HỌC VIỆN KỸ THUẬT MẬT MÃ
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO THỰC TẬP CƠ SỞ

ĐỀ TÀI:

XÂY DỰNG TRÌNH BIÊN DỊCH CHO NGÔN NGỮ LẬP TRÌNH PANDORA SỬ DỤNG NGÔN NGỮ RUST

Cán bộ hướng dẫn:

ThS. Nguyễn Văn Phác

Sinh viên thực hiện:

- Trần Lưu Dũng
- Nguyễn Hữu Nhật Quang

Lớp: **CT6D**

Hà Nội, 2024

LỜI NÓI ĐẦU

Trong bối cảnh phát triển mạnh mẽ của ngành công nghệ thông tin, ngôn ngữ lập trình và công cụ biên dịch (compiler) đóng vai trò quan trọng trong việc xây dựng các phần mềm hiện đại. Từ các công cụ phân tích dữ liệu đến các hệ thống điều khiển công nghiệp, các ngôn ngữ lập trình mới không ngừng được ra đời nhằm đáp ứng nhu cầu đa dạng và chuyên biệt của ngành.

Nhóm chúng em thực hiện đề tài "Xây dựng trình biên dịch cho ngôn ngữ lập trình Pandora sử dụng ngôn ngữ Rust" nhằm nghiên cứu, thiết kế ngôn ngữ lập trình Pandora và triển khai một trình biên dịch dành riêng cho ngôn ngữ này. Đề tài này được lựa chọn với mục tiêu khám phá các khía cạnh cốt lõi của quá trình biên dịch và mở ra hướng đi mới cho việc ứng dụng Rust vào các dự án phức tạp.

Qua quá trình thực hiện, nhóm đã tìm hiểu về kiến trúc của compiler, từ các giai đoạn phân tích từ vựng, phân tích cú pháp, phân tích ngữ nghĩa, đến sinh mã và tối ưu mã. Đồng thời, chúng em đã triển khai các kỹ thuật và thư viện mới của Rust nhằm đạt được hiệu quả cao trong việc biên dịch và xử lý lỗi.

Nội dung báo cáo được chia làm bốn chương cụ thể như sau:

CHƯƠNG 1. Tổng quan về ngôn ngữ và compiler - Giới thiệu về các đặc điểm cơ bản của ngôn ngữ lập trình Pandora và các khái niệm cơ bản liên quan đến trình biên dịch (compiler). Em sẽ trình bày vai trò và tầm quan trọng của compiler trong quá trình phát triển phần mềm. Đồng thời, chương này cũng cung cấp cái nhìn tổng quát về ngôn ngữ lập trình Rust, ngôn ngữ được sử dụng để xây dựng trình biên dịch Pandora, và lý do lựa chọn Rust cho dự án. Và phần cuối của chương sẽ là mục tiêu của đề tài do bọn em đặt ra.

CHƯƠNG 2. Kiến trúc tổng quan về compiler - Phân tích các thành phần chính cấu thành nên một compiler, bao gồm các bước quan trọng trong quá trình biên dịch: phân tích từ vựng (lexical analysis), phân tích cú pháp (syntax analysis), phân tích ngữ nghĩa (semantic analysis), sinh mã trung gian, tối ưu mã và sinh mã đích. Ngoài ra, chương này còn mô tả vai trò của bảng ký hiệu và cách thức xử lý lỗi trong quá trình biên dịch.

CHƯƠNG 3. Quá trình xây dựng compiler sử dụng ngôn ngữ Rust - Trình bày chi tiết từng bước trong quá trình xây dựng compiler Pandora bằng Rust. Em sẽ giới thiệu cách thiết lập môi trường phát triển, sau đó đi vào các phần chính của compiler như: xây dựng bảng ký hiệu, xây dựng bộ phân tích từ vựng (Lexer), bộ phân tích cú pháp (Parser), bộ phân tích ngữ nghĩa, và cuối cùng là sinh mã và xử lý lỗi. Các bước này sẽ được mô tả kỹ càng để làm rõ quy trình triển khai và các kỹ thuật đặc thù trong Rust.

