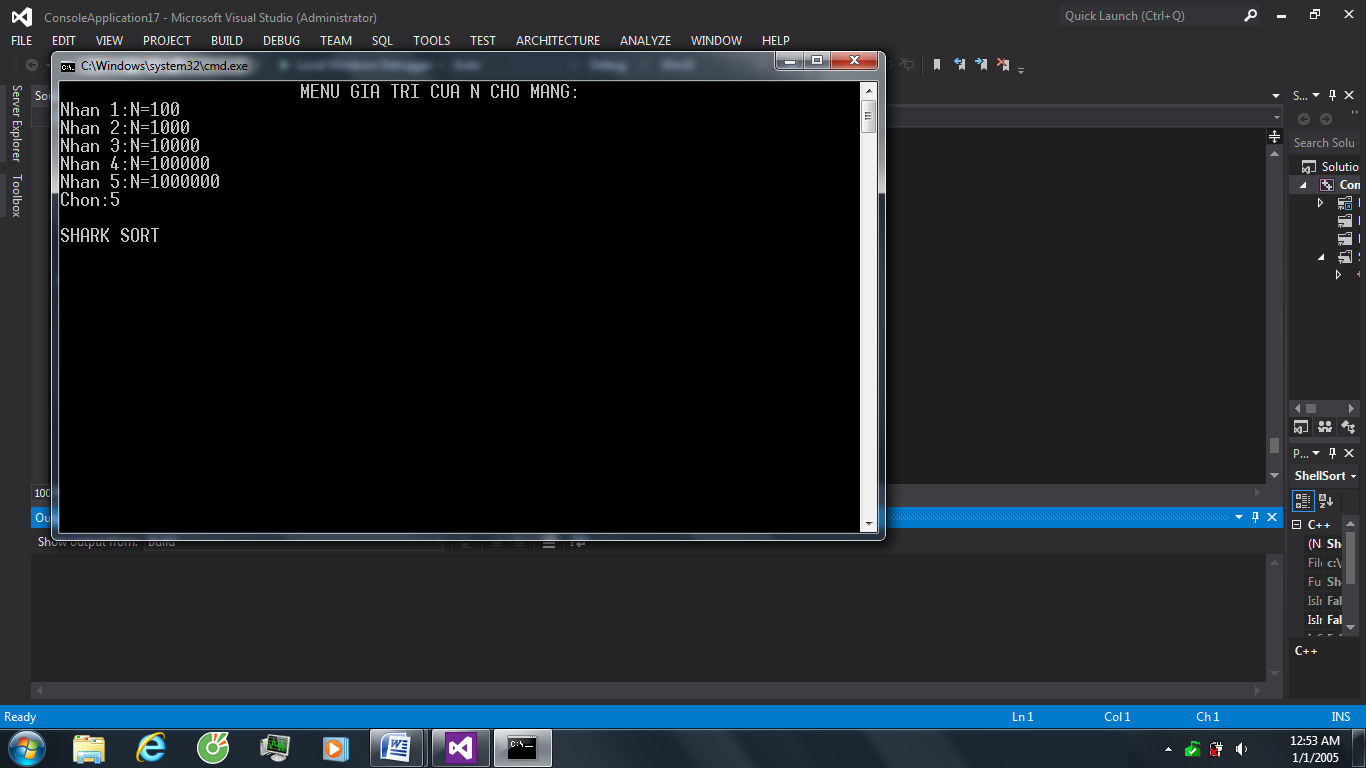
Thời gian quá lâu nhưng nếu nhìn vào kết quả của các N (100->1000000) thì ta có thể nhận xét được thời gian của cách INSERTION SORT sẽ ngắn hơn rất nhiều so với BUBBLE SORT rất nhiều ,không những vậy số phép so sánh và gán sẽ ít hơn nhiều.Vậy nên ta thấy sự quan trọng của thuật toán như thế nào

