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

NGUYÊN TẮC LẬP TRÌNH - CO3005

THÔNG SỐ KỸ THUẬT ZCODE

Phiên bản 1.2.2

THÀNH PHỐ HỒ CHÍ MINH, 12/2023

THÔNG SỐ KỸ THUẬT ZCODE

Phiên bản 1.1.0

1. Giới thiệu

ZCode là ngôn ngữ lập trình thủ tục có cấu trúc khối có mục đích chung nhằm phục vụ như một công cụ cho sinh viên thực hành triển khai trình biên dịch cơ bản được thiết kế cho ngôn ngữ lập trình đơn giản.

Mặc dù đơn giản nhưng ZCode vẫn kết hợp các yếu tố cốt lõi quan trọng đối với ngôn ngữ lập trình thủ tục, trong đó chương trình được tổ chức thành các hàm hoặc thủ tục.

2 Cấu trúc chương trình

Do tính đơn giản của nó, trình biên dịch ZCode thiếu khả năng biên dịch nhiều tệp. Điều này có nghĩa là chương trình ZCode phải được chứa trong một tệp duy nhất.

Trong tệp này có nhiều khai báo (như đã thảo luận trong phần 6). Điểm bắt đầu của chương trình ZCode là một hàm duy nhất có tên main. Hàm này không có tham số và không trả về gì. Tất cả các khai báo và câu lệnh trong ngôn ngữ lập trình này phải kết thúc bằng ít nhất một ký tự dòng mới.

3 Cấu trúc từ vựng

3.1 Bộ ký tự

Chương trình ZCode là một chuỗi các ký tự từ bộ ký tự ASCII. Trống (' '), tab ('\t'), backspace (('\b')), nguồn cấp dữ liệu biểu mẫu (tức là ASCII FF - '\f') là các ký tự khoảng trắng. '\n' được sử dụng làm ký tự dòng mới trong ZCode.

Định nghĩa về dòng này có thể được sử dụng để xác định số dòng được tạo bởi trình biên dịch ZCode.

3.2 Bình luận chương trình

Có một nhận xét một dòng trong ZCode bắt đầu bằng "##" bỏ qua tất cả các ký tự cho đến cuối dòng hiện tại, tức là khi đến cuối dòng hoặc cuối tệp. Ví dụ:

```
## Đây là một bình luận duy nhất.
một <- 5
```

3.3 Số nhận dạng

Mã định danh được sử dụng để đặt tên cho biến, hàm và tham số. Mã định danh bắt đầu bằng một chữ cái (AZ hoặc az) hoặc dấu gạch dưới (_) và có thể chứa các chữ cái, dấu gạch dưới và chữ số (0-9). Mã Z là phân biệt chữ hoa chữ thường, do đó các mã định danh sau là khác biệt: PrintLn, println và PRINTLN.

3.4 Từ khóa

Từ khóa phải bắt đầu bằng chữ cái viết thường (az). Các từ khóa sau được phép trong Mã Z:

ĐÚNG VẬY	SAI	con số	bool	sợi dây
trở lại	var	năng động	vui vẻ	
vì	cho đến khi	bởi	phá vỡ	Tiếp tục
nếu như	khác	yêu tinh	bắt đầu	kết thúc
không	Và	hoặc		

3.5 Toán tử

Sau đây là danh sách các toán tử hợp lệ:

+	-	*	/	%
không	Và	hoặc	=	<-
!=	<	<=	>	>=
...	==			

Ý nghĩa của các toán tử đó sẽ được giải thích ở phần sau.

3.6 Dấu phân cách

Các ký tự sau đây là dấu phân cách:

( ) [ ] , (một ký tự dòng mới)

3.7 Chữ

Một chữ là biểu diễn nguồn của một giá trị của một số, boolean, chuỗi.

1. Số: Tất cả các số trong ZCode đều được coi là số thực trong C/C++, giống như số thực những con số. Số ZCode bao gồm phần nguyên, phần thập phân và số mũ

phần. Phần nguyên là một dãy gồm một hoặc nhiều chữ số. Phần thập phân là số thập phân điểm theo sau tùy chọn là một số chữ số. Phần số mũ bắt đầu bằng ký tự 'e' hoặc 'E' theo sau là dấu '+' tùy chọn '-' dấu và sau đó là một hoặc nhiều chữ số. số thập phân hoặc một phần hoặc phần số mũ có thể được bỏ qua.

