Nhóm 6 Balanced Parentheses using Stack

Nghiêm Văn Quang 20203547

Nguyễn Tuấn Anh 20200038

Nguyễn Đức Lực 20203495

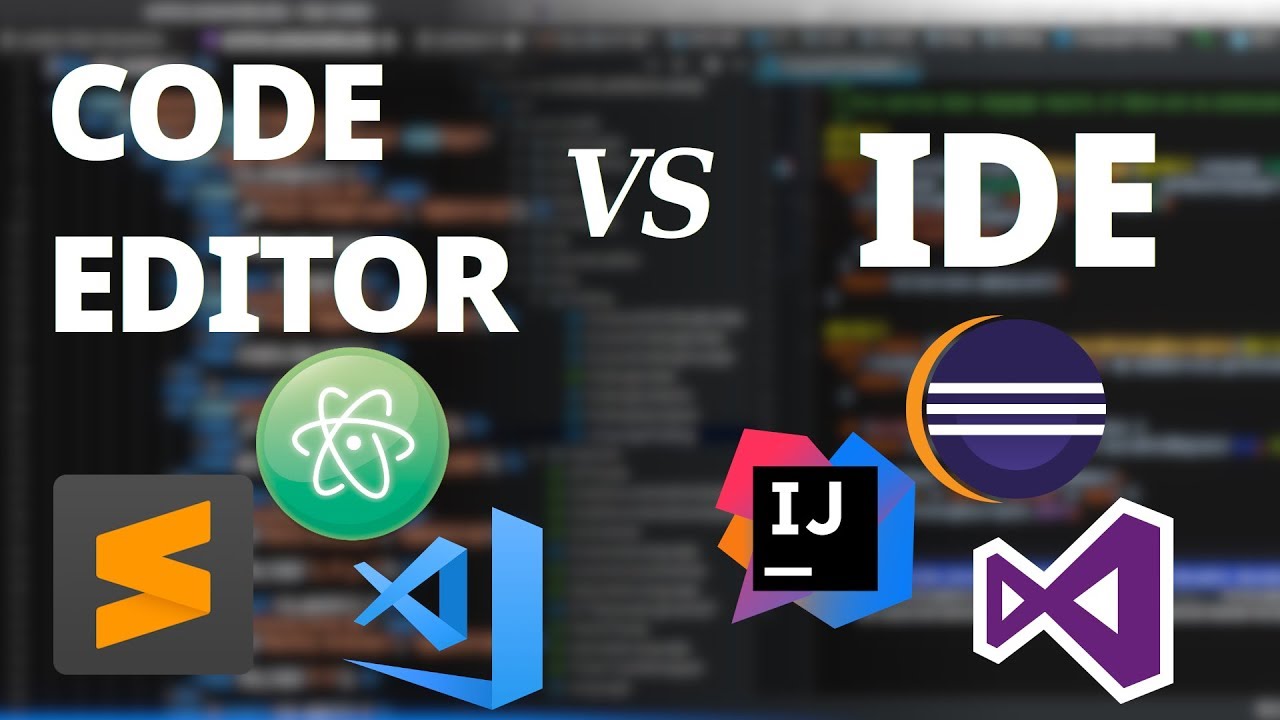
Chúng em xin lỗi cô vì thời gian gấp quá nên chưa hoàn thành kịp báo cáo 1 cách chỉnh chu nhất. Sau buổi bảo vệ thứ 7 bọn em sẽ hoàn thành lại và gửi cô bản final ạ

1. Giới thiệu
2. Lý do và động lực:

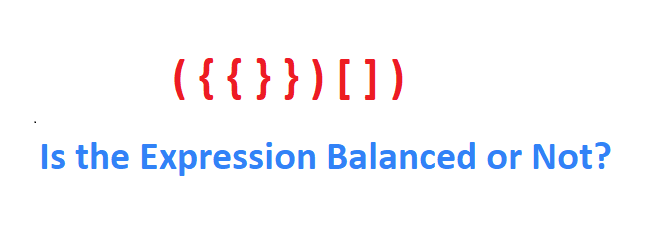
Thế giới ngày càng gắn liền với những sản phẩm công nghệ thông tin vì vậy nguồn nhân lực trong ngành này tăng lên mạnh trong những năm gần đây. Việc cũng dẫn đến sự ra đời và phát triển nhanh chóng của các công cụ hỗ trợ người dung lập trình, điển hình nhất là 2 kiểu công cụ: IDE(Intergrated Development Enviroment)[1] và Code Editor[2].

IDE bao gồm source code editor dung để viết mã, compiler hoặc interpreter để biên dịch hoặc thông dịch, debugger để hỗ trợ tìm lỗi. Điều này giúp cho người dung dễ dàng và thuận tiên hơn khi code và thực thi trực tiếp code trên công cụ.

