INITIAT CODES

*{****SYNTAX DEFINITIONS****}*

Un langage destiné académique.

Desc :

Est un langage destine au monde académique.

Pour permettre de promouvoir et donne le gout aux nouveaux apprenant de l’information en programmation la soupe qui se cache derrière cette merveille de l’ordinateur.

Le guide et le montre la structure d’un langage de programmation.

Bien les initier au langages informatique et les montre le chemin de la structure en français vers l’anglais.

LES MOTS CLE : il ne doivent pas être définie dans un code puis qu’ils le sont déjà et le compilateur les traite automatiquement

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| fonction | classe | var | methode | PRINCIPAL | si | sinon |
| pour | tanq | boucler | herite | dans | lire | ecrire |
| ecrirel | et | eg | affect | constructeur | arreter | lancer |
| parRef | finApp | allervers | pause | inst | retourner | native |
| entier | chaine | objet | reel | bool | caractere | sup |
| inf | eq | dif | non | vrais | faut | fait |
| alors | fois | comprend | contien | ff | fsi | fclasse |
| fpour | ftanq | fboucle | DEBUT | FIN | DEBUTPROGRAMME | FINPROGRAMME |
|  |  |  |  |  |  |  |

LA STRICTURES

|  |  |
| --- | --- |
|  | Le programme doit commencer avec une ligne vide ou un préprocesseur au cas où un préprocesseur ne pas fournis. |
| préprocesseurs | #mode console  DEBUTPROGRAMME : ‘information sur le programme’  //expressions  fonction PRINCIPAL *fait*  *ecrire* : ‘je suis un programme’  *ff*  FINPROGRAMME |
| Info sur le pourquoi |
| Fonction principal |
|  |
|  |
|  |

LES PREPOCESSEURS

|  |  |
| --- | --- |
| #mode [ console | graphique ] | Cette mode est en état CONSOLE par défaut  Donc Console va ouvrir la console quand le programme sera exécuté. GRAPHIQUE ouvrira des fenêtres de dialogue . |
| #indentation [ oui | non ] | Par défaut cette mode est activer OUI, si oui alors le compilateur va exiger des indentations a des adroits approprier.  IL NE FAUT PAS MELANGER LES TABS AVEC LES ESPACES DANS VOS IDENTATIONS sinon erreur |
| #inporter ‘nom du fichier à importer’ | Cette préprocesseur importe le fichier indique et le colle exactement à l’endroit où il était appelé. Donc le répertoire de travail de cette fichier importer sera le même que du fichier principale. |

LES STRICTURE DES CONTROLE

|  |  |  |
| --- | --- | --- |
| La clé | syntaxe | description |
| Si  Sinon | si *expression* *alors*  // *expressions*  sinon si expression *alors*  // expressions  sinon *alors*  // expressions  *fsi* | Expression sont tous les expressions valide.  Exp : les expressions conditionnelles  Arithmétique  Booléenne  Appel à une fonction  Appel à une objet |
| boucler | boucler expression *fois*  // expressions  *fboucle* | Le valeur de l’expression doit être une valeur Numérique en sachant que les expressions booléenne renvoi 1 ou 0 et celle conditionnel. |
| tanq | tanq expression *fait*  // expressions  *ftanq* |  |
| pour | pour index et valeur dans objet *fait*  // expressions  *fpour* | Objet doit être une valeur itterable comme un objet de tableau ou un objet des valeurs indexe par nom. |
| native | {  *// code native AHK*  } | Du code native AHK ( autohotkey ) |

FONCTIONS, CLASSES ET OBJETS

|  |  |  |
| --- | --- | --- |
| LES CLES | SYNTAXS | DESCRIPTIONS |
| classe  méthodes | classe nom [herite] [*contien|comprend*]  constructeur: param *fait*  inst variable = ‘’  *ff*  var inst chaine x  var local entier y  methode entier fn: param *fait*  // expressions  *ff*  *fclasse* | Params : paramètres :  type nom  type nom = expression  *parRef* type nom  parRef (par référence : cette variable est référencer a l’extérieur de la fonction vers l’intérieur |
| fonction | fonction entier nom: params *fait*  // expressions  *ff* |  |
| objet | . var objet x = new nom( params )  . var objet y = { cles : valeur, … } ou {}  . var objet z = [ valeur, … ] ou []  . x.fn()  . x.variable  . y.cle  . y[‘cle’]  . z[index] |  |

DECLARATION DE VARIABLE, TYPE ET AFFECTATIONS

|  |  |  |
| --- | --- | --- |
| CLES | SYNTAXS | DESCRIPTIONS |
| var | var type nom |  |
| types | entier  chaine  caractere  bool  reel |  |
| affectations | <--  =  eg  affect | Affecter dans.  Affecter dans.  Affecter dans (égale)  Affecter dans (affecter) |
| Variables du langage | IC\_nom  IC\_index | Nom est le nom approprier |
| IC\_index est une variable qui est déclarer automatiquement dans tout le boucles |