Ví dụ:

```
0          199        12.        12.3       12.3e3        12e3            12.3e-30
```

Tất cả các số đều ở dạng thập phân và không được biểu thị cho các hệ thống số khác.

2. Boolean: Một chữ boolean có thể đúng hoặc sai, được hình thành từ các chữ cái ASCII.

3. Chuỗi: Một chuỗi ký tự bao gồm 0 hoặc nhiều ký tự được bao quanh bởi dấu ngoặc kép ("). Sử dụng chuỗi thoát (được liệt kê bên dưới) để biểu thị các ký tự đặc biệt trong chuỗi. Hãy nhớ rằng dấu ngoặc kép kèm theo không phải là một phần của chuỗi. Đó là lỗi thời gian biên dịch khi một dòng mới hoặc ký tự EOF xuất hiện sau dấu mở (") và trước dấu khớp đóng (").

   Tất cả các chuỗi thoát được hỗ trợ như sau:

   \b lùi lại

   \f nguồn cấp dữ liệu biểu mẫu

   \r trả lại vận chuyển

   \n dòng mới

   \t tab ngang

   \' trích dẫn đơn

   \\ dấu gạch chéo ngược

   Đối với dấu ngoặc kép (") bên trong một chuỗi, phải viết một dấu ngoặc đơn (') trước nó: '" dấu ngoặc kép

   Ví dụ:

   ```
   "Đây là một chuỗi chứa tab \t"
   "Anh ấy hỏi tôi: '"John ở đâu?'""
   ```

4 Loại và Giá trị

Có ba kiểu dữ liệu nguyên thủy (hoặc vô hướng) trong ZCode (số, boolean, chuỗi) và mảng kiểu.

4.1 Kiểu Boolean

Mỗi giá trị của kiểu boolean có thể đúng hoặc sai.

if và các câu lệnh điều khiển khác hoạt động với các biểu thức boolean.

Toán hạng của các toán tử sau có kiểu boolean:

không	Và	hoặc

4.2 Loại số

Giá trị của số có thể dương hoặc âm. Chỉ những toán tử này mới có thể hành động trên các giá trị số:

+	-	*	/	%
=	!=			
>	>=	<	<=	

Trong trường hợp tính số dư r của phép chia hai số thực a và b thì
được tính bằng công thức:

r = a b a/b

trong đó a/b là phần nguyên của phép chia hai số thực a và b.

Ví dụ: 7,5%3,5 = 0,5, 7,8%3,38 = 1,04

4.3 Kiểu chuỗi

Toán hạng của các toán tử sau ở dạng chuỗi:

...	==

4.4 Kiểu mảng

ZCode hỗ trợ mảng đa chiều.

- Tất cả các phần tử của mảng phải có cùng kiểu, có thể là số, chuỗi, boolean.
- Trong khai báo mảng, một hằng số phải được sử dụng để biểu diễn số (hoặc length) của một chiều của mảng đó. Nếu mảng là nhiều chiều thì sẽ có

nhiều hơn một chữ số. Những chữ này sẽ được phân tách bằng dấu phẩy và kèm theo bằng dấu ngoặc vuông ([]).

Ví dụ:

```
số a[5] <- [1, 2, 3, 4, 5]
số b[2, 3] <- [[1, 2, 3], [4, 5, 6]]
```

Giá trị mảng là danh sách các biểu thức được phân tách bằng dấu phẩy được đặt trong '[' và ']'. Các các phần tử biểu thức có cùng kiểu.

Ví dụ: [1, 5, 7, 12] hoặc [[1, 2], [4, 5], [3, 5]].

- Giới hạn dưới của một chiều luôn bằng 0.

5 biểu thức

Biểu thức là các cấu trúc được tạo thành từ các toán tử và toán hạng. Biểu thức làm việc với dữ liệu hiện có và trả về dữ liệu mới.

Trong ZCode, tồn tại hai loại hoạt động: đơn nhất và nhị phân. Hoạt động đơn nhất làm việc với một toán hạng và các phép toán nhị phân hoạt động với hai toán hạng. Các toán hạng có thể là các biến, dữ liệu được trả về bởi toán tử khác hoặc dữ liệu được trả về bởi lệnh gọi hàm. Các nhà khai thác có thể được nhóm lại theo loại chúng hoạt động. Có năm nhóm toán tử: số học, logic, chuỗi, quan hệ, chỉ mục.

5.1 Toán tử số học

Các toán tử số học tiêu chuẩn được liệt kê dưới đây.