Code Editor không tích hợp sẵn trình biên dịch hoặc trình thông dịch bên trong nó, nghĩa là 1muốn chạy được thực thi, người dung phải dung riêng compiler bên ngoài. Điều này chính là khác biệt chính giữa IDE và Text editor. Ví dụ muốn viết và thực thi một chương trình viết bằng ngôn ngữ C++[3], người dung có thể sử dụng IDE Visual Studio[4] để thực hiện luôn, còn nếu sử dựng Visual Studio Code[5] để viết mã thì sau đó ta phải dùng thêm một compiler bên ngoài của C/C++ như g+[6] để biên dịch.



Ngày nay, nhiều code editor cũng đã có them những extensions[7] đi kèm và tiện dụng không kém gì IDE. Một trong số đó là tính năng debug, tính năng này tìm và phát hiện những lỗi sai về thư viện, về cú pháp, về dấu ngoặc… Với những kiến thức được học trong học phần Cấu trúc dữ liệu và giải thuật và khả năng hiện tại, nhóm đã quyết định mô phỏng lại tính năng tìm lỗi về dấu ngoặc(Balanced Brackets) để phục vụ cho bài tập lớn lần này.



1. Định nghĩa bài toán
2. Giới thiệu về bài toán

Bài toán “Cân bằng dấu ngoặc”(balanced parentheses[8]) là một bài toán kinh điển trong khoa học máy tính, liên quan đến việc xác định xem một chuỗi dấu ngoặc có cân bằng hay không. Trong ngữ cảnh này, “cân bằng” có nghĩa là mỗi dấu ngoặc mở đều có một dấu ngoắc đóng tương ứng, và chúng phải được ghép cặp theo đúng thứ tự. Ví dụ chuỗi “{[()]}” là cân bằng trong khi chuỗi “{()]” không phải.

|  |  |
| --- | --- |
| Balanced expression | Unbalanced Expression |
| {([])} | {[)} |
| {a+b} | {a+b |
| [(c-d)\*e] | [c-d)\*e] |

Việc kiểm tra xem một chuỗi dấu ngoặc có cân bằng hay không rất quan trọng trong nhiều lĩnh vực của khoa học máy tính, như thiết kế trình biên dịch, phân tích cú pháp và xử lí văn bản. Giải pháp cho vấn đề này phần lớn đều thực hiện thông qua việc sử dụng một cấu trúc dữ liệu dạng stack, có thể lưu trữ thông tin về các dấu ngoặc trong chuỗi và đói tượng tương ứng của chúng.

Bên cạnh đó, theo như Leetcode[9], đây cũng là một câu hỏi phỏng vấn phổ biến đối với ứng viên xin việc trong lĩnh vực khoa học máy tính vì nó đòi hỏi hiểu biết sâu sắc về các cấu trúc dữ liệu và thuật toán cơ bản, cũng như khả năng tư duy sáng tạo và giải quyết vấn đề.

1. Mô hình hóa bài toán theo dạng đầu vào và đầu ra

Nhóm đã xác định đầu vào và đầu ra của bài toán như sau

Input:

* File code c/c++ có đuôi .cpp hoặc .c

Output:

* Balanced or Not balanced
* If Balanced: phân bố các đoạn code (code block) được sắp xếp theo số dòng, đoạn code có nhiều dòng nhất và đồ thị phân bố các đoạn code theo số dòng
* If Not Balanced: các dấu ngoặc có thể chưa được cân bằng trong đoạn code và vị trí của chúng trong file .cpp, và the longest valid parentheses.

1. Bảng phân công nhiệm vụ và đánh giá mức độ hoàn thành

|  |  |  |
| --- | --- | --- |
| STT | Nhiệm vụ | Thực hiện |
| 1 | Lập trình Stack cơ bản | Nguyễn Tuấn Anh  Nguyễn Đức Lực |
| 2 | Thuật toán CheckingBalanced | Nghiêm Văn Quang  Nguyễn Tuấn Anh |
| 3 | Thao tác với file, chuyển đổi file.cpp hoặc .c thành string | Nghiêm Văn Quang  Nguyễn Đức Lực |
| 4 | Viết file utils.cpp hỗ trợ cho file main, các tính năng khác của chương trình(vẽ đồ thị, xác định vị trí thừa ngoặc) | Nghiêm Văn quang |
| 5 | Báo cáo(chưa xong) | Cả nhóm |

1. Phương pháp lựa chọn
2. Cấu trúc dữ liệu

Ở phần này , nhóm sẽ trình bày các cấu trúc dữ liệu chính và phụ được sử dụng chính cho bài toán Balanced Parathenses.

