**Projet: Compilation du langage C-Pascal**

Le but de ce projet est de réaliser un petit compilateur d'un langage s'inspirant de C et de Pascal, appelé C-Pascal.

1. **Spécification du langage C-Pascal**

* **Structure du programme** : <déclarations><fonctions>
* **Les variable:**  doivent être déclarées avant utilisation. Les variables sont de type entier ou tableau d'entiers à une dimension.
* **Les identificateurs** :

il n'y a aucune distinction entre les minuscules et les majuscules dans le cas des identificateurs,

ils doivent commencer par une lettre,

ils ne peuvent porter le nom d'un mot clé réservé,

elles ne peuvent être déclarées qu'en début de programme (globales) ou qu'au début de la définition d'une fonction (locales).

* **Les constantes :**

elles sont entières et exprimées en base 10,

* **Les commentaires :** le commentaire est sur une seule ligne et débute par **#.**
* **Types simples :**  Le langage C-Pascal connaît un seul type simple, le type entier.
* **Types dérivés:**  possède un unique type dérivé, le type « tableau d’entiers à un indice ». Une variable de type tableau est **nécessairement globale**. Une variable de ce type est déclarée à l’aide de crochets après le nom de la variable. La taille d’un tableau doit être un nombre entier positif.
* **Déclarations de variables:** Une variable doit être déclarée avant de pouvoir être utilisée. Une déclaration de variable est constituée d’un nom de type (entier) suivi d’un nom de variable suivi, optionnellement, d’une taille constante de tableau entre crochets.On peut déclarer plusieurs variables en séparant les déclarations les unes des autres par une virgule.
* **Opérateurs:** On s’en tiendra aux opérateurs arithmétiques ( +, -, \* et /) de comparaison (au moins: <, <= et =) et logiques (& (et), | (ou) et ! (non)).
* **Instructions:** Le langage C-Pascal connaît les instructions suivantes :
  + Affectation : a = b + 1; (contrairement à C, une affectation n’est pas une expression, elle ne correspond pas à une valeur).
  + **si** expression **alors** { ... } et **si** expression **alors** { ... } **sinon** { ... }
  + **tantque** expression **faire** { ... }
  + **retour** expression ;
  + Appel de fonction simple: nom\_fonction( liste d’expressions ); et à la fonction prédéfinie ecrire( expression );
  + Bloc d’instructions, délimité par des accolades { ... }
  + instruction vide ;

**NB:**Les instructions de type vide, affectation, retour et appel de fonction sont obligatoirement suivies d’un point-virgule.

Les accolades autour des blocs d’instruction du si et du tantque sont obligatoires.

* **Fonctions:** Le langage C-Pascal permet de définir des fonctions à résultat entier. Leurs arguments sont simples (i.e. des entiers, les tableaux sont des variables globales ) et le passage se fait par valeur. Les fonctions peuvent avoir des variables locales.

Les parties d’une fonction doivent être déclarées dans l’ordre suivant :

* + en-tête,
  + déclaration des variables locales,
  + bloc d’instructions du corps de la fonction.

La structure d'une fonction est donc :

  identificateur (<liste\_paramètres>)

<declarations>

{

<instructions>

}

* A la différence du C, les variables locales sont déclarées entre l’en-tête et le bloc d’instructions.
* L’en- tête d’une fonction est un identificateur pour nommer la fonction, suivi d’une liste de déclarations d’arguments entre parenthèses.
* Une fonction ne peut pas être déclarée à l’intérieur d’une autre : les fonctions sont toutes globales.
* Un programme en C-Pascal est une suite de déclarations de fonctions.
* On ne peut pas déclarer deux fonctions avec le même nom (pas de surcharge) et on doit déclarer toutes les fonctions appelées (sauf les fonctions spéciales lire et ecrire).
* La fonction appelée en premier dans un programme en C-Pascal est celle qui s’appelle main.
* L’appel d’une fonction comme si c’était une procédure – c’est-à-dire en ignorant le résultat rendu – est permis, comme en C.
* **Procédures prédéfinies:** Les entrées-sorties de base se présentent comme des appels à des fonctions prédéfinies lire et ecrire.