Nhà điều hành	Hoạt động	Kiểu toán hạng
-	Phủ định ký hiệu số	con số
+	Phép cộng số	con số
-	Phép trừ số	con số
*	Phép nhân số	con số
	Phân số	con số
/ %	Số còn lại	con số

Tất cả các phép tính số học đều dẫn đến một loại số.

5.2 Toán tử logic

Các toán tử logic bao gồm logic NOT, logic AND và logic OR. Tất cả các kết quả hoạt động logic theo kiểu boolean. Hoạt động của từng được tóm tắt dưới đây:

Toán tử Hoạt động Loại toán hạng		
không	Boolean phủ định	
Và	Sự liên kết	boolean
hoặc	Phân ly	boolean

5.3 Toán tử chuỗi

Các toán tử chuỗi tiêu chuẩn được liệt kê dưới đây.

Nhà điều hành	Hoạt động	Kiểu toán hạng
...	Nối chuỗi	sợi dây

Hoạt động này dẫn đến một loại chuỗi.

5.4 Toán tử quan hệ

Các toán tử quan hệ thực hiện so sánh số học và so sánh theo nghĩa đen. Tất cả các kết quả hoạt động quan hệ theo kiểu boolean. Các toán tử quan hệ bao gồm:

Nhà điều hành	Hoạt động	Kiểu toán hạng
=	Bình đẳng	con số
!=	Không công bằng	con số
<	Ít hơn	con số
>	Lớn hơn	con số
<=	Nhỏ hơn hoặc bằng	con số
>=	Lớn hơn hoặc bằng	con số
== So sánh hai chuỗi giống nhau		sợi dây

5.5 Toán tử chỉ mục

Toán tử chỉ mục được sử dụng để tham chiếu hoặc trích xuất các phần tử được chọn của một mảng. Nó phải mất dạng sau:

```
biểu thức phần tử -> biểu thức [index_operators]
index_operators -> biểu thức
                    | biểu thức, index_operators
```

Biểu thức giữa '[' và ']' phải thuộc loại số. Kiểu biểu thức (trong sản xuất đầu tiên) phải là kiểu mảng để biểu thức có thể là mã định danh hoặc lệnh gọi hàm.
Toán tử chỉ mục trả về phần tử của biến mảng có chỉ mục là biểu thức.
Toán tử có quyền ưu tiên cao nhất.

Ví dụ:

```
a[3 + foo(2)] <- a[b[2, 3]] + 4
```

5.6 Lệnh gọi hàm

Lệnh gọi hàm bắt đầu bằng một mã định danh (là tên của hàm), sau đó là dấu ngoặc đơn mở, danh sách các đối số được phân tách bằng dấu phẩy (danh sách này có thể trống) và dấu ngoặc đơn đóng.
Giá trị của lệnh gọi hàm là giá trị trả về của hàm callee.

5.7 Độ ưu tiên của toán tử và tính kết hợp

Thứ tự ưu tiên của các toán tử được liệt kê từ cao đến thấp:

Toán tử Loại Toán tử		Hiệp hội vị trí Arity		
Toán tử chỉ mục [,]		Hậu tố đơn nhất		Bên trái
Dấu hiệu	-	Tiền tố đơn nhất		Phải
Hợp lý	không	Tiền tố đơn nhất		Phải
nhân	*, /, %	nhị phân	trung tố	Bên trái
Thêm	+,	nhị phân	trung tố	Bên trái
Hợp lý	- và,	nhị phân	trung tố	Bên trái
quan hệ	hoặc =, ==, !=, <, >, <=, >=	nhị phân	trung tố	Không có
Sợi dây	...	nhị phân	trung tố	Không có

Biểu thức trong ngoặc đơn có độ ưu tiên cao nhất nên dấu ngoặc đơn được sử dụng để thay đổi quyền ưu tiên của các toán tử.

6 Biến và Hàm (Khai báo)

Trong chương trình ZCode, tất cả các biến phải được khai báo trước lần sử dụng đầu tiên. Một cái tên không thể được khai báo lại trong cùng một phạm vi, nhưng nó có thể hiển thị trong các phạm vi lồng nhau. Khi một cái tên được khai báo lại bởi một khai báo khác trong phạm vi lồng nhau, nó sẽ bị ẩn trong phạm vi lồng nhau.

6.1 Biến

Có ba loại biến: biến toàn cục, biến cục bộ và tham số của hàm. Khai báo biến bắt đầu bằng từ khóa làm tên loại (number, bool, string) hoặc từ khóa ẩn (var, Dynamic). Tiếp theo là một khai báo duy nhất, bao gồm một mã định danh và một giá trị khởi tạo tùy chọn.