LES SEPARATEUR DES VALEURS ET PARAMETRES

|  |  |
| --- | --- |
| et et , | Exp : fn(a , b et c ) |

LES OPERATEURS CONTIONNEL

|  |  |
| --- | --- |
| OPERATEURS |  |
| eq | équivaux |
| sup | supérieur |
| inf | inferieur |
| non | Non logique ou diffèrent |
| faux | Valeur booléenne qui est équivaut a 1 et 0 |
| vrais |
| dif | différent |
| Et | Et logique |
| ou | Ou logique |
| < > <= >= != == | commune |
| `n | Caractère de saut de ligne dans un chaine des caractères |

QUELQUES MOTS CLES ET SYNTAX

|  |  |  |
| --- | --- | --- |
| MOTS CLES | SYNTAX | DESC |
| ecrire | ecrire: … ; |  |
| ecrirel | ecrirel: … ; |  |
| lire | lire: … ; |  |
| pause | pause [milisec] |  |
| lancer | lancer: … | Lancer un programmer. Cette fonction est une substitue de la fonction divers.lancer(‘ ’) |
| : | Exp : lire: ‘valeur’ ;  Fonction  Methode  Lire  Ecrire  lancer | Cette operateur doit être lieu au nom de la fonction et a ces mots clés indiquer dans la syntaxe, sinon erreur lors de l’exécution. |
| ` | `n  `r  `’  `’’ | Nouvelle ligne.  Retour chariot (entrer).  Echapper une apostrophe.  Echapper un guillemet. |

LES CLASSES DU LANGAGE

ATTENTION : ceci est une liste des classes propre à cette langage. Ils doivent être écrite comme décrite car elles ne sont pas prises en charge par le PARSER Syntaxique mais néanmoins elles sont soumises à certains règles mais minime et parser ne le vérifie pas ce qui peut causer une erreur d’exécution.

S’ils sont males autographier elles peuvent générer une erreur d’exécution. Du côté académique ceci aide l’étudiant à bien tenir compte de l’autographe de ses codes.

