**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

*TRƯỜNG ĐIỆN – ĐIỆN TỬ*

A picture containing text, sign

Description automatically generated

**BÁO CÁO BÀI TẬP LỚN MÔN CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**Đề tài: Parenthesis checking using Stack**

**Giảng viên hướng dẫn:**  PGS.TS Trần Thị Thanh Hải

**Mã lớp :** 137284

**Sinh viên thực hiện:**

Nghiêm Văn Quang 20203547

Nguyễn Tuấn Anh 20200038

Nguyễn Đức Lực 20203495

**Mục Lục**

[I. Giới thiệu 3](#_Toc127574437)

[1. Lý do và động lực 3](#_Toc127574438)

[2. Định nghĩa bài toán 4](#_Toc127574439)

[a. Giới thiệu về bài toán 4](#_Toc127574440)

[b. Mô hình hóa bài toán theo dạng đầu vào và đầu ra 5](#_Toc127574441)

[3. Bảng phân công nhiệm vụ và đánh giá mức độ hoàn thành 5](#_Toc127574442)

[II. Phương pháp lựa chọn 6](#_Toc127574443)

[1. Cấu trúc dữ liệu 6](#_Toc127574444)

[a. Stack (LIFO) 6](#_Toc127574445)

[b. Các kiểu dữ liệu khác 9](#_Toc127574446)

[2. Tiền xử lí 9](#_Toc127574447)

[3. Giải thuật 10](#_Toc127574448)

[a. Giải thuật kiểm tra cân bằng đóng mở ngoặc (Checking balanced parentheses) 10](#_Toc127574449)

[b. Giải thuật tìm số ngoặc hợp lệ lớn nhất (The longest valid parentheses) 13](#_Toc127574450)

[c. Giải thuật sắp xếp các Code block tăng dần dự trến số dòng 15](#_Toc127574451)

[III. Triển khai cài đặt 16](#_Toc127574452)

[1. Ngôn ngữ lập trình và thư viện 16](#_Toc127574453)

[a. Ngôn ngữ lập trình 16](#_Toc127574454)

[b. Thư viện 16](#_Toc127574455)

[2. Tổ chức CT và đóng gói 16](#_Toc127574456)

[a. Tổ chức chương trình 16](#_Toc127574457)

[b. Đóng gói 17](#_Toc127574458)

[IV. Kết quả thực nghiệm 18](#_Toc127574459)

[1. Dữ liệu 18](#_Toc127574460)

[2. Các kết quả thử nghiệm 20](#_Toc127574461)

[a. Cách sử dụng chương trình 20](#_Toc127574462)

[b. Kết quả thử nghiệm trên sample1 và sample2 20](#_Toc127574463)

[c. Kết quả đánh giá 23](#_Toc127720498)

[V. Kết luận 24](#_Toc127574464)

[1. Đánh giá về mức độ hoàn thành 24](#_Toc127574465)

[2. Bài học rút ra 24](#_Toc127574466)

[3. Các khó khăn khi học tập môn này 24](#_Toc127574467)

[Tài liệu tham khảo 25](#_Toc127574468)

**Phụ Lục Hình Ảnh**

[Hình 1: Công cụ IDE(Intergrated Development Enviroment) và Code Editor 3](#_Toc127720490)

[Hình 2: Tính năng tìm lỗi về dấu ngoặc (Balanced Brackets) 4](#_Toc127720491)

Bảng 1: Ví dụ minh họa 4

[Hình 3: Khởi tạo mảng có dữ liệu T 6](#_Toc127720492)

[Hình 4: Trả về số phần tử của Stack 7](#_Toc127720493)

[Hình 5: Trả về phần tử thứ i 7](#_Toc127720494)

[Hình 6: Trả về số phần tử của Stack 7](#_Toc127720495)

[Hình 7: Kiểm tra Stack rỗng 7](#_Toc127720496)

[Hình 8: Kiểm tra Stack đầy 8](#_Toc127720497)

[Hình 9: Thêm một phần tử vào Stack 8](#_Toc127720498)

[Hình 10: Xóa phần tử ở đỉnh Stack 8](#_Toc127720499)

[Hình 11: Trả về phần tử ở đỉnh Stack 8](#_Toc127720500)

[Hình 12: Hiển thị giá trị các phần tử trong Stack 9](#_Toc127720501)

[Hình 13: Kiểm tra File .c hoặc .cpp và chuyển thành các string 9](#_Toc127720502)

[Hình 14: Minh họa thuật toán 10](#_Toc127720503)

[Hình 15: Khai báo Stack lưu trữ vị trí và số dòng hiện tại 11](#_Toc127720504)

[Hình 16: Duyệt string đầu vào 11](#_Toc127720505)

[Hình 17: Kiểm tra và thêm dấu mở ngoặc vào Stack 11](#_Toc127720506)

[Hình 18: Kiểm tra độ tương thích dấu mở và đóng ngoặc 12](#_Toc127720507)

[Hình 19: Kiểm tra dấu ngoặc chưa cân bằng 12](#_Toc127720508)

[Hình 20: Kiểm tra Stack rỗng 12](#_Toc127720509)

[Hình 21: Giải thuật tìm số ngoặc hợp lệ lớn nhất 13](#_Toc127720510)

[Hình 22: Output Code Block 15](#_Toc127720511)

[Hình 23: Giải thuật sắp xếp các Code Block tăng dần trên số dòng 15](#_Toc127720512)

[Hình 24: Tổ chức chương trình 16](#_Toc127720513)

[Hình 25: Chạy generatefile.exe 18](#_Toc127720514)

[Hình 26: Chạy hoàn tất 18](#_Toc127720515)

[Hình 27: Sample1 19](#_Toc127720516)

