**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**LÊ QUANG MINH – 20120329**

**BÁO CÁO ĐỒ ÁN 2**

**CỘNG TRỪ VÀ SẮP XẾP SỐ NGUYÊN LỚN BẰNG DANH SÁCH LIÊN KẾT**

**| Giáo viên hướng dẫn |**

**ThS. Phạm Minh Hoàng**

**Khoa Công Nghệ Thông Tin**

Thành phố Hồ Chí Minh – 2021

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**LÊ QUANG MINH – 20120329**

**BÁO CÁO ĐỒ ÁN 2**

**CỘNG TRỪ VÀ SẮP XẾP SỐ NGUYÊN LỚN BẰNG DANH SÁCH LIÊN KẾT**

**| Giáo viên hướng dẫn |**

**ThS. Phạm Minh Hoàng**

**Khoa Công Nghệ Thông Tin**

Thành phố Hồ Chí Minh – 2021

|  |
| --- |
| LỜI CẢM ƠN |

|  |
| --- |
| MỤC LỤC |

Contents

[LỜI CẢM ƠN 3](#_Toc77604917)

[MỤC LỤC 4](#_Toc77604918)

[DANH MỤC HÌNH 5](#_Toc77604919)

[DANH MỤC BẢNG 6](#_Toc77604920)

[DANH MỤC TỪ VIẾT TẮT 7](#_Toc77604921)

[GIỚI THIỆU ĐỒ ÁN 8](#_Toc77604922)

[HEADER VÀ NHẬP BIGNUM TỪ FILE .TXT 10](#_Toc77604923)

[CỘNG TRỪ SỐ NGUYÊN LỚN 13](#_Toc77604924)

[QUICKSORT VÀ MERGESORT 22](#_Toc77604925)

|  |
| --- |
| DANH MỤC HÌNH |

|  |
| --- |
| DANH MỤC BẢNG |

|  |
| --- |
| DANH MỤC TỪ VIẾT TẮT |

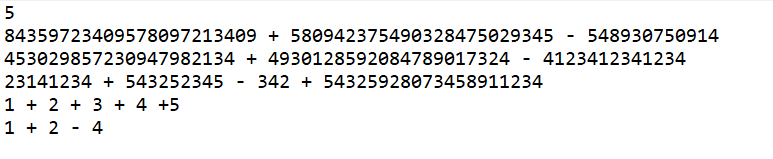
|  |
| --- |
| GIỚI THIỆU ĐỒ ÁN |

Đồ án được xây dựng để giải quyết ý tưởng rằng nếu chúng ta có một số nguyên quá lớn mà các kiểu dữ liệu bình thường không thể lưu trữ được ( int, long, long long,…) thì ta buộc phải nghĩ ra được một cấu trúc dữ liệu mới để có thể lưu trữ các số nguyên ngày và phát triển các thuật toán để có thể làm việc với loại cấu trúc dữ liệu mới này.

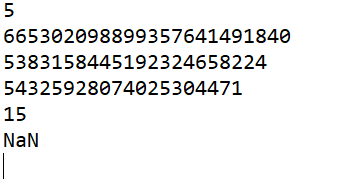
Để thực hiện ý trên, ta có thể sử dụng danh sách liên kết. Ta coi một số nguyên lớn là một danh sách liên kết, với mỗi node chứa k chữ số của số nguyên lớn trong đó. Lý do sử dụng danh sách liên kết thay vì string là vì mỗi node của danh sách liên kết nằm ở những vị trí khác nhau liên kết qua các con trỏ Next, và ta quản lý danh sách đó bằng hai con trỏ Head và Tail, vì vậy danh sách liên kết chỉ đầy khi mà heap đầy, khác với ý tưởng ta sử dụng string ( có thể dẫn tới StackOverflown hoặc HeapOverflown ) vì string là các mảng char liên tiếp nhau.