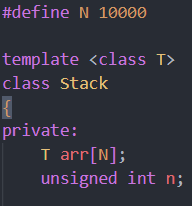
1. Stack(LIFO)

Stack là kiểu cấu trúc dữ liệu cơ bản trong khoa học máy tính cho phép quản lý thông tin một cách hiệu quả. Nó cho phép lưu trữ các phần tử và tổ chức dựa trên nguyên tắc “vào sau ra trước”(LIFO), trong đó phần tử được them vào gần nhất là phần tử đầu tiên bị xóa bỏ khi cần.

Một Stack có thể được hình dung như một chồng đĩa. Mỗi đĩa mới được them vào đầu của Stack, và chỉ có thể loại bỏ được đĩa đầu tiên ở đỉnh của Stack. Với những đặc tính trên, Stack rất hữu ích trong bài toán Balanced Parathenses. Vì vậy nhóm đã chọn loại cấu trúc dữ liệu này để mô phỏng chính thuật toán.

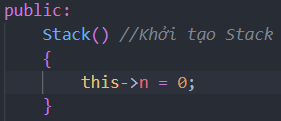
Dưới đây là các thuộc tính và phương thức của một Class Stack tổng quát được viết bằng ngôn ngữ C++ :

* Thuộc tính:

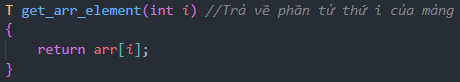


Thuộc tính đầu tiên là một mảng có kiểu dữ liệu là T với N=1000 phẩn tử, tiếp đó là một số nguyên không dấu n để lưu trữ số phần tử của Stack.

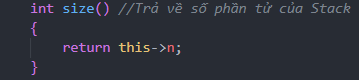
* Các phương thức:
* Khởi tạo một Stack rỗng



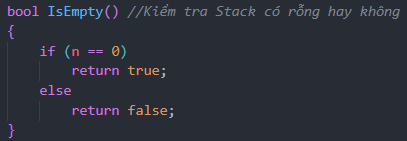
* Trả về phần tử thử i của mảng arr



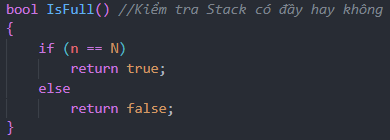
* Trả về số phần tử của Stack



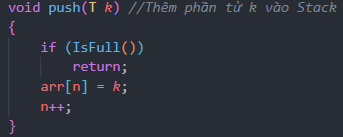
* Kiểm tra Stack có trống hay không



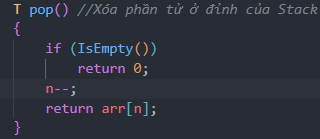
* Kiểm tra Stack có đầy hay không



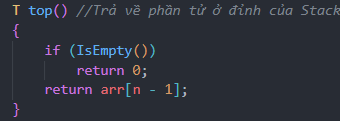
* Thêm một phần tử vào Stack



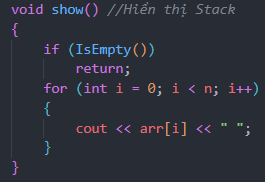
* Xóa phần tử ở đỉnh của Stack



* Trả về phần tử ở đỉnh của Stack



* Hiển thị giá trị của các phần tử trong Stack



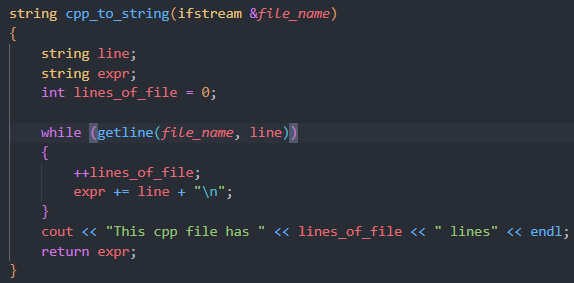
1. Các kiểu dữ liệu khác:

Ngoài việc kiểm tra xem file code đã đầy đủ việc đóng mở ngoặc hay chưa, nhóm còn hướng đến các chức năng khác của chương trình, vì vậy nhóm có sử dụng thêm một số các cấu trúc dữ liệu khác đã tích hợp trong C++:

* list[9]
* pair[10]
* vector[11]
* map[12]

1. Tiền xử lí

Với mục tiêu là kiểm tra xem 1 file code có đuôi .c hoặc .cpp đã đảm bảo cân bằng về các dấu ngoặc hay chưa, nhóm đã chuyển các file này thành các string để tiện xử lí và thực thi giải thuật.