[Hình 28: Sample2 19](#_Toc127720517)

[Hình 29, 30, 31: Kết quả thử nghiệm trên Sample1 20](#_Toc127720518)

[Hình 32: Kết quả thử nghiệm trên Sample2 21](#_Toc127720521)

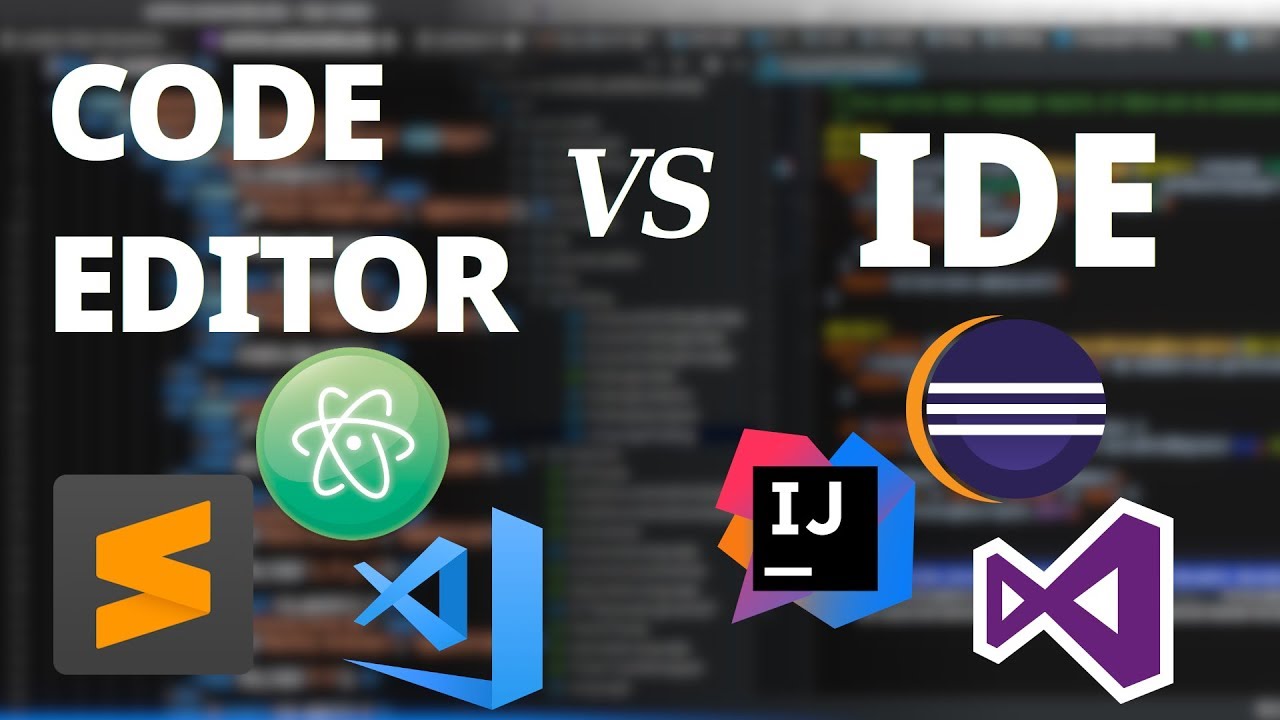
Bảng 2: Kết quả đánh giá 24

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].

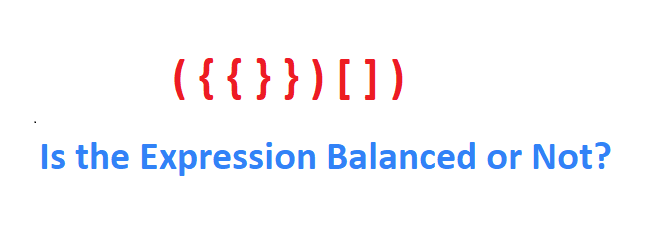
IDE bao gồm source code editor dung để viết mã, compiler hoặc interpreter để biên dịch hoặc thông dịch, debugger để hỗ trợ tìm lỗi. Điều này giúp cho người dung dễ dàng và thuận tiên hơn khi code và thực thi trực tiếp code trên công cụ.

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.



Hình 1: Công cụ IDE(Intergrated Development Enviroment) và Code Editor

Ngày nay, nhiều code editor cũng đã có thêm những extensions 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.



Hình 2: Tính năng tìm lỗi về dấu ngoặc (Balanced Brackets)

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

Bài toán “Cân bằng dấu ngoặc”(balanced parentheses[7]) 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] |

**Bảng 1: Ví dụ minh họa bài toán**

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[8], 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 | Thu thập data | Cả nhóm |
| 2 | Lập trình Stack cơ bản | Nguyễn Tuấn Anh  Nguyễn Đức Lực |
| 3 | Thuật toán CheckingBalanced | Nghiêm Văn Quang  Nguyễn Tuấn Anh |
| 4 | Thao tác với file, chuyển đổi file.cpp hoặc .c thành string | Nghiêm Văn Quang  Nguyễn Đức Lực |
| 4 | Gộp các phần code, 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 | Cả nhóm |
| 6 | Thuyết trình | Nghiêm Văn Quang |

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.

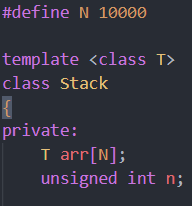
1. ***Stack (LIFO)***

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 thêm 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 thêm 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++ :

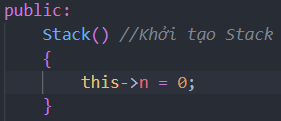
* Thuộc tính:



Hình 3: Khởi tạo mảng có dữ liệu T

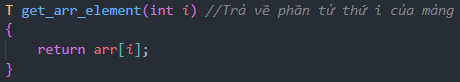
Thuộc tính đầu tiên là một mảng có kiểu dữ liệu là T với N = 10000 phẩn tử, tiếp đó là một số nguyên không dấu n để lưu trữ số phần tử của Stack.