**Demo của cộng trừ số nguyên cực lớn ( được cắt ra từ quá trình test của em )**

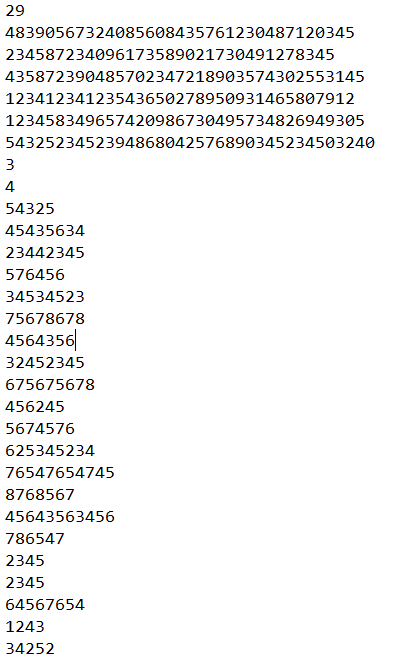
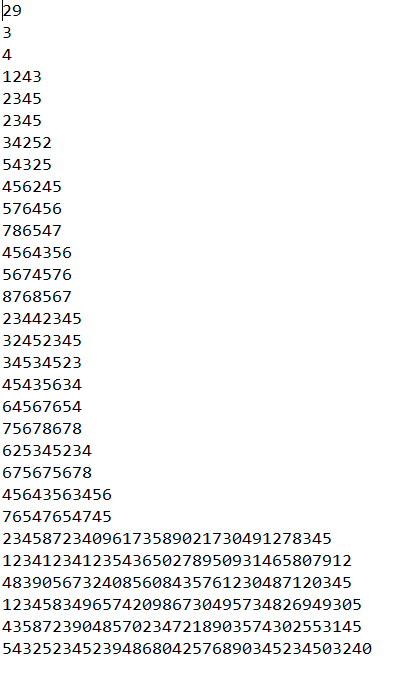
***Input ( với k = 7 )***