Bên cạnh đó nhóm còn viết các hàm với nhiều chức năng khác như đánh dấu và lưu trữ các code block( nằm trong cặp ngoặc ‘{}’ hoàn chỉnh), chuyển đổi lẫn nhau giữa các dạng giữ liệu như vector, map để tiện cho việc xử lí. Tất cả các hàm chức năng này đều được lưu trong file utils.cpp, sẽ được nhắc đến và đề cập khi nêu cấu trúc tổ chức của thư mục.

1. Giải thuật

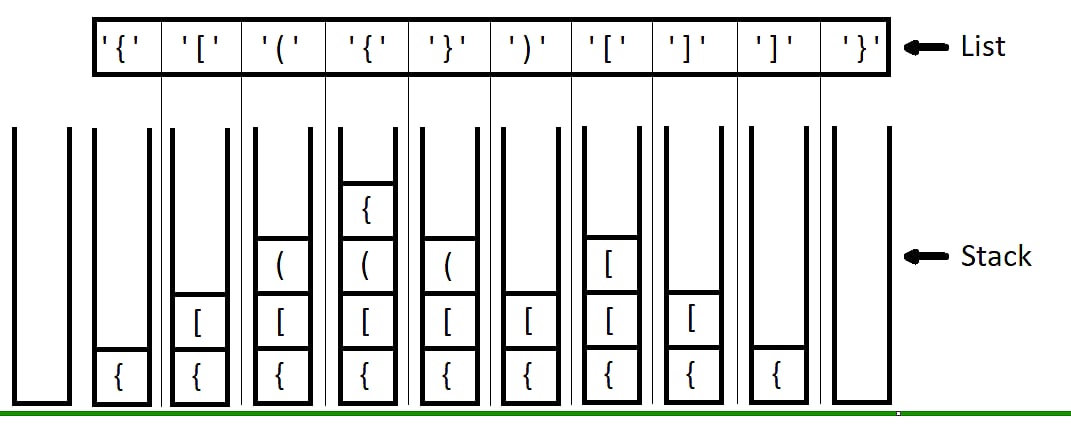
Bài toán được sử dụng nhiều giải thuật nhưng trong đó có giải thuật chính dựa trên các thuộc tính và phương thức của Stack để kiểm tra xem đoạn code đã được đầy đủ đóng mở ngoặc chưa. Bên cạnh đó nhóm cũng dùng Stack để tìm xem số dấu ngoặc hợp lệ dài nhất có thể trong trường hợp file code chưa được câng bằng ngoặc và giải thuật sắp xếp các Code Block theo thứ tự tăng dần dựa trên số dòng code.

1. Giải thuật kiểm tra cân bằng đóng mở ngoặc(Checking balanced parentheses)

* Ý tưởng:

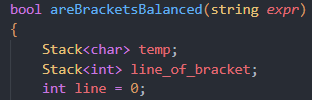
Duyệt một chuỗi các kí tự và đặt tất cả các dấu ngoặc mở vào trong Stack. Bất cứ khi nào gặp một dấu ngoặc đóng trong chuỗi, hãy kiểm tra xem đỉnh của Stack có phải là dấu ngoắc mở cùng loại hay không. Nếu đúng thì lấy phần tử đó ra khỏi Stack và tiếp tục vòng duyệt, cuối cùng nếu Stack trống, điều đó nghĩa là tất cả các dấu ngoặc đã được đóng mở đầy đủ đúng cách. Ngược lại, nếu Stack không rỗng thì các dấu ngoặc chưa được câng bằng.

* Minh họa cho thuật toán:



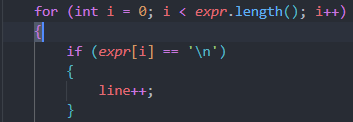
* Các bước thực hiện và triển khai trên ngôn ngữ C++:

Bước 1: Khai báo một Stack ký tự(char) tên là temp, một Stack kiểu int line\_of\_bracket để lưu trữ vị trí của các dấu ngoặc, một biến kiểu nguyên line để lưu trữ số dòng hiện tại.

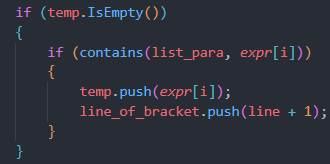


Bước 2: Duyệt string exp đầu vào:

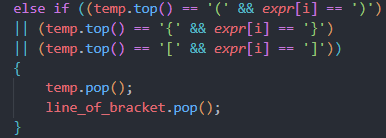
* Nếu kí tự hiện tại của exp là xuống dòng, thì biến line tăng thêm 1.

