ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

Cấu trúc dữ liệu và giải thuật - CO2003

Bài tập lớn 2

MÔ PHỎNG SYMBOL TABLE BẰNG CÂY SPLAY

Tác giả: ThS. Trần Ngọc Bảo Duy

TP. HỒ CHÍ MINH, THÁNG 08/2021

ĐẶC TẢ BÀI TẬP LỚN

Phiên bản 1.0

1 Chuẩn đầu ra

Sau khi hoàn thành bài tập lớn này, sinh viên ôn lại và sử dụng thành thục:

- Thiết kế và sử dụng đệ quy.
- Lập trình hướng đối tượng.
- Các cấu trúc dữ liệu cây.

2 Dẫn nhập

Symbol table (tạm gọi là bảng ghi đối tượng) là một cấu trúc dữ liệu quan trọng được tạo ra, duy trì và sử dụng bởi các trình biên dịch (compiler) nhằm lưu vết các ngữ nghĩa của các danh hiệu (identifiers) như lưu thông tin về tên (name), thông tin về kiểu (type), thông tin về tầm vực (scope), v.v...

Trong bài tập lớn trước, sinh viên đã được yêu cầu hiện thực một mô phỏng về bảng ghi đối tượng sử dụng cấu trúc dữ liệu danh sách. Tuy nhiên, tốc độ truy xuất dùng để kiểm tra của loại cấu trúc dữ liệu này là không cao. Khi chương trình nguồn có quá nhiều biến và lưu thành nhiều tầm vực khác nhau, chương trình trở nên thiếu hiệu quả. Mặt khác, trên thực tế, lập trình viên thường có xu hướng sử dụng các danh hiệu vừa khai báo hoặc các danh hiệu vừa sử dụng gần đây để tiếp tục sử dụng cho các dòng lệnh tiếp theo làm cho quá trình truy xuất các danh hiệu đó trở nên phổ biến hơn.

Trong bài tập lớn, sinh viên được yêu cầu hiện thực một mô phỏng về bảng ghi đối tượng sử dụng các cấu trúc dữ liệu **cây splay** để đáp ứng các nhược điểm nêu trên.

3 Mô tả

3.1 Đầu vào

Mỗi testcase là một tập tin đầu vào bao gồm các dòng lệnh tương tác với bảng ghi đối tượng. Các dòng lệnh mô tả được mô tả ở mục 3.5. Sinh viên có thể thấy được ví dụ về các testcase

thông qua mục này.

3.2 Yêu cầu

Để hoàn thành bài tập lớn này, sinh viên phải:

1. Đọc toàn bộ tập tin mô tả này.
2. Tải xuống tập tin initial.zip và giải nén nó. Sau khi giải nén, sinh viên sẽ nhận được các tập tin: main.h, main.cpp, SymbolTable.h, SymbolTable.cpp, error.h, trong đó, sinh viên không được phép sửa đổi các tập tin vì nó sẽ không nằm trong các danh mục dùng để nộp bài.
3. Sửa đổi các file SymbolTable.h, SymbolTable.cpp để hoàn thành bài tập lớn này nhưng đảm bảo hai yêu cầu sau:
   - Ít nhất có một lớp SymbolTable có phương thức đối tượng (instance method) public **void run(string testcase)** vì phương thức này là đầu vào cho lời giải. Đối với mỗi testcase, một đối tượng của lớp này được tạo và phương thức run của đối tượng này sẽ được gọi với tham số là tên file của tập tin văn bản (chứa một đoạn tương tác với bảng ghi đối tượng).
   - Chỉ có một lệnh **include** trong file SymbolTable.h là **#include "main.h"** và một include trong file SymbolTable.cpp đó là **#include "SymbolTable.h"**. Ngoài ra, không cho phép có một **#include** nào khác trong các tập tin này.
4. Sinh viên được yêu cầu thiết kế và sử dụng các cấu trúc dữ liệu dựa trên cấu trúc dữ liệu cây splay đã học.
5. Sinh viên phải giải phóng toàn bộ vùng nhớ đã xin cấp phát động khi chương trình kết thúc.

