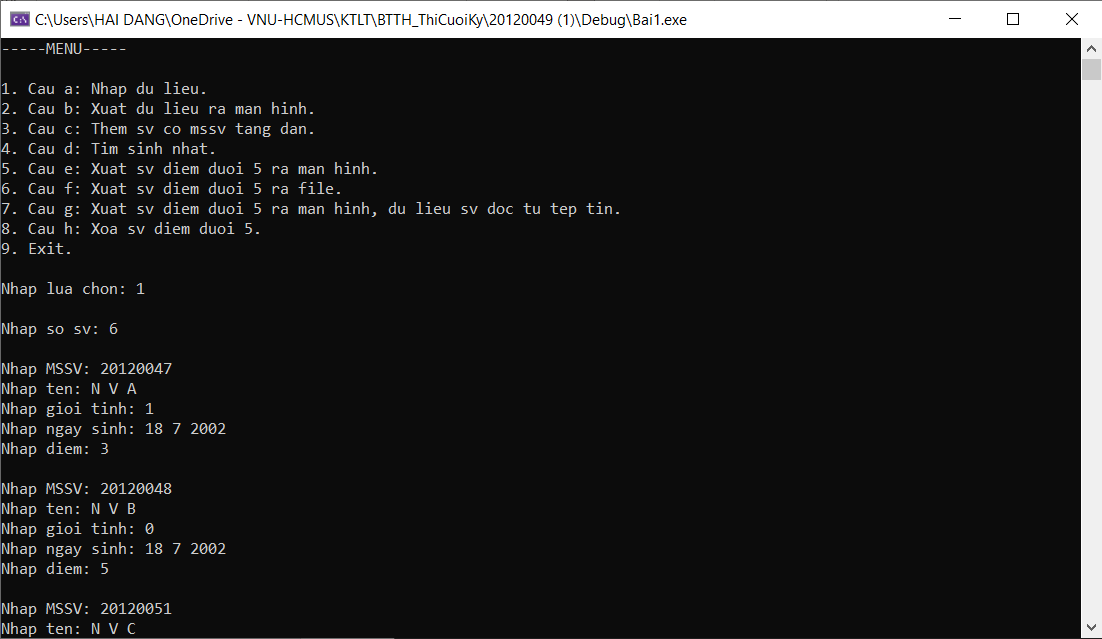
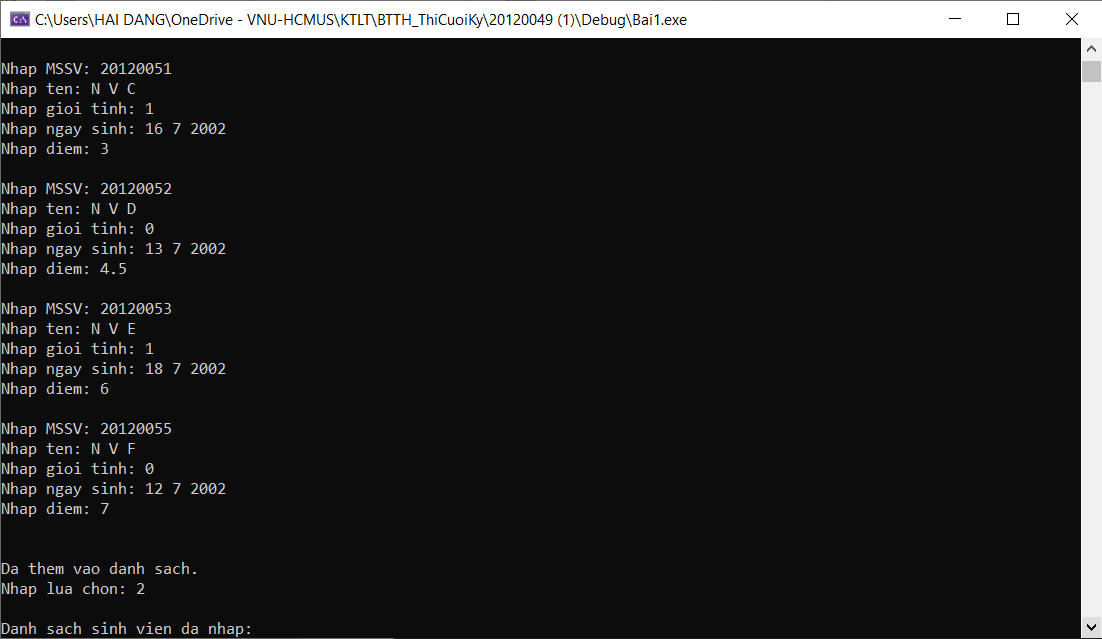
|  |  |  |
| --- | --- | --- |
| Họ và tên: Nguyễn Hải Đăng. | MSSV: 20120049. | Nhóm: N1. |