****

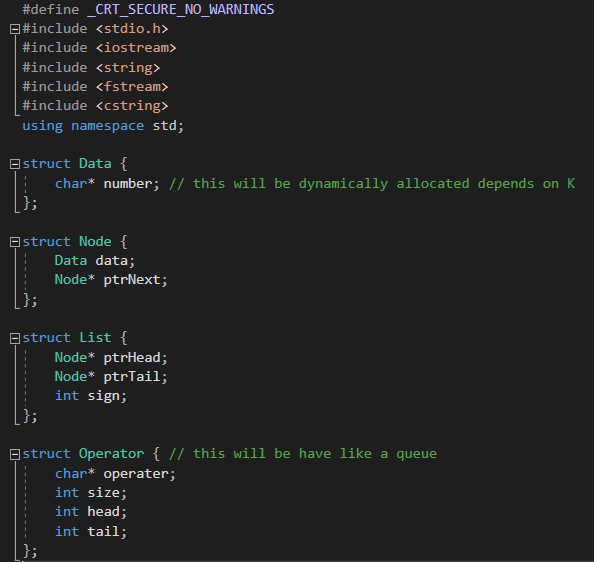
***Output***

****

**Demo của sort các số nguyên cực lớn ( quicksort với mergesort cho ra kết quả như nhau, k = 7 )**

***Input*  *Output***

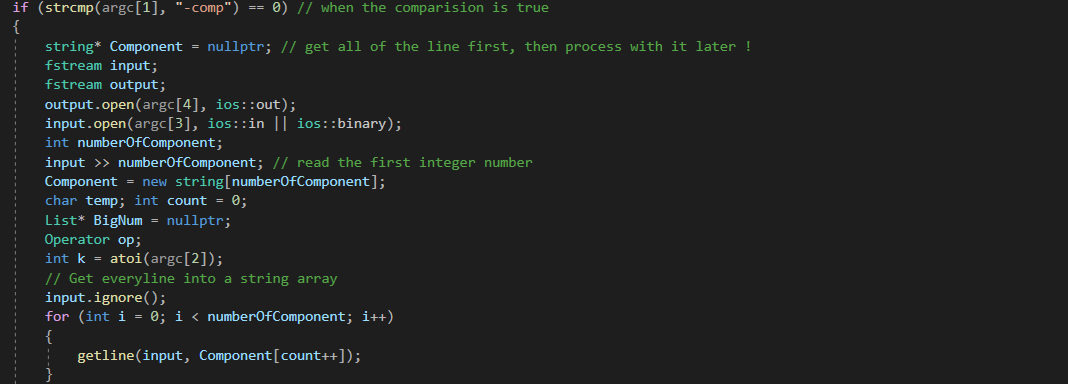
|  |
| --- |
| HEADER VÀ NHẬP BIGNUM TỪ FILE .TXT |

****

Ở phần Header, ta cài đặt một danh sách liên kết như bình thường, với struct Data chứa con trỏ trỏ tới một mảng char được cấp phát động ( chứa k phần tử )

Struct Operator ( toán tử ) sẽ hoạt động như một hàng đợi. Ta dễ nhận thấy việc cộng trừ số nguyên giống như việc t lấy từng toán tử ra và cộng với số nguyên đi kèm nó.

Về việc nhập số nguyên lớn từ file .txt, em thực hiện việc nhập ở main, em chia ra thành 2 loại cấu trúc file là file dành cho lệnh -comp và file dành cho hai lệnh sort còn lại



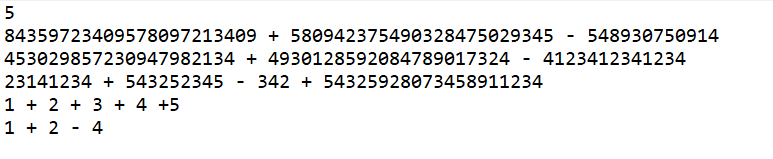
**Cách thức lấy các mảng string từ file .txt cho lệnh -comp**

Để giải thích thì em tạo ra một con trỏ trỏ tới một mảng string với số phần tử là số dòng được cho dựa trên số nguyên ở đầu file

Sau khi có các tác vụ như vậy, con trỏ string của em sẽ chứa dữ liệu như sau

string Component = {dòng 1, dòng 2, dòng 3,…, dòng n}

Để dễ trực quan, em sẽ sử dụng ví dụ demo ở đầu bài

****

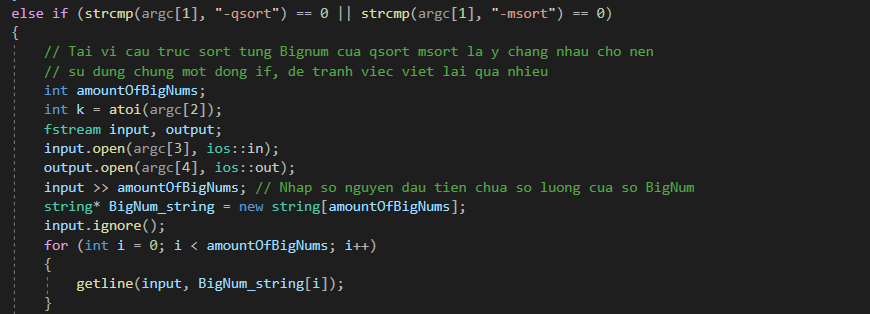
Ta có numberOfComponent sẽ chứa giá trị bằng 5

Sau các lệnh trên, con trỏ Component sẽ trỏ tới mảng chứa các giá trị như sau

string Component = {84359723409578097213409 + 580942375490328475029345 – 548930750914, 453029857230947982134 + 4930128592084789017324 – 4123412341234, 23141234 + 543252345 - 342 + 54325928073458911234, 1 + 2 + 3 + 4 +5, 1 + 2 - 4}