* Các phương thức:
* Khởi tạo một Stack rỗng



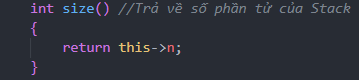
Hình 4: Trả về số phần tử của Stack

* Trả về phần tử thử i của mảng arr[]



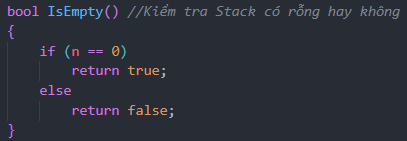
Hình 5: Trả về phần tử thứ i

* Trả về số phần tử của Stack



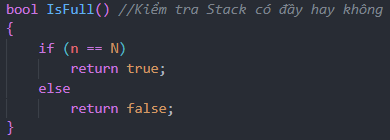
Hình 6: Trả về số phần tử của Stack

* Kiểm tra Stack có trống hay không



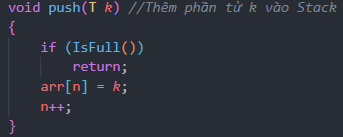
Hình 7: Kiểm tra Stack rỗng

* Kiểm tra Stack có đầy hay không



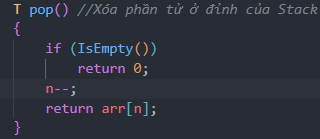
Hình 8: Kiểm tra Stack đầy

* Thêm một phần tử vào Stack



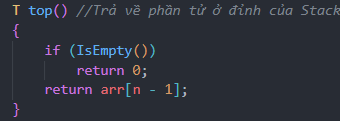
Hình 9: Thêm một phần tử vào Stack

* Xóa phần tử ở đỉnh của Stack



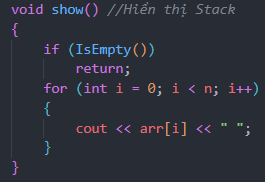
Hình 10: Xóa phần tử ở đỉnh Stack

* Trả về phần tử ở đỉnh của Stack



Hình 11: Trả về phần tử ở đỉnh Stack

* Hiển thị giá trị của các phần tử trong Stack



Hình 12: Hiển thị giá trị các phần tử trong Stack

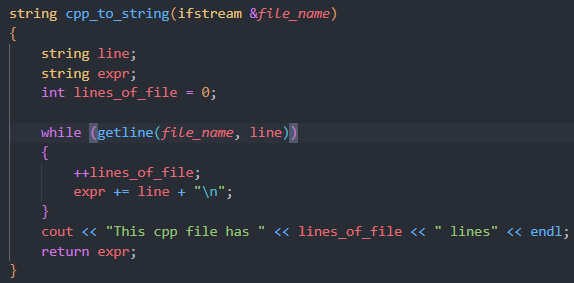
1. ***Các kiểu dữ liệu khác:***

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.



Hình 13: Kiểm tra File .c hoặc .cpp và chuyển thành các string

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**

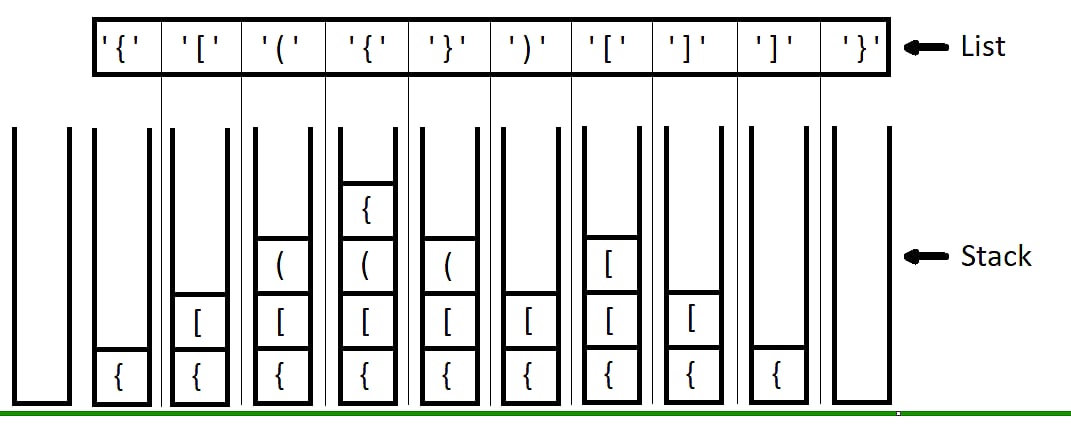
Bài toán được sử dụng nhiều giải thuật nhưng trong đó có giải thuật chính dựa trên các thuộc tính và phương thức của Stack để kiểm tra xem đoạn code đã được đầy đủ đóng mở ngoặc chưa. Bên cạnh đó nhóm cũng dùng Stack để tìm xem số dấu ngoặc hợp lệ dài nhất có thể trong trường hợp file code chưa được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

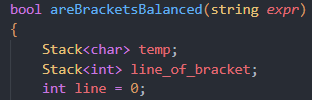
* Minh họa cho thuật toán:



Hình 14: Minh họa thuật toán

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

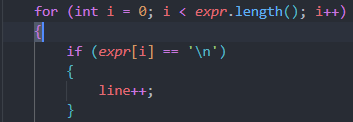
Bước 1: Khai báo một Stack ký tự(char) tên là temp, một Stack kiểu int line\_of\_bracket để lưu trữ vị trí của các dấu ngoặc, một biến kiểu nguyên line để lưu trữ số dòng hiện tại.