QUICK SORT



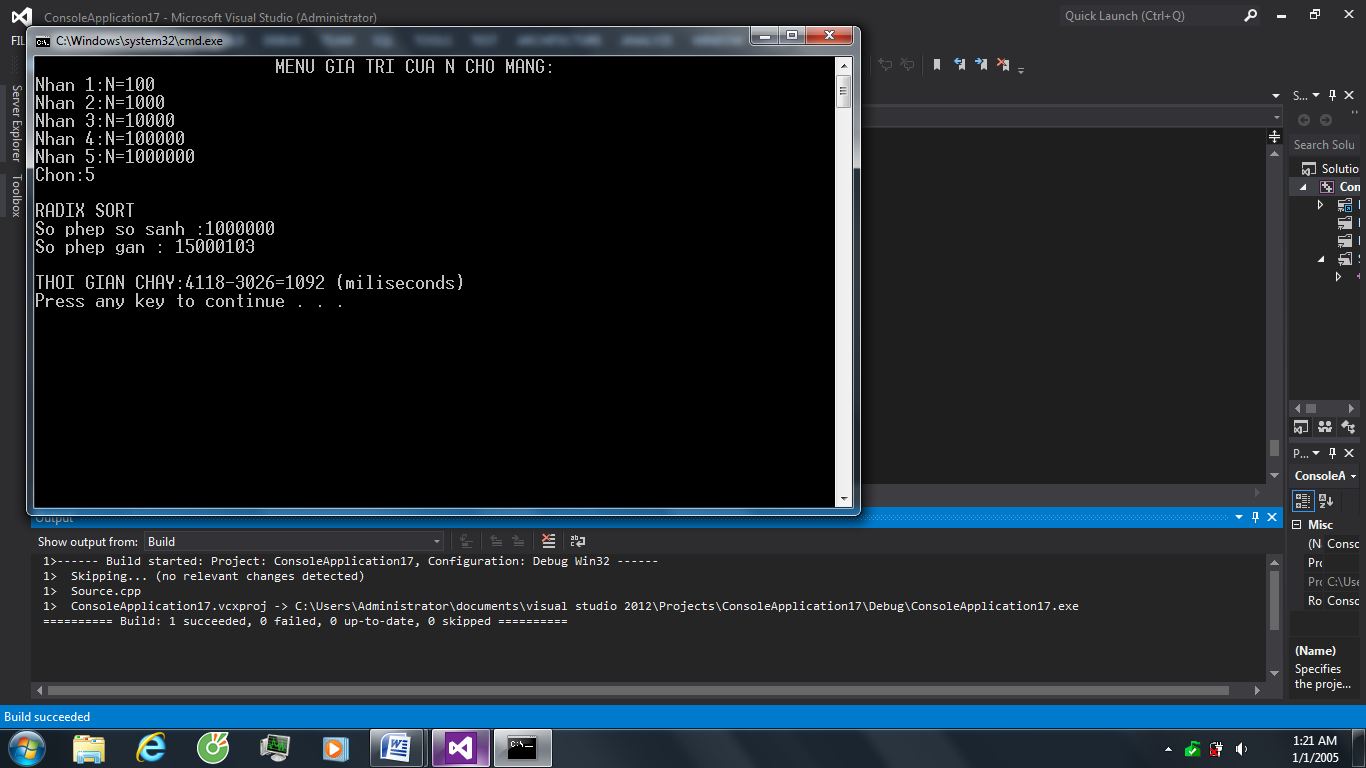
Ta thấy với N=1000000 mà thuật toán Quick chỉ chạy 1891s,nhanh hơn rất nhiều so với thuật toán INSERTION SORT và BUBBLE SORT,còn tối ưu hơn về số phép so sánh và số phép gán so với 2 thuật toán trên.

