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


## Lập trình nâng cao - CO2039

---

## Bài tập lớn 3

# MÔ PHỎNG SYMBOL TABLE BẰNG DANH SÁCH

---

**Tác giả: ThS. Trần Ngọc Bảo Duy, CN. Thi Khắc Quân**

TP. HỒ CHÍ MINH, THÁNG 03/2025

# 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:

- Các khái niệm cơ bản của lập trình hàm (functional programming).
- Sử dụng hàm bậc cao (high-order function).
- Cách sử dụng các cấu trúc dữ liệu danh sách trong lập trình hàm.

## 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 giai đoạn mà trình biên dịch thực hiện để chuyển từ mã nguồn (source code) sang mã máy có thể thực thi được (executable code), giai đoạn phân tích ngữ nghĩa (semantic analysis) là một trong những giai đoạn quan trọng để kiểm tra tính chính xác của đoạn mã nguồn, ví dụ như kiểm tra một biến khi sử dụng đã khai báo chưa, việc gán giá trị vào một biến có phù hợp về kiểu hay không, v.v.. Giai đoạn **phân tích ngữ nghĩa** đòi hỏi phải có bảng ghi tối tượng này để truy vết các thông tin mà việc kiểm tra đòi hỏi.

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 danh sách.

## 3 Mô tả

### 3.1 Đầu vào

Mỗi testcase là một danh sách 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.py, Symbol.py, SymbolTable.py, TestSuite.py, TestUtils.py, trong đó, sinh viên không được phép sửa đổi tên các tập tin vì nó sẽ không nằm trong các danh mục dùng để nộp bài.
3. Sửa đổi file **SymbolTable.py** để hoàn thành bài tập lớn này nhưng đảm bảo hai yêu cầu sau:
    - Ít nhất có một hàm def simulate(list_of_commands) tồn tại trong file này vì hàm là đầu vào cho lời giải. Đối với mỗi testcase, hàm này sẽ được gọi với tham số đầu vào là một mảng các lệnh (tham khảo tại file TestUtils.py).
    - Sinh viên chỉ có thể sử dụng duy nhất ba lần thêm module ngoài vào file này, cụ thể **from StaticError import *** , **from Symbol import *** và **from functools import *** . Ngoài ra, không cho phép có một **import** nào khác trong các tập tin này.
4. Viết ít nhất 50 testcase trong file **TestSuite.py** để thực hiện kiểm tra cho bài làm của mình.

## 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. Kiểu tương ứng của danh hiệu (type)

## 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 **raise** trong ngôn ngữ lập trình Python) nếu tìm thấy:

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

Các lỗi này đều đi kèm lệnh tương ứng bằng chuỗi kí tự trong 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 và các quy định về định dạng của các lệnh được mô tả trong các nội dung con dưới đây, 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>**
  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 hai loại kiểu là **number** hoặc **string** để khai báo kiểu số và kiểu chuỗi ký tự.
- 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: **success** nếu thêm thành công vào bảng, ngược lại thì ném lỗi tương ứng ra.
- Các lỗi có thể xảy ra: **Redeclared**.

**Ví dụ 1:** Với đầu vào gồm các dòng:

```
INSERT a1 number
INSERT b2 string
```

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

```
success
success
```

**Ví dụ 2:** Với đầu vào gồm các dòng:

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

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:

```
Redeclared: INSERT x string
```

### 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: là 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 và kết thúc bằng một dấu nháy đơn. Ví dụ: ’abc’, ’a12C’ là các hằng chuỗi, còn ’abc_1’, ’abC@u’, ’a 12 C’ không là các hằng chuỗi. Hằng chuỗi được xem có kiểu chuỗi (string).
    * Một danh hiệu khác đã được khai báo trước.
- Ý nghĩa: Kiểm tra sự phù hợp cho việc gán một giá trị đơn giản cho danh hiệu.
- Giá trị in ra màn hình: **success** 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 một trong hai phần <identifier_name> hoặc <value>.
  - **TypeMismatch** nếu kiểu của giá trị được gán và danh hiệu khác nhau.