CHƯƠNG 4. Triển khai, thực nghiệm và đánh giá - trình bày các thử nghiệm thực tế nhằm đánh giá khả năng và hiệu quả của trình biên dịch Pandora. Em sẽ đưa ra ví dụ cụ thể, như việc biên dịch một chương trình in "Hello, world", sau đó thử nghiệm với các đoạn mã khác để kiểm tra tính chính xác và độ ổn định của compiler. Phần này cũng bao gồm việc đánh giá hiệu năng và những cải tiến có thể thực hiện trong tương lai.

Cuối cùng, báo cáo sẽ tổng kết lại những kết quả đạt được, phân tích về những hạn chế và đề xuất một số hướng cải tiến, phát triển tiếp theo cho ngôn ngữ Pandora và trình biên dịch trong tương lai, chẳng hạn như tối ưu hóa hệ thống để tăng tốc độ biên dịch hoặc mở rộng chức năng của ngôn ngữ Pandora, giúp ngôn ngữ lập trình Pandora có thể được ứng dụng thực tế tốt hơn

Chúng em hy vọng đề tài này sẽ đóng góp một phần nhỏ vào công cuộc nghiên cứu và phát triển các công cụ biên dịch, đồng thời tạo tiền đề cho các hướng nghiên cứu mới về compiler và ngôn ngữ lập trình trong tương lai.

LỜI CAM ĐOAN

Nhóm chúng em là nhóm 5 môn Thực tập cơ sở gồm Trần Lưu Dũng và Nguyễn Hữu Nhật Quang, sinh viên lớp CT6D, khóa 18. Người hướng dẫn là ThS.Nguyễn Văn Phác. Chúng em xin cam đoan toàn bộ nội dung được trình bày trong báo cáo đề tài "Xây dựng trình biên dịch cho ngôn ngữ lập trình Pandora sử dụng ngôn ngữ Rust" là hoàn toàn trung thực, phản ảnh đúng kết quả tìm hiểu thực tế. Mọi thông tin trích dẫn đều tuân thủ các quy định về sở hữu trí tuệ; các tài liệu tham khảo được liệt kê rõ ràng. Chúng em xin chịu hoàn toàn trách nhiệm với những nội dung được viết trong bài báo cáo này.

MỤC LỤC

BẢNG KÝ HIỆU

Từ viết tắt	*Từ gốc*

DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH VẼ

CHƯƠNG 1. TỔNG QUAN VỀ NGÔN NGỮ VÀ COMPILER

1.1. Giới thiệu về ngôn ngữ Pandora

Nguồn gốc tên Pandora: Tên gọi Pandora bắt nguồn từ thần thoại Hy Lạp cổ đại. Trong câu chuyện, Pandora là người phụ nữ đầu tiên được các vị thần tạo ra, người đã mở chiếc hộp chứa đựng mọi điều xấu xa tràn vào thế giới, nhưng đồng thời hy vọng vẫn còn ở lại trong đó.

Ý nghĩa và lí do chọn Pandora để đặt tên cho ngôn ngữ: Chiếc hộp Pandora là biểu tượng của sự tò mò, những điều không thể lường trước, và cũng là khởi đầu cho những thay đổi to lớn. Nhóm chúng em chọn tên Pandora cho ngôn ngữ lập trình mới này để truyền tải ý tưởng về những thử thách, khó khăn và rủi ro tiềm ẩn khi khám phá một ngôn ngữ hoàn toàn mới. Tuy nhiên, ẩn sau đó cũng là hy vọng cho một khởi đầu mới mẻ, một hành trình đi tìm tri thức và giải pháp sáng tạo.

1.1.1. Mô tả ngôn ngữ Pandora

1.1.2. Định nghĩa ngôn ngữ Pandora

1.1.2.1. Từ tố (*token*)

Định nghĩa từ tố thường được biểu diễn ở dạng biểu thức chính quy[3]. Ở đây, ta sẽ sử dụng 1 số quy ước cú pháp của biểu thức chính quy mở rộng (Extended regular expression [2]).