Hình 15: Khai báo Stack lưu trữ vị trí và số dòng hiện tại

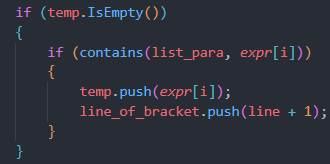
Bước 2: Duyệt string exp đầu vào:

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



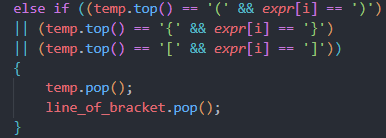
Hình 16: Duyệt string đầu vào

* 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.



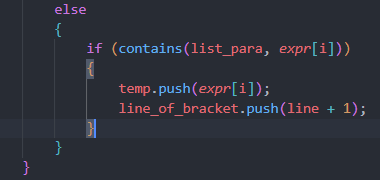
Hình 17: Kiểm tra và thêm dấu mở ngoặc vào Stack

* 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.



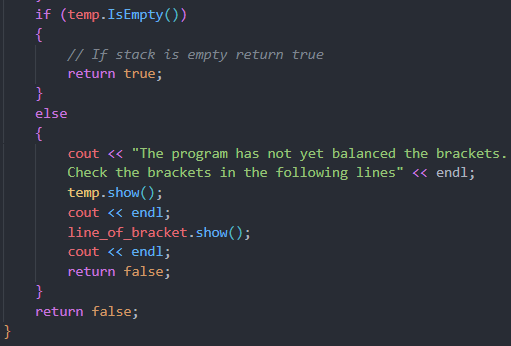
Hình 18: Kiểm tra độ tương thích dấu mở và đóng ngoặc

* 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.



Hình 19: Kiểm tra dấu ngoặc chưa cân bằng

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



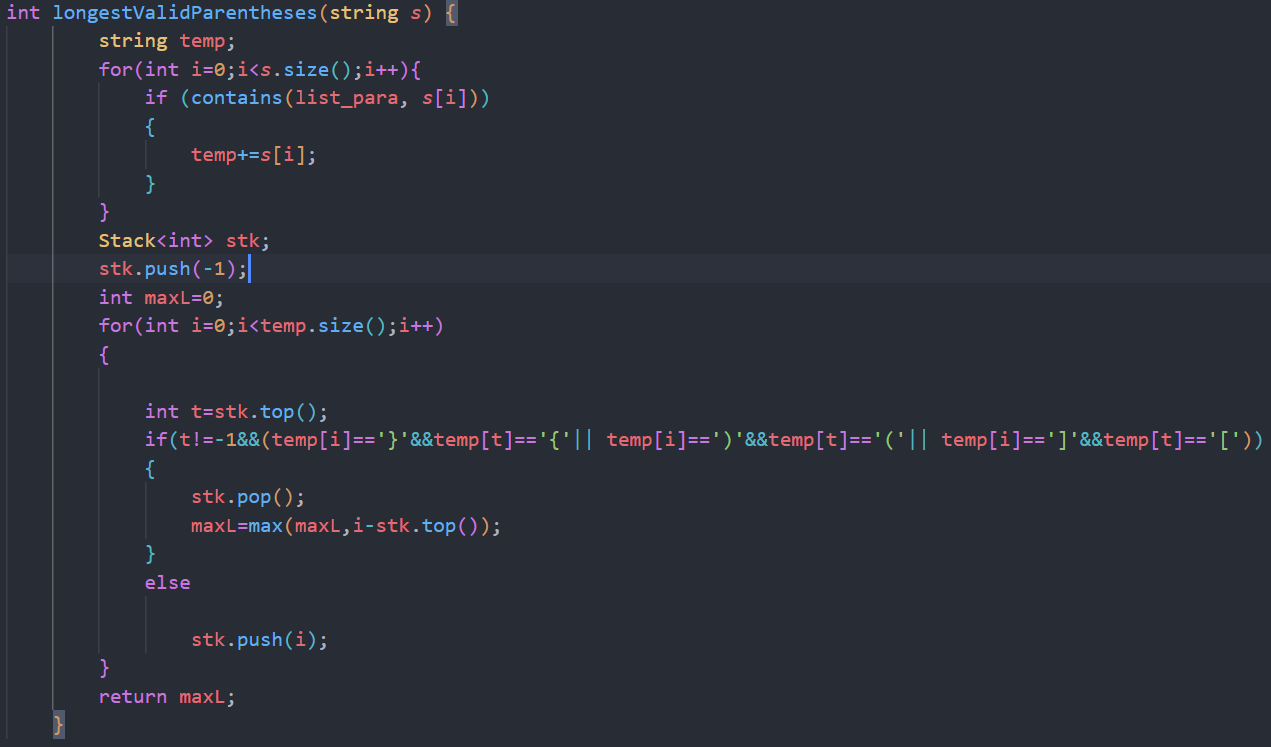
Hình 20: Kiểm tra Stack rỗng

*Độ phức tạp giải thuật: O(n)*

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++:



Hình 21: Giải thuật tìm số ngoặc hợp lệ lớn nhất

*Độ phức tạp giải thuật: O(n)*

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.