**Ví dụ 3:** Với đầu vào gồm các dòng:

```
INSERT x number
INSERT y string
ASSIGN x 15
ASSIGN y 17
ASSIGN x ’abc’
```

Việc gán ở dòng thứ 3 không gây ra lỗi, nhưng ở dòng thứ 4, lỗi kiểu đã xảy ra do gán hằng số vào danh hiệu kiểu chuỗi. Chương trình chạy đến dòng này và kết thúc.

```
TypeMismatch: ASSIGN y 17
```

### 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 đinh với khối toàn cục được xác định ở mức 0 và tăng dần với các khối con.
- 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 đầu vào gồm các dòng:

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

Chương trình này sẽ in ra:

```
success
success
success
success
```

Vì khi bắt đầu block có mức 1 ở dòng 3 thì sau đó được phép khai báo lại biến x ở dòng 4. Block có mức 2 bắt đầu ở dòng 5 nên sau đó biến y cũng được phép báo lại mà không có lỗi. Tuy nhiên, nếu ta xóa lệnh END ở dòng cuối cùng, chương trình sẽ in ra:
`UnclosedBlock: 1`
vì ta chưa có lệnh để đóng lại block có mức là 1.

### 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 đầu vào gồm các dòng:

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

Chương trình này sẽ in ra:

```
success
success
success
1
0
```

Vì x được tìm thấy trong block con trước, còn y không tìm thấy trong block con nên tìm thấy ở block cha.

### 3.5.5 In thuận các danh hiệu đang hoạt động ở tầm vực hiện tại - PRINT

- Định dạng chung: **PRINT**
- Ý nghĩa: In hết tất cả các danh hiệu có thể tìm thấy ở tầm vực hiện tại theo thứ tự được khai báo từ dòng đầu tiên đến dòng hiện tại.
- 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 đầu vào gồm các dòng:

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

Chương trình này sẽ in ra:

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

Dòng PRINT đó được nằm trong khối có mức 1 và biến x được khai báo lại nên chỉ có x trong tầm vực 1 này được thấy, còn x ở tầm vực cha (khối có mức 0) là không thấy.

### 3.5.6 In ngược các danh hiệu đang hoạt động ở tầm vực hiện tại - RPRINT

- Định dạng chung: **RPRINT**
- Ý nghĩa: In hết tất cả các danh hiệu có thể tìm thấy ở tầm vực hiện tại theo thứ tự được từ tầm vực con đến tầm vực cha.
- 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 7:** Với đầu vào gồm các dòng:

```
INSERT x number
INSERT y string
BEGIN
INSERT x number
INSERT z number
RPRINT
END
```

Chương trình này sẽ in ra:

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

Dòng RPRINT đó được nằm trong khối có mức 1 và biến x được khai báo lại nên chỉ có x trong tầm vực 1 này được thấy, còn x ở tầm vực cha (khối có mức 0) là không thấy, tất cả được in ngược lại.

# 4 Quy tắc về việc viết mã

Trong bài tập lớn này, sinh viên được yêu cầu phải sử dụng tối đa các khái niệm của lập trình hàm, vì vậy, bài nộp của sinh viên phải thỏa các yêu cầu sau đây:

1. Không được phép sử dụng thêm bất kỳ mô-đun nào khác ngoài các mô-đun đã được thêm vào.
2. Chương trình của sinh viên chỉ bao gồm các hàm được định nghĩa bằng từ khóa def mà không được phép định nghĩa các biến toàn cục, các kiểu dữ liệu khác (thông qua khai báo các lớp).
3. Không được phép sử dụng các vòng lặp for, while, ... mà thay vào đó phải sử dụng các hàm bậc cao, list comprehension.
4. Trong mỗi thân hàm được định nghĩa, sinh viên được phép khai báo biến và mỗi biến chỉ gắn liền với một phép khởi gán, không được phép gán lại giá trị khác cho biến để đảm bảo tính bất biến của dữ liệu.

Bài làm của sinh viên sẽ được hệ thống kiểm tra và trả kết quả cho sinh viên, nếu vi phạm, bài

làm sẽ bị từ chối trực tiếp trên hệ thống.

# 5 Nộp bài

Sinh viên chỉ nộp 2 tập tin: SymbolTable.py và TestSuite.py, 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 sử dụng để chấm và 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.

# 6 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.

# 7 Thay đổi so với phiên bản trước

**——————HẾT——————**