



SPÉCIFICATIONS DÉTAILLÉES INTERPRÉTEUR DU LANGAGE LIR

PROJET PROPOSÉ PAR FRÉDÉRIQUE BARRIOS

Nicolas CAMINADE, Sylvan COURTIOL,
Pierre DEBAS, Heïa DEXTER,
Lucàs VABRE

Sommaire

1	Première itération	3
1.1	Commande	4
1.2	Commande debut	5
1.3	Commande fin	5
1.4	Commande defs	6
1.5	Commande affiche	6
1.6	Commande affiche avec une expression	7
1.7	Commande var pour une chaîne de caractères	7
1.8	Commande var pour un entier	8
1.9	Expression concaténation sur chaîne de caractères	8
1.10	Expression logique	9
1.11	Expression arithmétique	9
2	Deuxième itération	11
2.1	Commande efface	12
2.2	Commande lance	13
2.3	Commande stop	13
2.4	Etiquette	14
2.5	Instruction	14
2.6	Instruction vaen	15
2.7	Commande lance à partir d'une étiquette	15
2.8	Instruction procédure	16
2.9	Instruction retour	16
2.10	Commande liste	17
2.11	Instruction	17
3	Troisième itération	19
3.1	Commande sauve	20
3.2	Commande charge	21
3.3	Instruction si... vaen	21

Introduction

Le projet interpréteur LIR a été réalisé selon un modèle de cycle de vie itératif. Dans ce document des spécifications détaillées du projet seront présentées les fonctionnalités ajoutées à l'interpréteur au cours de chaque itération avec les récits d'utilisation des fonctionnalités ajoutées ou des éléments nécessaires au bon fonctionnement de l'interpréteur LIR.

Chapitre 1

Première itération

Contenu de la première itération

Dans cette première itération, l'objectif est d'avoir un prototype de l'interpréteur avec des fonctionnalités de base. Ces premières fonctionnalités sont les commandes et instructions suivantes :

- Commande `debut` qui efface toutes les lignes de programme mémorisées ainsi que tous les identificateurs mémorisés.
- Commande `fin` qui quitte l'interpréteur.
- Commande `defs` qui affiche le contexte de l'interpréteur, i.e. affiche la liste des identificateurs définis durant la session avec leur valeur.
- Instruction `affiche` qui évalue la valeur de l'expression et l'affiche sur la sortie texte courante ou alors provoque un saut de ligne sur la sortie texte courante.
- Instruction `var` qui affecte la valeur de l'expression à la variable nommée par l'identificateur.

Récits d'utilisation proposés lors de l'itération 1

1.1 Commande

Récit d'utilisation

Titre : Exécution d'une commande

Récit : Exécution d'une commande

En tant que : programmeur avec l'interpréteur LIR

Je souhaite : exécuter une commande

Afin de : obtenir le résultat de cette commande ou une confirmation de son exécution

Critères d'acceptation

À partir du fait : l'interpréteur affiche un invite

Alors : j'entre une ligne de commande

Enfin : j'obtiens le résultat de cette commande ou un retour m'informant du bon déroulé de l'exécution de la commande ou de son échec.

1.2 Commande debut

Récit d'utilisation

Titre : debut

Récit : Réinitialiser l'environnement de l'interpréteur LIR

En tant que : programmeur

Je souhaite : vider l'intégralité du contexte d'exécution

Afin de : obtenir un environnement de travail vierge

Critères d'acceptation

À partir de : d'une session de l'interpréteur LIR

Alors : j'entre la commande debut

Enfin : L'interpréteur efface toutes les lignes de programme mémorisées ainsi que tous les identificateurs mémorisés

1.3 Commande fin

Récit d'utilisation

Titre : Commande fin

Récit : Quitter l'interpréteur

En tant que : programmeur avec l'interpréteur LIR

Je souhaite : quitter l'interpréteur LIR et avoir un message m'informant de la fermeture de session

Afin de : fermer la session courante de l'interpréteur LIR

Critères d'acceptation

À partir de : une session de l'interpréteur LIR ouverte

Alors : je souhaite quitter l'interpréteur et fermer la session courante en exécutant la commande fin

Enfin : le processus courant de l'interpréteur LIR s'arrête

1.4 Commande defs

Récit d'utilisation

Titre : Affichages du contexte courant (commande defs)

Récit : Affichages du contexte courant (commande defs)