- Nếu từ bắt đầu là var thì việc khởi tạo giá trị là bắt buộc. Nó bắt đầu bằng dấu gán (<-) và một biểu thức. Loại biểu thức được gán cho biến tại thời điểm biên dịch.

- Khi một biến được khai báo kiểu động, loại của biến có thể được biết tại thời điểm biên dịch bằng kỹ thuật suy luận kiểu. Khả năng tự động suy ra các kiểu giúp nhiều tác vụ lập trình trở nên dễ dàng hơn, giúp người lập trình có thể thoải mái bỏ qua các chú thích kiểu trong khi vẫn duy trì một số mức độ an toàn về kiểu.

Nếu một biến được khai báo theo kiểu mảng, danh sách kích thước cố định sẽ được hiển thị sau tên và được đặt trong dấu ngoặc vuông. Dạng sau của nó là <type> <tên biến>[size-1, size-n]. Mỗi kích thước phải là một chữ số. Từ khóa ẩn size-2, size-3, ..., không thể được sử dụng cho kiểu mảng.

1. Biến toàn cục: Biến toàn cục là biến được khai báo bên ngoài bất kỳ hàm nào trong chương trình. Các biến toàn cục được hiển thị từ nơi chúng được khai báo cho đến cuối chương trình.

2. Biến cục bộ: Biến cục bộ là biến được khai báo bên trong hàm (tức là bên trong thân hàm). Chúng hiển thị bên trong khối nơi chúng được khai báo và tất cả các khối lồng nhau.

   Đoạn mã sau đây là hợp pháp:

```
func foo(số a[5], chuỗi b) bắt đầu

    var tôi <- 0
    for i cho đến khi tôi >= 5 x 1
    bắt
        đầu a[i] <- i * i + 5
    kết thúc
    trở lại -1
kết thúc
```

3. Tham số: Trong ZCode, việc khai báo tham số bắt đầu bằng từ khóa là tên loại (số, bool, chuỗi), theo sau là một khai báo duy nhất, bao gồm một mã định danh. Các

từ khóa ẩn không thể được sử dụng để khai báo tham số. Một tham số trong kiểu mảng sẽ được truyền theo tham chiếu trong khi tham số ở kiểu khác sẽ được truyền theo giá trị.

Trong trường hợp truyền theo giá trị, hàm callee được cho giá trị trong các tham số của nó. Do đó, hàm callee không thể thay đổi các đối số trong hàm gọi theo bất kỳ cách nào. Khi một hàm được gọi, mỗi tham số của nó được khởi tạo với giá trị của đối số tương ứng được truyền từ hàm gọi.

Trong trường hợp truyền theo tham chiếu của mảng, tham số trong hàm callee sẽ được cung cấp địa chỉ của đối số tương ứng. Do đó, việc sửa đổi một phần tử của tham số thực sự xảy ra trong phần tử tương ứng của đối số.

Tham số hình thức là các biến có phạm vi hàm (trong phần 8.2).

6.2 Chức năng

Hàm là đơn vị cấu trúc chương trình trong ZCode. Mọi hàm đều bắt đầu bằng từ khóa func, theo sau là mã định danh, danh sách các tham số được phân tách bằng dấu phẩy, được đặt trong dấu ngoặc tròn () và tùy chọn, kết thúc bằng câu lệnh return hoặc câu lệnh khối. Danh sách các ký tự dòng mới có thể rỗng có thể được sử dụng để phân tách phần khai báo tham số và phần thân của hàm.

Tương tự như C/C++, ngôn ngữ cho phép các hàm chỉ có phần khai báo, nhưng sau đó phải có phần triển khai hoàn chỉnh. Kiểu của hàm sẽ được suy ra từ câu lệnh return; nếu không có câu lệnh return thì hàm sẽ không trả về bất cứ thứ gì.

Như phần giới thiệu đầu tiên, ZCode phải có chức năng main để bắt đầu điểm vào.

7 phát biểu

Một câu lệnh chỉ ra hành động mà chương trình thực hiện. Có nhiều loại phát biểu, được mô tả như sau:

7.1 Câu lệnh khai báo biến

Câu lệnh khai báo biến phải giống với câu khai báo biến (được mô tả ở phần trước).