3.3 Thông tin một đối tượng trong bảng ghi

Thông tin một đối tượng (symbol) bao gồm:

1. Tên của danh hiệu (identifier)
2. Mức của khối mà danh hiệu thuộc về (level of block)
3. Kiểu tương ứng của danh hiệu (type)

Trên cây nhị phân tìm kiếm, việc so sánh khóa của các nút diễn ra thường xuyên. Để so sánh các khóa của một đối tượng, ta lần lượt so sánh:

- So sánh mức của khối mà danh hiệu thuộc về, nếu lớn hơn thì khóa của nút được xem là lớn hơn và ngược lại.
- Trong trường hợp mức của khối mà danh hiệu thuộc về bằng nhau, ta so sánh tên của danh hiệu (chuỗi) theo các bước:
  - Bằng nhau nếu tên của danh hiệu trùng nhau hoàn toàn.
  - Lớn hơn nếu ký tự đầu tiên không trùng nhau của chuỗi thứ nhất lớn hơn ký tự tương ứng của chuỗi thứ hai.
  - Nhỏ hơn cho các trường hợp ngược lại.

  Sinh viên nên sử dụng phương thức **compare** của thư viện lớp **string** để hiện thực khi so sánh chuỗi.

3.4 Các lỗi ngữ nghĩa

Trong quá trình tương tác, có thể kiểm tra được một số lỗi ngữ nghĩa và sẽ được ném ra (thông qua lệnh **throw** trong ngôn ngữ lập trình C/C++) nếu tìm thấy:

1. Lỗi không khai báo **Undeclared**.
2. Lỗi khai báo lại **Redeclared**.
3. Lỗi khai báo không hợp lệ **InvalidDeclaration**
4. Lỗi không đúng kiểu **TypeMismatch**.
5. Lỗi không đóng lại khối **UnclosedBlock** đi kèm với mức của khối không đóng (được mô tả ở mục 3.5.3).
6. Lỗi không tìm thấy khối tương ứng **UnknownBlock**.

Các lỗi này đều đi kèm lệnh tương ứng bằng chuỗi kí tự trong tập tin đầu vào trừ các lỗi UnclosedBlock và UnknownBlock. Chương trình sẽ dừng lại và không tiếp tục tương tác nếu có bất kỳ lỗi nào xảy ra.

3.5 Các lệnh tương tác

Một lệnh được viết trên một dòng và luôn bắt đầu bằng một mã. Ngoài ra, một lệnh có thể không có hoặc có một hoặc hai tham số. Tham số đầu tiên trong lệnh, nếu có, sẽ cách mã bằng đúng một khoảng trắng (space). Tham số thứ hai của mã, nếu có, sẽ cách với tham số đầu tiên bằng một khoảng trắng. Ngoài ra, không có ký tự phân cách và theo sau nào khác.

Ngược với quy định trên, đều là các lệnh sai, mô phỏng lập tức ném ra lỗi InvalidInstruction kèm với dòng lệnh sai và kết thúc.

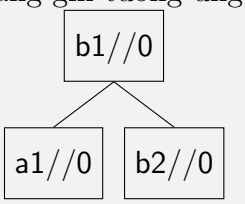
3.5.1 Thêm một đối tượng vào trong bảng ghi hoạt động - INSERT