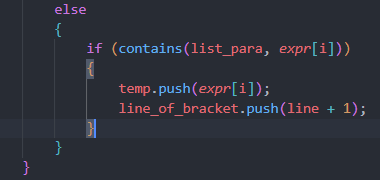


* Nếu kí tự hiện tại của exp là một dấu mở ngoặc(‘(’ hoặc ‘{‘ hoặc ‘[‘) thì đầy nó vào Stack temp đồng thời đẩy line+1 vào Stack line\_of\_bracket.

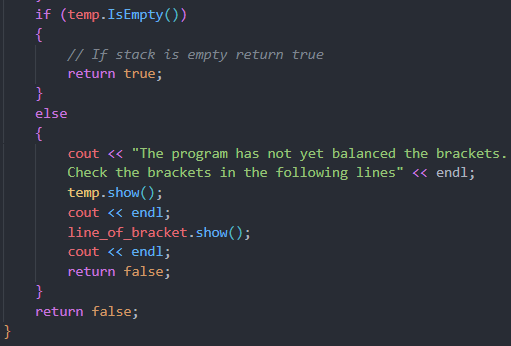


* Nếu kí tự hiện tại của exp là một dấu đóng ngoặc(‘)’,’}’,’]’) và nó tương thích với dấu mở ngoặc là đỉnh của Stack temp thì lấy ra đỉnh của Stack temp và đỉnh của Stack line\_of\_bracket.



* Còn lại, các dấu ngoặc chưa được cân bằng, kiểm tra xem kí tự hiện tại của exp có phải là dấu ngoặc hay không, nếu có thì tiếp tục cho vào Stack temp và đầy line +1 vào Stack line\_of\_bracket.
* 

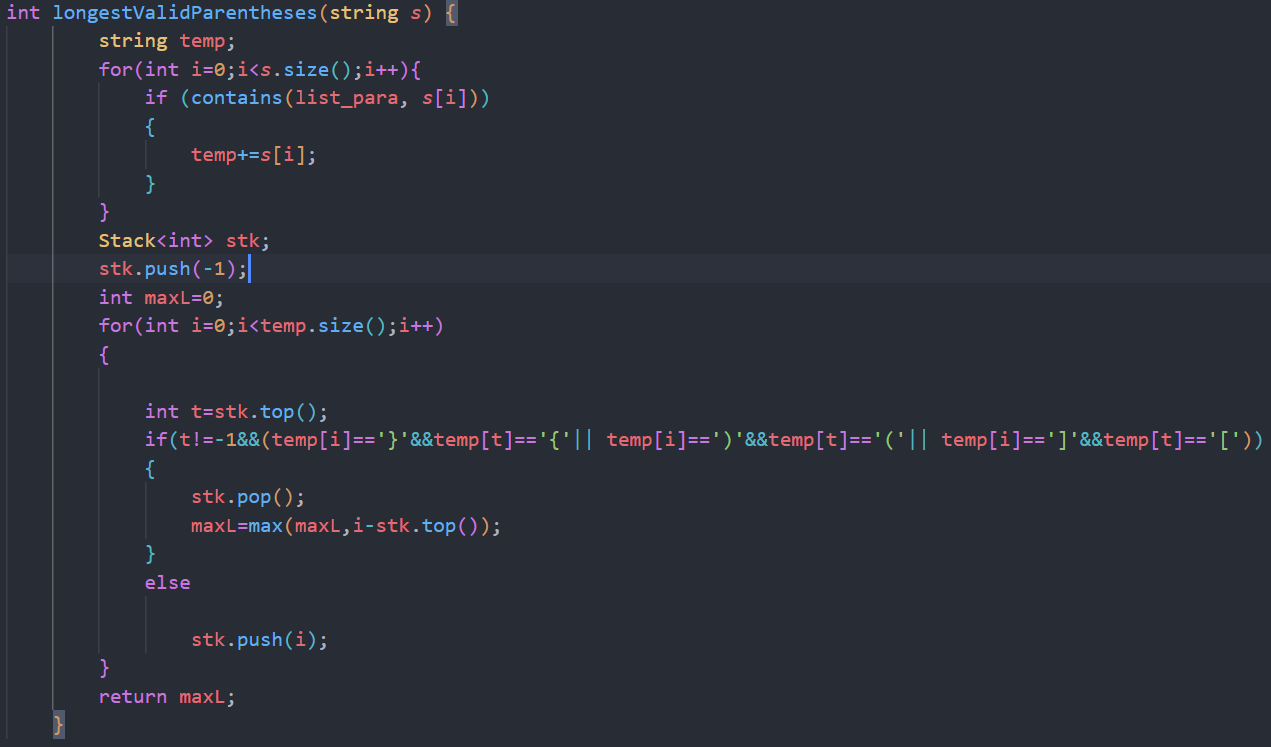
Bước 3:Sau khi duyệt xong chuỗi exp, nếu Stack temp không rỗng thì chuỗi exp đầu vào không được cân bằng về ngoặc.