**SHARK SORT**

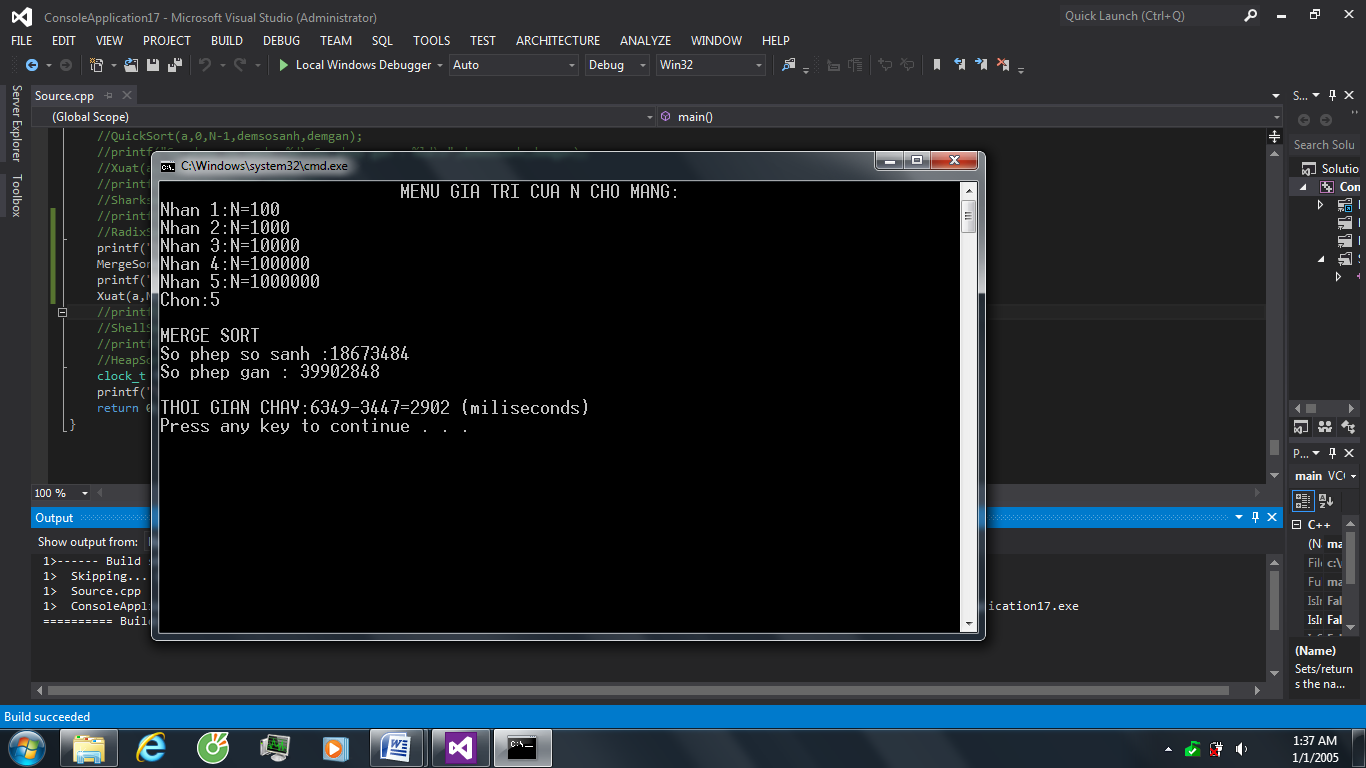


Thời gian tương đối lâu,chỉ nhanh hơn BUBBLE SORT.

**RADIX SORT**

****

**MERGE SORT**

****

**2.3.Nhận xét**

Nhưng như ta đã nói,độ nhanh chậm của thuật toán tùy thuộc vào thứ tự các phần tử cần sắp xếp và độ xử lý của máy tính,với mỗi lần chạy sẽ cho ra kết quả khác nhau nếu radom .Những xét về mặt chung thì thuật toán BubleSort là thuật toán có số phép gán, số phép so sánh và thời gian chạy lâu nhất => là thuật toán kém hiệu quả nhất. QuickSort là thuật toán chạy hiệu quả nhất ,chạy khá nhanh và số phép gán, so sánh ít nhất .

**2.4.Tự chấm điểm:** 8/10 (