Tức là vẫn chứa đủ các dấu trên một dòng, ta sẽ phát triển thuật toán đưa các toán tử vô một queue và các số lớn vô một danh sách liên kết ở nội dung sau

Tương tự với lệnh tính toán, ta có cách thức nhập file với 2 lệnh sort



Mặc dù khác tên, nhưng cách thức hoạt động vẫn tương tự với biến con trỏ string BigNum\_string, chỉ khác rằng ở đây sau khi thực thi hết các lệnh này thì mỗi phần tử trong string sẽ là một bignum ( và ta sẽ chuyển nó vô dslk )

**\*\*Các mảng string sau khi sử dụng xong đều sẽ được xóa ngay lập tức\*\***

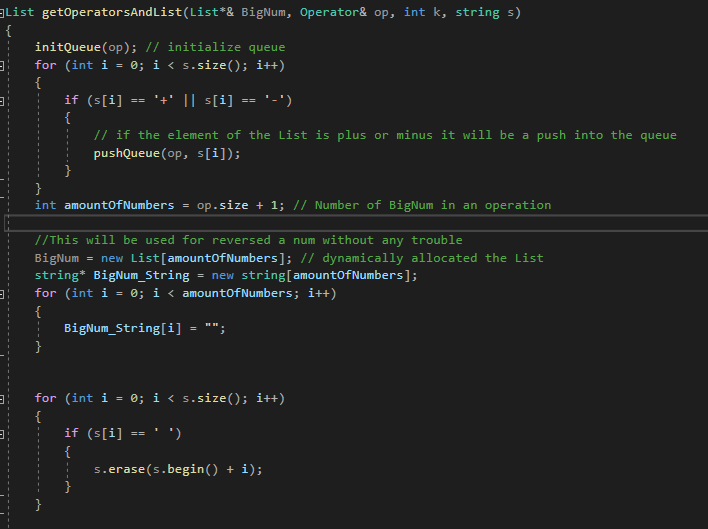
|  |
| --- |
| CỘNG TRỪ SỐ NGUYÊN LỚN |

**Vì tỉ lệ ảnh quá lớn nên em xin được phép gửi link ảnh chứa đủ source code của em ( hoặc thầy có thể xem trong source code hàm List GetOperatorAndList(…) )**

[**https://ibb.co/T2NrB4B**](https://ibb.co/T2NrB4B)

Giải thích source code:

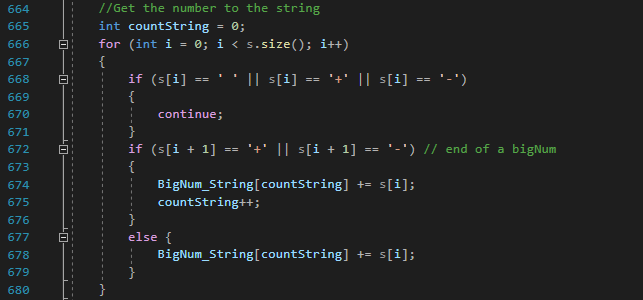
Đây là một hàm nhận vào một con trỏ List trỏ tới các mảng danh sách liên kết rỗng, một hàng đợi các toán tử rỗng, số k đầu vào và một string ( là lần lượt các phần tử của mảng Component đã nhập vô ở trước )



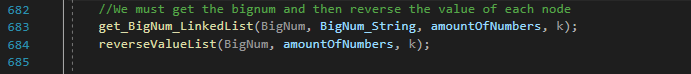
Đầu tiên là em từ string nhập vô từ tham trị, em sẽ lọc ra được các toán tử ( và đồng thời đếm số toán tử ) để bỏ và một queue ( FIFO )

Sau đó, số các bignum trong một string s nhập vô sẽ bằng số toán tử + 1 ( biến amountOfNumbers )