7.2 Tuyên bố nhiệm vụ

Câu lệnh gán gán một giá trị cho vế trái, giá trị này có thể là biến vô hướng, biểu thức chỉ số. Một bài tập có dạng sau:

```
lhs <- biểu thức
```

trong đó giá trị được biểu thức trả về được lưu trữ ở vế trái lhs, có thể là biến cục bộ hoặc một phần tử của mảng.

Loại giá trị được biểu thức trả về phải tương thích với loại lhs. Đoạn mã sau chứa các ví dụ về phép gán:

```
API <- 3,14
l[3] <- giá trị * aPi
```

7.3 Câu lệnh if

Câu lệnh if thực hiện có điều kiện một trong một số danh sách câu lệnh dựa trên giá trị của một số biểu thức boolean. Câu lệnh if có dạng sau:

```
if (biểu thức-1) <câu lệnh-1> [elif (biểu
 thức-2) <câu lệnh-2>]? [elif (biểu thức-3) <câu
 lệnh-3>]? [elif (biểu thức-4) <câu lệnh-4>]?

...
 [khác <tuyên bố khác>]?
```

trong đó biểu thức đầu tiên ước tính thành giá trị boolean. Nếu biểu thức cho kết quả đúng thì câu lệnh-1 sẽ được thực thi.

Nếu biểu thức-1 có giá trị sai thì biểu thức-2, nếu có, sẽ được tính. Câu lệnh tương ứng được thực thi nếu giá trị của biểu thức là đúng.

Nếu tất cả các biểu thức trước đó đều trả về false thì câu lệnh else theo sau, nếu có, sẽ được thực thi. Nếu không có mệnh đề else tồn tại và biểu thức sai thì câu lệnh if sẽ được bỏ qua.

Không có hoặc có nhiều phần elif trong khi không có hoặc có một phần khác. Danh sách các ký tự dòng mới có thể rỗng có thể được sử dụng để phân tách biểu thức điều kiện và các ký tự sau
tuyên bố.

7.4 Đối với tuyên bố

Nói chung, câu lệnh for cho phép thực hiện lặp đi lặp lại <câu lệnh>. Câu lệnh For thực hiện một vòng lặp với số lần lặp được xác định trước. Đối với các báo cáo có dạng sau:

```
cho <biến số> cho đến <biểu thức điều kiện> bởi <biểu thức cập nhật>
      <tuyên bố>
```

<number-variable> phải là biến vô hướng của kiểu dữ liệu number. Vòng lặp sẽ lặp lại biến <number-variable>, kiểm tra <biểu thức điều kiện> và nếu nó có giá trị đúng thì nó sẽ chấm dứt; nếu không, nó sẽ thực thi <câu lệnh>. Sau mỗi lần lặp, giá trị của <biến số> sẽ được tăng thêm <biểu thức cập nhật>. Sau khi hoàn thành hoặc chấm dứt vòng lặp đột ngột bằng lệnh ngắt, giá trị của <number-variable> sẽ giữ nguyên giá trị như trước vòng lặp. Danh sách các ký tự dòng mới có thể rỗng có thể được sử dụng để phân tách <update-biểu thức> và <statement>.

```
var tôi <- 0
cho tôi cho đến khi tôi >= 10 x 1
    viếtNumbe(i)
```

7.5 Tuyên bố nghỉ giải lao

Sử dụng câu lệnh break, chúng ta có thể thoát khỏi vòng lặp ngay cả khi điều kiện kết thúc vòng lặp không được đáp ứng. Nó có thể được sử dụng để kết thúc một vòng lặp vô hạn hoặc buộc nó phải kết thúc trước khi kết thúc tự nhiên. Nó phải nằm trong một vòng lặp. Nếu không, sẽ xảy ra lỗi (Điều này sẽ được thảo luận trong giai đoạn Phân tích ngữ nghĩa). Câu lệnh break có dạng sau:

```
phá vỡ
```

7.6 Tiếp tục tuyên bố

Câu lệnh continue làm cho chương trình bỏ qua phần còn lại của thân vòng lặp trong lần lặp hiện tại như thể đã đạt đến phần cuối của khối câu lệnh. Nó phải nằm trong một vòng lặp.
Nếu không, sẽ xảy ra lỗi (Điều này sẽ được thảo luận trong giai đoạn Phân tích ngữ nghĩa). Câu lệnh continue có dạng sau:

```
Tiếp tục
```

7.7 Tuyên bố trả lại

Câu lệnh return nhằm mục đích chuyển quyền điều khiển và dữ liệu cho người gọi hàm chứa nó. Câu lệnh return bắt đầu bằng từ khóa return, theo sau là một biểu thức tùy ý.