Exemple :

— lecture : variable = lire();

— écriture : ecrire( expression );

À part le fait qu’elles soient prédéfinies, ces deux fonctions sont traitées comme des fonctions ordinaires, écrites par les programmeurs C-Pascal. En particulier, le passage de l’argument de la fonction ecrire et le retour du résultat de la fonction lire se fait comme pour les fonctions ordinaires.

* **Exemple: Un programme en C-Pascal:**

entier t[10], entier a; # déclaration de variables globales

f(entier a, entier b) # déclaration d'une fonction à deux args

entier c, entier k; # déclaration de deux variables locales

{ # début d'un bloc d'instruction

k = a + b; # affectation et expression arithmétique

retour k; # valeur de retour de la fonction

} # fin du bloc d'instruction

main() # point d'entrée dans le programme

entier d;

{

a = lire(); # appel à la fonction lire

tantque 0 < a faire { # instruction tantque avec expression

d = f(d, 2 \* a); # affectation et appel de fonction

ecrire(d + 1); # appel de la fonction prédéfinie ecrire

a = a - 1;

}

}

**Expressions arithmétiques :**

Les expressions arithmétiques en C-Pascal sont composées de variables simples, d’accès à une case d’un tableau, de nombres entiers, d’appels de fonctions (dont la fonction spéciale lire), d’opérateurs arithmétiques et logiques et de parenthèses. Il n’y a pas de distinction entre les expressions arithmétiques et logiques : comme en C, les opérateurs logiques renvoient zéro pour représenter FAUX, et un autre entier différent de zéro pour représenter VRAI. Dans un même niveau de précédence, tous les opérateurs sont associatifs à gauche (c’est-à-dire, 5 − 3 + 2 équivaut à (5 − 3) + 2). Voici dans l’ordre décroissant la priorité des opérateurs :

**1 ( expression )**

**2 ! (non)**

**3 \*** , **/**

**4 +** ,  **-**

**5 =** ,  **< , < =**

**6 & (et)**

**7 | (ou)**

**Personnalisation du langage**

Chaque équipe intégrera un nouvel élément dans le langage C—Pascal, le choix de l'ajout se fera par tirage au sort  parmi les options 1, 2, 3, 4 , 5 et 6.

**Option 1 :**Intégrer à la grammaire le type caractère **Car.** On manipulera donc les caractères à l’aide des variables simples, des variables tableaux ou des constantes. Le caractère est considérer comme un entier (code ASCII) comme en C. remarquez que les instructions lire et écrire se basent sur le type pour traiter leurs arguments comme caractère ou comme entier.

**Option 2** :Intégrer à la grammaire le **tableau d'entiers à deux dimensions.**

**Option 3** : Intégrer à la grammaire le type **structure avec des champs de type entier.**

**Option 4** :Intégrer à la grammaire le type **reel.** On manipulera donc les réels à l’aide des variables simples, des variables tableaux ou des constantes (12.18, - 15 e3, ….). Remarquez que les instructions lire et écrire se basent sur le type pour traiter leur argument comme réel ou comme entier.

**Option 5** : Intégrer à la grammaire le type **chaine.** On manipulera donc les chaînes à l’aide des variables simples, des variables tableaux ou des constantes (˝Bonjour˝, ˝chaîne longue˝, ….). Remarquez que les instructions lire et écrire se basent sur le type pour traiter leurs arguments comme chaîne de caractères ou comme entier.

**Option 6 :**Intégrer à la grammaire le type **Bool.** On manipulera donc les booléens à l’aide des variables simples, des variables tableaux ou des constantes ( **faux**, **vrai**). Remarquez que les instructions lire et écrire se basent sur le type pour traiter leurs arguments comme Booléen ou comme entier.

**II.Le Langage intermédiaire P-machine**

La mémoire de la machine P-Machine est faite de cellules numérotées, organisées comme le montre la figure. Les registres suivants ont un rôle essentiel dans le fonctionnement de la machine :

• CO (Compteur Ordinal) indique constamment la cellule contenant l'instruction que la machine est en train d'exécuter ;

• BEG (Base de l'Espace Global) indique la première cellule de l'espace réservé aux variables globales (autrement dit, BEG pointe la variable globale d'adresse 0) ;

• BEL (Base de l'Espace Local) indique la cellule autour de laquelle est organisé l'espace local de la fonction en cours d'exécution ; la valeur de BEL change lorsque l'activation d'une fonction commence ou finit ;

• SP (Sommet de la Pile) indique constamment le sommet de la pile, ou plus exactement la première cellule libre au-dessus de la pile, c'est-à-dire le nombre total de cellules occupées dans la mémoire.