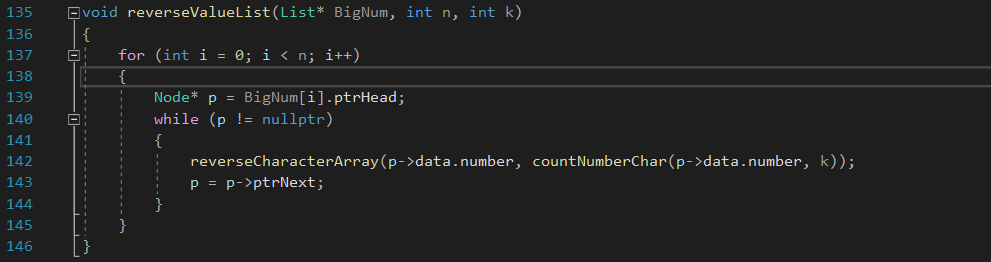
Từ đó, em khởi tạo tạm thời để tạo ra tiếp một mảng BigNum\_string, tách các dslk thành các số nguyên lớn rời rạc nhau



Đây là cách em lấy các BigNum\_String[i] tạm thời đó để rồi chuyển các string này thành các danh sách liên kết



Sau đó em thực hiện lời gọi hai hàm này, hàm get\_BigNum\_LinkedList là ta sẽ tạo ra các dslk từ BigNum\_String





Giải thích hàm get\_BigNum\_LinkedList

Ta nhận string đầu vào ( là string đã tách thành các số riêng ) ví dụ là:

84359723409578097213409, 580942375490328475029345, 548930750914

Ta nhập ngược các số vào danh sách

Thì output của ta khi đó sẽ là một danh sách liên kết với mỗi node là một k như sau

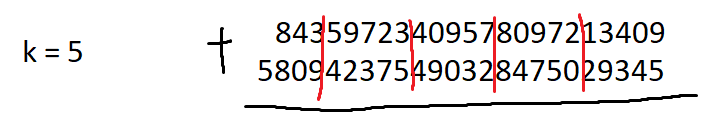
490431279087590432795348, 543920574823094573249085, 419057039845

Sẽ có trường hợp số tại node cuối cùng của danh sách lk lúc này ( tức là ở những số đầu tiên của số nguyên lớn vì ta nhập ngược , na ná việc reversed string ) sẽ có một trường hợp node đó có thành phần trong mảng char\* không đủ k phần tử, thì ta sẽ xử lý bằng cách thêm ký tự ‘\0’ vào đầu mảng char\* của node đó

Ví dụ số nguyên nhận vào là 1, mà k = 5, thì theo tính toán thì dslk chứa số 1 sẽ có 1 node với char\* là 5 phần tử lần lượt là {\0,\0,\0,\0,1}

Vì vấn đề đọc ngược mà ta phát sinh ra hàm reverseValueList, tức là là ta sẽ reverse dữ liệu char\* của từng node

Để minh họa rõ vì sao em đọc ngược từ đầu và lại reverse từng giá trị, em xin trình bày thuật toán cộng trừ của em như sau



Ta sẽ nhập ngược các chuỗi số nguyên lớn vào các dslk theo từng node, vì vậy ở node đầu của dslk của em ở số nguyên lớn thứ 1 trên hình mình họa, node đó sẽ là {9,0,4,3,1}, nhưng mà để tính toán, thì ta cần phải lấy số ngược của số ở trên, vì vậy em phải reverse giá trị char trong từng node lại thành {1,3,4,0,9} để có thể dịch nó về đúng ngôn ngữ tính toán như từ trước chúng ta đã học

Và ta thấy, ở trường hợp trên, ở node cuối của danh sách liên kết đầu tiên lúc này là

{\0,\0,3,4,8}, khi ta reverse value, nó sẽ trở thành {8,4,3,\0,\0}, lợi dụng được việc đó khiến việc tính toán cộng trừ của chúng ta trở nên dễ dàng