1. Giải thuật tìm số ngoặc hợp lệ lớn nhất( The longest valid parentheses)
2. Ý tưởng:

Sử dụng Stack có kiểu số nguyên nhằm tìm ra số ngoặc hợp lệ lớn nhất.

1. Cài đặt trên C++:



1. Minh họa ví dụ:

Input: “({()}]”

Output: 4. Vì số ngoặc hợp lệ lớn nhất là {()}

* Trước vòng lặp:

stk:

|  |
| --- |
| -1 |

🡨Chiều vào

maxL=0

* i=0, s[i]=s[0]=’(‘, t=-1,s[t]=’’

stk:

|  |  |
| --- | --- |
| -1 | 0 |

maxL=0

* i=1, s[i]=s[1]=’{‘, t=0,s[t]=’(’

stk:

|  |  |  |
| --- | --- | --- |
| -1 | 0 | 1 |

maxL=0

* i=2, s[i]=s[2]=’(‘, t=1,s[t]=’{’

stk:

|  |  |  |  |
| --- | --- | --- | --- |
| -1 | 0 | 1 | 2 |

maxL=0

* i=3, s[i]=s[3]=’)‘, t=2,s[t]=’(’

stk:

|  |  |  |
| --- | --- | --- |
| -1 | 0 | 1 |

maxL=max(0,3-1)=2

* i=4, s[i]=s[4]=’}‘, t=1,s[t]=’}’

stk:

|  |  |
| --- | --- |
| -1 | 0 |

maxL=max(2,4-0)=4

* i=5, s[i]=s[5]=’]‘, t=-1,s[t]=’(’

stk:

|  |  |  |
| --- | --- | --- |
| -1 | 0 | 5 |

maxL=4

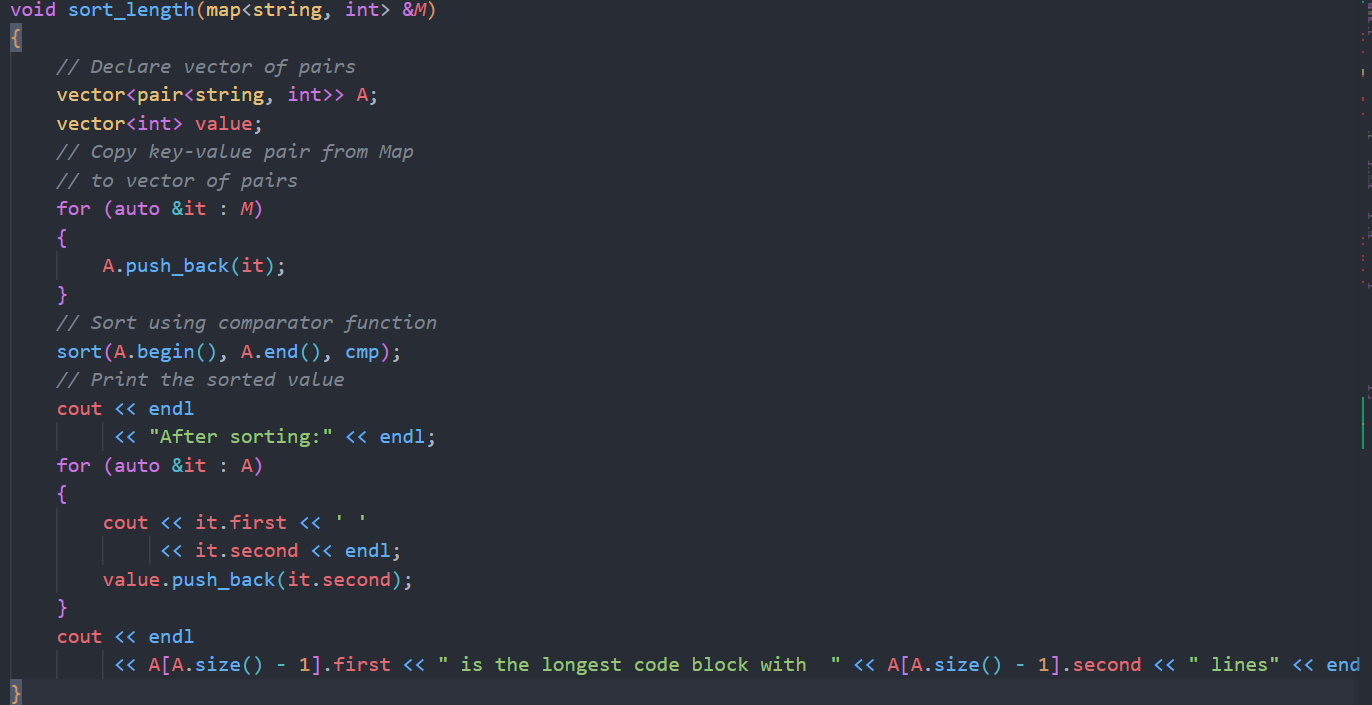
1. Giải thuật sắp xếp các Code block tăng dần dự trến số dòng:
2. Đặt vấn đề