P-Machine une machine à pile.

**Exemple en P-Machine**: Programme qui lit deux entiers et affiche le carré de leur somme

LIRE # lit le premier entier et le met en pile

LIRE # lit le premier entier et le met en pile

ADD # dépile deux fois empile la somme

DUP # duplique le sommet de pile

MUL # dépile deux fois empile le produit

ECRV # affiche le résultat

STOP

**Répertoire d'instructions de P-Machine**

| opcode | opérande | explication |
| --- | --- | --- |
| EMPC | valeur | EMPiler Constante. Empile la valeur indiquée. |
| EMPL | adresse | EMPiler la valeur d'une variable Locale. Empile la valeur de la variable déterminée par le déplacement relatif à BEL donné par adresse (entier relatif). |
| DEPL | adresse | DEPiler dans une variable Locale. Dépile la valeur qui est au sommet et la range dans la variable déterminée par le déplacement relatif à BEL donné par adresse (entier relatif). |
| EMPG | adresse | EMPiler la valeur d'une variable Globale. Empile la valeur de la variable déterminée par le déplacement (relatif à BEG) donné par adresse. |
| DEPG | adresse | DEPiler dans une variable Globale. Dépile la valeur qui est au sommet et la range dans la variable déterminée par le déplacement (relatif à BEG) donné par adresse. |
| EMPT | adresse | EMPiler la valeur d'un élément de Tableau. Empile la valeur qui est au sommet de la pile, soit i cette valeur. Empile la valeur de la cellule qui se trouve i cases au-delà de la variable déterminée par le déplacement (relatif à BEG) indiqué par adresse. |
| DEPT | adresse | DEPiler dans un élément de Tableau. Dépile une valeur v, puis une valeur i. Ensuite range v dans la cellule qui se trouve i cases au-delà de la variable déterminée par le déplacement (relatif à BEG) indiqué par adresse. |
| ADD |  | ADDition. Dépile deux valeurs et empile le résultat de leur addition. |
| SOUS |  | SOUStraction. Dépile deux valeurs et empile le résultat de leur soustraction. |
| MUL |  | MULtiplication. Dépile deux valeurs et empile le résultat de leur multiplication. |
| DIV |  | DIVision. Dépile deux valeurs et empile le quotient de leur division euclidienne. |
| MOD |  | MODulo. Dépile deux valeurs et empile le reste de leur division euclidienne. |
| EGAL |  | Dépile deux valeurs et empile 1 si elles sont égales, 0 sinon. |
| INF |  | INFerieur. Dépile deux valeurs et empile 1 si la première est inférieure à la seconde, 0 sinon. |
| INFEG |  | INFerieur ou EGal. Dépile deux valeurs et empile 1 si la première est inférieure ou égale à la seconde, 0 sinon. |
| NON |  | Dépile une valeur et empile 1 si elle est nulle, 0 sinon. |
| LIRE |  | Obtient de l'utilisateur un nombre et l'empile. |
| ECRIV |  | ECRIre Valeur. Extrait la valeur qui est au sommet de la pile et l'affiche. |
| SAUT | adresse | Saut inconditionnel. L'exécution continue par l'instruction ayant l'adresse indiquée. |
| SIVRAI | adresse | Saut conditionnel. Dépile une valeur et si elle est non nulle, l'exécution continue par l'instruction ayant l'adresse indiquée. Si la valeur dépilée est nulle, l'exécution continue normalement. |
| SIFAUX | adresse | Comme ci-dessus, en permutant nul et non nul. |
| APPEL | adresse | Appel de sous-programme. Empile l'adresse de l'instruction suivante, puis fait la même chose que SAUT. |
| RETOUR |  | Retour de sous-programme. Dépile une valeur et continue l'exécution par l'instruction dont c'est l'adresse. |
| ENTREE |  | Entrée dans un sous-programme. Empile la valeur courante de BEL, puis copie la valeur de SP dans BEL. |
| SORTIE |  | Sortie d'un sous-programme. Copie la valeur de BEL dans SP, puis dépile une valeur et la range dans BEL. |
| PILE | nbreMots | Allocation et restitution d'espace dans la pile. Ajoute nbreMots, qui est un entier positif ou négatif, à SP |
| DUP |  | Duplique le sommet de pile (erreur si pile vide) |
| POP |  | Dépile le sommet de la pile (il est perdu) |
| STOP |  | STOP |