|  |  |
| --- | --- |
| Objet/méthode | DESC |
| console | La classe console est disponible dans le mode console. Activable par le préprocesseur :  #mode console  Cette préprocesseur est activer automatiquement par défaut.  Car elle active tous les fonctionnalités de la console même si certaine fonctionnalité sont automatiquement prises en charge.  Ex : console.Ecrire(‘language IL’) |
| NLinge(n) | Insère plusieurs lignes vide dans la console. |
| Met(txt:="") | Ecrire sur la console. sur une seul ligne. |
| Ecrire(txt:="") | Ecrire du texte dans la console. Cette fonction supporter l’affichage sur plusieurs lignes.  Utiliser la caractère de saut de ligne **`n** |
| Lire(ByRef variable:="") | Cette fonction lie le clavier jusqu’à ce que l’utilisateur appuie sur la touche Entrer |
| Cle() | Lie un caractère du clavier. Cette fonction attend le retour utilisateur d’un touche du clavier. |
| Efacer() | Effacer l’écran de la console |
| tableCouleurs() | Affiche un tableau des couleurs dans la console. |
| Couleur(avantPlan, arrierPlan) | Changer la couleur de la console. *En hexadécimal* |
| CouleurFG(fg) | Changer la couleur d’avant plan d’un marceau de texte dans la console |
| CouleurBG(bg) | Changer la couleur d’arrière-plan d’un marceau de texte dans la console. |
| Attend(milisec) | Mettre la console en attente dans une délai définie |
| Pause(n:=1)  BarPorgession(longeur, actuel, Max, Unlock = 0, fixed=1,  lp="|", lba="[", lbb="]") | Mettre la console en pause. Ça évite que le console se ferme après une exécution.  Cette appel fait que la console attend une entre du clavier pour ce fermer. |
| Ajouter un bar de progression dans la console.  Ex : console.BarProgression(100, 10, 100) |
| MATH  Les fonctions mathématique | |
| Méthode de la classe math | Ex : math.abs(2.6) |
| abs(nombre) | Renvoie la valeur absolu |
| arrondd(nombre) | Arrondir simplement |
| exp(nombre) | exponentiel |
| aronde(nombre) | Arrondir par excès |
| log(nombre) | logarithme |
| ln(nombre) | ln |
| max(nombre, n\*) | Valeur max |
| mod(diviseur, dividende) | Modulo : la valeur du reste d’une division |
| arond(nombre, n:=0) | Renvoie un valeur aléatoire entre nombre et n |
| rcarre(nombre) | Racine carre |
| cos(nombre) | cosinus |
| sin(nombre) | sinus |
| tan(nombre) | tangente |
| DIALOGUE  Boites de dialogues | |
| Méthodes de la classe divers | Ex : dialogue.Msg(‘une langage très simple’) |
| Msg(txt, titre:="Boite de dialogue", timeout:=2147483, opt:=0) | Affiche une boite de dialogue |
| Entrer(Title:="Boite d'Entrer", Prompt:="", holdplace:="", pwd:=faux, hauteur:=189, largeur:=375) | Affiche une boite de dialogue qui demande un saisi d’entrer.  Prompt : est un texte qui sera afficher comme message.  Holdplace : est un message exemplaire qui soutient le champ de texte.  Psw : si le champs est en mode mot de passe. |
| SYSTEME DES FICHIERS | |
| La classe système de fichier contient des sous classes | Ex : sf.rep.copier(‘mon\_fichier.txt’, ‘vers\_ici’)  Ex : sf.fichier.ecrire(‘cette langage est très simple’, ‘mon\_fichier.txt’)  Ex : sf.recycle(‘mon\_fichier.txt’) |
| sf | Classe principale système de fichier. |
| rep | Sous classe de gestion des répertoire |
| copier(src, dest, opt:=true) | Copier un répertoire entier avec les sous répertoires. Cette appel affiche la boite de dialogue Windows de copie. |
| cree(chemain) | Cree un dossier ou répertoire |
| sup(chemin, rec:=true) | Supprimer un dossier avec tout ce que il contient |
| dep(src, dest, opt:=true) | Déplacer un dossier entière |
| select(chemain, aid:="", opt:=true) | Ouvrir la boite de dialogue Windows qui demande de sélectionner un dossier.  Cette fonction renvoie le chemin absolue du dossier sélectionner. |
| fichier | Sous classe de gestion des fichiers |
| ecrire(txt, chemain) | Ecrire dans un fichier ou crée une si le fichier n’existe pas. |
| copier(src, dest, opt:=true) | Copier un fichier. Cette appel affiche la boite de dialogue Windows de copie |
| sup(chemain) | Supprimer un fichier |
| dep(src, dest, opt:=true) | Déplacer un fichier |
| Contenus(fichier) | Cette fonction renvoie le contenue d’un fichier. |
| select(chemain:="", opt:="M", filtre:="(\*.\*)", aide:="" ) | Cette appel affiche la fenetre de sélection d’un fichier ou plusieurs si opt et régler a M : multiple et S : single. |
| Ces méthodes sont des méthodes de la classe principale sf  Ex : sf.recycler(‘mon\_fichier.txt’) | |
| recycler(chemain) | Recycler un fichier ou un répertoire dans la Corbeil |
| recycler0(lecteur) | Vider la Corbeil |
| racourcie(chemin, lien , WorkingDir , Args:="", Description:="") | Cree un raccourcie pointant vers un programme ou un fichier ou dossier.  Lien : nom |
| DIVERS  Des fonctions divers | |
| divers | Ex : divers.lancer(‘mon\_programme.exe’)  Ex : divers.lancer(‘shutdown /h’) |
| lancer(Target) | Elle agis comme la commande Exécuter ou cmd ( #+R ) |
| editer(fichier) | Edit un fichier. Cette appel ouvre le programme par défaut pour l’édition du fichier en question. Ex : Word |
| SON  Gestion de son , de la volume et jouer de la musique ou des sons | |
| son | Ex : son.bip(666, 2023) |
| bip(freq:=523, dure:=150) | Faire sonner un bip avec des dure |
| jouer(chemain, attend:=false) | Jouer de la musique ou une voix.  Cette fonction supporte les format : mp3, aac, wave |
| vol(Percent) | Jouer avec la volume de Windows en pourcentage. |
| LECTEUR  Lecteur amovible, CD, HDD | |
| lecteur | Ex : lecteur.eject() |
| eject() | Ejecter un lecteur CD |
| capaciter(chemain) | Renvoie la capacité du dossier ou lecteur en Ko |
| libeler(lecteur) | Affiche la libeller du lecteur spécifier |
| liste() | Renvoie la liste des lecteurs disponible et visible sur le système |
| chLibeler(lecteur, valeur) | Changer la libeller d’un lecteur |
| RECOURCIS  Cree des raccourcis clavier comme bon vous plaise mais tenant compte que cette classe est à utiliser seulement pour des personne expérimenter ou avec de notion du langage AHK | |
| racourcis | Ex : racourcis.clavier(‘#R’, ‘mon\_function’) |
| clavier(cles, libele) |  |
| chaine(chaine, replacent) |  |