- Định dạng chung: **INSERT <identifier_name> <type> <static>**
  trong đó:
  - <identifier_name> là tên của một danh hiệu, là một chuỗi ký tự bắt đầu bằng một ký tự chữ thường và tiếp theo là các ký tự bao gồm các ký tự chữ thường, in hoa, ký tự gạch dưới _ và ký tự số.
  - <type> là kiểu tương ứng của danh hiệu. Có ba loại kiểu là **number** hoặc **string** hoặc **kiểu hàm** để khai báo kiểu số và kiểu chuỗi ký tự.
    Kiểu hàm được chia thành hai phần: kiểu của danh sách có thể rỗng các tham số và kiểu trả về được phân cách với nhau bằng một dấu mũi tên ->. Kiểu của danh sách tham số bắt đầu bằng một dấu ngoặc tròn, tiếp theo các danh sách kiểu number hoặc string được phân cách với nhau bằng 1 dấu phẩy duy nhất. Kiểu trả về là một trong hai kiểu number hoặc string. Kiểu hàm chỉ được phép khai báo trong khối toàn cục (mức bằng 0). Ví dụ về kiểu hàm: (number,number)->string tức đây là một hàm có hai tham số đầu vào đều là kiểu number và hàm này trả ra một giá trị kiểu string.
  - <static> là một giá trị **true** hoặc **false**. Nếu nhận giá trị **true** thì danh hiệu vừa thêm thuộc tầm vực toàn cục (global scope), tức mức của khối mà danh hiệu này thuộc về luôn là 0.
- Ý nghĩa: Đưa một danh hiệu mới vào bảng ghi đối tượng. So sánh với C/C++, tương tự như việc khai báo một biến mới.
- Giá trị in ra màn hình: **<num_comp> <num_splay>** trong đó <num_comp> là số phép so sánh với các nút hiện có trên cây, <num_splay> là số thao tác splay phải thực hiện nếu thêm thành công vào bảng, ngược lại thì ném lỗi tương ứng ra.
- Các lỗi có thể xảy ra:
  - **Redeclared** nếu khai báo lại một danh hiệu đã khai báo trước.
  - **InvalidDeclaration** nếu khai báo hàm trong các khối có mức khác 0.

Ví dụ 1: Với tập tin đầu vào gồm các dòng:

```
INSERT a1 number false
INSERT b2 string false
INSERT b1 (number,number)->string false
```

Cây splay thể hiện bảng ghi tương ứng:



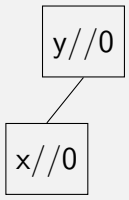
Do không có lỗi trùng nhau về tên (khai báo lại) nên chương trình in ra:

```
0 0
1 1
2 1
```

Ví dụ 2: Với tập tin đầu vào gồm các dòng:

```
INSERT x number false
INSERT y string false
INSERT x string false
```

Cây splay thể hiện bảng ghi tương ứng (trước khi bị bắt lỗi):



Do đã thêm danh hiệu x ở dòng số 1 mà còn tiếp tục thêm danh hiệu x ở dòng số 3 nên gây ra lỗi Redeclared nên chương trình in ra:

```
0 0
1 1
Redeclared: INSERT x string false
```

3.5.2 Gán giá trị cho đối tượng - ASSIGN

- Định dạng chung: **ASSIGN <identifier_name> <value>**
  trong đó:
  - <identifier_name> là tên của một danh hiệu và phải tuân theo luật đã nêu ở mục 3.5.1.
  - <value> là một giá trị được gán vào biến, có thể bao gồm ba dạng:
    - Hằng số: một dãy các số. Ví dụ: 123, 456, 789 là các hằng số đúng, còn 123a, 123.5, 123.8.7 không là các hằng số. Hằng số được xem có kiểu số (number).
    - Hằng chuỗi: được bắt đầu bằng dấu nháy đơn (’), tiếp theo là chuỗi bao gồm ký tự số, ký tự chữ, khoảng trắng và kết thúc bằng một dấu nháy đơn. Ví dụ: ’abc’, ’a 12 C’ là các hằng chuỗi, còn ’abc_1’, ’abC@u’ không là các hằng chuỗi.

Hằng chuỗi được xem có kiểu chuỗi (string).

* Một danh hiệu khác đã được khai báo trước.
* Một lời gọi hàm bao bắt đầu tên danh hiệu kiểu hàm, tiếp theo là dấu mở ngoặc tròn, một danh sách có thể rỗng các tham số thực (chỉ có thể là hằng số, hằng chuỗi hoặc một danh hiệu đã được khai báo trước đó) phân cách với nhau một bằng một dấu phẩy và kết thúc bằng một dấu đóng ngoặc tròn. Ví dụ, foo(1, 2) hay baz(a, 1) là cách lời gọi hàm hợp lệ.