**III. Travail demandé et consignes à respecter**

Le projet est à faire par équipe de trois personnes. Il doit être remis par email à l’adresse **elghazis@gmail.com**. Votre projet doit se présenter sous forme d'une archive compressée (rar ou zip). Le nom de l'archive compressé est défini ainsi: vos\_noms\_famille\_GI2.rar( ex : Nom1\_Nom2\_Nom3\_GI2.rar).

**Consistance des livrables**

Dans cet archive doivent se trouver :

* Les sources du votre programme compilateur.
* Quelques exemples de codes source acceptés par votre programme et démontrant ses possibilités, ainsi que la sortie code cible correspondant.
* Quelques exemples de codes source erronés et refusés par votre compilateur pour montrer la gestion d’erreurs.
* Un rapport au format Word (version papier et version numérique) décrivant :
* Le but que s’est fixé chaque membre de l’équipe
* Les différents choix techniques qui ont été faits
* Les langages source et objet choisis
* Les fonctionnalités implémentées
* Des indications de prise en main (comment utiliser votre compilateur. . .
* Éventuellement, les difficultés rencontrées ou les éléments non réalisés. . .
* La description des solutions choisies et leurs argumentations

**livrables et délais**

**À rendre version papier et version numérique par mail** :

* Lundi 11 Novembre 2019 :
  + la désignation du chef d'équipe et la répartition des taches dans l'équipe
  + la grammaire du C-Pascal initiale en BNF
  + l'analyseur lexical.
  + programme de test du lexical en suivant le modèle de l'exemple fourni en fin de document.
* Lundi 18 Novembre 2019
  + La grammaire Prédictive (LL1) après transformation
  + la justification de LL(1):
    - tableau des premiers et des suivants
    - la vérification des productions
* Lundi 25 Novembre 2019:
  + L'analyseur syntaxique du langage C-Pascal,
  + L'affichage de l'arbre de dérivation en XML,
  + L'affichage des erreurs détectées par le syntaxique
* Lundi 2 Décembre 2019:
  + l'analyseur sémantique avec le contrôle de type
* Lundi 9 Décembre 2019:
  + Générateur du code intermédiaire P-Machine
  + L'interpréteur de P-Machine.
* Lundi 16 Décembre 2019
  + projet complet (lexical, syntaxique, sémantique et génération du code)
  + jeu complet de programmes en C-Pascal (source, compilé et exécution).

* Mardi 24 Décembre 2019 : soutenances et démonstrations.

**Critères d’évaluation**

Ce projet est un travail d’équipe de 3 ou 4 personnes, chaque membre doit avoir une partie importante(~ 1/3 ou 1/4) du projet à réaliser. Une soutenance aura pour objectif de démontrer que chaque membre est réellement l’auteur de sa propre partie et être capable d’expliquer la problématique des autres et apporter des éléments de réponse quant à sa résolution.

Le projet sera évalué selon les critères suivants :

* Qualité du projet (choix techniques, lisibilité, présentation,..)
* Prise en main (Est-ce que tout marche du premier coup ; qualité et complétude des exemples;. . .)
* Le non-respect des consignes (délais, livrables, . . .)
* la note de chaque membre est pondérée par sa participation dans le projet et la qualité de sa soutenance

**Sortie du programme de test du lexical du langage C-Pascal**

programme en C-Pascal

entier e;

main()

{

e = 153;

}

Output du lexical

**Lexème lu Unité Lexicale(Code) Attribut(valeur du lexème)**

entier ENTIER ?

e IDENT e

; POINT\_VIRGULE ?

main MAIN ?

( PARENTHESE\_OUVRANTE ?

) PARENTHESE\_FERMANTE ?

{ ACCOLADE\_OUVRANTE ?

e IDENT e

= EGAL ?

153 NOMBRE 153

; POINT\_VIRGULE ?

} ACCOLADE\_FERMANTE ?

EOF FIN ?