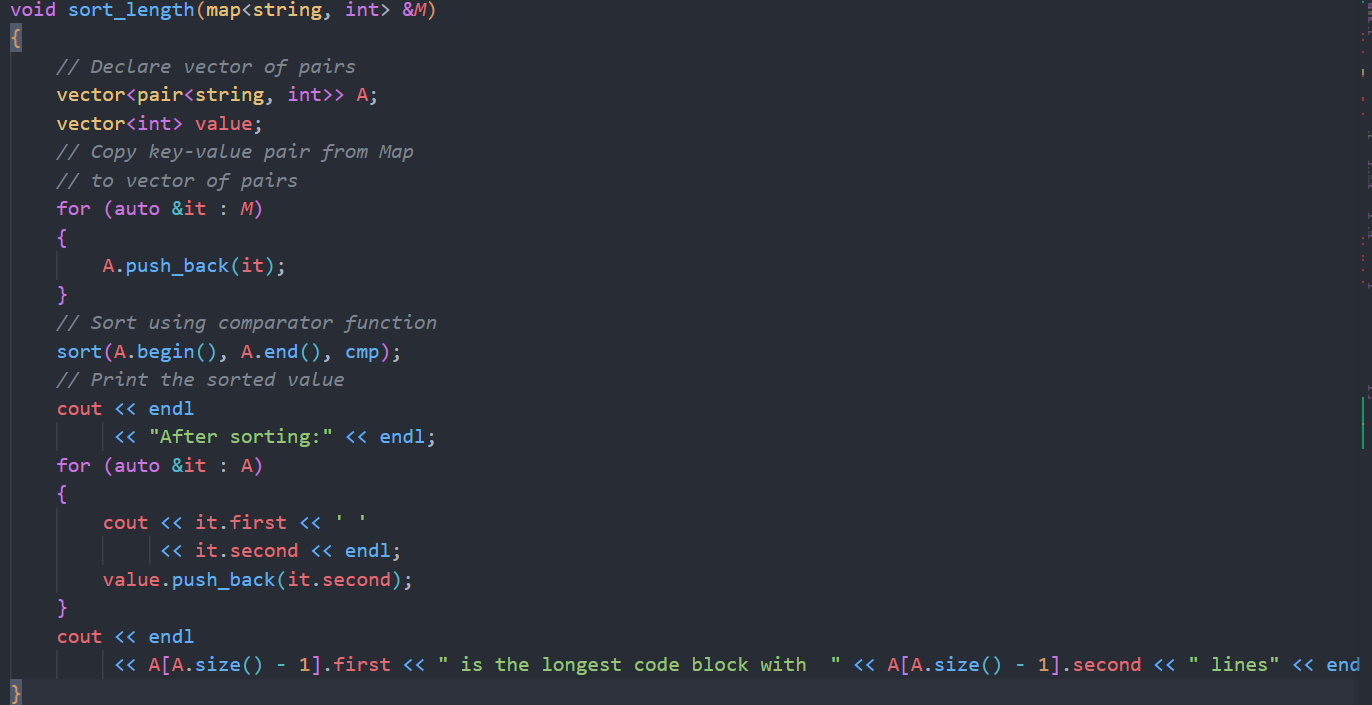
Sau khi đoạn file code đầu vào đã được kiểm tra và cân bằng về các dấu ngoặc, các code block trong file đó sẽ được lưu trữ dưới dạng dữ liệu map, bao gồm 2 thuộc tính là key và value. Trong đó key sẽ cho biết thông tin đó vị trí của Code Block, value sẽ cho biết số dòng có trong Code Block đó.



Hình 22: Output Code Block

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+**



Hình 23: Giải thuật sắp xếp các Code Block tăng dần trên số dòng

*Độ phức tạp giải thuật: O(nlogn)*

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[13].

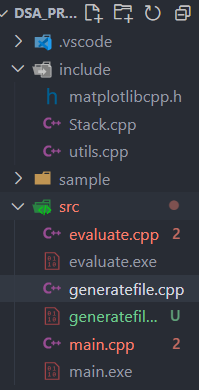
* 1. ***Thư viện***

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

* iostream[14]: hỗ trợ việc nhập xuất, thao tác với file.
* list: hỗ trợ cấu trúc dữ liệu linked list.
* string[15]: hỗ trợ kiểu dữ liệu string.
* map: hỗ trợ cấu trúc dữ liệu map.
* vector: hỗ trợ cấu trúc dữ liệu vector
* matplotlibcpp[16]: một thư viện hỗ trợ vẽ đồ thị sừ dụng Matplotlib[17] của Python thông qua C++.
* algorithm[18]: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, file generatefile.cpp để tạo ra các file sample, file evaluate.cpp để thực hiện đánh giá trên toàn bộ dataset và ghi vào 1 file text kết quả. 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.



Hình 24: Tổ chức chương trình

* 1. ***Đóng gói***

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

* Chương trình main

g++ src\main.cpp -o src\main.exe -std=c++17 -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

* Chương trình evaluate:

g++ src\evaluate.cpp -o src\evaluate.exe -std=c++17 -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

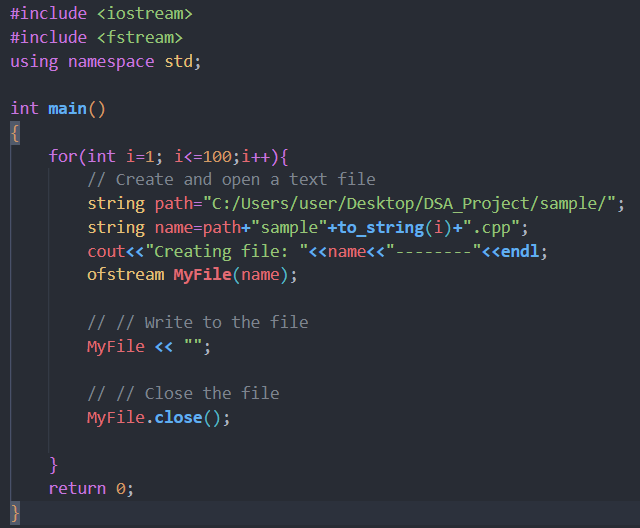
* Chương trình generatefile:

g++ src\generatefile.cpp -o src\generatefile.exe

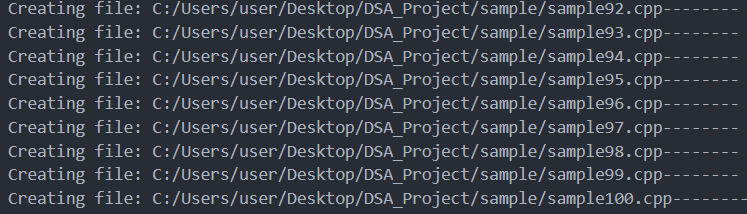
1. **Kết quả thực nghiệm**
2. **Dữ liệu**