Quy ước của biểu thức chính quy mở rộng:

- | nghĩa là hoặc (or)
- ( ) nhóm các thành phần
- [ ] khớp bất kì 1 thành phần nào bên trong nó
- ^ để phủ định thành phần
- [^] khớp bất kì 1 thành phần nào không bên trong nó
- * lặp lại không hoặc nhiều lần
- . khớp với 1 ký tự bất kì
- ? để chỉ thành phần trước nó có thể có hoặc không

Những từ tố đơn giản được định nghĩa bằng cách đưa trực tiếp mẫu của nó (như các định nghĩa về từ khoá, dấu toán học, ...). Những từ tố phức tạp hơn sẽ

được định nghĩa bằng biểu thức chính quy như tên, xâu, số và chú thích. Phần phân tích từ vựng (lexical analyzer) phải có khả năng phân tích chương trình nguồn và đưa ra được các từ tố như sau:

Bảng 1.1: Bảng các từ tố

Từ tố	***Kí hiệu***
Whitespace	WHITESPACE
Ident	IDENTIFIER
RawIdent	RAW_IDENTIFIER
Char	CHAR_LITERAL
Str	STRING_LITERAL
RawStr	RAW_STRING_LITERAL
Int	INTEGER_LITERAL
Float	FLOAT_LITERAL
LineComment	LINE_COMMENT
BlockComment	BLOCK_COMMENT
Unknown	UNKNOWN
Eof	EOF
Colon	:
Comma	,
Dot	.
Semicolon	;
Question	?
OpenParen	(
CloseParen	)
OpenBrace	{
CloseBrace	}
OpenBracket	[

CloseBracket	]
Bang	!
BangEq	! =
Eq	=
EqEq	==
Gt	>
Ge	>=
Lt	<
Le	<=
Tilde	
Plus	+
Minus	−
Star	*
Slash	/
Percent	%
Caret	^
And	&
AndAnd	&&
Or	|
OrOr	||
Shl	<<
Shr	>>

Từ vựng của Pandora được xác định không chỉ từ chữ cái (alphabet) và chữ số (digit), mà là hầu hết các ký tự trong bảng mã Unicode [1], cụ thể như sau:

```
DIGIT -> 0 | 1 | 2 | ... | 9
ALPHABET -> a | b | c | ... | z
          | A | B | C | ... | Z
```

và các ký tự khác trong bảng mã Unicode.

Khoảng trắng (**whitespace**) là bất kỳ chuỗi không trống nào chỉ chứa các ký tự có thuộc tính Unicode Pattern_White_Space [4], cụ thể là:

- U+0009 (horizontal tab, '\t')
- U+000A (line feed, '\n')
- U+000B (vertical tab)
- U+000C (form feed)
- U+000D (carriage return, '\r')
- U+0020 (space, ' ')
- U+0085 (next line)
- U+200E (left-to-right mark)
- U+200F (right-to-left mark)
- U+2028 (line separator)
- U+2029 (paragraph separator)

Pandora là một ngôn ngữ "dạng tự do", có nghĩa là tất cả các dạng khoảng trắng chỉ dùng để phân tách các từ tố trong ngữ pháp và không có ý nghĩa ngữ nghĩa.

Tên ***identifier*** trong Pandora được phân làm 2 loại: **non keyword identifier** hoặc **raw identifier**. **non keyword identifier** được tạo thành từ tập các kí tự nêu trên và không được là từ khóa (keyword). Trong khi đó, **raw identifier** có thể là tên hoặc từ khóa (có **r#** ở phía trước để phân biệt với tên thường hay từ khóa), cụ thể như sau:

```
IDENTIFIER -> NON_KEYWORD_IDENTIFIER
            | RAW_IDENTIFIER
NON_KEYWORD_IDENTIFIER -> IDENTIFIER_OR_KEYWORD
                                        (without keyword)
RAW_IDENTIFIER -> r#IDENTIFIER_OR_KEYWORD
IDENTIFIER_OR_KEYWORD -> ID_START ID_CONTINUE*
                       | _ID_CONTINUE*
ID_START -> XID_Start | EMOJI_SYMBOL
ID_CONTINUE -> XID_Continue | EMOJI_SYMBOL
```

Trong đó **XID_Start** và **XID_Continue** là các thuộc tính của ký tự trong Unicode liên quan đến tên (định danh). Chúng thường được sử dụng để xác định liệu một ký tự có thể là phần đầu hoặc phần thân của 1 định danh trong các ngôn ngữ lập trình hay không. Ngoài ra, 1 tên còn có thể chứa các biểu tượng cảm xúc (**EMOJI_SYMBOL**).