En tant que : programmeur avec l'interpréteur LIR

Je souhaite : voir toutes les variables définies dans la session courante (identificateur et valeur)

Afin de : connaître le contexte actuel de la session courante de l'interpréteur

Critères d'acceptation

À partir du fait : des variables sont définies dans la session courante de l'interpréteur

Alors : je souhaite connaître le contexte actuel en exécutant la commande defs

Enfin : l'interpréteur affiche chaque variable ligne par ligne avec son identificateur et sa valeur

1.5 Commande affiche

Récit d'utilisation

Titre : Faire un saut de ligne avec la commande affiche

Récit : Provoquer le saut de ligne sur la sortie de texte courante

En tant que : Programmeur

Je souhaite : que l'interpréteur LIR saute une ligne sur la sortie de texte courante

Afin de : Provoquer un saut de ligne sur cette sortie

Critères d'acceptation

À partir du fait : que j'ai une sortie de texte courante

Alors : j'entre la commande affiche

Enfin : l'interpréteur saute une ligne sur la sortie de texte courante

1.6 Commande affiche avec une expression

Récit d'utilisation

Titre : Commande affiche (expression)

Récit : Afficher le contenu d'une expression sur la console de l'interpréteur

En tant que : Programmeur

Je souhaite : que l'interpréteur LIR évalue et affiche le contenu de l'expression que l'on lui donne

Afin de : d'afficher le résultat de l'expression en argument

Critères d'acceptation

À partir de : l'interpréteur affichant un invite

Alors : j'entre la commande affiche et écrit l'expression dont je veux le résultat affiché

Enfin : l'interpréteur affiche le résultat de l'expression

1.7 Commande var pour une chaîne de caractères

Récit d'utilisation

Titre : Commande var (Chaîne de caractères)

Récit : Initialiser une chaîne de caractère dans variable / Changer sa valeur

En tant que : Programmeur

Je souhaite : que l'interpréteur LIR stock une chaîne dans une variable

Afin de : pouvoir récupérer/manipuler cette chaîne plus tard dans le programme

Critères d'acceptation

À partir du fait : que j'ai la possibilité de saisir une ligne de commande

Alors : je tape la commande var et met une chaîne de caractère entre double guillemets comme valeur : var <nomVariable>=<chaîne>

Enfin : l'interpréteur enregistre dans la variable spécifiée la chaîne de caractère voulue et renvoie la variable suivie de sa valeur (en tant que feed-back)

1.8 Commande var pour un entier

Récit d'utilisation

Titre : Commande var (Entier)

Récit : Initialiser un entier dans variable / Changer sa valeur

En tant que : Programmeur

Je souhaite : que l'interpréteur LIR stocke un entier dans une variable

Afin de : pouvoir récupérer/manipuler cet entier plus tard dans le programme

Critères d'acceptation

À partir du fait : que j'ai la possibilité de saisir une ligne de commande

Alors : je tape la commande var et mets un entier comme valeur : | var<nomVariable>=<entier>
|

Enfin : l'interpréteur enregistre dans la variable spécifiée l'entier voulu et renvoie la variable suivie de sa valeur (en tant que feed-back)

1.9 Expression concaténation sur chaîne de caractères

Récit d'utilisation

Titre : Opérateur + sur les chaînes de caractères

Récit : Concaténation de chaînes

En tant que : Programmeur

Je souhaite : accoler deux chaînes l'une à la suite de l'autre

Afin de : créer des messages dépendant du contexte d'exécution sur la sortie standard. Représenter une valeur entière par son écriture chiffrée en base 10.

Critères d'acceptation

À partir de : deux chaînes de caractères ou une chaîne et un entier, en tant qu'identificateurs déclarés ou expressions littérales.

Alors : En utilisant une expression de type `var nouvelleChaine = opeGauche + opeDroite`, j'obtiens la concaténation de deux chaînes.

Enfin : L'identificateur `nouvelleChaine` contient la chaîne constituée des deux primordiales concaténées. L'interpréteur confirme en affichant la nouvelle valeur ou m'informe d'une erreur. L'opération peut être récursive mais n'est pas commutative. Une concaténation s'effectue toujours par la droite.

