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 3

MÔ PHỎNG SYMBOL TABLE BẰNG BẢNG BĂM

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 giải thuật tìm kiếm và cấu trúc dữ liệu bảng băm.

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 các bài tập lớn trước, sinh viên đã được yêu cầu hiện thực các mô phỏng về bảng ghi hoạt động bằng danh sách và cây. Để tối ưu hóa cho quá trình tìm kiếm, bảng băm là một trong những cấu trúc dữ liệu phù hợp để làm việc này. Ngoài ra, trong bài tập lớn này, sinh viên cũng được giới thiệu cách xây dựng bảng ghi đối tượng cho các ngôn ngữ có sử dụng suy diễn kiểu (type inference). Ngôn ngữ suy diễn kiểu, nói nôm na, là ngôn ngữ lập trình khi khai báo các danh hiệu thì không khai báo tường minh về kiểu, việc áp đặt kiểu của danh hiệu là kiểu gì gắn với các phát biểu (statements) hoặc các biểu thức (expressions) sử dụng danh hiệu.

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 **bảng băm**.

3 Mô tả

3.1 Đầu vào

Mỗi testcase là một tập tin đầu vào bao gồm các dòng lệnh thiết lập tham số cho bảng băm (được mô tả ở mục 3.5) và các lệnh tương tác với bảng ghi đối tượng (được mô tả ở mục 3.6).

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 bảng băm 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) có thể bao gồm:

1. Tên của danh hiệu (identifier)
2. Mức của khối mà danh hiệu thuộc về (level of block)

Sinh viên phải thiết kế lại thông tin lưu trữ để phù hợp với đề bài.

Trong quá trình tương tác với bảng băm, ta cần mã hóa thông tin một đối tượng thành khóa. Khóa được mã hóa là một số nguyên dạng $\overline{ca_1a_2a_3a_4...a_n}$ trong đó:

- c là mức của khối mà danh hiệu thuộc về.
- a_1, a_2, ..., a_n lần lượt là giá trị số thập phân trên bảng mã ASCII của từng ký tự trong danh hiệu sau khi trừ đi 48.

Ví dụ 1: Ta có một danh hiệu xB nằm ở mức 1 thì khóa của đối tượng này là

- Mức của khối là 1 nên $c = 1$.
- $x = 120$ nên ta mã hóa thành $120 - 48 = 72$
  $B = 66$ nên ta mã hóa thành $66 - 48 = 18$

Như vậy, xB//1 sẽ được mã hóa thành 17218.

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** đi kèm với danh hiệu không được khai báo.
2. Lỗi khai báo lại **Redeclared** đi kèm với danh hiệu bị khai báo lại.
3. Lỗi khai báo không hợp lệ **InvalidDeclaration** đi kèm với danh hiệu bị khai báo không hợp lệ.
4. Lỗi không đúng kiểu **TypeMismatch** đi kèm với lệnh gây ra lỗi.
5. Lỗi không suy diễn được kiểu **TypeCannotBeInfered** đi kèm với lệnh gây ra lỗi.
6. Lỗi tràn bảng **Overflow** đi kèm với lệnh gây ra lỗi.
7. Lỗi không đóng lại khối **UnclosedBlock** đi kèm với mức của khối không đóng (được mô tả ở mục 3.6.4).
8. Lỗi không tìm thấy khối tương ứng **UnknownBlock**.

Chương trình sẽ dừng lại và không tiếp tục tương tác nếu có bất kỳ lỗi nào xảy ra.

3.5 Các thiết lập tham số cho bảng băm

Băng bảm được sử dụng trong bài tập lớn này là bảng băm sử dụng phương pháp giải quyết đụng độ địa chỉ mở (open addressing). Vì vậy, các thiết lập về tham số cho bảng băm bao gồm thiết lập kích thước, thiết lập hàm băm và thiết lập hàm dò tìm là cần thiết cho bài tập. Dòng thiết lập này luôn luôn là dòng đầu tiên trong testcase và xuất hiện đúng một lần. Một thiết

lập được viết trên một dòng và luôn bắt đầu bằng một mã tương ứng với một phương pháp dò tìm (probing). Ngoài ra, một lệnh có nhiều tham số. Tham số đầu tiên cách mã thiết lập một khoảng trắng và các tham số cũng cách nhau bằng một khoảng trắng. Tất cả các tham số đều phải thỏa định dạng là một số tự nhiên tối đa có 6 chữ số. Ngoài ra, không có ký tự phân cách và theo sau nào khác.