Sau khi hoàn thành việc kiểm tra xem file code đầu vào đã là đầy đủ về ngoặc chưa, nhóm tiếp tục phát triển các chức năng khác của chương trình. Một trong số đó là việc sắp xếp các Code Block theo thứ tự tăng dần dựa trên số dòng của mỗi Code Block.

Sau khi đoạn file code đầu vào đã được kiểm tra và cân bằng về các dấu ngoặc, các code block trong file đó sẽ được lưu trữ dưới dạng dữ liệu map, bao gồm 2 thuộc tính là key và value. Trong đó key sẽ cho biết thông tin đó vị trí của Code Block, value sẽ cho biết số dòng có trong Code Block đó.



Việc sắp xếp lại các code block này theo tứ tự tăng dần trên cấu trúc dữ liệu map là một khó khăn lớn đối với nhóm, vì vậy nhóm quyết định sẽ sử dụng hàm sort() trong thư viện stl\_algo.h và tìm hiểu xem hàm sort() này dựa trên giải thuật sắp xếp nào. Theo đó hàm sort() này được dựa trên giải thuật insertion sort.

1. Cài đặt trên C++: 

1. Triển khai cài đặt
2. Ngôn ngữ lập trình và thư viện
   1. Ngôn ngữ lập trình:

Để tiện cho việc cài đặt và hiểu bản chất của các cấu trúc dữ liệu và giải thuật cũng như ưu tiên thời gian thực thi, nhóm quyết định chọn ngôn ngữ lập trình C++, một ngôn ngữ đã được học ở kì trước, làm ngôn ngữ chính cho việc minh họa và thực thi. Vì là một ngôn ngữ đã ra mắt và phát triển từ rất lâu nên C++ cũng có một vài hạn chế về các các thư viện chức năng cũng như tạo giao diện đồ họa. Vì vậy nhóm quyết định mô phỏng và thực thi chương trình đơn thuần và chủ yếu trên cmd[].

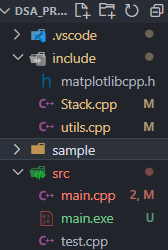
* 1. Thư viện

Trong project lần này, các thư viện sau đã được sử dụng:

* iostream[]: hỗ trợ việc nhập xuất, thao tác với file.
* list[]: hỗ trợ cấu trúc dữ liệu linked list.
* string[]: hỗ trợ kiểu dữ liệu string.
* map[]: hỗ trợ cấu trúc dữ liệu map.
* vector[]: hỗ trợ cấu trúc dữ liệu vector
* matplotlibcpp[]: một thư viện hỗ trợ vẽ đồ thị sừ dụng Matplotlib[] của python thông qua c++.
* algorithm[]:hỗ trợ hàm sort() cho cấu trúc dữ liệu map

1. Tổ chức CT và đóng gói
   1. Tổ chức chương trình

Chương trình bao gồm file thực thi chính có tên là main.cpp nằm trong folder src, bên cạnh đó trong thư mục include gồm các file có đuôi .h và .cpp bao gồm các thư viện và hàm hỗ trợ cho chương trình main.



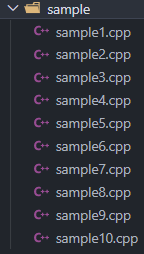
* 1. Đóng gói

Nhóm sử dụng g++ để đóng gói chương trình thành file .exe. Cú pháp đóng gói:

g++ src\main.cpp -o src\main.exe -std=c++11 -I C:\Users\user\AppData\Local\Programs\Python\Python310\include -I include -I C:\Users\user\AppData\Local\Programs\Python\Python310\Lib\site-packages\numpy\core\include -L C:\Users\user\AppData\Local\Programs\Python\Python310\libs -lpython310

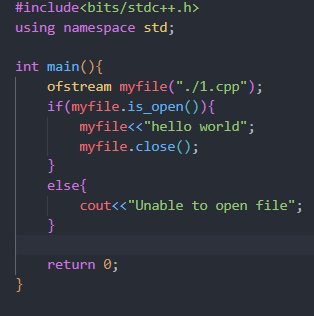
1. Kết quả thực nghiệm
2. Dữ liệu

Dữ liệu là các file có đuôi cpp trong thư mục sample

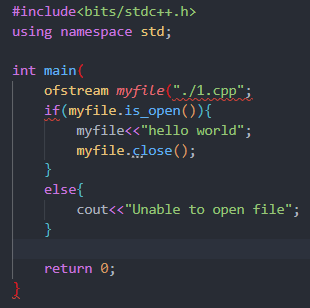


Nhóm thu thập được hơn 100 sample từ chính các đoạn code các thành viên viết trong suốt quá trình từ khi học tập môn KTLT C/C++. Ở đây nhóm chỉ mới upload lên 10 samples với đủ các trường hợp bao quát: đầy đủ đóng mở ngoặc, không đầy đủ dấu ngoặc để kiểm thử chương trình.