1.10 Expression logique

Récit d'utilisation

Titre : Expression logique dans un branchement conditionnel

Récit : Opérations relationnelles sur deux entiers ou sur deux chaînes de caractères

En tant que : Programmeur

Je souhaite : que l'Interpréteur LIR compare deux entiers avec une relation d'ordre ou d'équivalence

Afin que : d'exécuter ou non une branche du code avec l'instruction `si <expression> va en`

Critères d'acceptation

À partir de : d'une ligne de programme à mémoriser et d'identificateurs auxquels une valeur aura été affectée préalablement ou de constantes littérales de même type.

Alors : j'entre une expression composée de deux opérandes de même type et l'interpréteur évalue l'expression.

Les opérandes peuvent être :

- deux constantes littérales
- deux identificateurs
- une constante littérale et un identificateur

Enfin : si l'expression (condition dans l'instruction) est vraie alors l'exécution continuera à partir du numéro de ligne spécifié par l'étiquette, sinon l'exécution continuera en séquence.

1.11 Expression arithmétique

Récit d'utilisation

Titre : Expression arithmétique

Récit : Calculer à l'aide d'expression arithmétique

En tant que : Programmeur

Je souhaite : que l'Interpréteur LIR effectue une opération arithmétique courante (addition, soustraction, multiplication, quotient ou reste d'une division entière)

Afin que : j'en exploite ou vois le résultat

Critères d'acceptation

À partir de : d'une ligne de l'interpréteur ou d'une ligne de programme à mémoriser et d'identificateurs auxquels une valeur aura été affectée préalablement ou de constantes littérales numériques.

Alors : j'entre une expression composée de deux opérandes de type entier signé et d'un opérateur.

Les opérandes peuvent être :

- deux constantes littérales
- deux identificateurs
- une constante littérale et un identificateur

Enfin : j'obtiens le résultat de l'opération ou un message d'erreur m'informant que l'opération est impossible pour les identificateurs ou constantes littérales saisies.

Chapitre 2

Deuxième itération

Contenu de la deuxième itération

Dans cette deuxième itération, l'objectif est d'ajouter des fonctionnalités permettant l'écriture de programmes simple en LIR, à savoir :

- Commande `efface` qui efface toutes les lignes de programme dont le numéro d'étiquette est dans la plage comprise entre `<etiquette_debut>` et `<etiquette_fin>`.
- Commande `lance` qui démarre l'exécution d'un programme à partir de son plus petit numéro d'étiquette ou du numéro d'étiquette indiqué par l'utilisateur.
- Commande `liste` qui affiche toutes les lignes de programme mémorisées dans l'ordre croissant des numéros de ligne.
- Instruction `stop` qui arrête l'exécution du programme.
- Instruction `vaen` qui continue l'exécution à partir du numéro spécifié par étiquette.
- Instruction `procedure` qui transfère l'exécution du programme au numéro d'étiquette spécifié et qui reprendra en séquence lorsque la procédure sera terminée.
- Instruction `retour` qui, rencontrée après un appel de procédure, provoque un retour à l'instruction qui suit son appel.

Récits d'utilisation proposés lors de l'itération 2

2.1 Commande efface

Récit d'utilisation

Titre : Commande efface

Récit : Utilisation de la commande efface

En tant que : Programmeur

Je souhaite : Supprimer une ou plusieurs lignes d'un programme

Afin de : Effacer les instructions d'un bloc de code

Critères d'acceptation

À partir de : une ou plusieurs lignes de programme mémorisé et leur étiquettes

Alors : on tape la commande : efface <etiquette_debut> : <etiquette_fin>

Enfin : l'interpréteur efface les lignes de programme dont le numéro d'étiquette est compris dans la plage, comprise entre etiquette_debut et etiquette_fin

2.2 Commande lance

Récit d'utilisation

Titre : Commande lance sans argument

Récit : Exécuter le programme à partir de l'étiquette la plus petite

En tant que : Programmeur avec l'interpréteur LIR

Je souhaite : Exécuter le programme chargé avec la commande lance

Afin de : obtenir le comportement du programme chargé pour atteindre son objectif

Critères d'acceptation

À partir de : lignes d'instructions chargé dans la session courante de l'interpréteur LIR

Alors : lorsque j'entre la commande lance sans arguments et la valide le programme s'exécute à partir de l'étiquette la plus petite

Enfin : le contexte de l'interpréteur contient le contexte final du programme exécuté

2.3 Commande stop

Récit d'utilisation

Titre : Commande stop

Récit : Utilisation de la commande stop

En tant que : Programmeur

Je souhaite : Arrêter un programme

Afin de : terminer son execution

Critères d'acceptation

À partir du fait : Qu'un programme comporte au moins une instruction

Alors : on tape la commande : <etiquette> stop

Enfin : À son exécution, le programme s'arrête lorsqu'il a atteint l'étiquette indiquée. Puis l'interpréteur affiche de nouveau un invite.