7.8 Câu lệnh gọi hàm

Câu lệnh gọi hàm bắt đầu bằng một mã định danh (là tên của hàm), sau đó là dấu ngoặc đơn mở, danh sách các đối số được phân tách bằng dấu phẩy (danh sách này có thể trống) và dấu ngoặc đơn đóng. Giá trị của lệnh gọi hàm là giá trị trả về của hàm callee.

7.9 Khối tuyên bố

Câu lệnh khối bắt đầu bằng từ khóa bắt đầu và kết thúc bằng từ khóa kết thúc. Trong phần nội dung của câu lệnh khối, có thể có một danh sách các câu lệnh có thể rỗng.

Ví dụ:

```
số bắt
    đầu r
    số s
    r <- 2.0
    số a[5] số b[5]

    s <- r * r * 3,14
    một[0] <- s
kết thúc
```

Sau khi từ khóa bắt đầu và kết thúc, người lập trình phải sử dụng ít nhất một ký tự dòng mới.

8 Phạm vi

Có 3 cấp độ phạm vi: toàn cầu, chức năng và khối.

8.1 Phạm vi toàn cầu

Tất cả tên hàm và tên biến bên ngoài khối hàm đều có phạm vi toàn cục.

8.2 Phạm vi chức năng

Tất cả các tham số được khai báo trong hàm đều có phạm vi hàm. Chúng hiển thị từ những nơi chúng được khai báo cho đến cuối hàm kèm theo trừ khi một biến khác được khai báo có cùng tên trong câu lệnh khối nội dung và nó là các khối lồng nhau.

8.3 Phạm vi khối

Tất cả các biến được khai báo trong một khối đều có phạm vi khối, tức là chúng có thể nhìn thấy được từ nơi chúng được khai báo cho đến cuối khối.

9 IO

Để thực hiện các thao tác đầu vào và đầu ra, Để thuận tiện, ZCode cung cấp các chức năng tích hợp sau:

Hàm	Ngữ nghĩa
readNumber	Đọc một số từ bàn phím và trả về nó.
writeNumber(anArg)	Viết một số lên màn hình.
readBool()	Đọc một giá trị boolean từ bàn phím và trả về nó.
writeBool(anArg)	Viết một giá trị boolean ra màn hình.
readString()	Đọc một chuỗi từ bàn phím và trả về nó.
writeString(anArg)	Viết một chuỗi ra màn hình.

Các tên hàm này nằm trong phạm vi toàn cục.

10 ví dụ về mã nguồn chính xác trong ZCode

10.1 Mã nguồn 1

```
func areDivisors(số num1, số num2)
	trả về ((num1 % num2 = 0) hoặc (num2 % num1 = 0))

chức năng chính()
	bắt đầu
		var num1 <- readNumber() var num2
		<- readNumber()

		if (areDivisors(num1, num2)) writeString("Yes") else writeString("No")

	kết thúc
```

10.2 Mã nguồn 2

```
func isPrime(số x)

func main() số
	bắt đầu
		x <- readNumber() if (isPrime(x))
		writeString("Yes") else writeString("No")

	kết thúc
```

```
func isPrime(số x)
      bắt đầu
            nếu (x <= 1) trả về sai
            var tôi <- 2
            cho i cho đến khi i > x / 2 x 1 bắt đầu
            nếu (x
                  % i = 0) trả về sai
            kết thúc
            trả lại sự thật
      kết thúc
```

11 Nhật ký thay đổi

11.1 Phiên bản 1.2.0 (24/01/2024)

- Thay đổi phần cuối của tất cả các khai báo và câu lệnh từ chỉ một thành ít nhất một dòng mới tính cách.
- bắt đầu và kết thúc bằng câu lệnh khối.
- Thay đổi dạng câu lệnh for và if.
- Thay đổi hình thức báo cáo hoàn trả.
- Thay đổi đoạn mã chứa ví dụ về câu lệnh gán.
- Cho phép kết hợp bình luận và các tuyên bố, phát biểu khác.
- Sửa đổi các ví dụ về mã nguồn.
- Làm rõ việc khai báo tham số.

11.2 Phiên bản 1.2.1 (26/01/2024)

- Cập nhật giá trị mảng.
- Sửa đổi các ví dụ về mã nguồn.

11.3 Phiên bản 1.2.2 (28/01/2024)

- Được coi là dòng mới, dấu xuống dòng không phải là ký tự khoảng trắng.
- Chi tiết biểu thức con