**BÀI 1:** (hoàn thành: 100%).

a. Hàm nhập thông tin 1 sinh viên rồi đưa vào danh sách liên kết (hoàn thành: 100%).

**Input**:



****

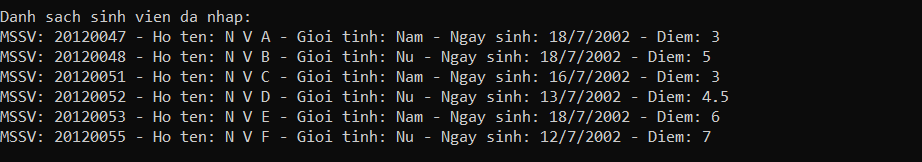
**Output**:



b. Xuất thông tin sinh viên ra màn hình (hoàn thành: 100%).

**Input**: Như câu 1a.

**Output**:



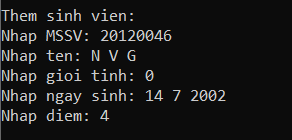
c. Giả sử danh sách sinh viên đã có thứ tự tăng dần theo mã số sinh viên, viết hàm  
thêm một sinh viên vào danh sách, mà vẫn duy trì thứ tự sau khi thêm (hoàn thành: 100%).

**Input danh sách ban đầu**: Như câu 1a.

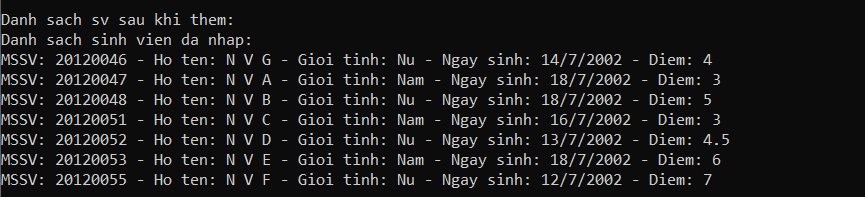
Xét 3 trường hợp:

Trường hợp 1: Nếu MSSV của sinh viên mới nhỏ hơn MSSV của sinh viên đầu tiên thì thêm sinh viên đó vào đầu danh sách (đặt node sinh viên mới là key, node tiếp theo là next); key->next = node đầu tiên của danh sách cũ; node đầu tiên của danh sách cũ = key.