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++. Đầu tiên, nhóm trưởng hoặc chạy file generatefile.exe theo cứ pháp:

generatefile.exe



Hình 25: Chạy generatefile.exe

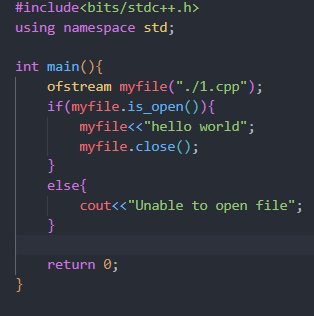


Hình 26: Chạy hoàn tất

Sau khi chạy file này xong, 100 files sample với tên được đánh theo thứ tự đã được sinh ra trong folder sample. Sau đó sẽ upload project lên trên GitHub[19], các thành viên update nội dung những file này rồi push lại lên GitHub.

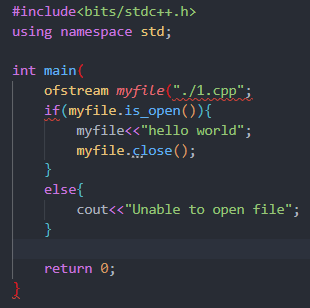
Cuối cùng nhóm thu được dataset với 100 mẫu, mỗi mẫu là 1 file có đuôi .cpp. Trong 100 files, có 50 files là những files code không mắc lỗi về dấu ngoặc(những file có số thự tự lẻ), 50 files còn lại là những files có nội dung tương tự 50 files trên nhưng lại có lỗi về dầu ngoặc(những file có số thứ tự chẵn). Việc thiết kế dataset như vậy giúp tiện và dễ so sánh hơn trong quá trình đánh giá.

Sample1.cpp:



Hình 27: Sample1

Sample2.cpp:



Hình 28: Sample2

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

Trước tiên, người dùng cần clone lại project này từ nguồn: [highquanglity/DSA\_Project (github.com)](https://github.com/highquanglity/DSA_Project)

* **Chạy kiểm nghiệm trên 1 file:**

./src/main.exe filename argument

Ví dụ:

./src/main.exe ./sample/sample1.cpp 0

Nếu muốn hiển thị đồ thị phân bố các đoạn code dựa trên số dòng:

\* Plot:

./src/main.exe ./sample/sample1.cpp 1

\* Bar:

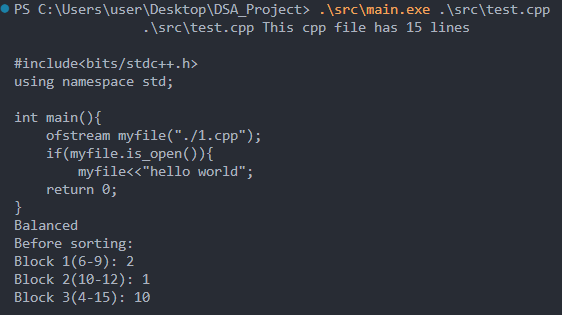
./src/main.exe ./sample/sample1.cpp 2

* **Đánh giá trên toàn bộ tập dataset và lưu vào 1 file text:**

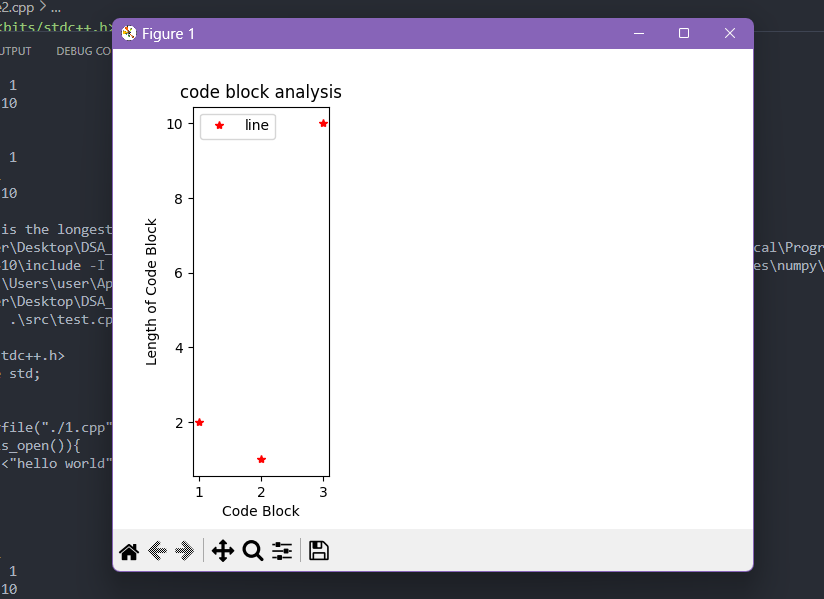
.\src\evaluate.exe > .\output.txt

1. ***Kết quả sample1 và sample2*:**

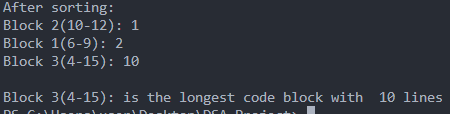
* Sample 1:



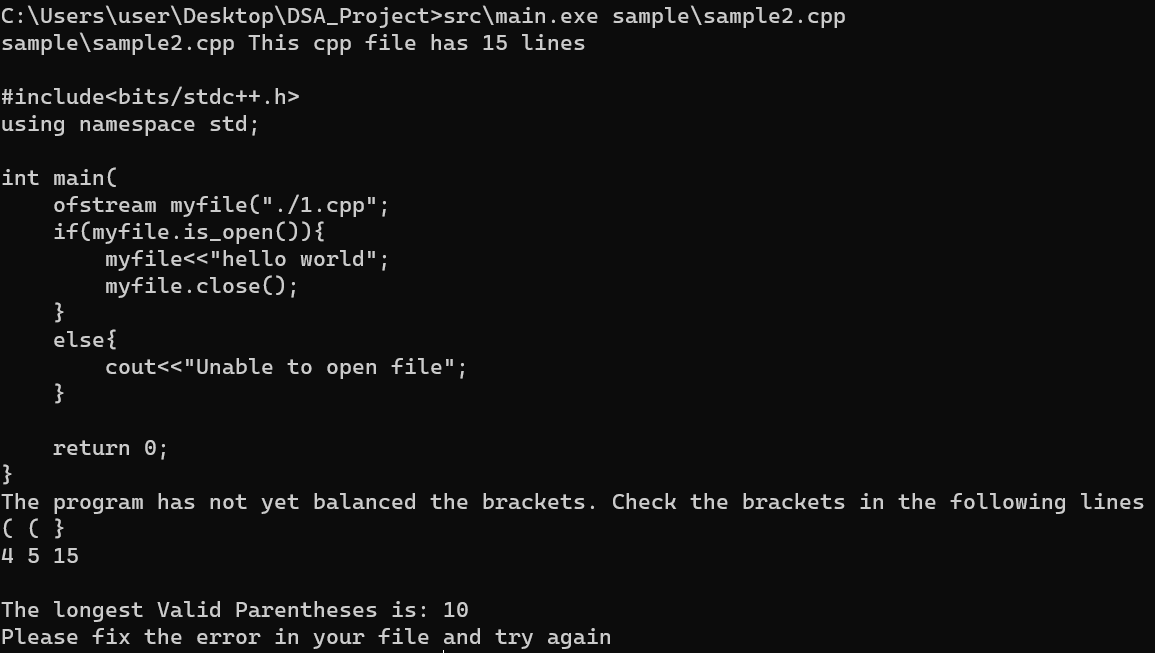
Hình 29: Kết quả thử nghiệm trên Sample1



Hình 30: Kết quả thử nghiệm trên Sample1



Hình 31: Kết quả thử nghiệm trên Sample1

* Sample 2: 

Hình 32: Kết quả thử nghiệm trên Sample2

1. ***Kết quả đánh giá***

Nhóm đã chạy và kiểm thử trên toàn bộ 100 samples. Với những samples không có lỗi về dấu ngoặc, chương trình chạy tốt, thực hiện đầy đủ và đúng các chức năng kiểm tra, vẽ đồ thị, sắp xếp và định vị các Codeblock dựa trên số dòng.

Còn với những samples có lỗi về dấu ngoặc, chương trình chạy tốt nhiệm vụ phát hiện và định vị những dấu ngoặc có vẫn đề, tìm ra số dấu ngoặc hợp lệ dài nhất có thể.

Sau đây là một vài những trường hợp nhóm chọn ra để làm so sánh, nhóm sẽ chỉ nêu kết quả chương trình về việc đã cân bằng ngoặc hay chưa, đoạn codeblock dài nhất trong trườn hợp cân bằng ngoặc, các dấu ngoặc còn thiếu và số ngoặc hợp lệ dài nhất có thể cho trường hợp chưa cân bằng ngoặc. Các đầu ra của các chức năng khác và các samples khác được lưu trong file output.txt.

|  |  |  |
| --- | --- | --- |
| Sample | Các dầu ngoặc có trong file | Kết quả của chương trình |
| 1 | (){()(()){()}{}} | Balanced  The longest code block: Block 3(4-15):11 lines |
| 2 | (((()){()}{}} | Unbalanced  ( ( }  4 5 15  The longest Valid Parentheses is: 10 |
| 3 | (){} | Balanced  The longest code block: Block 1(5-8):3 lines |
| 4 | ({ | Unbalanced  ( {  5 5  The longest Valid Parentheses is: 0 |
| 14 | (){)())()(())()}(){(}{()())}} | Unbalanced  { ) ) } { ( } { ) } }  5 12 13 18 20 22 24 26 28 32 35  The longest Valid Parentheses is: 8 |
| 20 | ()[](){([])}(){(){([]]{[][[][}}}(){[]}} | Unbalanced  { { ( ] { [ [ } } } }  15 16 17 17 17 19 20 21 22 23 32  The longest Valid Parentheses is: 14 |
| 26 | {}(){){(} | Unbalanced  { ) { ( }  9 26 26 31 32  The longest Valid Parentheses is: 4 |
| 42 | ({){(){}} | Unbalanced  ( { )  4 4 11  The longest Valid Parentheses is: 6 |
| 82 | ()()(){({})((()))(()){[]})(())()()()()()}  ()()()(())(()){(()()())(())}()} | Unbalanced  { ) } }  12 21 31 46  The longest Valid Parentheses is: 30 |
| 100 | (){(()){([]){}(()){(([])){([])()}}((()([]))  ((){[]})(()[[]])){()()}{(([])){([])()}}}  (()){}{()()}}(){(()){}}(){}(){()()}(){()  {()}()({{}})()()()()()}(){(){()}()()(){()}((()(){()}[()][()]} | Unbalanced  { ( ( }  94 115 115 127  The longest Valid Parentheses is: 148 |

**Bảng 2: Kết quả đánh giá**

1. **Kết luận**
2. **Đánh giá về mức độ hoàn thành**

Nhóm tự đánh giá đã hoàn thành những yêu cầu cơ bản của bài toán, nắm rõ và cài đặt được cấu trúc Stack. Trong quá trình làm việc nhóm cũng dựa trên các problems của LeetCode để hình thành các chức năng cho chương trình. Trong đó với bài toán Checking balanced parentheses using Stack, đây là một problem mức easy của LeetCode, còn với bài toán The longest valid parathenses, đây là vấn đề ở mức Hard của LeetCode.