Vì vậy sau khi đã xác định được thuật toán rõ ràng, ta chỉ việc cộng hai danh sách liên kết lại với nhau

Hai hình dưới sẽ là trình bày code hàm Add,Subtraction và cách thức cộng dựa trên hàng đợi

Bản chất em sử dụng kỹ thuật cơ bản của hai con trỏ, em sẽ cho lần lượt hai con trỏ đi hết hai dslk, nếu con trỏ nào mà chưa đi hết dslk của chính nó thì sẽ cộng/trừ cho 0 tương ứng

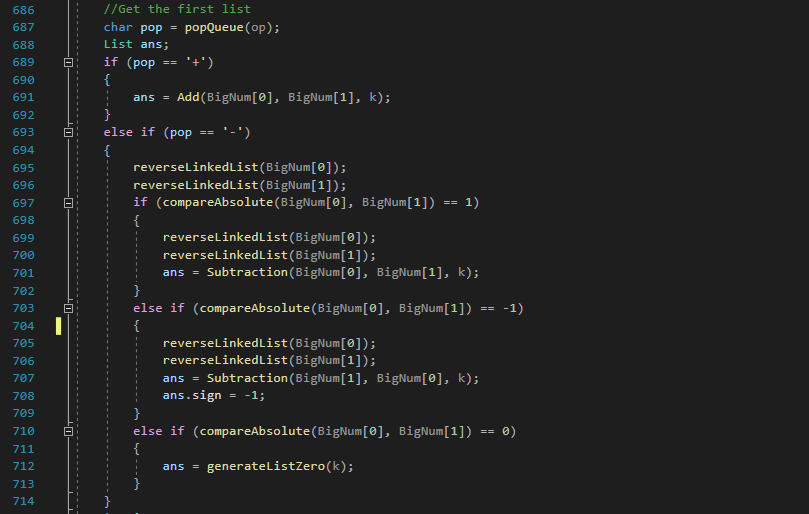
Ở hàm Add, mỗi một node ta xem như một số nguyên nhỏ, ta cộng hai số đó lại ở mỗi node của dslk, nếu mà số tính ra có số chữ lớn hơn k thì t áp dụng như cách tính tổng hai số, đó là ghi dư nhớ thương, ghi dư của số đó với 10^k, nhớ thương của số đó với 10^k

Ta áp dụng qui tắc còn lại tương tự với hàm Subtraction

Ở đây xuất hiện hàm so sánh hai số nguyên lớn ( compare2BigNum ), cơ chế của nó hoạt động sẽ được giải thích ở phần sort, ở đây là trả về 1 nếu danh sách A lớn hơn B, trả về -1 khi B lớn hơn A và trả về 0 khi cả hai danh sách bằng nhau