**Input sinh viên mới:**

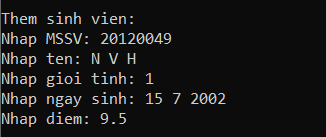
****

**Output:**

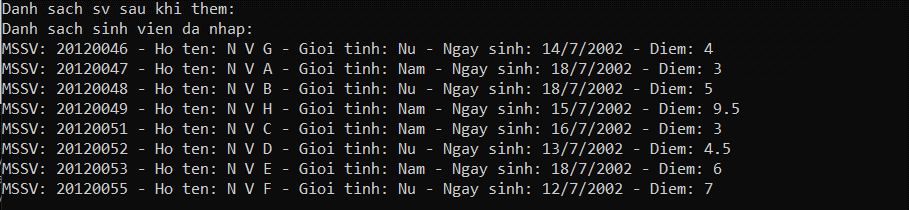
****

Trường hợp 2: Nếu MSSV của sinh viên mới **không** nhỏ hơn MSSV của sinh viên đầu tiên và nếu MSSV mới lớn MSSV của sinh viên hiện tại thì: key->next = node tiếp theo của sinh viên hiện tại, node tiếp theo của sinh viên hiện tại = node của sinh viên mới, kết thúc vòng lặp. Nếu không thoả mãn điều kiện đầu thì sẽ xét tiếp của node tiếp theo, cho đến khi kết thúc vòng lặp.

**Input sinh viên mới:**

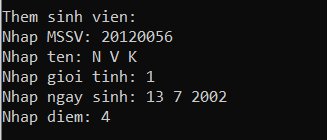


**Output:**

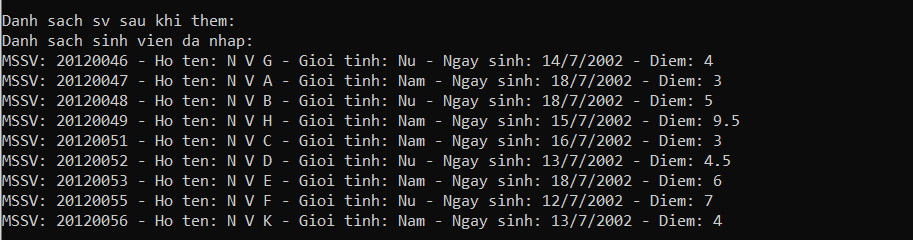


Trường hợp 3: Nếu kết thúc vòng lặp mà vẫn chưa thêm được sinh viên mới thì có nghĩa là MSSV của sinh viên mới lớn hơn tất cả MSSV của các sinh viên đã có trong danh sách. Vì vậy ta tiến hành thêm sinh viên đó vào cuối danh sách.

**Input sinh viên mới:**



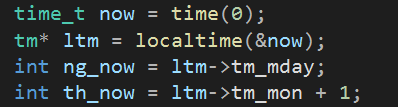
**Output:**



d. Tìm sinh viên có ngày sinh nhật hiện tại (hoàn thành: 100%).

**Input:** Như output của câu 1c.

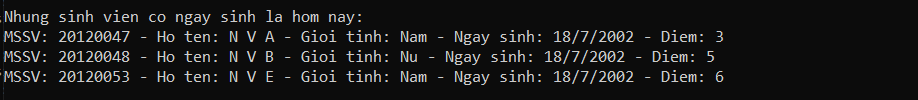
Cách lấy thời gian từ hệ thống máy tính**:**

****

Biến ng\_now: là ngày hiện tại.

Biến th\_now: là tháng hiện tại.

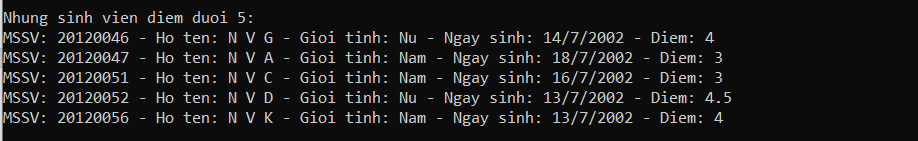
**Output:**



e. Tìm sinh viên điểm trung bình dưới 5.

**Input:** Như output của câu 1c.

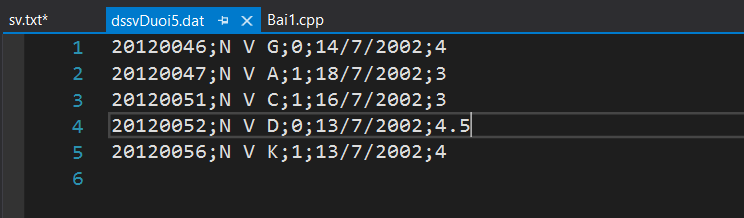
**Output:**



f. Xuất toàn bộ sinh viên điểm dưới 5 ra tệp tin văn bản (hoàn thành: 100%).