1 ký tự chữ (**character**) là 1 ký tự Unicode đơn được đặt trong 2 ký tự ' (dấu nháy đơn - U+0027), ngoại trừ chính U+0027, ký tự này phải được thoát bằng ký tự U+005C trước đó (\), cụ thể như sau:

```
CHAR_LITERAL ->
        '([^' \ \n \r \t] | QUOTE_ESC | ASCII_ESC)'
QUOTE_ESC -> \' | \"
ASCII_ESC -> \n | \r | \t | \\ | \0
```

Chuỗi ký tự (**string literal**) là một chuỗi gồm bất kỳ ký tự Unicode nào được đặt trong hai ký tự U+0022 (dấu ngoặc kép), ngoại trừ chính U+0022, ký tự này phải được thoát bằng ký tự U+005C trước đó (\). Chuỗi ký tự cho phép có các line-breaks (cho phép ngắt dòng, được biểu thị bởi ký tự U+000A). Khi ký tự U+005C không thoát (\) xuất hiện ngay trước ngắt dòng, ngắt dòng sẽ không xuất hiện trong xâu được biểu diễn trong từ tố, cụ thể như sau:

```
STRING_LITERAL ->
"([" \] | QUOTE_ESC | ASCII_ESC | STRING_CONTINUE)*"
STRING_CONTINUE -> \ \n
```

Ta có thể thấy, các ký tự chữ hoặc các xâu có thể chứa 1 hoặc 1 vài loại **escape** (thoát ký tự). Một escape bắt đầu bằng U+005C (\) và tiếp tục bằng một trong các dạng sau:

- **Whitespace escape** (thoát khoảng trắng) là 1 trong các ký tự U+006E (n), U+0072 (r) hoặc U+0074 (t), biểu thị các giá trị Unicode U+000A (LF), U+000D (CR) hoặc U+ 0009 (HT) tương ứng
- **Null escape** (thoát null) là ký tự U+0030 (0) và biểu thị giá trị Unicode U+0000 (NUL)
- **Backslash escape** (thoát gạch chéo ngược) là ký tự U+005C (\) phải được thoát để biểu thị chính nó

Chuỗi ký tự thô (**raw string literal**) không xử lý bất kỳ escape nào. Chúng bắt đầu bằng ký tự U+0072 (r), theo sau là ít hơn 256 ký tự U+0023 (#) và ký tự U+0022 (dấu nháy kép). Raw string có thể chứa chuỗi bất kì các ký tự nào trong Unicode. Nó chỉ được kết thúc bởi một ký tự U+0022 (dấu nháy kép) khác, theo sau là các ký tự U+0023 (#) có số lượng giống với các ký tự U+0023 (#) đứng trước ký tự U+0022 (dấu nháy kép) mở đầu. Tất cả các ký tự Unicode đều có trong phần thân raw string thể hiện chính chúng, các ký tự U+0022 (dấu nháy kép) (trừ khi được theo sau bởi ít nhất nhiều ký tự U+0023 (#) như đã được sử dụng để bắt đầu raw string) hoặc U+005C (\) không có ý nghĩa gì đặc biệt. Cụ thể như sau:

```
RAW_STRING_LITERAL -> rRAW_STRING_CONTENT
RAW_STRING_CONTENT -> ".*" | #RAW_STRING_CONTENT#
```

Một hằng số nguyên (**integer literal**) có một trong bốn dạng sau:

- Một hằng thập phân bắt đầu bằng một chữ số thập phân và tiếp tục bằng bất kỳ hỗn hợp nào của các chữ số thập phân và dấu gạch dưới
- Một hằng thập lục phân bắt đầu bằng chuỗi ký tự U+0030 U+0078 (0x)

và tiếp tục dưới dạng bất kỳ hỗn hợp nào (có ít nhất một chữ số) gồm các chữ số thập lục phân và dấu gạch dưới
- Một chữ bát phân bắt đầu bằng chuỗi ký tự U+0030 U+006F (0o) và tiếp tục dưới dạng bất kỳ hỗn hợp nào (có ít nhất một chữ số) gồm các chữ số bát phân và dấu gạch dưới
- Một chữ số nhị phân bắt đầu bằng chuỗi ký tự U+0030 U+0062 (0b) và tiếp tục dưới dạng bất kỳ hỗn hợp nào (có ít nhất một chữ số) gồm các chữ số nhị phân và dấu gạch dưới

```
INTEGER_LITERAL -> DEC_LITERAL
                 | BIN_LITERAL
                 | OCT_LITERAL
                 | HEX_LITERAL
DEC_LITERAL -> DEC_DIGIT(DEC_DIGIT | _)*
BIN_LITERAL -> 0bBIN_DIGIT(BIN_DIGIT | _)*
OCT_LITERAL -> 0oOCT_DIGIT(OCT_DIGIT | _)*
HEX_LITERAL -> 0hHEX_DIGIT(HEX_DIGIT | _)*
DEC_DIGIT -> 0 | 1 | ... | 9
BIN_DIGIT -> 0 | 1
OCT_DIGIT -> 0 | 1 | ... | 7
HEX_DIGIT -> 0 | 1 | ... | 9
           | a | b | ... | f
           | A | B | ... | F
```

Một hằng số thực (**float literal**) có một trong hai dạng:

- Một số thập phân theo sau là ký tự dấu chấm U+002E (.). Theo sau có thể là một số thập phân khác (sau số thập phân đó có thể có số mũ
- Một số thập phân theo sau là số mũ

```
FLOAT_LITERAL -> DEC_LITERAL.
       | DEC_LITERAL(.DEC_LITERAL)?FLOAT_EXPONENT
FLOAT_EXPONENT -> (e | E)(+ | -)?
       (DEC_DIGIT | _)*DEC_DIGIT(DEC_DIGIT | _)*
```

Chú thích không phải tài liệu (**non-doc comment**) có thể là dòng (//) hoặc là khối (/* ... */). Pandora có hỗ trợ các chú thích khối lồng nhau. Các chú thích kiểu này được hiểu là một dạng khoảng trắng.

Chú thích tài liệu (**doc comment**) được chia làm 2 loại chính: chú thích tài liệu ngoài (**outer doc comment**) và chú thích tài liệu trong (**inner doc comment**).

Chú thích tài liệu dòng bên ngoài sẽ bắt đầu bằng //@, còn khối bên ngoài sẽ có dạng /*@ ... */. Trong khi đó, chú thích tài liệu dòng bên trong là //!, khối bên trong có dạng /*! ... */.

Các chú thích được biểu diễn như sau:

```
LINE_COMMENT -> //([^!@\n])^\n*
              | //
BLOCK_COMMENT -> /*([@!]
               | BLOCK_COMMENT_OR_DOC)
              (BLOCK_COMMENT_OR_DOC | ^(*/))**/
               | /**/
BLOCK_COMMENT_OR_DOC -> BLOCK_COMMENT
                      | OUTER_BLOCK_DOC
                      | INNER_BLOCK_DOC
INNER_LINE_DOC -> //!^\n*
OUTER_LINE_DOC -> //@^\n*
INNER_BLOCK_DOC ->
          /*!(^(*/) | BLOCK_COMMENT_OR_DOC)**/
OUTER_BLOCK_DOC ->
          /*@(^(*/) | BLOCK_COMMENT_OR_DOC)**/
```

1.1.2.2. Cú pháp và ngữ nghĩa

1.2. Khái niệm và vai trò của compiler

1.2.1. Khái niệm về compiler

Chương trình dịch là 1 chương trình dùng để chuyển 1 chương trình từ ngôn ngữ này (ngôn ngữ nguồn) thành 1 chương trình tương đương ở ngôn ngữ khác (ngôn ngữ đích).