Ngược với quy định trên và định dạng được nêu dưới đây, đều là các thiết lập sai, mô phỏng lập tức ném ra lỗi InvalidInstruction kèm với dòng thiết lập sai và kết thúc.

3.5.1 Phương pháp dò tìm tuyến tính - LINEAR

- Định dạng chung: LINEAR <m> <c>
  trong đó:
  - <m> là kích thước của bảng băm sử dụng.
  - <c> là một hằng số được sử dụng trong quá trình dò tìm.
- Ý nghĩa: Khi đó, hàm băm và hàm dò tìm tương ứng là

$$h(k) = k \mod m$$
$$hp(k,i) = (h(k) + ci) \mod m$$

  với k là khóa và i là số lần dò tìm.

3.5.2 Phương pháp dò tìm bậc hai - QUADRATIC

- Định dạng chung: QUADRATIC <m> <c_1> <c_2>
  trong đó:
  - <m> là kích thước của bảng băm sử dụng.
  - <c_1>, <c_2> là các hằng số được sử dụng trong quá trình dò tìm.
- Ý nghĩa: Khi đó, hàm băm và hàm dò tìm tương ứng là

$$h(k) = k \mod m$$
$$hp(k,i) = (h(k) + c_1 i + c_2 i^2) \mod m$$

  với k là khóa và i là số lần dò tìm.

3.5.3 Phương pháp băm đôi - DOUBLE

- Định dạng chung: DOUBLE <m> <c>
  trong đó:
  - <m> là kích thước của bảng băm sử dụng.
  - <c> là một hằng số được sử dụng trong quá trình dò tìm.
- Ý nghĩa: Khi đó, các hàm băm và hàm dò tìm tương ứng là

$$h_1(k) = k \mod m$$
$$h_2(k) = 1 + (k \mod (m-2))$$
$$hp(k,i) = (h_1(k) + cih_2(k)) \mod m$$

  với k là khóa và i là số lần dò tìm.

3.6 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.

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

- Định dạng chung: **INSERT <identifier_name> <num_of_parameters>?**
  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.
  - <num_of_parameters>? là số tham số cần phải truyền cho hàm nếu danh hiệu được khai báo đại diện cho môt hàm. Thành phần này chỉ có khi danh hiệu là hàm, ngược lại sẽ không có thành phần này. Việc khai báo danh hiệu chỉ được thực hiện ở mức toàn cục (mức 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, tuy nhiên khi khai báo biến, ta chưa định ra một kiểu dữ liệu cho biến, gần giống với khái niệm **auto** và **decltype**.

- Giá trị in ra màn hình: Số khe (slot) phải đi qua **trước** khi đến vị trí khe trống có thể đặt giá trị vào.
- 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.
  - **Overflow** nếu khai báo không thể tìm được vị trí trống phù hợp trong bảng băm.

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

```
LINEAR 19 1
INSERT a1
INSERT b2
INSERT rj 2
```

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

```
0
0
1
```

Bảng ghi đối tượng lúc đó là

Khe	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Giá trị									b2//0	rj//0							a1//0		

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

```
LINEAR 19 1
INSERT x
INSERT y
INSERT x
```

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
Redeclared: x
```

3.6.2 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 3.6.1.
  - <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 biểu thức 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:
  - Nếu cả hai thành phần của phép gán đều đã có kiểu thì ta 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.
  - Nếu ít nhất một trong hai thành phần của phép gán chưa tìm được kiểu, ta thực hiện suy diễn kiểu theo các quy tắc sau:
    * Kiểu của thành phần chưa biết sẽ được suy diễn là kiểu của thành phần đã biết.
    * Kiểu của các tham số hình thức trong danh hiệu được khai báo hàm phải được suy diễn thành kiểu của tham số thực truyền vào nếu tham số thực đã có kiểu và ngược lại.
    * Kiểu trả về của danh hiệu được khai báo hàm phải được suy diễn thành kiểu của danh hiệu được gán và ngược lại.

- Giá trị in ra màn hình: Tổng số khe phải đi qua trước khi tìm thấy khe chứa thông tin về đối tượng cần tìm của mỗi danh hiệu xuất hiện trong lệnh nếu thành công kiểm tra hoặc suy diễn.
- Các lỗi có thể xảy ra:
  - **Undeclared** nếu một danh hiệu chưa được khai báo xuất hiện ít nhất một trong hai thành phần <identifier_name> hoặc <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.
* Kiểu trả về của hàm không phù hợp với kiểu của danh hiệu được gán.

– **TypeCannotBeInferred** nếu:

* Cả hai thành phần của phép gán (đối với danh hiệu được khai báo là hàm thì đó là kiểu trả về của hàm) đều chưa được suy diễn kiểu.
* Tham số hình thức và tham số thực tương ứng đều chưa được suy diễn kiểu.

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