2.4 Etiquette

Récit d'utilisation

Titre : Étiquettes

Récit : Ordonner les lignes d'un programme avec les étiquettes

En tant que : Programmeur avec l'interpréteur LIR

Je souhaite : ajouter des instruction au programmes dans un ordre précis

Afin de : que les instructions puissent être exécutées dans le bon ordre

Critères d'acceptation

À partir de : l'interpréteur LIR et des instructions définies

Alors : lorsque j'entre une instruction précédée d'une étiquette alors celle-ci est enregistrée avec son étiquette pour pouvoir être exécutée plus tard.

Enfin : lorsque le programme est lancé alors les instructions s'exécutent l'ordre des étiquettes.

2.5 Instruction

Récit d'utilisation

Titre : Instructions

Récit : Consulter et modifier le contexte d'exécution

En tant que : programmeur

Je souhaite : faire réaliser des actions par l'interpréteur

Afin de : déclarer des variables, des fonctions, effectuer des sauts conditionnels, des itérations, connaître et manipuler le contexte d'un programme.

Critères d'acceptation

À partir de : ligne de commande ou programme

Alors : J'entre une instruction pour effectuer une action précise

Enfin : Le contexte est modifié en fonction de cette instruction. L'interpréteur m'informe en cas d'erreur de syntaxe

2.6 Instruction `vaen`

Récit d'utilisation

Titre : Instruction `vaen`

Récit : Sauts inconditionnels

En tant que : programmeur

Je souhaite : effectuer un saut vers une ligne spécifique d'un programme.

Afin de : Créer des branchements ou des itérations dans mes programmes.

Critères d'acceptation

À partir de : la saisie d'un programme

Alors : j'entre la commande `vaen` suivie du numéro de la ligne où je veux effectuer le saut.

Enfin : lors de l'exécution de l'instruction, le programme ignorera les lignes suivantes et sautera directement à la ligne indiquée.

2.7 Commande `lance` à partir d'une étiquette

Récit d'utilisation

Titre : Commande `lance` <Étiquette>

Récit : Exécuter le programme à partir de l'étiquette argument

En tant que : Programmeur avec l'interpréteur LIR

Je souhaite : Exécuter le programme chargé avec la commande `lance` <étiquette>

Afin de : obtenir le comportement et objectif du programme chargé

Critères d'acceptation

À partir de : lignes d'instructions chargé dans la session courante de l'interpréteur LIR

Alors : lorsque j'entre la commande lance sans arguments et la valide le programme s'exécute à partir de l'étiquette passé en argument

Enfin : le contexte de l'interpréteur contient le contexte final du programme exécuté à partir de l'étiquette spécifiée

2.8 Instruction procédure

Récit d'utilisation

Titre : Procédure

Récit : Ordonner a l'interpréteur à exécuter des lignes de code à partir de l'étiquette de l'instruction.

En tant que : Programmeur

Je souhaite : transférer l'exécution au numéro d'étiquette spécifié.

Afin de : exécuter le programme puis reprendre en séquence une fois le procédure terminée.

Critères d'acceptation

À partir de : Plusieurs lignes de code et d'identificateurs déclarés, dont la portée est globale.

Alors : En utilisant l'instruction `procedure <etiquette>`

Enfin : Alors l'interpréteur va chercher la ligne qui a pour identificateur celui référencé en étiquette et va l'exécuter jusqu'à la fin de la séquence.

2.9 Instruction retour

Récit d'utilisation

Titre : retour

Récit : Ordonner a l'interpréteur de retourner à la suite de l'instruction qui suit son appel.

En tant que : Programmeur

Je souhaite : retourner à la suite de la ligne de code qui a précédé l'appel de procédure.

Afin de : d'exécuter le programme qui allais s'exécuter si l'appel de procédure n'avait pas été fais.

Critères d'acceptation

À partir de : Plusieurs lignes de code et a la suite d'une instruction procédure.

Alors : En utilisant l'instruction `retour`

Enfin : Alors l'interpréteur va chercher la ligne qui suivait l'instruction procédure et va l'exécuter jusqu'à la fin de la séquence.