**Input:** Như output của câu c.

**Output:**



g. Đọc thông tin sinh viên dưới 5 từ tệp tin và in ra màn hình.

**Input:**

- Tạo biến f có kiểu fstream và biến f mở file sv.txt, một danh sách liên kết K;

- Sử dụng vòng lặp while để đọc tệp tin cho đến khi con trỏ tệp tin trỏ đến cuối tệp tin thì dừng lại, nội dung đọc như sau:

+ Đọc MSSV của sinh viên bằng f.getline, cho đến khi gặp ký tự “;” thì dừng đọc, lưu MSSV và nhảy 1 ký tự; nếu MSSV là rỗng thì kết thúc việc đọc file.

+ Đọc tên của sinh viên f.getline, cho đến khi gặp ký tự “;” thì dừng đọc, lưu tên và nhảy 1 ký tự.

+ Đọc giới tính của sinh viên, sau đó sử dụng f.seekg(1,1) để nhảy 1 ký tự.

+ Đọc ngày sinh của sinh viên, sau đó sử dụng f.seekg(1,1) để nhảy 1 ký tự.

+ Đọc tháng sinh của sinh viên, sau đó sử dụng f.seekg(1,1) để nhảy 1 ký tự.

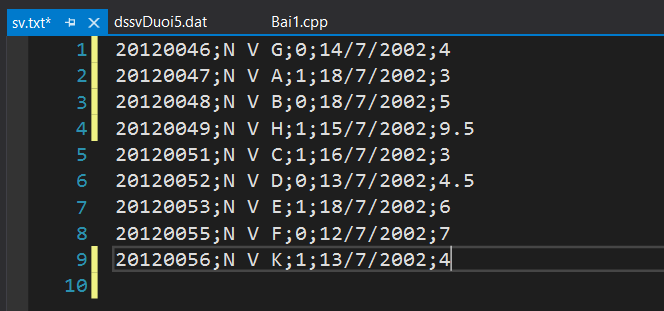
+ Đọc năm sinh của sinh viên, sau đó sử dụng f.seekg(1,1) để nhảy 1 ký tự.

+ Đọc điểm của sinh viên, sau đó sử dụng f.ignore() để xoá ký tự “\n” trong bộ nhớ đệm.

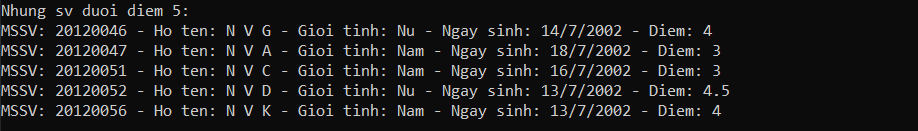
+ Tạo 1 node với dữ liệu sinh viên đã đọc ở trên.

+ Thêm node đó vào cuối danh sách liên kết K.

- Kết thúc vòng lặp, sau đó đọc các sinh viên có điểm dưới 5 (sử dụng while và if).



**Output:**

****

h. Xoá sinh viên có điểm dưới 5 (hoàn thành: 100%).

**Input:** Như output của câu 1c.

Xét 3 trường hợp:

Trường hợp 1: Nếu node đầu tiên có dữ liệu điểm số dưới 5.

- Node cần xoá là node đầu tiên.

- Node kế tiếp sau node đầu tiên sẽ trở thành node đầu tiên.

- Delete node cần xoá.

- Node cần xoá có địa chỉ NULL.

- Trỏ đến node kế tiếp.

**2 trường hợp tiếp theo sẽ đưa vô vòng lặp while cho đến khi node kế tiếp của node đang trỏ có địa chỉ NULL thì sẽ dừng lại.**

Trường hợp 2: Nếu node có điểm số dưới 5 nằm ở giữa danh sách.