1. **Bài học rút ra và hướng cải tiến chương trình trong tương lai**

Chắc hẳn code của nhóm vẫn chưa ‘clean’ và tối ưu nhất do thời gian chuẩn bị và làm bài tập lớn là không nhiều. Với một vài thành viên trong nhóm, việc tìm và đọc Document về các thư viện, các hàm vẫn là một khó khăn, điều này các thành viên phải tự cố gắng là đọc và code nhiều hơn. Trong lần này nhóm có sử dụng Git để quản lí project nhưng chưa hiệu quả khi mà commit vẫn chưa đầy đủ những thay đổi của project, đây sẽ là thứ nhóm phải cải thiện để tiến tới việc đủ khả năng làm các dự án lớn hơn và đi làm.

Thêm đó với dự án này, nhóm có thể cải tiến thành 1 ứng dụng có đồ họa trên máy tính và điện thoại, một trang web hay một API[20] để gần gũi, dễ dàng và thân thiện hơn cho người sử dụng.

1. **Các khó khăn khi học tập môn này**

CTDLvGT bản thân đã là một môn học tương đối khó với sinh viên ngành ĐTVT nói chung. Ở khóa của chúng em là từ k65 đổ đi, học phần CTDLvGT và học phần KTLC C/C++ được đổi chỗ cho nhau vì vậy ở kì trước chúng em được học và sử dụng C++ (Class) nhiều hơn nên khi vào học phần này, các code minh họa lại dựa trên phần lớn là C(struct) với các kiểu khai báo cấp phát động và một vài vấn đè liên quan đến con trả khá là khác cú pháp so với C++. Mong rằng học phần này sớm chấp nhập trình bày các giải thuật và minh họa các giải thuật trên nhiều ngôn ngữ khác nhau. Bên cạnh đó trong quá trình học tập, bọn em cũng không gặp quá nhiều khó khăn. Một phần là nhờ cách giảng dạy của cô Trần Thị Thanh Hải phù hợp với bọn em. Cô luôn đi vào những ví dụ, step by step mô phỏng giải thuật trước sau đó mới đi vào mã giải, việc này đã cho bọn em một cái nhìn trước nhất về giải thuật và biết giải thuật này làm gì, nắm rõ hơn các giải thuật trước khi sử dụng. Chúng em xin gửi lời cảm ơn đến cô rất nhiều vì đã giảng dạy và hướng dẫn chúng em trong quá trình làm và bảo vệ BTL môn học.

# 

# Tài liệu tham khảo:

[1]Wikipedia Contributors, “Integrated development environment,” *Wikipedia*. <https://en.wikipedia.org/wiki/Integrated_development_environment>

[2]Wikipedia Contributors, “Source-code editor,” *Wikipedia*. <https://en.wikipedia.org/wiki/Source-code_editor>

[3]Wikipedia Contributors, “C++,” *Wikipedia*. <https://en.wikipedia.org/wiki/C%2B%2B>

[4]Microsoft, “Visual Studio,” *Visual Studio*. <https://visualstudio.microsoft.com/>

[5]Microsoft, “Visual Studio Code,” *Visualstudio.com*, Apr. 14, 2016. <https://code.visualstudio.com/>

[6]“Compiling with g++,” *GeeksforGeeks*, Feb. 19, 2019. <https://www.geeksforgeeks.org/compiling-with-g-plus-plus/>

[7]“LeetCode - The World’s Leading Online Programming Learning Platform,” *LeetCode*. <https://leetcode.com/problems/valid-parentheses/>

[8]“LeetCode - The World’s Leading Online Programming Learning Platform,” *Leetcode.com*. <https://leetcode.com/>

[9]“Pair in C++ Standard Template Library (STL),” *GeeksforGeeks*. <https://www.geeksforgeeks.org/pair-in-cpp-stl/>

[10]“vector - C++ Reference,” *cplusplus.com*. <https://cplusplus.com/reference/vector/vector/>

[11]*Cplusplus.com*. <https://cplusplus.com/reference/map/map/>

[12]“cmd.exe,” *Wikipedia*. <https://vi.wikipedia.org/wiki/Cmd.exe>

[13]*Cplusplus.com*. <https://cplusplus.com/reference/iostream/>

[14]“list - C++ Reference,” *cplusplus.com*. <https://cplusplus.com/reference/list/list/>

[15]“string - C++ Reference,” *cplusplus.com*. <https://cplusplus.com/reference/string/string/>

[16]“Home — Matplotlib for C++ documentation,” *matplotlib-cpp.readthedocs.io*. <https://matplotlib-cpp.readthedocs.io/en/latest/>

[17]Matplotlib, “Matplotlib: Python plotting — Matplotlib 3.1.1 documentation,” *Matplotlib.org*, 2012. <https://matplotlib.org/>

[18]*Cplusplus.com*. <https://cplusplus.com/reference/algorithm/>

[19]GitHub, “GitHub,” *GitHub*. <https://github.com/>

[20]“API,” *Wikipedia*. <https://en.wikipedia.org/wiki/API>