2.10 Commande liste

Récit d'utilisation

Titre : Commande liste

Récit : Utilisation de la commande liste avec argument

En tant que : Programmeur

Je souhaite : que l'Interpréteur LIR affiche toutes les lignes de programme mémorisées dans l'ordre croissant des numéros de ligne dans un intervalle donné.

Afin que : je visualise uniquement les lignes de cet intervalle dans l'ordre croissant.

Critères d'acceptation

À partir de : aucune ou plusieurs lignes de programme à mémoriser et de leurs étiquettes et d'un intervalle d'entier passé en argument

Alors : j'entre la commande `liste <etiquette_debut>:<etiquette_fin>`

Enfin : l'interpréteur affiche toutes les lignes de programme mémorisées, s'il y en a, dans l'ordre croissant de leur étiquette et dont les étiquettes sont situées dans cet intervalle donné.

2.11 Instruction

Récit d'utilisation

Titre : Commande liste

Récit : Utilisation de la commande liste sans argument

En tant que : Programmeur

Je souhaite : que l'Interpréteur LIR affiche toutes les lignes de programme mémorisées dans l'ordre croissant des numéros de ligne.

Afin que : je visualise ces lignes dans leur ordre d'exécution

Critères d'acceptation

À partir de : d'aucune ou plusieurs lignes de programme à mémoriser et de leurs étiquettes

Alors : j'entre la commande `liste`

Enfin : l'interpréteur affiche toutes les lignes de programme mémorisées, s'il y en a, dans l'ordre croissant de leur étiquette.

Chapitre 3

Troisième itération

Contenu de la première itération

Dans cette troisième itération, l'objectif est de couvrir toutes les fonctionnalités attendues. Ces dernières concernent la lecture et l'écriture de fichier et l'ajout d'une structure de contrôle à l'interpréteur :

- Commande `save` qui sauvegarde les lignes de programme dans le fichier texte indiqué en argument.
- Commande `charge` qui charge dans le contexte les lignes de programme sauvegardées dans le fichier texte indiqué en argument.
- Instruction `si . . . va en` : si la condition est vraie alors l'exécution continuera à partir du numéro de ligne spécifié par l'étiquette, sinon l'exécution continuera en séquence.

Récits d'utilisation proposés lors de l'itération 3

3.1 Commande sauve

Récit d'utilisation

Titre : Commande sauve

Récit : Sauvegarde d'un programme dans un fichier

En tant que : Programmeur dans l'interpréteur LIR

Je souhaite : sauvegarder un programme LIR dans un fichier

Afin de : Pourvoir reprendre mon travail où je m'étais arrêté

Critères d'acceptation

À partir du fait : Qu'un programme (avec des étiquettes) ai été saisi

Alors : lorsque j'entre la commande sauve avec en argument le chemin du fichier (dans lequel on souhaite sauvegarder le travail) sauve <cheminFichier>

Enfin : les lignes de codes tapées dans l'interpréteur s'enregistrent dans le fichier passé en argument de la commande pour pouvoir être rechargées plus tard par l'interpréteur LIR avec la commande charge <cheminFichier>

3.2 Commande charge

Récit d'utilisation

Titre : Commande charge

Récit : Chargement d'un programme à partir d'un fichier

En tant que : Programmeur avec l'interpréteur LIR

Je souhaite : charger un programme LIR préalablement enregistré dans un fichier

Afin de : je puisse réutiliser un programme LIR sans repartir de zéro.

Critères d'acceptation

À partir du fait : un fichier contenant un programme LIR sur mon ordinateur

Alors : lorsque j'entre la commande charge avec en argument le chemin de ce fichier

Enfin : les lignes de codes enregistrées dans le fichier sont chargée dans le programme pour pouvoir être exécutées et/ou modifiées par l'interpréteur LIR

3.3 Instruction si... vaen

Récit d'utilisation

Titre : Instruction Si...vaen

Récit : Sauts conditionnels

En tant que : programmeur

Je souhaite : effectuer un saut vers une ligne spécifique d'un programme si la condition est remplie.

Afin de : Créer des branchements ou des itérations dans mes programmes.

Critères d'acceptation

À partir de : la saisie d'un programme

Alors : j'entre la commande `si` suivie de la condition à remplir `va en` suivie du numéro de la ligne où je veux effectuer le saut.

Enfin : lors de l'exécution de l'instruction, le programme ignorera les lignes suivantes et sautera directement à la ligne indiquée si il valide la condition imposée.