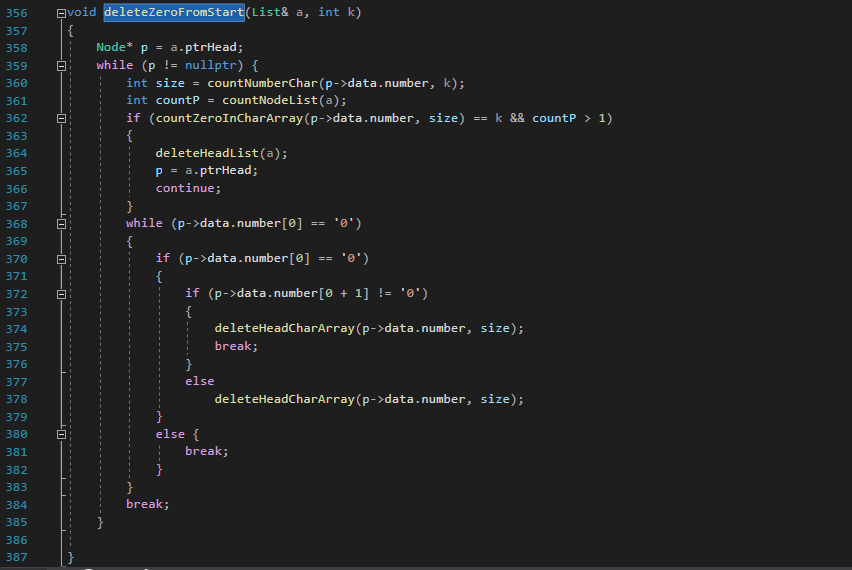


Đầu tiên ta phải lấy được tổng/hiệu của hai danh sách đầu tiên, sau đó ta liên tục pop các toán tử trong hàng đợi ra để làm phép toán, kết quả là ta sẽ return về một cái list Ans là đáp án của phép toán

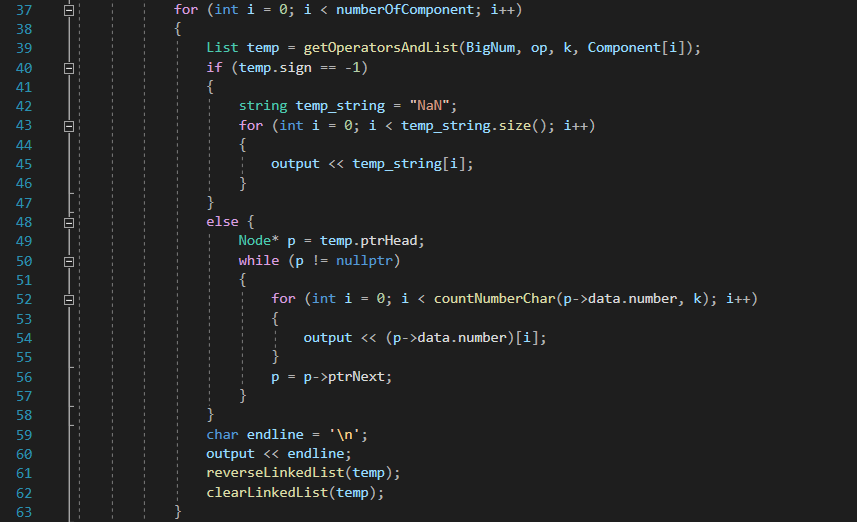


Ta vẫn sẽ gặp một vài trường hợp để dễ tính toán thì các số mà nếu ở node đầu không đủ k phần tử trong mảng char, thì em đã thêm vô các số “0” cho đủ

Vì vậy ta phát sinh thêm một hàm xóa các số 0 ở đầu node đi

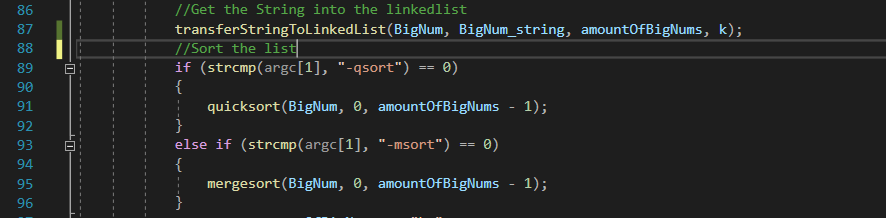


Ở hàm main, ta sẽ đi một vòng lặp từ dòng thứ 0 tới dòng thứ n-1 dựa trên số nguyên đầu nhập vô ở file .txt, và ta sẽ có kết quả tương tự như demo