- Ý nghĩa: Kiểm tra sự phù hợp cho việc gán một giá trị đơn giản cho danh hiệu. Quá trình kiểm tra được diễn ra ở <value> trước rồi <identifier_name> sau.
- Giá trị in ra màn hình: **<num_comp> <num_splay>** trong đó <num_comp> là số phép so sánh với các nút hiện có trên cây, <num_splay> là số thao tác splay phải thực hiện nếu thành công, ngược lại thì ném lỗi tương ứng ra.
- Các lỗi có thể xảy ra:
  - **Undeclared** nếu một danh hiệu chưa được khai báo xuất hiện trong cả phần <identifier_name> và <value>.
  - **TypeMismatch** nếu:
    * Kiểu của giá trị được gán và danh hiệu khác nhau.
    * Lời gọi hàm có tên danh hiệu không phải là kiểu hàm.
    * Các kiểu của tham số thực không tương ứng với kiểu của tham số hình thức được khai báo.

Ví dụ 3: Với tập tin đầu vào gồm các dòng:

```
INSERT x number false
INSERT sum (number,number)->number false
ASSIGN x sum(1,1)
ASSIGN z sum(12,'abc')
```

Việc gán ở dòng thứ 3 gây ra lỗi TypeMismatch, nên chương trình in ra như sau:

```
0 0
1 1
3 1
TypeMismatch: ASSIGN z sum(12,'abc')
```

Mặc dù chưa có danh hiệu z được khai báo như sum(12,'abc') có tham số thực thứ 2 là hằng chuỗi 'abc' không phù hợp về kiểu đã khai báo là number.

3.5.3 Mở và đóng khối (block) - BEGIN/ END

- Định dạng chung: **BEGIN/ END**.
- Ý nghĩa: Mở và đóng một khối mới tương tự với việc mở đóng { } trong C/C++. Khi mở một khối mới, có một số quy tắc như sau:
  - Được phép khai báo lại tên danh hiệu đã khai báo trước đó.
  - Khi tìm kiếm một danh hiệu, ta phải tìm với khối trong cùng. Nếu không tìm được thì tìm ra khối cha và làm cho đến khi gặp khối toàn cục.
  - Các khối có một mức (level) xác định với khối toàn cục được xác định ở mức 0 và tăng dần với các khối con.
  - Khi ra khỏi một khối, ta phải xóa hết đi các danh hiệu không được khai báo static khỏi bảng ghi danh hiệu theo thứ tự xuất hiện của chúng.
- Giá trị in ra: Chương trình không in ra với việc đóng mở block.
- Các lỗi có thể xảy ra: UnclosedBlock có thể được ném ra nếu ta không đóng lại một khối đã mở hoặc UnknownBlock nếu đóng lại nhưng không tìm dược khối bắt đầu của nó.

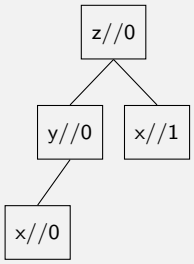
Ví dụ 4: Với tập tin đầu vào gồm các dòng:

```
INSERT x number false
INSERT y string false
BEGIN
INSERT x number false
BEGIN
INSERT z string true
END
END
```

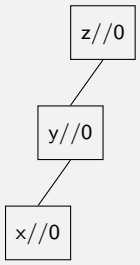
Chương trình này sẽ in ra:

```
0 0
1 1
1 1
2 1
```

Sau khi thêm z ở dòng thứ 6, cây splay thể hiện bảng ghi như sau:



Sau khi ra khỏi khối có mức 2 ở dòng 7, do z là danh hiệu được khai báo static nên z không bị xóa đi khỏi cây. Khi rời khỏi khối có mức 1 ở dòng 8, do x//1 không phải là danh hiệu static nên bị xóa đi khỏi cây:



3.5.4 Tìm đối tượng tương ứng với danh hiệu - LOOKUP

- Định dạng chung: **LOOKUP <identifier_name>**
  trong đó, <identifier_name> là tên của một danh hiệu và phải tuân theo luật đã nêu ở mục 3.5.1.
- Ý nghĩa: Tìm kiếm một danh hiệu có nằm trong bảng hoạt động hay không. So sánh với C/C++, tương tự như tìm và sử dụng một biến.
- Giá trị in ra màn hình: mức của block chứa danh hiệu nếu tìm thấy, ngược lại thì ném lỗi tương ứng ra.
- Các lỗi có thể xảy ra: **Undeclared** nếu không tìm thấy danh hiệu trong tất cả các tầm vực của bảng ghi đối tượng.

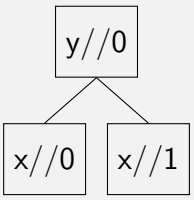
Ví dụ 5: Với tập tin đầu vào gồm các dòng:

```
INSERT x number false
INSERT y string false
BEGIN
INSERT x number false
LOOKUP y
END
```

Chương trình này sẽ in ra:

```
0 0
1 1
1 1
0
```

Sau khi đã tìm y ở dòng 5, cây splay thể hiện bảng ghi đối tượng:



3.5.5 In tiền thứ tự cây splay thể hiện bảng ghi hoạt động - PRINT

- Định dạng chung: **PRINT**
- Ý nghĩa: In tiền thứ tự cây splay thể hiện bảng ghi hoạt động.
- Giá trị in ra: các danh hiệu và kèm theo mức của khối tương ứng được in ra cách nhau một khoảng trắng trên cùng một dòng và không có khoảng trắng ở cuối dòng.

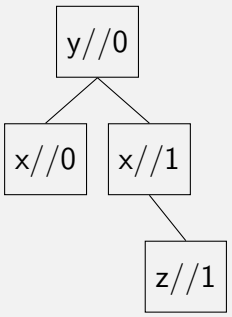
Ví dụ 6: Với tập tin đầu vào gồm các dòng:

```
INSERT x number false
INSERT y string false
BEGIN
INSERT x number false
INSERT z number false
LOOKUP y
PRINT
END
```

Chương trình này sẽ in ra:

```
0 0
1 1
1 1
1 1
0
y//0 x//0 x//1 z//1
```

Sau khi thực hiện tìm kiếm y ở dòng 6, cây splay thể hiện bảng ghi hoạt động là:



4 Nộp bài

Sinh viên chỉ nộp 2 tập tin: SymbolTable.h và SymbolTable.cpp, trước thời hạn được đưa ra trong đường dẫn "Assignment 2 - Submission". Có một số testcase đơn giản được sử dụng để kiểm tra bài làm của sinh viên nhằm đảm bảo rằng kết quả của sinh viên có thể biên dịch và chạy được. Sinh viên có thể nộp bài bao nhiêu lần tùy ý nhưng chỉ có bài nộp cuối cùng được tính điểm. Vì hệ thống không thể chịu tải khi quá nhiều sinh viên nộp bài cùng một lúc, vì vậy sinh viên nên nộp bài càng sớm càng tốt. Sinh viên sẽ tự chịu rủi ro nếu nộp bài sát hạn chót. Khi quá thời hạn nộp bài, hệ thống sẽ đóng nên sinh viên sẽ không thể nộp nữa. Bài nộp qua các phương thức khác đều không được chấp nhận.

5 Other regulations

- Sinh viên phải tự mình hoàn thành bài tập lớn này này và phải ngăn không cho người khác đánh cắp kết quả của mình. Nếu không, sinh viên sẽ bị xử lý theo quy định của trường vì gian lận.
- Mọi quyết định của giảng viên phụ trách bài tập lớn là quyết định cuối cùng.
- Sinh viên không được cung cấp testcase sau khi chấm bài mà chỉ được cung cấp thông tin về chiến lược thiết kế testcase và phân bố số lượng sinh viên đúng theo từng testcase.

——————————END——————————