```
LINEAR 19 1
INSERT x
INSERT sum 2
ASSIGN x 1
ASSIGN x sum(5,x)
INSERT z
INSERT foo 1
ASSIGN z foo('abc')
```

Danh hiệu sum được khai báo là hàm có 2 tham số, foo là hàm có 1 tham số. Quá trình suy diễn diễn ra như sau:

- Dòng thứ 3 giúp ta suy diễn được kiểu của x là kiểu **number**.
- Dòng thứ 4 giúp ta suy diễn:
  – Tham số hình thức thứ nhất của hàm cũng có kiểu của hằng số 5, tức là kiểu **number**.
  – Tham số hình thức thứ hai của hàm cũng có kiểu của x, tức là kiểu **number**.
  – Kiểu trả về của hàm sum cũng có kiểu của x, tức là **number**.
- Dòng thứ 7 giúp ta suy diễn:
  – Tham số hình thức thứ nhất của hàm cũng có kiểu của hằng chuỗi 'abc', tức là kiểu **string**.
  – Kiểu trả về của foo và kiểu của z chưa có nên lỗi **TypeCannotBeInferred** được ném ra.

3.6.3 Lời gọi hàm - CALL

- Định dạng chung: **CALL** <call_exp>
  trong đó, <call_exp> là biểu thức gọi hàm như đã mô tả ở mục 3.6.2.
- Ý nghĩa: lời gọi hàm không gán vào một danh hiệu nào. Quá trình suy diễn cho lời gọi này tương tự với biểu thức nào, tuy nhiên, kiểu trả về của hàm phải là (hoặc được suy diễn là) **void**.
- Giá trị in ra màn hình: Tổng số khe phải đi qua trước khi tìm thấy khe chứa thông tin về đối tượng cần tìm của mỗi danh hiệu xuất hiện trong lệnh nếu thành công kiểm tra hoặc suy diễn.
- Các lỗi có thể xảy ra: **Undeclared**, **TypeMismatch**, **TypeCannotBeInferred**.

3.6.4 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 đã được khai báo trong khối ra khỏi bảng ghi đối tượng.
- 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 5: Với tập tin đầu vào gồm các dòng:

```
LINEAR 19 1
INSERT x1
INSERT y1
BEGIN
INSERT x1
BEGIN
INSERT u7
```

```
END
END
```

Bảng ghi đối tượng sau khi thêm u7 là:

Khe	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Giá trị	u7//0									y1//0		x1//1							x1//0

3.6.5 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 3.6.1.
- 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: Chỉ số của khe hiện tại đang chứa danh hiệu đó.
- 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 6: Với tập tin đầu vào gồm các dòng:

```
LINEAR 19 1
INSERT x1
INSERT y1
BEGIN
INSERT x1
BEGIN
INSERT u7
LOOKUP x1
END
END
```

Dòng LOOKUP x1 sẽ tìm thấy x1//1, nên dòng này in ra 11.

3.6.6 In bảng ghi hoạt động - PRINT

- Định dạng chung: **PRINT**

- Ý nghĩa: In chỉ số và giá trị đang lưu tương ứng của những vị trí đang chứ dữ liệu trên bảng ghi hoạt động.
- Giá trị in ra: chỉ số khe và các danh hiệu kèm theo mức của khối tương ứng được in ra cách nhau một khoảng trắng trên cùng một dòng, các khe cách nhau bằng dấu ; và không có khoảng trắng ở cuối dòng.

Ví dụ 7: Với tập tin đầu vào gồm các dòng:

```
LINEAR 19 1
INSERT x1
INSERT y1
BEGIN
INSERT x1
BEGIN
INSERT u7
PRINT
END
END
```

Dòng PRINT sẽ in ra:

```
0 u7//0;9 y1//0;11 x1//1;18 x1//0
```

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 3 - 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 Một số quy định khác

- 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.

——————————HẾT——————————