1.2.2. Vai trò của compiler

Trong quá trình dịch, chương trình dịch có thể phát hiện lỗi cú pháp, ngữ nghĩa, đồng thời có thể tối ưu hóa chương trình (giảm thiểu số câu lệnh, tôi ưu bộ nhớ,...). Ngoài ra, trình biên giúp cho các chương trình nguồn độc lập với phần cứng, chạy được trên nhiều hệ thống khác nhau.

1.3. Giới thiệu về ngôn ngữ Rust

Rust là một ngôn ngữ lập trình hệ thống hiện đại, nổi bật với khả năng đảm bảo hiệu năng cao, tính an toàn bộ nhớ, và sự quản lý tài nguyên tối ưu. Với sự kết hợp giữa tốc độ tương đương C/C++ và các tính năng hiện đại như quản lý bộ nhớ an toàn mà không cần garbage collector, Rust mang lại lợi thế rõ rệt trong việc phát triển những phần mềm đòi hỏi độ tin cậy và hiệu suất cao. Ngoài ra, Rust có hệ thống kiểu mạnh mẽ, giúp hạn chế lỗi tiềm ẩn ngay từ khâu biên dịch. Điều này đặc biệt quan trọng trong việc xây dựng trình biên dịch – một loại phần mềm cần xử lý chính xác và hiệu quả các thao tác liên quan đến cú pháp, ngữ nghĩa, và quản lý tài nguyên.

Lý do chọn Rust làm ngôn ngữ cho trình biên dịch:

- Hiệu năng cao: Trình biên dịch thường cần xử lý lượng lớn dữ liệu nhanh chóng, và Rust có khả năng tối ưu hóa tương tự C/C++.
- An toàn bộ nhớ: Việc quản lý tài nguyên trong Rust giúp giảm thiểu các lỗi phổ biến như tràn bộ nhớ hoặc lỗi truy cập null pointer, điều này rất cần thiết cho sự ổn định của trình biên dịch.
- Cộng đồng và thư viện phong phú: Rust có hệ sinh thái mạnh với nhiều thư viện hữu ích hỗ trợ phân tích cú pháp, giúp đẩy nhanh quá trình phát triển trình biên dịch.

Với sự kết hợp giữa hiệu năng, độ tin cậy và khả năng mở rộng, Rust là sự lựa chọn lý tưởng cho việc phát triển trình biên dịch cho ngôn ngữ mới của chúng em.

1.4. Mục tiêu của đề tài

Thiết kế ngôn ngữ Pandora với các đặc điểm thân thiện, phù hợp với người lập trình, xu hướng lập trình hiện tại. Đồng thời phải đảm bảo được hiệu suất cũng như việc dễ bảo trì, mở rộng.

Thiết kế và phát triển trình biên dịch cho ngôn ngữ Pandora.Trình biên dịch sẽ đảm bảo chuyển đổi mã nguồn của ngôn ngữ Pandora thành mã đích, có thể thực thi trên nhiều hệ thống khác nhau.

Tài liệu tham khảo

[1] Julie D Allen, Deborah Anderson, Joe Becker, Richard Cook, Mark Davis, Peter Edberg, Michael Everson, Asmus Freytag, Laurentiu Iancu, Richard Ishida, et al. The unicode standard. *Mountain view, CA*, pages 660–664, 2012.

[2] Dominik D Freydenberger. Extended regular expressions: Succinctness and decidability. *Theory of Computing Systems*, 53:159–193, 2013.

[3] Yunyao Li, Rajasekar Krishnamurthy, Sriram Raghavan, Shivakumar Vaithyanathan, and HV Jagadish. Regular expression learning for information extraction. In *Proceedings of the 2008 conference on empirical methods in natural language processing*, pages 21–30, 2008.

[4] Robin Leroy Mark Davis. Unicode identifiers and syntax. Last accessed 31 October 2024. URL: https://www.unicode.org/reports/tr31.