- Xác định node trước node cần xoá (gọi là node A).

- Node cần xoá là node kế tiếp node A.

- Node kế tiếp node A sẽ trở thành node **kế tiếp kế tiếp** node A.

- Địa chỉ node kế tiếp node cần xoá sẽ là NULL.

- Delete node cần xoá.

- Địa chỉ node đã xoá sẽ là NULL.

Trường hợp 3: Nếu node có điểm số dưới 5 nằm ở cuối danh sách.

- Xác định node trước node cần xoá (gọi là node A).

- Node A sẽ trở thành node cuối cùng của danh sách.

- Node kế tiếp node A sẽ trỏ tới NULL.

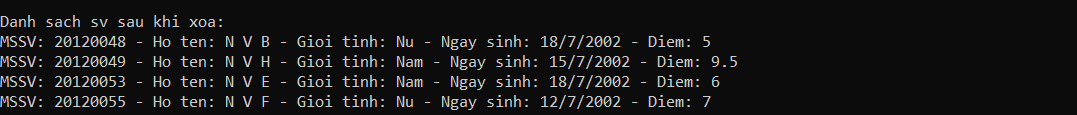
- Node kế tiếp node cần xoá sẽ trỏ tới NULL.

- Delete node cần xoá.

- Địa chỉ node đã xoá sẽ là NULL.

**Nếu node đang trỏ đều không thoả mãn cả 3 trường hợp trên thì chương trình sẽ trỏ đến node kế tiếp.**

**Output:**



**BÀI 2:**

a. Viết công thức tính và viết hàm tính  theo  và .



**Input:**



**Output:**



b. Tính .

**Input:** như câu 2a.

**Output:**



c. Tính .

**Input:** như câu 2a.

**Output:**



d. Viết hàm tính  không sử dụng vòng lặp.

**Input:** như câu 2a.

- Đầu tiên gọi biến kq để lưu kết quả.

- Sử dụng vòng lặp for chạy từ i = 1 -> n – 1, trong vòng lặp tính .

- Trả về giá trị hàm là giá trị biến kq.

**Output:**

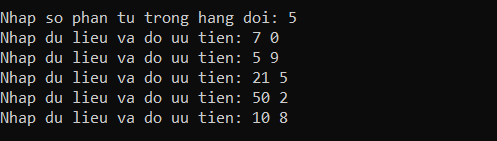


**BÀI 3:** (hoàn thành: 100%).

a. Hàm tạo hàng đợi ưu tiên (hoàn thành: 100%).

**Input:**

Khi tạo một node mới vào hàng đợi, thì node mới sẽ tự động được đưa vào cuối hàng đợi. Rồi sau đó sử dụng thuật toán Bubble Sort để sắp xếp dữ liệu theo thứ tự độ ưu tiên của node đó.



**Output:**



b. Hàm xuất hàng đợi ưu tiên ra màn hình (hoàn thành: 100%).

**Input:** như câu 3a.

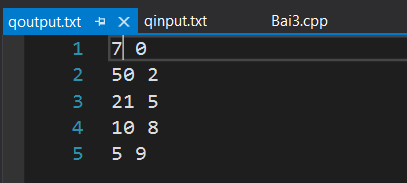
**Output:**



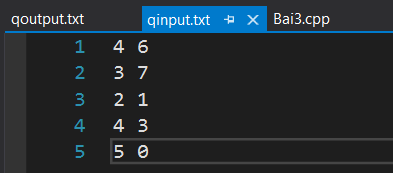
c. Hàm xuất hàng đợi ưu tiên ra tệp tin (hoàn thành: 100%).

**Input:** như câu 3a.

**Output:**



d. Hàm đọc hàng đợi ưu tiên từ tệp tin (hoàn thành: 100%).



**Output:**



e. Lấy một phần tử hàng đợi ưu tiên (hoàn thành: 100%).

**Input:** như câu 3a.

**Output:** Phần tử được lấy là phần tử đầu tiên hàng đợi.