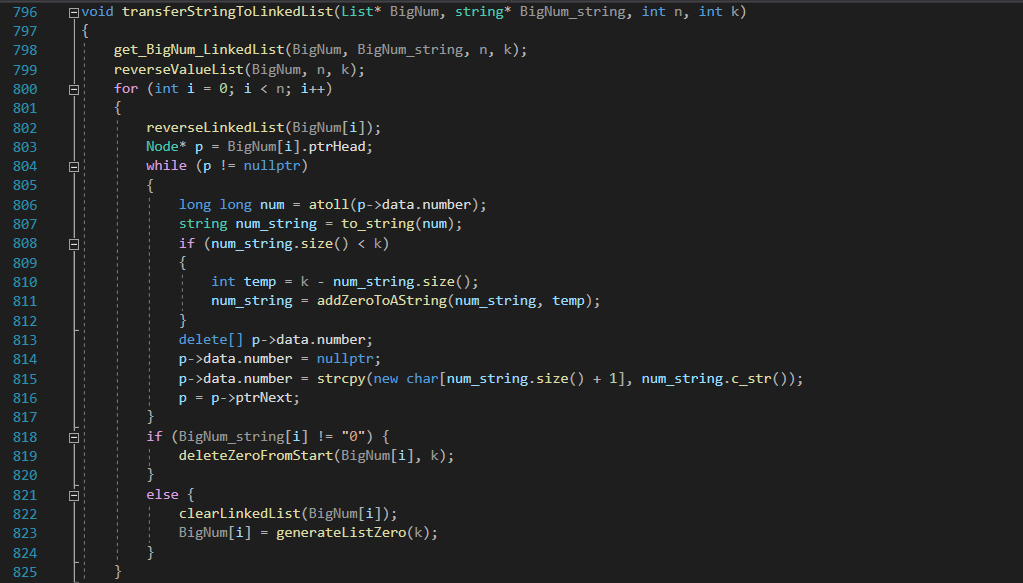


|  |
| --- |
| QUICKSORT VÀ MERGESORT |

Tương tự như đã nói ở trước, ta sẽ nhập các số nguyên lớn bằng việc lợi dùng hàm get\_BigNum\_LinkedList để nhập các dslk từ string vô, sau đó ta viết hàm so sánh hai số nguyên cực lớn, khi đã viết được hàm so sánh hai số nguyên cực lớn thì việc áp Quicksort và Mergesort rất là đơn giản.



Cơ chế của hàm transferStringToLinkedList



Cơ chế chuyển từ string sang dslk tương tự như ta làm với lệnh -comp

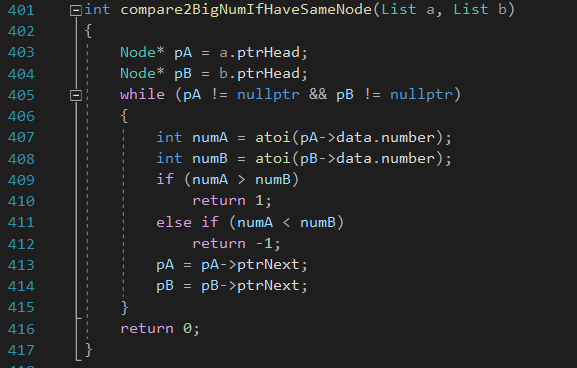
Còn dưới đây là hình của hàm compare2BigNum



Ý tưởng của hàm này là, ta sẽ so sánh dấu của hai số đó đầu tiên ( mặc dù thật sự không cần thiết ở phần sort , nhưng em viết chung để so sánh cộng trừ hai số nguyên lớn ở phần trước).

Sau khi ta so sánh được dấu của hai số đó, nếu cả hai số cùng dấu thì ta sẽ tiếp tục so sánh rằng trong danh sách đang có bao nhiêu Node, danh sách nào có Node nhiều hơn thì sẽ lớn hơn.

Nếu trong danh sách có số Node bằng nhau thì ta tiến hành vô hàm compare2BigNumIfHaveSameNode(...)



Ý tưởng của hàm này là ta sẽ dùng kỹ thuật hai con trỏ tương tự như cộng trừ ( mặc dù ở đây ta k cần xét dư hay không vì số node bằng nhau ở hai danh sách ), ta tách từng số nhỏ ra và so sánh.