Sample1.cpp:



Sample2.cpp:



1. Các kết quả thử nghiệm
2. Cách sử dụng chương trình

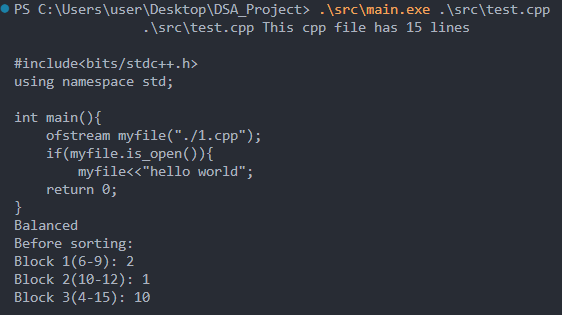
.\src\main.exe duong\_dan\_den\_sample

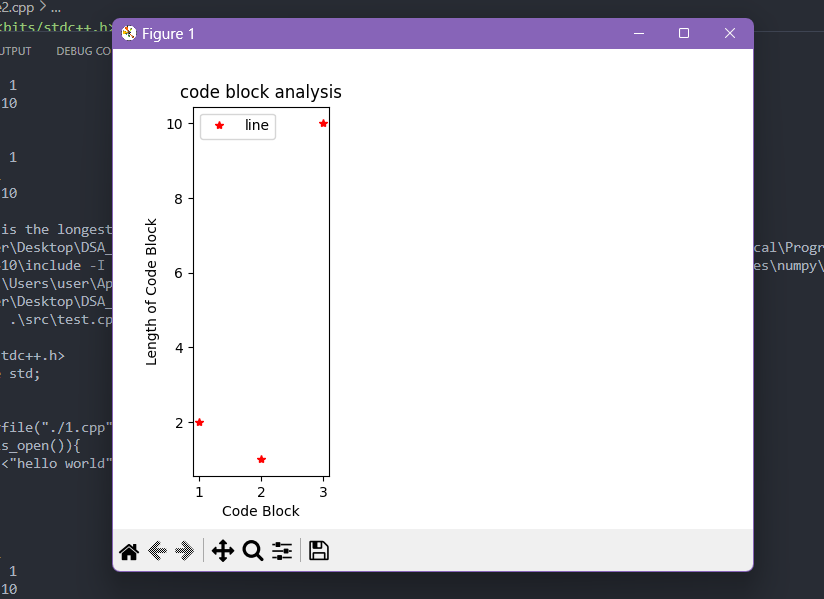
Ví dụ:

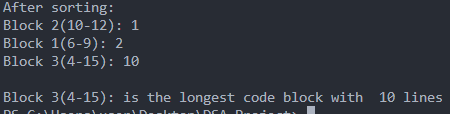
.\src\main.exe .\sample\sample1.cpp

.\src\main.exe .\sample\sample2.cpp

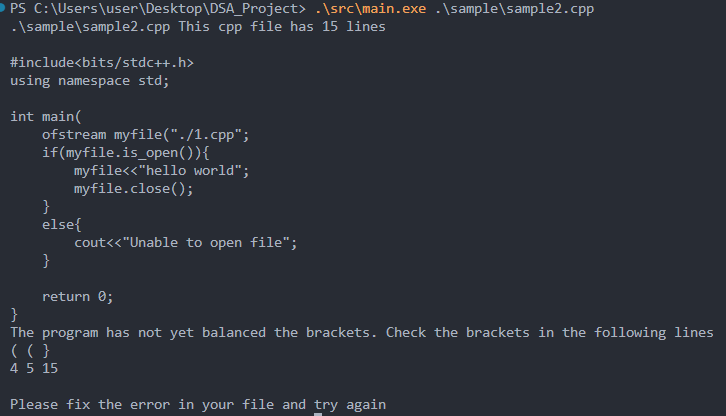
1. Kết quả thử nghiệm trên sample1 và sample2:

* Sample 1: 





* Sample 2:



1. Kết luận
2. Đánh giá về mức độ hoàn thành
3. Bài học rút ra
4. Các khó khăn khi học tập môn này